Code-Bibliothek

AWS SDK-Codebeispiele

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK-Codebeispiele: Code-Bibliothek

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

..... xxxi
Was ist die Codebeispielbibliothek? 1
Arbeitet mit AWS SDKs 1
Codebeispiele nach Service 3
ACM 8
Grundlagen 9
API Gateway 113
Grundlagen 114
Szenarien 163
AWS Beiträge der Gemeinschaft 188
Management-API von API Gateway 190
Grundlagen 191
Application Auto Scaling 193
Grundlagen 194
Application Recovery Controller 231
Grundlagen 232
Athena 242
Szenarien 243
Audit Manager 259
Szenarien 259
Aurora 272
Grundlagen 274
Szenarien 665
Auto Scaling 671
Grundlagen 674
Szenarien 1016
AWS Batch 1184
Grundlagen 1186
Szenarien 1251
Amazon Bedrock 1265
Grundlagen 1266
Szenarien 1293
Amazon Bedrock Runtime 1294
Grundlagen 1297

Szenarien ... 1305
Amazon Nova ... 1370
Amazon Nova Canvas ... 1444
Amazon Nova Reel ... 1455
Amazon Titan Image Generator ... 1462
Amazon Titan Text ... 1468
Amazon Titan Text Embeddings ... 1470
Anthropic Claude ... 1475
Cohere Command ... 1530
DeepSeek ... 1556
Meta Llama ... 1558
Mistral AI ... 1579
Stable Image Core ... 1596
Agenten für Amazon Bedrock ... 1608
Grundlagen ... 1610
Szenarien ... 1674
Runtime der Agenten für Amazon Bedrock ... 1708
Grundlagen ... 1710
Szenarien ... 1728
AWS Budgets ... 1746
Grundlagen ... 1746
AWS Cloud Map ... 1750
Szenarien ... 1751
AWS Cloud9 ... 1793
Grundlagen ... 1794
CloudFormation ... 1815
Grundlagen ... 1817
Szenarien ... 1853
CloudFront ... 1877
Grundlagen ... 1879
Szenarien ... 1948
CloudFront Funktionen, Beispiele ... 1989
CloudTrail ... 2005
Grundlagen ... 2006
CloudWatch ... 2031
Grundlagen ... 2033

Szenarien 2297
CloudWatch Events 2344
Grundlagen 2344
CloudWatch Protokolle 2357
Grundlagen 2359
Szenarien 2424
CodeBuild 2493
Grundlagen 2494
CodeCommit 2510
Grundlagen 2511
CodeDeploy 2529
Grundlagen 2531
CodePipeline 2593
Grundlagen 2594
Amazon Cognito Identity 2651
Grundlagen 2652
Szenarien 2677
Amazon Cognito Identity Provider 2679
Grundlagen 2681
Szenarien 2840
Amazon Cognito Sync 3017
Grundlagen 3018
Amazon Comprehend 3020
Grundlagen 3022
Szenarien 3105
AWS Config 3137
Grundlagen 3139
Szenarien 3177
Kunde Connect 3188
Szenarien 3189
AWS Control Tower 3201
Grundlagen 3202
Firehose 3364
Grundlagen 3365
Szenarien 3377
Device Farm 3390

Grundlagen 3391
Szenarien 3394
Direct Connect 3405
Szenarien 3406
Directory Service 3415
Grundlagen 3416
AWS DMS 3421
Grundlagen 3422
Amazon DocumentDB 3425
Szenarien 3426
Serverless-Beispiele 3441
DynamoDB 3453
Grundlagen 3457
Szenarien 4033
Serverless-Beispiele 4703
AWS Beiträge der Community 4724
Amazon EBS 4726
Grundlagen 4726
Amazon EC2 4730
Grundlagen 4737
Szenarien 5907
Amazon ECR 6359
Grundlagen 6360
Szenarien 6477
Amazon ECS 6485
Grundlagen 6486
Szenarien 6570
Amazon EFS 6627
Grundlagen 6628
Amazon EKS 6644
Grundlagen 6645
Szenarien 6708
Elastic Beanstalk 6721
Grundlagen 6722
Elastic Load Balancing – Version 1 6731
Grundlagen 6733

Elastic Load Balancing – Version 2 6800
Grundlagen 6802
Szenarien 6943
ElastiCache 7120
Szenarien 7120
MediaStore 7128
Grundlagen 7129
Amazon EMR 7150
Grundlagen 7151
Szenarien 7175
AWS Entity Resolution 7189
Grundlagen 7190
OpenSearch Dienst 7271
Grundlagen 7272
Szenarien 7308
EventBridge 7322
Grundlagen 7324
Szenarien 7444
EventBridge Scheduler 7467
Grundlagen 7468
Szenarien 7496
AWS FIS 7546
Szenarien 7546
Forecast 7562
Grundlagen 7563
Amazon Glacier 7577
Grundlagen 7578
Szenarien 7644
AWS Glue 7656
Grundlagen 7657
Szenarien 7959
AWS Health 7967
Grundlagen 7967
HealthImaging 7975
Grundlagen 7976
Szenarien 8134

HealthLake 8210
Grundlagen 8211
IAM 8257
Grundlagen 8263
Szenarien 9052
Amazon Inspector 9543
Grundlagen 9544
AWS IoT 9596
Grundlagen 9597
Szenarien 9826
AWS IoT data 9856
Grundlagen 9857
AWS IoT SiteWise 9873
Grundlagen 9874
Amazon Keyspaces 10006
Grundlagen 10007
Kinesis 10125
Grundlagen 10126
Serverless-Beispiele 10187
AWS KMS 10211
Grundlagen 10213
Szenarien 10461
Lambda 10463
Grundlagen 10466
Szenarien 10748
Serverless-Beispiele 10895
AWS Beiträge der Gemeinschaft 11021
Amazon Lex 11023
Szenarien 11023
Lightsail 11025
Szenarien 11025
Amazon Location 11033
Grundlagen 11034
Location Service Places 11147
Grundlagen 11148
Amazon Managed Grafana 11154

Szenarien 11154
Managed Service für Apache Flink 11162
Grundlagen 11164
Datengenerator 11179
AWS Marketplace Katalog-API 11193
AMI-Produkte 11198
Angebote von Channel-Partnern 11257
Container-Produkte 11297
Entitäten 11312
Angebote 11323
Produkte 11462
Wiederverkaufsgenehmigung 11474
SaaS--Produkte 11573
Dienstprogramme 11631
AWS Marketplace Vereinbarungs-API 11638
Vereinbarungen 11640
MediaConnect 11899
Szenarien 11899
MediaConvert 11908
Grundlagen 11909
MediaLive 11957
Grundlagen 11957
MediaPackage 11959
Grundlagen 11959
Migration Hub 11965
Grundlagen 11966
Amazon MSK 11980
Szenarien 11980
Serverless-Beispiele 12007
Neptune 12017
Grundlagen 12018
Szenarien 12109
Organizations 12119
Grundlagen 12121
Szenarien 12162
Partner Central 12169

Grundlagen 12171
Szenarien 12236
AWS Payment Cryptography 12242
Szenarien 12242
Amazon Personalize 12247
Grundlagen 12249
Amazon Personalize Events 12298
Grundlagen 12299
Amazon Personalize Runtime 12306
Grundlagen 12307
Amazon Pinpoint 12315
Grundlagen 12316
Szenarien 12406
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API 12419
Grundlagen 12420
Amazon Polly 12439
Grundlagen 12440
Szenarien 12490
AWS-Preisliste 12505
Grundlagen 12506
Amazon RDS 12515
Grundlagen 12517
Szenarien 12786
Serverless-Beispiele 12798
Amazon RDS Data Service 12818
Grundlagen 12819
Szenarien 12821
Amazon Redshift 12826
Grundlagen 12827
Szenarien 12957
Amazon Rekognition 12986
Grundlagen 12988
Szenarien 13161
Resource Groups 13241
Grundlagen 13242
Resource Groups Tagging API 13269

Grundlagen 13270
Route 53 13283
Grundlagen 13284
Route-53-Domainregistrierung 13312
Grundlagen 13313
Amazon S3 13394
Grundlagen 13400
Szenarien 14089
Serverless-Beispiele 14542
Amazon S3 Control 14554
Grundlagen 14555
S3-Verzeichnis-Buckets 14634
Grundlagen 14636
Szenarien 14759
SageMaker KI 14761
Grundlagen 14763
Szenarien 14809
Secrets Manager 14925
Grundlagen 14926
Szenarien 14944
Amazon SES 14993
Grundlagen 14995
Szenarien 15126
Amazon SES API v2 15174
Grundlagen 15175
Szenarien 15257
Amazon SNS 15314
Grundlagen 15316
Szenarien 15511
Serverless-Beispiele 15733
Amazon SQS 15743
Grundlagen 15745
Szenarien 15931
Serverless-Beispiele 16151
Step Functions 16169
Grundlagen 16171

Szenarien 16282
AWS STS 16305
Grundlagen 16307
Szenarien 16339
Support 16590
Grundlagen 16591
Szenarien 16731
Systems Manager 16740
Grundlagen 16744
Szenarien 17144
Amazon Textract 17187
Grundlagen 17188
Szenarien 17224
Amazon Transcribe 17243
Grundlagen 17244
Szenarien 17315
Amazon Transcribe Streaming 17337
Grundlagen 17338
Szenarien 17353
Amazon Translate 17373
Grundlagen 17374
Szenarien 17392
VPC-Gitter 17404
Szenarien 17404
AWS WAFV2 17415
Grundlagen 17416
Szenarien 17420
WorkSpaces 17433
Grundlagen 17434
Szenarien 17467
X-Ray 17480
Grundlagen 17481
Codebeispiele von SDK 17509
SDK für .NET 17510
ACM 17512
API Gateway 17517

Aurora 17518
Auto Scaling 17520
Amazon Bedrock Runtime 17572
CloudWatch Protokolle 17613
Amazon Cognito Identity Provider 17627
Amazon Comprehend 17651
Amazon DocumentDB 17663
DynamoDB 17667
Amazon EC2 17733
Amazon ECS 17856
Elastic Load Balancing – Version 2 17867
EventBridge 17925
EventBridge Scheduler 17965
Amazon Glacier 17995
AWS Glue 18005
IAM 18036
Amazon Keyspaces 18133
Kinesis 18160
AWS KMS 18178
Lambda 18190
MediaConvert 18241
Amazon MSK 18252
Organizations 18254
Partner Central 18272
Amazon Pinpoint 18279
Amazon Polly 18285
Amazon RDS 18298
Amazon RDS Data Service 18337
Amazon Rekognition 18338
Route-53-Domainregistrierung 18369
Amazon S3 18396
SageMaker KI 18538
Secrets Manager 18572
Amazon SES 18575
Amazon SES API v2 18590
Amazon SNS 18628

Amazon SQS 18673
Schrittfunktionen 18717
AWS STS 18744
Support 18747
Amazon Textract 18774
Amazon Transcribe 18775
Amazon Translate 18787
SDK für .NET (v4) 18800
Aurora 18801
Auto Scaling 18843
Amazon Bedrock 18878
Amazon Bedrock Runtime 18882
CloudFormation 18901
CloudWatch 18905
CloudWatch Logs 18962
Amazon Cognito Identity Provider 18988
AWS Control Tower 18989
DynamoDB 19028
Amazon EC2 19069
Amazon ECS 19071
AWS IoT 19073
AWS IoT data 19132
Amazon Redshift 19135
Amazon S3 19165
AWS CLI mit Bash-Skript 19201
API Gateway 19203
Athena 19211
AWS Batch 19227
AWS Cloud Map 19240
CloudFormation 19282
CloudFront 19304
CloudWatch 19327
CloudWatch Logs 19362
Amazon Cognito Identity Provider 19391
AWS Config 19401
Kunden verbinden 19413

Direct Connect 19424
Amazon DocumentDB 19434
DynamoDB 19448
Amazon EC2 19554
Amazon ECR 19937
Amazon ECS 19945
Amazon EKS 19994
Elastic Load Balancing – Version 2 20006
ElastiCache 20015
Amazon EMR 20023
OpenSearch Bedienung 20033
AWS FIS 20047
AWS Glue 20063
HealthImaging 20071
IAM 20079
AWS IoT 20407
AWS KMS 20437
Lambda 20439
Lightsail 20479
Amazon Managed Grafana 20487
MediaConnect 20495
Amazon MSK 20503
Neptune 20529
AWS Payment Cryptography 20539
Amazon Pinpoint 20544
Amazon Polly 20556
Amazon RDS 20563
Amazon Redshift 20571
Amazon S3 20598
SageMaker KI 20691
Secrets Manager 20705
Amazon SES 20752
Amazon SNS 20760
Step-Funktionen 20798
AWS STS 20819
Support 21051

Systems Manager 21059
Amazon Textract 21101
VPC-Gitter 21109
AWS WAFV2 21119
WorkSpaces 21132
SDK für C++ 21144
ACM 21146
API Gateway 21176
Aurora 21177
Auto Scaling 21221
CloudTrail 21252
CloudWatch 21257
CloudWatch Protokolle 21267
CodeBuild 21271
Amazon Cognito Identity Provider 21276
DynamoDB 21299
Amazon EC2 21378
EventBridge 21412
AWS Glue 21416
HealthImaging 21455
IAM 21489
AWS IoT 21531
AWS IoT data 21573
Lambda 21576
MediaConvert 21600
Amazon RDS 21609
Amazon RDS Data Service 21646
Amazon Rekognition 21647
Amazon S3 21653
Secrets Manager 21735
Amazon SES 21737
Amazon SNS 21759
Amazon SQS 21801
AWS STS 21837
Amazon Transcribe Streaming 21839
SDK für Go V2 21847

API Gateway 21848
Aurora 21849
Amazon Bedrock 21889
Amazon Bedrock Runtime 21892
CloudFormation 21900
CloudWatch Protokolle 21902
Amazon Cognito Identity Provider 21904
Amazon DocumentDB 21986
DynamoDB 21988
IAM 22061
Kinesis 22122
Lambda 22125
Amazon MSK 22237
Partner Central 22238
Amazon RDS 22241
Amazon Redshift 22277
Amazon S3 22296
Amazon SNS 22385
Amazon SQS 22414
SDK für JavaScript (v2) 22446
CloudWatch 22447
CloudWatch Events 22456
CloudWatch Protokolle 22459
DynamoDB 22462
AWS Entity Resolution 22478
EventBridge 22496
Amazon Glacier 22499
IAM 22503
Lambda 22523
Amazon Pinpoint 22524
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API 22529
Amazon SNS 22532
Amazon SQS 22533
AWS STS 22543
SDK für JavaScript (v3) 22545
API Gateway 22548

Aurora 22549
Auto Scaling 22550
Amazon Bedrock 22593
Amazon Bedrock Runtime 22598
Agenten für Amazon Bedrock 22641
Runtime der Agenten für Amazon Bedrock 22654
CloudWatch 22659
CloudWatch Events 22668
CloudWatch Protokolle 22672
CodeBuild 22689
Amazon Cognito Identity 22692
Amazon Cognito Identity Provider 22693
Amazon Comprehend 22734
Amazon DocumentDB 22739
DynamoDB 22741
Amazon EC2 22935
Elastic Load Balancing – Version 2 23031
AWS Entity Resolution 23080
EventBridge 23093
Amazon Glacier 23099
AWS Glue 23101
HealthImaging 23127
IAM 23189
AWS IoT SiteWise 23282
Kinesis 23314
Lambda 23320
Amazon Lex 23377
Amazon Location 23377
AWS Marketplace API für Vereinbarungen 23409
Amazon MSK 23453
Amazon Personalize 23455
Amazon Personalize Events 23471
Amazon Personalize Runtime 23475
Amazon Pinpoint 23479
Amazon Polly 23484
Amazon RDS 23488

Amazon RDS Data Service 23493
Amazon Redshift 23494
Amazon Rekognition 23499
Amazon S3 23501
SageMaker KI 23635
Secrets Manager 23673
Amazon SES 23675
Amazon SNS 23701
Amazon SQS 23742
Step-Funktionen 23773
AWS STS 23774
Support 23776
Systems Manager 23794
Amazon Textract 23821
Amazon Transcribe 23827
Amazon Translate 23836
SDK für Java 2.x 23842
ACM 23845
API Gateway 23863
Application Auto Scaling 23869
Application Recovery Controller 23878
Aurora 23880
Auto Scaling 23916
AWS Batch 23980
Amazon Bedrock 24029
Amazon Bedrock Runtime 24034
CloudFront 24116
CloudWatch 24147
CloudWatch Events 24222
CloudWatch Loggt 24228
Amazon Cognito Identity 24247
Amazon Cognito Identity Provider 24254
Amazon Comprehend 24281
AWS Control Tower 24294
Firehose 24355
Amazon DocumentDB 24365

DynamoDB 24366
Amazon EC2 24629
Amazon ECR 24726
Amazon ECS 24766
Elastic Load Balancing – Version 2 24780
MediaStore 24824
AWS Entity Resolution 24839
OpenSearch Dienst 24887
EventBridge 24917
EventBridge Scheduler 24952
Forecast 24977
Amazon Glacier 24990
AWS Glue 25006
HealthImaging 25038
IAM 25068
Amazon Inspector 25152
AWS IoT 25200
AWS IoT data 25239
AWS IoT SiteWise 25242
Amazon Keyspaces 25281
Kinesis 25307
AWS KMS 25320
Lambda 25376
Amazon Lex 25414
Amazon Location 25415
Location Service Places 25462
AWS Marketplace Katalog-API 25467
AWS Marketplace API für Vereinbarungen 25662
MediaConvert 25771
Migration Hub 25792
Amazon MSK 25805
Neptune 25806
Partner Central 25849
Amazon Personalize 25884
Amazon Personalize Events 25913
Amazon Personalize Runtime 25917

Amazon Pinpoint 25921
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API 25965
Amazon Polly 25969
Amazon RDS 25976
Amazon RDS Data Service 26019
Amazon Redshift 26020
Amazon Rekognition 26059
Route-53-Domainregistrierung 26130
Amazon S3 26153
Amazon S3 Control 26343
S3-Verzeichnis-Buckets 26386
SageMaker KI 26468
Secrets Manager 26497
Amazon SES 26499
Amazon SES API v2 26515
Amazon SNS 26536
Amazon SQS 26605
Step-Funktionen 26704
AWS STS 26728
Support 26732
Systems Manager 26755
Amazon Textract 26798
Amazon Transcribe 26810
Amazon Transcribe Streaming 26821
Amazon Translate 26840
X-Ray 26842
SDK für Kotlin 26854
API Gateway 26856
Aurora 26857
Auto Scaling 26885
Amazon Bedrock 26903
Amazon Bedrock Runtime 26904
CloudWatch 26908
CloudWatch Protokolle 26948
Amazon Cognito Identity Provider 26952
Amazon Comprehend 26967

DynamoDB 26968
Amazon EC2 26997
Amazon ECR 27027
OpenSearch Dienst 27057
EventBridge 27060
AWS Glue 27092
IAM 27103
AWS IoT 27122
AWS IoT data 27147
Amazon Keyspaces 27149
AWS KMS 27174
Lambda 27184
Amazon Location 27193
MediaConvert 27224
Amazon Pinpoint 27229
Amazon RDS 27239
Amazon RDS Data Service 27257
Amazon Redshift 27258
Amazon Rekognition 27262
Route-53-Domainregistrierung 27281
Amazon S3 27300
SageMaker KI 27322
Secrets Manager 27348
Amazon SES 27349
Amazon SNS 27351
Amazon SQS 27378
Step-Funktionen 27401
Support 27422
Amazon Translate 27440
X-Ray 27441
SDK für PHP 27447
API Gateway 27448
Aurora 27455
Auto Scaling 27456
Amazon Bedrock 27471
Amazon Bedrock Runtime 27473

Amazon DocumentDB 27476
DynamoDB 27478
Amazon EC2 27511
AWS Glue 27516
IAM 27536
Kinesis 27553
AWS KMS 27557
Lambda 27593
Amazon MSK 27623
Amazon RDS 27625
Amazon RDS Data Service 27634
Amazon Rekognition 27635
Amazon S3 27636
S3-Verzeichnis-Buckets 27659
Amazon SES 27675
Amazon SNS 27676
Amazon SQS 27697
Tools für V5 PowerShell 27700
ACM 27702
Application Auto Scaling 27707
WorkSpaces Anwendungen 27714
Aurora 27741
Auto Scaling 27742
AWS Budgets 27779
AWS Cloud9 27780
CloudFormation 27787
CloudFront 27799
CloudTrail 27807
CloudWatch 27812
CodeCommit 27816
CodeDeploy 27821
CodePipeline 27841
Amazon Cognito Identity 27859
AWS Config 27863
Device Farm 27883
Directory Service 27884

AWS DMS 27910
DynamoDB 27911
Amazon EC2 27926
Amazon ECR 28060
Amazon ECS 28062
Amazon EFS 28068
Amazon EKS 28075
Elastic Load Balancing – Version 1 28088
Elastic Load Balancing – Version 2 28108
Amazon FSx 28132
Amazon Glacier 28140
AWS Glue 28144
AWS Health 28146
IAM 28147
Kinesis 28222
Lambda 28226
Amazon ML 28239
Macie 28245
AWS-Preisliste 28246
Resource Groups 28249
Resource Groups Tagging API 28257
Route 53 28262
Amazon S3 28277
Security Hub CSPM 28317
Amazon SES 28319
Amazon SES API v2 28320
Amazon SNS 28321
Amazon SQS 28322
AWS STS 28335
Support 28339
Systems Manager 28346
Amazon Translate 28420
AWS WAFV2 28421
WorkSpaces 28422
SDK für Python 28438
DynamoDB 28439

SDK für Python (Boto3) 28440
ACM 28443
API Gateway 28466
Application Recovery Controller 28498
Audit Manager 28501
Aurora 28513
Auto Scaling 28560
Amazon Bedrock 28637
Amazon Bedrock Runtime 28643
Amazon Bedrock Agents 28712
Runtime der Agenten für Amazon Bedrock 28788
CloudFormation 28814
CloudFront 28815
CloudWatch 28819
CloudWatch Loggt 28839
Amazon Cognito Identity 28854
Amazon Cognito Identity Provider 28855
Amazon Comprehend 28897
AWS Config 28940
AWS Control Tower 28944
Firehose 28984
Device Farm 28994
Amazon DocumentDB 29005
DynamoDB 29006
Amazon EC2 29207
Amazon ECR 29350
Elastic Load Balancing – Version 2 29379
Amazon EMR 29436
EventBridge 29449
EventBridge Scheduler 29450
Amazon Glacier 29469
AWS Glue 29495
HealthImaging 29531
HealthLake 29584
IAM 29603
AWS IoT 29729

AWS IoT data 29765
AWS IoT SiteWise 29768
Amazon Keyspaces 29809
Kinesis 29841
AWS KMS 29850
Lambda 29905
Managed Service für Apache Flink 29942
AWS Marketplace Katalog-API 29969
AWS Marketplace API für Vereinbarungen 30207
Amazon MSK 30309
Neptune 30310
Organizations 30355
Partner Central 30361
Amazon Pinpoint 30387
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API 30396
Amazon Polly 30399
Amazon RDS 30410
Amazon RDS Data Service 30450
Amazon Redshift 30452
Amazon Rekognition 30475
Amazon S3 30530
Amazon S3 Control 30621
S3-Verzeichnis-Buckets 30643
Secrets Manager 30665
Amazon SES 30669
Amazon SES API v2 30724
Amazon SNS 30765
Amazon SQS 30816
Step-Funktionen 30871
AWS STS 30902
Support 30921
Systems Manager 30952
Amazon Textract 30987
Amazon Transcribe 30997
SDK für Ruby 31019
Aurora 31020

Auto Scaling 31021
CloudTrail 31023
CloudWatch 31027
Amazon Cognito Identity Provider 31040
Amazon Comprehend 31041
Amazon DocumentDB 31043
DynamoDB 31044
Amazon EC2 31072
Elastische Beanstalk 31107
EventBridge 31112
AWS Glue 31130
IAM 31158
Kinesis 31215
AWS KMS 31217
Lambda 31221
Amazon MSK 31251
Amazon Polly 31252
Amazon RDS 31256
Amazon S3 31264
Amazon SES 31296
Amazon SES API v2 31302
Amazon SNS 31303
Amazon SQS 31313
AWS STS 31326
Amazon Textract 31328
Amazon Translate 31329
SDK für Rust 31330
API Gateway 31332
Management-API von API Gateway 31335
Application Auto Scaling 31336
Aurora 31337
Auto Scaling 31485
Amazon Bedrock Runtime 31519
Runtime der Agenten für Amazon Bedrock 31545
Amazon Cognito Identity Provider 31549
Amazon Cognito Sync 31551

Firehose 31552
Amazon DocumentDB 31553
DynamoDB 31555
Amazon EBS 31574
Amazon EC2 31576
Amazon ECR 31638
Amazon ECS 31640
Amazon EKS 31643
AWS Glue 31645
IAM 31660
AWS IoT 31687
Kinesis 31689
AWS KMS 31697
Lambda 31706
MediaLive 31757
MediaPackage 31758
Amazon MSK 31760
Amazon Polly 31762
Amazon RDS 31767
Amazon RDS Data Service 31771
Amazon Rekognition 31772
Route 53 31774
Amazon S3 31776
SageMaker KI 31826
Secrets Manager 31829
Amazon SES API v2 31830
Amazon SNS 31846
Amazon SQS 31853
AWS STS 31858
Systems Manager 31859
Amazon Transcribe 31862
SDK für SAP ABAP 31863
ACM 31865
API Gateway 31874
Application Recovery Controller 31882
Aurora 31887

Auto Scaling 31893
Amazon Bedrock Runtime 31902
Amazon Bedrock Agents Runtime 31905
CloudFront 31907
CloudWatch 31909
CloudWatch Protokolle 31919
Amazon Cognito Identity Provider 31921
Amazon Comprehend 31927
AWS Config 31941
AWS Control Tower 31943
Firehose 31955
DynamoDB 31957
Amazon EC2 31972
Amazon ECR 31993
Amazon EMR 32000
EventBridge Scheduler 32005
AWS Glue 32010
HealthImaging 32023
HealthLake 32040
IAM 32054
AWS IoT 32078
AWS IoT data 32089
AWS IoT SiteWise 32091
Amazon Keyspaces 32100
Kinesis 32108
AWS KMS 32118
Lambda 32137
Organizations 32151
Amazon Pinpoint 32156
Amazon-Pinpoint-SMS- und -Sprachnachrichten-API 32161
Amazon Polly 32171
Amazon RDS 32179
Amazon Redshift 32184
Amazon Rekognition 32192
Amazon S3 32206
Amazon S3 Control 32232

SageMaker KI 32240
Secrets Manager 32257
Amazon SES 32260
Amazon SES API v2 32276
Amazon SNS 32286
Amazon SQS 32297
Step-Funktionen 32307
Systems Manager 32315
Amazon Textract 32325
Amazon Transcribe 32335
Amazon Translate 32343
SDK für Swift 32351
Amazon Bedrock 32352
Amazon Bedrock Runtime 32358
Amazon Cognito Identity 32374
Amazon Cognito Identity Provider 32379
DynamoDB 32407
Amazon EC2 32446
AWS Glue 32499
IAM 32540
Lambda 32561
Amazon RDS 32592
Amazon S3 32629
Amazon SNS 32654
Amazon SQS 32681
AWS STS 32709
Amazon Transcribe Streaming 32710

Weitere AWS SDK-Beispiele sind im AWS Doc SDK Examples GitHub Repo verfügbar.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist die Codebeispielbibliothek?

Die Codebeispielbibliothek ist eine Sammlung von Codebeispielen, die Ihnen zeigen, wie Sie AWS Software Development Kits (SDKs) mit verwenden AWS.

Die Beispiele sind nach AWS-Service und nach AWS SDK geordnet. Die gleichen Beispiele finden Sie in beiden Bereichen.

- Codebeispiele nach Service— Eine Liste AWS-Services davon enthält Beispiele für die Verwendung der einzelnen Dienste mit AWS SDKs. Verwenden Sie diesen Bereich, wenn Sie wissen, welchen Service Sie nutzen möchten.
- Codebeispiele von SDK— Eine Liste AWS SDKs davon enthält Beispiele für die Verwendung AWS-Services mit den einzelnen SDKs. Verwenden Sie diesen Bereich, wenn Sie wissen, welches SDK Sie verwenden möchten.

Innerhalb jedes Bereichs sind die Beispiele in die folgenden Kategorien unterteilt:

- Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.
- Aktionen sind Code-Auszüge, die Ihnen zeigen, wie Sie einzelne Servicefunktionen aufrufen.
- Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.
- Für einige Dienste sind zusätzliche Beispielkategorien definiert. Diese Codebeispiele zeigen Ihnen spezielle Aspekte eines Services oder einer Integration.

Alle Beispiele in dieser Bibliothek finden Sie auch im GitHub Repository für AWS Codebeispiele. Das GitHub Repository enthält auch Anweisungen zum Einrichten, Ausführen und Testen der Beispiele.

Verwendung AWS-Services mit einem AWS SDK

AWS Software Development Kits (SDKs) sind für viele beliebte Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Codebeispiele
AWS CLI	AWS CLI Codebeispiele
AWS SDK für Go	AWS SDK für Go Codebeispiele
AWS SDK für Java	AWS SDK für Java Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Codebeispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Codebeispiele
AWS SDK für .NET	AWS SDK für .NET Codebeispiele
AWS SDK für PHP	AWS SDK für PHP Codebeispiele
AWS -Tools für PowerShell	AWS -Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Codebeispiele
AWS SDK für Ruby	AWS SDK für Ruby Codebeispiele
AWS SDK für Rust	AWS SDK für Rust Codebeispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Codebeispiele
AWS SDK für Swift	AWS SDK für Swift Codebeispiele

Beispiel für die Verfügbarkeit

Sie können nicht finden, was Sie brauchen? Fordern Sie ein Codebeispiel an, indem Sie unten den Link Provide feedback (Feedback geben) auswählen.

Codebeispiele nach Service unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS-Services mit einem AWS Software Development Kit (SDK) verwenden. Dieser Abschnitt ist organisiert von AWS-Service.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Codebeispiele für die Verwendung von ACM AWS SDKs
- Codebeispiele für API Gateway mit AWS SDKs
- Codebeispiele für die API Gateway Management API mit AWS SDKs
- Codebeispiele für Application Auto Scaling mit AWS SDKs
- Codebeispiele für Application Recovery Controller mit AWS SDKs
- Codebeispiele für Athena mit AWS SDKs
- Codebeispiele für Audit Manager mit AWS SDKs
- Codebeispiele für Aurora mit AWS SDKs
- Codebeispiele für Auto Scaling mit AWS SDKs
- Codebeispiele für AWS Batch verwenden AWS SDKs

- Codebeispiele für Amazon Bedrock mit AWS SDKs
- Codebeispiele für Amazon Bedrock Runtime mit AWS SDKs
- Codebeispiele für Amazon Bedrock Agents mit AWS SDKs
- Codebeispiele für Amazon Bedrock Agents Runtime mit AWS SDKs
- Codebeispiele für AWS Budgets verwenden AWS SDKs
- Codebeispiele für AWS Cloud Map verwenden AWS SDKs
- Codebeispiele für AWS Cloud9 verwenden AWS SDKs
- Codebeispiele für CloudFormation verwenden AWS SDKs
- Codebeispiele für die CloudFront Verwendung AWS SDKs
- Codebeispiele für die CloudTrail Verwendung AWS SDKs
- Codebeispiele für die Verwendung CloudWatch AWS SDKs
- Codebeispiele für CloudWatch Ereignisse mit AWS SDKs
- Codebeispiele für CloudWatch Logs mit AWS SDKs
- Codebeispiele für die CodeBuild Verwendung AWS SDKs
- Codebeispiele für die CodeCommit Verwendung AWS SDKs
- Codebeispiele für die Verwendung CodeDeploy AWS SDKs
- Codebeispiele für die Verwendung CodePipeline AWS SDKs
- Codebeispiele für Amazon Cognito Identity mit AWS SDKs
- Codebeispiele für Amazon Cognito Identity Provider mit AWS SDKs
- Codebeispiele für Amazon Cognito Sync mit AWS SDKs
- Codebeispiele für Amazon Comprehend mit AWS SDKs
- Codebeispiele für AWS Config verwenden AWS SDKs
- Codebeispiele für Connect Customer mit AWS SDKs
- Codebeispiele für AWS Control Tower verwenden AWS SDKs
- Codebeispiele für Firehose mit AWS SDKs
- Codebeispiele für Device Farm mit AWS SDKs
- Codebeispiele für Direct Connect verwenden AWS SDKs
- Codebeispiele für Directory Service verwenden AWS SDKs
- Codebeispiele für AWS DMS verwenden AWS SDKs

- Codebeispiele für Amazon DocumentDB mit AWS SDKs
- Codebeispiele für DynamoDB mit AWS SDKs
- Codebeispiele für Amazon EBS mit AWS SDKs
- Codebeispiele für Amazon EC2 mit AWS SDKs
- Codebeispiele für Amazon ECR mit AWS SDKs
- Codebeispiele für Amazon ECS mit AWS SDKs
- Codebeispiele für Amazon EFS mit AWS SDKs
- Codebeispiele für Amazon EKS mit AWS SDKs
- Codebeispiele für Elastic Beanstalk mit AWS SDKs
- Codebeispiele für Elastic Load Balancing — Version 1 mit AWS SDKs
- Codebeispiele für Elastic Load Balancing — Version 2 mit AWS SDKs
- Codebeispiele für die ElastiCache Verwendung AWS SDKs
- Codebeispiele für die MediaStore Verwendung AWS SDKs
- Codebeispiele für Amazon EMR mit AWS SDKs
- Codebeispiele für AWS Entity Resolution verwenden AWS SDKs
- Codebeispiele für die Verwendung von OpenSearch Diensten AWS SDKs
- Codebeispiele für die EventBridge Verwendung AWS SDKs
- Codebeispiele für die Verwendung von EventBridge Scheduler AWS SDKs
- Codebeispiele für AWS FIS verwenden AWS SDKs
- Codebeispiele für Forecast mit AWS SDKs
- Codebeispiele für Amazon Glacier mit AWS SDKs
- Codebeispiele für AWS Glue verwenden AWS SDKs
- Codebeispiele für AWS Health verwenden AWS SDKs
- Codebeispiele für die Verwendung HealthImaging AWS SDKs
- Codebeispiele für die HealthLake Verwendung AWS SDKs
- Codebeispiele für die Verwendung von IAM AWS SDKs
- Codebeispiele für Amazon Inspector mit AWS SDKs
- Codebeispiele für AWS IoT verwenden AWS SDKs
- Codebeispiele für AWS IoT data verwenden AWS SDKs

- Codebeispiele für AWS IoT SiteWise verwenden AWS SDKs
- Codebeispiele für Amazon Keyspaces mit AWS SDKs
- Codebeispiele für Kinesis mit AWS SDKs
- Codebeispiele für AWS KMS verwenden AWS SDKs
- Codebeispiele für Lambda mit AWS SDKs
- Codebeispiele für Amazon Lex mit AWS SDKs
- Codebeispiele für Lightsail mit AWS SDKs
- Codebeispiele für Amazon Location mit AWS SDKs
- Codebeispiele für Location Service Places mit AWS SDKs
- Codebeispiele für Amazon Managed Grafana mit AWS SDKs
- Codebeispiele für Managed Service für Apache Flink mit AWS SDKs
- Codebeispiele für AWS Marketplace Katalog-API verwenden AWS SDKs
- Codebeispiele für AWS Marketplace Vereinbarungs-API unter Verwendung AWS SDKs
- Codebeispiele für die Verwendung MediaConnect AWS SDKs
- Codebeispiele für die MediaConvert Verwendung AWS SDKs
- Codebeispiele für die Verwendung MediaLive AWS SDKs
- Codebeispiele für die MediaPackage Verwendung AWS SDKs
- Codebeispiele für Migration Hub mit AWS SDKs
- Codebeispiele für Amazon MSK mit AWS SDKs
- Codebeispiele für Neptune mit AWS SDKs
- Codebeispiele für Organizations, die AWS SDKs
- Codebeispiele für die Verwendung von Partner Central AWS SDKs
- Codebeispiele für AWS Payment Cryptography verwenden AWS SDKs
- Codebeispiele für Amazon Personalize mit AWS SDKs
- Codebeispiele für Amazon Personalize Events mit AWS SDKs
- Codebeispiele für Amazon Personalize Runtime mit AWS SDKs
- Codebeispiele für Amazon Pinpoint mit AWS SDKs
- Codebeispiele für Amazon Pinpoint SMS und Voice API unter Verwendung AWS SDKs
- Codebeispiele für Amazon Polly mit AWS SDKs

- Codebeispiele für AWS-Preisliste verwenden AWS SDKs
- Codebeispiele für Amazon RDS mit AWS SDKs
- Codebeispiele für Amazon RDS Data Service mit AWS SDKs
- Codebeispiele für Amazon Redshift mit AWS SDKs
- Codebeispiele für Amazon Rekognition mit AWS SDKs
- Codebeispiele für Resource Groups mit AWS SDKs
- Codebeispiele für die Ressourcengruppen-Tagging-API mit AWS SDKs
- Codebeispiele für Route 53 mit AWS SDKs
- Codebeispiele für die Route 53-Domainregistrierung mit AWS SDKs
- Codebeispiele für Amazon S3 mit AWS SDKs
- Codebeispiele für Amazon S3 Control mit AWS SDKs
- Codebeispiele für S3 Directory Buckets mit AWS SDKs
- Codebeispiele für SageMaker KI unter Verwendung AWS SDKs
- Codebeispiele für Secrets Manager mit AWS SDKs
- Codebeispiele für Amazon SES mit AWS SDKs
- Codebeispiele für Amazon SES API v2 mit AWS SDKs
- Codebeispiele für Amazon SNS mit AWS SDKs
- Codebeispiele für Amazon SQS mit AWS SDKs
- Codebeispiele für Step Functions mit AWS SDKs
- Codebeispiele für AWS STS verwenden AWS SDKs
- Codebeispiele für Support verwenden AWS SDKs
- Codebeispiele für Systems Manager mit AWS SDKs
- Codebeispiele für Amazon Textract mit AWS SDKs
- Codebeispiele für Amazon Transcribe mit AWS SDKs
- Codebeispiele für Amazon Transcribe Streaming mit AWS SDKs
- Codebeispiele für Amazon Translate mit AWS SDKs
- Codebeispiele für VPC Lattice mit AWS SDKs
- Codebeispiele für AWS WAFV2 verwenden AWS SDKs
- Codebeispiele für die WorkSpaces Verwendung AWS SDKs
- Codebeispiele für die X-Ray Verwendung AWS SDKs

Codebeispiele für die Verwendung von ACM AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Certificate Manager mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- ACM-Benutzerhandbuch – Weitere Informationen zu ACM.
- ACM-API-Referenz – Details zu allen verfügbaren ACM-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von ACM AWS SDKs
  - Erlernen Sie die Grundlagen von ACM mit einem AWS SDK
  - Aktionen für die Verwendung von ACM AWS SDKs
    - Verwenden Sie AddTagsToCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie DescribeCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie ExportCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie GetCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie ImportCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI
    - Verwenden Sie ListTagsForCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie RemoveTagsFromCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie RenewCertificatemit einem AWS SDK

- Verwenden Sie RequestCertificatemit einem AWS SDK oder CLI
- Verwenden Sie ResendValidationEmailmit einem AWS SDK oder CLI
- Verwenden Sie UpdateCertificateOptionsmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung von ACM AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Certificate Manager mit AWS SDKs verwendet werden.

Beispiele

- Erlernen Sie die Grundlagen von ACM mit einem AWS SDK
- Aktionen für die Verwendung von ACM AWS SDKs
  - Verwenden Sie AddTagsToCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie DescribeCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie ExportCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie GetCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie ImportCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI
  - Verwenden Sie ListTagsForCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie RemoveTagsFromCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie RenewCertificatemit einem AWS SDK
  - Verwenden Sie RequestCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie ResendValidationEmailmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateCertificateOptionsmit einem AWS SDK oder CLI

Erlernen Sie die Grundlagen von ACM mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Anfordern eines Zertifikats von ACM
- Importieren eines selbstsignierten Zertifikats
- Auflisten und Beschreiben von Zertifikaten

- Entfernen Sie Zertifikate.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die ACM umschließt.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being
 issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
```

```
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
when
                                   the email domain differs from the primary
domain of
                                   the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn
```

```
    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.

        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in
PEM
                            format.
        :return: The ARN of the imported certificate.
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
            return certificate_arn


    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None,
all
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If
None,
                          all key usages are included.
```

```
        :param extended_key_usage: Filters the results to the specified extended
key
                                   usages. If None, all extended key usages are
                                   included.
        :param key_types: Filters the results to the specified key types. If
None, all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the
certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
            )
```

```
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
certificate["DomainName"]
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
certificate_arn)
            raise
        else:
            return certificate


    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in
tags.items()],
```

```
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
certificate_arn)
            raise


    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
certificate_arn)
            raise
        else:
            return tags


    def remove_tags(self, certificate_arn, tags):
        """
        Removes tags from a certificate. If the value of a tag is specified, the
tag is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
```

```
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags),
 certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
            )
            raise


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

Verwenden Sie die Wrapper-Klasse, um Zertifikate für Ihr Konto zu verwalten.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the AWS Certificate Manager (ACM) demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    acm_certificate = AcmCertificate(boto3.client("acm"))
```

```
    domain = "example.com"
    sub_domains = [f"{sub}.{domain}" for sub in ["test", "dev"]]
    print(f"Request a certificate for {domain}.")
    certificate_arn = acm_certificate.request_validation(domain, sub_domains,
"DNS")
    print(f"Started validation, got certificate ARN: {certificate_arn}.")

    import_cert_arn = None
    cert_file_name = input(
        "Enter the file name for a self-signed certificate in PEM format. "
        "This certificate will be imported to ACM. Press Enter to skip: "
    )
    if cert_file_name:
        pk_file_name = input(
            "Enter the file name for the unencrypted private key of the
certificate. "
            "This file must also be in PEM format: "
        )
        if pk_file_name:
            with open(cert_file_name, "rb") as cert_file:
                import_cert = cert_file.read()
            with open(pk_file_name, "rb") as pk_file:
                import_pk = pk_file.read()
            import_cert_arn = acm_certificate.import_certificate(import_cert,
import_pk)
            print(f"Certificate imported, got ARN: {import_cert_arn}")
        else:
            print("No private key file entered. Skipping certificate import.")
    else:
        print("Skipping self-signed certificate import.")

    print("Getting the first 10 issued certificates.")
    certificates = acm_certificate.list(10, statuses=["ISSUED"])
    print(f"Found {len(certificates)} issued certificates.")

    print(f"Getting metadata for certificate {certificate_arn}")
    cert_metadata = acm_certificate.describe(certificate_arn)
    pprint(cert_metadata)

    if import_cert_arn is not None:
        print(f"Getting certificate for imported certificate {import_cert_arn}")
        import_cert_data = acm_certificate.get(import_cert_arn)
        pprint(import_cert_data)
```

```
    print(f"Adding tags to certificate {certificate_arn}.")
    acm_certificate.add_tags(certificate_arn, {"purpose": "acm demo", "color":
"green"})
    tags = acm_certificate.list_tags(certificate_arn)
    print(f"Found tags: {tags}")
    acm_certificate.remove_tags(certificate_arn, {key: None for key in tags})
    print("Removed tags.")

    print("Removing certificates added during the demo.")
    acm_certificate.remove(certificate_arn)
    if import_cert_arn is not None:
        acm_certificate.remove(import_cert_arn)

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AddTagsToCertificate
  - DeleteCertificate
  - DescribeCertificate
  - GetCertificate
  - ImportCertificate
  - ListCertificates
  - ListTagsForCertificate
  - RemoveTagsFromCertificate
  - RequestCertificate
  - ResendValidationEmail

Aktionen für die Verwendung von ACM AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne ACM-Aktionen mit SDKs ausgeführt werden. AWS Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Certificate Manager -API-Referenz.

Beispiele

- Verwenden Sie AddTagsToCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DescribeCertificatemit einem AWS SDK oder CLI
- Verwenden Sie ExportCertificatemit einem AWS SDK oder CLI
- Verwenden Sie GetCertificatemit einem AWS SDK oder CLI
- Verwenden Sie ImportCertificatemit einem AWS SDK oder CLI
- Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI
- Verwenden Sie ListTagsForCertificatemit einem AWS SDK oder CLI
- Verwenden Sie RemoveTagsFromCertificatemit einem AWS SDK oder CLI
- Verwenden Sie RenewCertificatemit einem AWS SDK
- Verwenden Sie RequestCertificatemit einem AWS SDK oder CLI
- Verwenden Sie ResendValidationEmailmit einem AWS SDK oder CLI
- Verwenden Sie UpdateCertificateOptionsmit einem AWS SDK oder CLI

Verwenden Sie **`AddTagsToCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AddTagsToCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add tags to an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::addTagsToCertificate(const Aws::String &certificateArn,
                                       const Aws::String &tagKey,
                                       const Aws::String &tagValue,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::AddTagsToCertificateRequest request;
    Aws::Vector<Aws::ACM::Model::Tag> tags;
    Aws::ACM::Model::Tag tag;

    tag.WithKey(tagKey).WithValue(tagValue);
    tags.push_back(tag);

    request.WithCertificateArn(certificateArn).WithTags(tags);

    Aws::ACM::Model::AddTagsToCertificateOutcome outcome =
            acmClient.AddTagsToCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: addTagsToCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey <<
                  "' and value '" << tagValue <<
                  "' added to certificate with ARN '" <<
                  certificateArn << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AddTagsToCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie einem vorhandenen ACM-Zertifikat Tags hinzu

Der folgende `add-tags-to-certificate`-Befehl fügt dem angegebenen Zertifikat zwei Tags hinzu. Verwenden Sie ein Leerzeichen, um mehrere Tags voneinander zu trennen:

```
aws acm add-tags-to-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
 --tags Key=Admin,Value=Alice Key=Purpose,Value=Website
```

- Einzelheiten zur API finden Sie AddTagsToCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTagsToCertificate {
```

```
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add
 tags to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie AddTagsToCertificatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in
 tags.items()],
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
 certificate_arn)
```

```
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
 certificate_arn)
            raise
```

- Einzelheiten zur API finden Sie AddTagsToCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->addtagstocertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags added to certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
      CATCH /aws1/cx_acmtoomanytagsex.
        MESSAGE 'Too many tags for certificate.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AddTagsToCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::deleteCertificate(const Aws::String &certificateArn,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::DeleteCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DeleteCertificateOutcome outcome =
            acmClient.DeleteCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: The certificate with the ARN '" <<
```

```
                    certificateArn << "' is deleted." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie ein ACM-Zertifikat aus Ihrem Konto

Der folgende `delete-certificate`-Befehl löscht das Zertifikat mit dem angegebenen ARN:

```
aws acm delete-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        deleteCertificate(certArn);
    }

    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht das Zertifikat, das durch den angegebenen ARN und den zugehörigen privaten Schlüssel identifiziert wurde. Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, fügen Sie den Schalter -Force hinzu, um die Prompts zu unterdrücken.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter DeleteCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht das Zertifikat, das durch den angegebenen ARN und den zugehörigen privaten Schlüssel identifiziert wurde. Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, fügen Sie den Schalter -Force hinzu, um die Prompts zu unterdrücken.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter DeleteCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
```

```
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

- Einzelheiten zur API finden Sie DeleteCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->deletecertificate( iv_certificatearn = iv_certificate_arn ).
        MESSAGE 'Certificate deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
```

```
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmresourceinuseex.
        MESSAGE 'Certificate is in use and cannot be deleted.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;

namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
```

```
        // certificate using the AWS Certificate Manager (ACM) service.

        // Specify your AWS Region (an example Region is shown).
        private static readonly RegionEndpoint ACMRegion =
 RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new
 Amazon.CertificateManager.AmazonCertificateManagerClient(ACMRegion);

            var describeCertificateReq = new DescribeCertificateRequest();
            // The ARN used here is just an example. Replace it with the ARN of
            // a certificate that exists on your account.
            describeCertificateReq.CertificateArn =
                "arn:aws:acm:us-
east-1:123456789012:certificate/8cfd7dae-9b6a-2d07-92bc-1c309EXAMPLE";

            var certificateDetailResp =
                DescribeCertificateResponseAsync(client: _client, request:
 describeCertificateReq);
            var certificateDetail = certificateDetailResp.Result.Certificate;

            if (certificateDetail is not null)
            {
                DisplayCertificateDetails(certificateDetail);
            }
        }

        /// <summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        /// </summary>
        /// <param name="certificateDetail">The object that contains details
        /// returned from the call to DescribeCertificateAsync.</param>
        static void DisplayCertificateDetails(CertificateDetail
 certificateDetail)
        {
            Console.WriteLine("\nCertificate Details: ");
            Console.WriteLine($"Certificate Domain:
 {certificateDetail.DomainName}");
            Console.WriteLine($"Certificate Arn:
 {certificateDetail.CertificateArn}");
```

```
            Console.WriteLine($"Certificate Subject:
 {certificateDetail.Subject}");
            Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
            foreach (var san in certificateDetail.SubjectAlternativeNames)
            {
                Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with the ACM service certificate.
        /// </summary>
        /// <param name="client">An AmazonCertificateManagerClient object
        /// used to call DescribeCertificateResponse.</param>
        /// <param name="request">The DescribeCertificateRequest object that
        /// will be passed to the method call.</param>
        /// <returns></returns>
        static async Task<DescribeCertificateResponse>
 DescribeCertificateResponseAsync(
            AmazonCertificateManagerClient client, DescribeCertificateRequest
 request)
        {
            var response = new DescribeCertificateResponse();

            try
            {
                response = await client.DescribeCertificateAsync(request);
            }
            catch (InvalidArnException)
            {
                Console.WriteLine($"Error: The ARN specified is invalid.");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine($"Error: The specified certificate could not be
 found.");
            }

            return response;
        }
    }

}
```

- Einzelheiten zur API finden Sie DescribeCertificatein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::describeCertificate(const Aws::String &certificateArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::DescribeCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DescribeCertificateOutcome outcome =
            acm_client.DescribeCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DescribeCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::ACM::Model::CertificateDetail certificate =
                outcome.GetResult().GetCertificate();

        std::cout << "Success: Information about certificate "
                     "with ARN '" << certificateArn << "':" << std::endl <<
 std::endl;
```

```
        std::cout << "ARN:                 " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << "Authority ARN:       " <<
                  certificate.GetCertificateAuthorityArn() << std::endl;
        std::cout << "Created at (GMT):    " <<
                  certificate.GetCreatedAt().ToGmtString(
                          Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
        std::cout << "Domain name:         " << certificate.GetDomainName()
                  << std::endl;

        Aws::Vector<Aws::ACM::Model::DomainValidation> options =
                certificate.GetDomainValidationOptions();

        if (!options.empty()) {
            std::cout << std::endl << "Domain validation information: "
                      << std::endl << std::endl;

            for (auto &validation: options) {
                std::cout << "  Domain name:              " <<
                          validation.GetDomainName() << std::endl;

                const Aws::ACM::Model::ResourceRecord &record =
                        validation.GetResourceRecord();

                std::cout << "  Resource record name:     " <<
                          record.GetName() << std::endl;

                Aws::ACM::Model::RecordType recordType = record.GetType();
                Aws::String type;

                switch (recordType) {
                    case Aws::ACM::Model::RecordType::CNAME:
                        type = "CNAME";
                        break;
                    case Aws::ACM::Model::RecordType::NOT_SET:
                        type = "Not set";
                        break;
                    default:
                        type = "Cannot determine.";
                        break;
                }
```

```
                std::cout << "  Resource record type:     " << type <<
                          std::endl;

                std::cout << "  Resource record value:    " <<
                          record.GetValue() << std::endl;

                std::cout << "  Validation domain:        " <<
                          validation.GetValidationDomain() << std::endl;

                Aws::Vector<Aws::String> emails =
                        validation.GetValidationEmails();

                if (!emails.empty()) {
                    std::cout << "  Validation emails:" << std::endl <<
                              std::endl;

                    for (auto &email: emails) {
                        std::cout << "    " << email << std::endl;
                    }

                    std::cout << std::endl;
                }

                Aws::ACM::Model::ValidationMethod validationMethod =
                        validation.GetValidationMethod();
                Aws::String method;

                switch (validationMethod) {
                    case Aws::ACM::Model::ValidationMethod::DNS:
                        method = "DNS";
                        break;
                    case Aws::ACM::Model::ValidationMethod::EMAIL:
                        method = "Email";
                        break;
                    case Aws::ACM::Model::ValidationMethod::NOT_SET:
                        method = "Not set";
                        break;
                    default:
                        method = "Cannot determine";
                }

                std::cout << "  Validation method:        " <<
                          method << std::endl;
```

```
                Aws::ACM::Model::DomainStatus domainStatus =
                        validation.GetValidationStatus();
                Aws::String status;

                switch (domainStatus) {
                    case Aws::ACM::Model::DomainStatus::FAILED:
                        status = "Failed";
                        break;
                    case Aws::ACM::Model::DomainStatus::NOT_SET:
                        status = "Not set";
                        break;
                    case Aws::ACM::Model::DomainStatus::PENDING_VALIDATION:
                        status = "Pending validation";
                        break;
                    case Aws::ACM::Model::DomainStatus::SUCCESS:
                        status = "Success";
                        break;
                    default:
                        status = "Cannot determine";
                }

                std::cout << "  Domain validation status: " << status <<
                          std::endl << std::endl;

            }
        }

        Aws::Vector<Aws::ACM::Model::ExtendedKeyUsage> usages =
                certificate.GetExtendedKeyUsages();

        if (!usages.empty()) {
            std::cout << std::endl << "Extended key usages:" <<
                      std::endl << std::endl;

            for (auto &usage: usages) {
                Aws::ACM::Model::ExtendedKeyUsageName usageName =
                        usage.GetName();
                Aws::String name;

                switch (usageName) {
                    case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
                        name = "Any";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
```

```
                        name = "Code signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
                        name = "Custom";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
                        name = "Email protection";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
                        name = "IPSEC end system";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
                        name = "IPSEC tunnel";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
                        name = "IPSEC user";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
                        name = "None";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
                        name = "Not set";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
                        name = "OCSP signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
                        name = "Time stamping";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
                        name = "TLS web client authentication";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
                        name = "TLS web server authentication";
                        break;
                    default:
                        name = "Cannot determine";
                }

                std::cout << "  Name: " << name << std::endl;
                std::cout << "  OID:  " << usage.GetOID() <<
                          std::endl << std::endl;
```

```
            }

            std::cout << std::endl;
        }

        Aws::ACM::Model::CertificateStatus certificateStatus =
                certificate.GetStatus();
        Aws::String status;

        switch (certificateStatus) {
            case Aws::ACM::Model::CertificateStatus::EXPIRED:
                status = "Expired";
                break;
            case Aws::ACM::Model::CertificateStatus::FAILED:
                status = "Failed";
                break;
            case Aws::ACM::Model::CertificateStatus::INACTIVE:
                status = "Inactive";
                break;
            case Aws::ACM::Model::CertificateStatus::ISSUED:
                status = "Issued";
                break;
            case Aws::ACM::Model::CertificateStatus::NOT_SET:
                status = "Not set";
                break;
            case Aws::ACM::Model::CertificateStatus::PENDING_VALIDATION:
                status = "Pending validation";
                break;
            case Aws::ACM::Model::CertificateStatus::REVOKED:
                status = "Revoked";
                break;
            case Aws::ACM::Model::CertificateStatus::VALIDATION_TIMED_OUT:
                status = "Validation timed out";
                break;
            default:
                status = "Cannot determine";
        }

        std::cout << "Status:              " << status << std::endl;

        if (certificate.GetStatus() ==
            Aws::ACM::Model::CertificateStatus::FAILED) {
            Aws::ACM::Model::FailureReason failureReason =
                    certificate.GetFailureReason();
```

```
            Aws::String reason;

            switch (failureReason) {
                case
 Aws::ACM::Model::FailureReason::ADDITIONAL_VERIFICATION_REQUIRED:
                    reason = "Additional verification required";
                    break;
                case Aws::ACM::Model::FailureReason::CAA_ERROR:
                    reason = "CAA error";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_NOT_ALLOWED:
                    reason = "Domain not allowed";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_VALIDATION_DENIED:
                    reason = "Domain validation denied";
                    break;
                case Aws::ACM::Model::FailureReason::INVALID_PUBLIC_DOMAIN:
                    reason = "Invalid public domain";
                    break;
                case Aws::ACM::Model::FailureReason::NOT_SET:
                    reason = "Not set";
                    break;
                case Aws::ACM::Model::FailureReason::NO_AVAILABLE_CONTACTS:
                    reason = "No available contacts";
                    break;
                case Aws::ACM::Model::FailureReason::OTHER:
                    reason = "Other";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_ACCESS_DENIED:
                    reason = "PCA access denied";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARGS:
                    reason = "PCA invalid args";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARN:
                    reason = "PCA invalid ARN";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_DURATION:
                    reason = "PCA invalid duration";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_STATE:
                    reason = "PCA invalid state";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_LIMIT_EXCEEDED:
```

```
                    reason = "PCA limit exceeded";
                    break;
                case
Aws::ACM::Model::FailureReason::PCA_NAME_CONSTRAINTS_VALIDATION:
                    reason = "PCA name constraints validation";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_REQUEST_FAILED:
                    reason = "PCA request failed";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_RESOURCE_NOT_FOUND:
                    reason = "PCA resource not found";
                    break;
                default:
                    reason = "Cannot determine";
            }

            std::cout << "Failure reason:      " << reason << std::endl;
        }

        if (certificate.GetStatus() ==
Aws::ACM::Model::CertificateStatus::REVOKED) {
            std::cout << "Revoked at (GMT):    " <<
                      certificate.GetRevokedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;

            Aws::ACM::Model::RevocationReason revocationReason =
                    certificate.GetRevocationReason();
            Aws::String reason;

            switch (revocationReason) {
                case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
                    reason = "Affiliation changed";
                    break;
                case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
                    reason = "AA compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::CA_COMPROMISE:
                    reason = "CA compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::CERTIFICATE_HOLD:
                    reason = "Certificate hold";
                    break;
                case Aws::ACM::Model::RevocationReason::CESSATION_OF_OPERATION:
```

```
                    reason = "Cessation of operation";
                    break;
                case Aws::ACM::Model::RevocationReason::KEY_COMPROMISE:
                    reason = "Key compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::NOT_SET:
                    reason = "Not set";
                    break;
                case Aws::ACM::Model::RevocationReason::PRIVILEGE_WITHDRAWN:
                    reason = "Privilege withdrawn";
                    break;
                case Aws::ACM::Model::RevocationReason::REMOVE_FROM_CRL:
                    reason = "Revoke from CRL";
                    break;
                case Aws::ACM::Model::RevocationReason::SUPERCEDED:
                    reason = "Superceded";
                    break;
                case Aws::ACM::Model::RevocationReason::UNSPECIFIED:
                    reason = "Unspecified";
                    break;
                default:
                    reason = "Cannot determine";
            }

            std::cout << "Revocation reason:   " << reason << std::endl;
        }

        if (certificate.GetType() == Aws::ACM::Model::CertificateType::IMPORTED)
{
            std::cout << "Imported at (GMT):   " <<
                      certificate.GetImportedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        Aws::Vector<Aws::String> inUseBys = certificate.GetInUseBy();

        if (!inUseBys.empty()) {
            std::cout << std::endl << "In use by:" << std::endl << std::endl;

            for (auto &in_use_by: inUseBys) {
                std::cout << "  " << in_use_by << std::endl;
            }
```

```
            std::cout << std::endl;
        }

        if (certificate.GetType() ==
Aws::ACM::Model::CertificateType::AMAZON_ISSUED &&
            certificate.GetStatus() ==
Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Issued at (GMT):     " <<
                      certificate.GetIssuedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        std::cout << "Issuer:              " << certificate.GetIssuer() <<
                  std::endl;

        Aws::ACM::Model::KeyAlgorithm keyAlgorithm =
                certificate.GetKeyAlgorithm();
        Aws::String algorithm;

        switch (keyAlgorithm) {
            case Aws::ACM::Model::KeyAlgorithm::EC_prime256v1:
                algorithm = "P-256 (secp256r1, prime256v1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp384r1:
                algorithm = "P-384 (secp384r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp521r1:
                algorithm = "P-521 (secp521r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::NOT_SET:
                algorithm = "Not set";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_1024:
                algorithm = "RSA 1024";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_2048:
                algorithm = "RSA 2048";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
                algorithm = "RSA 4096";
                break;
            default:
                algorithm = "Cannot determine";
```

```
        }

        std::cout << "Key algorithm:       " << algorithm << std::endl;

        if (certificate.GetStatus() ==
Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Not valid after (GMT): " <<
                      certificate.GetNotAfter().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
            std::cout << "Not valid before (GMT): " <<
                      certificate.GetNotBefore().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        Aws::ACM::Model::CertificateTransparencyLoggingPreference
loggingPreference =

certificate.GetOptions().GetCertificateTransparencyLoggingPreference();
        Aws::String preference;

        switch (loggingPreference) {
            case
Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED:
                preference = "Disabled";
                break;
            case
Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED:
                preference = "Enabled";
                break;
            case
Aws::ACM::Model::CertificateTransparencyLoggingPreference::NOT_SET:
                preference = "Not set";
                break;
            default:
                preference = "Cannot determine";
        }

        std::cout << "Logging preference:  " << preference << std::endl;

        std::cout << "Serial:              " << certificate.GetSerial() <<
                  std::endl;
        std::cout << "Signature algorithm: "
```

```
                << certificate.GetSignatureAlgorithm() << std::endl;
        std::cout << "Subject:             " << certificate.GetSubject() <<
                std::endl;

        Aws::ACM::Model::CertificateType certificateType = certificate.GetType();
        Aws::String type;

        switch (certificateType) {
            case Aws::ACM::Model::CertificateType::AMAZON_ISSUED:
                type = "Amazon issued";
                break;
            case Aws::ACM::Model::CertificateType::IMPORTED:
                type = "Imported";
                break;
            case Aws::ACM::Model::CertificateType::NOT_SET:
                type = "Not set";
                break;
            case Aws::ACM::Model::CertificateType::PRIVATE_:
                type = "Private";
                break;
            default:
                type = "Cannot determine";
        }

        std::cout << "Type:                " << type << std::endl;

        Aws::Vector<Aws::String> altNames =
                certificate.GetSubjectAlternativeNames();

        if (!altNames.empty()) {
            std::cout << std::endl << "Alternative names:" <<
                      std::endl << std::endl;

            for (auto &alt_name: altNames) {
                std::cout << "  " << alt_name << std::endl;
            }

            std::cout << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die in einem ACM-Zertifikat enthaltenen Felder ab

Der folgende `describe-certificate`-Befehl ruft alle Felder für das Zertifikat mit dem angegebenen ARN ab:

```
aws acm describe-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

Die Ausgabe sieht in etwa wie folgt aus:

```
{
  "Certificate": {
    "CertificateArn":
 "arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012",
    "CreatedAt": 1446835267.0,
    "DomainName": "www.example.com",
    "DomainValidationOptions": [
      {
        "DomainName": "www.example.com",
        "ValidationDomain": "www.example.com",
        "ValidationEmails": [
          "hostmaster@example.com",
          "admin@example.com",
          "owner@example.com.whoisprivacyservice.org",
          "tech@example.com.whoisprivacyservice.org",
          "admin@example.com.whoisprivacyservice.org",
          "postmaster@example.com",
          "webmaster@example.com",
          "administrator@example.com"
        ]
      },
      {
        "DomainName": "www.example.net",
        "ValidationDomain": "www.example.net",
```

```
        "ValidationEmails": [
          "postmaster@example.net",
          "admin@example.net",
          "owner@example.net.whoisprivacyservice.org",
          "tech@example.net.whoisprivacyservice.org",
          "admin@example.net.whoisprivacyservice.org",
          "hostmaster@example.net",
          "administrator@example.net",
          "webmaster@example.net"
        ]
      }
    ],
    "InUseBy": [],
    "IssuedAt": 1446835815.0,
    "Issuer": "Amazon",
    "KeyAlgorithm": "RSA-2048",
    "NotAfter": 1478433600.0,
    "NotBefore": 1446768000.0,
    "Serial": "0f:ac:b0:a3:8d:ea:65:52:2d:7d:01:3a:39:36:db:d6",
    "SignatureAlgorithm": "SHA256WITHRSA",
    "Status": "ISSUED",
    "Subject": "CN=www.example.com",
    "SubjectAlternativeNames": [
      "www.example.com",
      "www.example.net"
    ]
  }
}
```

- Einzelheiten zur API finden Sie DescribeCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to
 describe
     * @throws AcmException if an error occurs while describing the certificate
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
```

```
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " +
 response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeCertificatein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details zum angegebenen Zertifikat zurück.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Ausgabe:

```
CertificateArn          : arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt               : 1/21/2016 5:55:59 PM
DomainName              : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy                 : {}
IssuedAt                : 1/1/0001 12:00:00 AM
Issuer                  :
KeyAlgorithm            : RSA-2048
```

```
NotAfter                : 1/1/0001 12:00:00 AM
NotBefore               : 1/1/0001 12:00:00 AM
RevocationReason        :
RevokedAt               : 1/1/0001 12:00:00 AM
Serial                  :
SignatureAlgorithm      : SHA256WITHRSA
Status                  : PENDING_VALIDATION
Subject                 : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- Einzelheiten zur API finden Sie unter DescribeCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details zum angegebenen Zertifikat zurück.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Ausgabe:

```
CertificateArn          : arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt               : 1/21/2016 5:55:59 PM
DomainName              : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy                 : {}
IssuedAt                : 1/1/0001 12:00:00 AM
Issuer                  :
KeyAlgorithm            : RSA-2048
NotAfter                : 1/1/0001 12:00:00 AM
NotBefore               : 1/1/0001 12:00:00 AM
RevocationReason        :
RevokedAt               : 1/1/0001 12:00:00 AM
Serial                  :
SignatureAlgorithm      : SHA256WITHRSA
Status                  : PENDING_VALIDATION
Subject                 : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- Einzelheiten zur API finden Sie unter DescribeCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the
 certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
            )
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
 certificate["DomainName"]
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
 certificate_arn)
```

```
            raise
        else:
            return certificate
```

- Einzelheiten zur API finden Sie DescribeCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->describecertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate details retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ExportCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ExportCertificate` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Export an AWS Certificate Manager (ACM)  certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param passphrase: A passphrase to decrypt the exported certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::exportCertificate(const Aws::String &certificateArn,
                                    const Aws::String &passphrase,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ExportCertificateRequest request;
    Aws::Utils::CryptoBuffer cryptoBuffer(
            reinterpret_cast<const unsigned char *>(passphrase.c_str()),
            passphrase.length());
    request.WithCertificateArn(certificateArn).WithPassphrase(cryptoBuffer);

    Aws::ACM::Model::ExportCertificateOutcome outcome =
            acm_client.ExportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ExportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();
```

```
        std::cout << "Certificate:       " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl << std::endl;
        std::cout << "Private key:       " << std::endl << std::endl <<
                  result.GetPrivateKey() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie ExportCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So exportieren Sie ein privates Zertifikat, das von einer privaten Zertifizierungsstelle ausgegeben wurde.

Mit dem folgenden `export-certificate`-Befehl werden ein privates Zertifikat, eine Zertifikatskette und ein privater Schlüssel auf Ihr Display exportiert:

```
aws acm export-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
 --passphrase file://path-to-passphrase-file
```

Verwenden Sie den folgenden Befehl, um das Zertifikat, die Kette und den privaten Schlüssel in eine lokale Datei zu exportieren:

```
aws acm export-certificate --certificate-
arn arn:aws:acm:region:sccount:certificate/12345678-1234-1234-1234-123456789012
 --passphrase file://path-to-passphrase-file > c:\temp\export.txt
```

- Einzelheiten zur API finden Sie ExportCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and
 certificate chain from AWS Certificate Manager (ACM).
```

```
    *
    * @param certArn The Amazon Resource Name (ARN) of the certificate that you
want to export.
    * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
    */
   public static void exportCert(String certArn) throws IOException {
       AcmClient acmClient = AcmClient.create();

       // Initialize a file descriptor for the passphrase file.
       RandomAccessFile filePassphrase = null;
       ByteBuffer bufPassphrase = null;

       // Create a file stream for reading the private key passphrase.
       try {
           filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
       } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
           throw ex;
       }

       // Create a channel to map the file.
       FileChannel channelPassphrase = filePassphrase.getChannel();

       // Map the file to the buffer.
       try {
           bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY,
0, channelPassphrase.size());
           channelPassphrase.close();
           filePassphrase.close();
       } catch (IOException ex) {
           throw ex;
       }

       // Create a request object.
       ExportCertificateRequest req = ExportCertificateRequest.builder()
           .certificateArn(certArn)
           .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
           .build();

       // Export the certificate.
       ExportCertificateResponse result = null;
       try {
           result = acmClient.exportCertificate(req);
```

```
        } catch (InvalidArnException | InvalidTagException |
 ResourceNotFoundException ex) {
            throw ex;
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);

        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- Einzelheiten zur API finden Sie ExportCertificatein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::getCertificate(const Aws::String &certificateArn,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::GetCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::GetCertificateOutcome outcome =
            acmClient.GetCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: GetCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate: " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie ein ACM-Zertifikat ab

Der folgende `get-certificate`-Befehl ruft das Zertifikat für den angegebenen ARN und die Zertifikatskette ab:

```
aws acm get-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

Die Ausgabe sieht in etwa wie folgt aus:


```
{
  "Certificate": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",

  "CertificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
```

```
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
"-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
"-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
}
```

- Einzelheiten zur API finden Sie GetCertificatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie Sie ein Zertifikat und dessen Zertifikatskette mithilfe des ARN des Zertifikats zurückgeben.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter GetCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie Sie ein Zertifikat und dessen Zertifikatskette mithilfe des ARN des Zertifikats zurückgeben.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- Einzelheiten zur API finden Sie unter GetCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
```

```
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie GetCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
```

```
        oo_result = lo_acm->getcertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate body and chain retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmrequestinprgssex.
        MESSAGE 'Certificate request is in progress.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ImportCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ImportCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Import an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateFile: Path to certificate to import.
  \param privateKeyFile: Path to file containing a private key.
  \param certificateChainFile: Path to file containing a PEM encoded certificate
 chain.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                    const Aws::String &privateKeyFile,
                                    const Aws::String &certificateChainFile,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream privateKeyInstream(privateKeyFile.c_str());
    if (!privateKeyInstream) {
        std::cerr << "Error: The private key file '" << privateKeyFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream certificateChainInStream(certificateChainFile.c_str());
    if (!certificateChainInStream) {
        std::cerr << "Error: The certificate chain file '"
                  << certificateChainFile << "' does not exist." << std::endl;

        return false;
    }

    Aws::String certificate;
    certificate.assign(std::istreambuf_iterator<char>(certificateInStream),
                       std::istreambuf_iterator<char>());

    Aws::String privateKey;
    privateKey.assign(std::istreambuf_iterator<char>(privateKeyInstream),
                      std::istreambuf_iterator<char>());

    Aws::String certificateChain;

 certificateChain.assign(std::istreambuf_iterator<char>(certificateChainInStream),
                         std::istreambuf_iterator<char>());
```

```
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ImportCertificateRequest request;

    request.WithCertificate(Aws::Utils::ByteBuffer((unsigned char *)
                                                        certificate.c_str(),
                                                    certificate.size()))
            .WithPrivateKey(Aws::Utils::ByteBuffer((unsigned char *)
                                                        privateKey.c_str(),
                                                    privateKey.size()))
            .WithCertificateChain(Aws::Utils::ByteBuffer((unsigned char *)

  certificateChain.c_str(),

  certificateChain.size()));

    Aws::ACM::Model::ImportCertificateOutcome outcome =
            acmClient.ImportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ImportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Certificate associated with ARN '" <<
                  outcome.GetResult().GetCertificateArn() << "' imported."
                  << std::endl;

        return true;
    }
}
```

- Einzelheiten zur API finden Sie ImportCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So importieren Sie ein Zertifikat in ACM.

Der folgende `import-certificate`-Befehl importiert ein Zertifikat in ACM. Ersetzen Sie die Dateinamen durch Ihre eigenen:

```
aws acm import-certificate --certificate file://Certificate.pem --certificate-
chain file://CertificateChain.pem --private-key file://PrivateKey.pem
```

- Einzelheiten zur API finden Sie ImportCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ImportCert {

    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <certificateKey> <privateKeyKey>

            Where:
                bucketName - The name of the S3 bucket containing the certificate
 and private key.
                certificateKey - The object key for the SSL/TLS certificate file
 in S3.
                privateKeyKey - The object key for the private key file in S3.
            """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String certificateKey = args[1];
        String privateKeyKey = args[2];

        String certificateArn = importCertificate(bucketName, certificateKey,
privateKeyKey);
        System.out.println("Certificate imported with ARN: " + certificateArn);
    }

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS
Certificate Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
    public static String importCertificate(String bucketName, String
certificateKey, String privateKeyKey) {
        AcmClient acmClient = AcmClient.create();
        S3Client s3Client = S3Client.create();

        try {
            byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))

.privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
acmClient.importCertificate(request);
            return response.certificateArn();
```

```
        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
 S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " +
 e.awsErrorDetails().errorMessage());
        }
        return "";
    }

    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String
 bucketName, String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
 s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- Einzelheiten zur API finden Sie ImportCertificatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.

        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in
 PEM
                            format.
        :return: The ARN of the imported certificate.
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
```

```
            return certificate_arn
```

- Einzelheiten zur API finden Sie ImportCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass certificate chain if it's provided (it's optional)
        IF iv_certificate_chain IS NOT INITIAL.
          DATA(lo_result) = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
            iv_certificatechain = iv_certificate_chain
          ).
        ELSE.
          lo_result = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
          ).
        ENDIF.
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate imported successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
      CATCH /aws1/cx_acmlimitexceededex.
        MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ImportCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListCertificates`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListCertificates` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;

namespace ListCertificates
{
    // The following example retrieves and displays a list of the
    // certificates defined for the default account using the AWS
    // Certificate Manager (ACM) service.
    class ListCertificates
    {
        // Specify your AWS Region (an example Region is shown).

        private static readonly RegionEndpoint ACMRegion =
 RegionEndpoint.USEast1;
```

```
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new AmazonCertificateManagerClient(ACMRegion);
            var certificateList = ListCertificatesResponseAsync(client: _client);

            Console.WriteLine("Certificate Summary List\n");

            foreach (var certificate in
 certificateList.Result.CertificateSummaryList)
            {
                Console.WriteLine($"Certificate Domain:
 {certificate.DomainName}");
                Console.WriteLine($"Certificate ARN:
 {certificate.CertificateArn}\n");
            }
        }

        /// <summary>
        /// Retrieves a list of the certificates defined in this Region.
        /// </summary>
        /// <param name="client">The ACM client object passed to the
        /// ListCertificateResAsync method call.</param>
        /// <param name="request"></param>
        /// <returns>The ListCertificatesResponse.</returns>
        static async Task<ListCertificatesResponse>
 ListCertificatesResponseAsync(
            AmazonCertificateManagerClient client)
        {
            var request = new ListCertificatesRequest();

            var response = await client.ListCertificatesAsync(request);
            return response;
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the AWS Certificate Manager (ACM) certificates in an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ListCertificatesRequest request;
    Aws::Vector<Aws::ACM::Model::CertificateSummary> allCertificates;
    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::ACM::Model::ListCertificatesOutcome outcome =
                acmClient.ListCertificates(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListCertificates: " <<
                      outcome.GetError().GetMessage() << std::endl;

            return false;
        }
        else {
            const Aws::ACM::Model::ListCertificatesResult &result =
 outcome.GetResult();

            const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates
 =
```

```
                    result.GetCertificateSummaryList();
            allCertificates.insert(allCertificates.end(), certificates.begin(),
                                   certificates.end());

            nextToken = result.GetNextToken();
        }
    } while (!nextToken.empty());

    if (!allCertificates.empty()) {
        for (const Aws::ACM::Model::CertificateSummary &certificate:
 allCertificates) {
            std::cout << "Certificate ARN: " <<
                      certificate.GetCertificateArn() << std::endl;
            std::cout << "Domain name:     " <<
                      certificate.GetDomainName() << std::endl << std::endl;
        }
    }
    else {
        std::cout << "No available certificates found in account."
                  << std::endl;
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um die ACM-Zertifikate für ein AWS Konto aufzulisten

Der folgende `list-certificates`-Befehl listet die ARNs der Zertifikate in Ihrem Konto auf:

```
aws acm list-certificates
```

Der vorhergehende Befehl erzeugt eine Ausgabe, die der folgenden ähnelt:

```
{
    "CertificateSummaryList": [
```

```
        {
            "CertificateArn":
 "arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012",
            "DomainName": "www.example.com"
        },
        {
            "CertificateArn": "arn:aws:acm:region:account:certificate/aaaaaaaa-
bbbb-cccc-dddd-eeeeeeeeeeee",
            "DomainName": "www.example.net"
        }
    ]
}
```

Sie können festlegen, wie viele Zertifikate bei jedem Aufruf von `list-certificates` angezeigt werden sollen. Wenn Sie beispielsweise vier Zertifikate haben und jeweils höchstens zwei gleichzeitig anzeigen möchten, setzen Sie das Argument `max-items` auf 2, wie im folgenden Beispiel gezeigt:

```
aws acm list-certificates --max-items 2
```

Es werden zwei Zertifikats-ARNs und ein `NextToken`-Wert angezeigt:

```
"CertificateSummaryList": [
  {
    "CertificateArn": "arn:aws:acm:region:account: \
            certificate/12345678-1234-1234-1234-123456789012",
    "DomainName": "www.example.com"
  },
  {
    "CertificateArn": "arn:aws:acm:region:account: \
             certificate/aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "DomainName": "www.example.net"
  }
  ],
    "NextToken": "9f4d9f69-275a-41fe-b58e-2b837bd9ba48"
```

Um die nächsten beiden Zertifikate in Ihrem Konto anzuzeigen, legen Sie den `NextToken`-Wert in Ihrem nächsten Aufruf fest:

```
aws acm list-certificates --max-items 2 --next-token 9f4d9f69-275a-41fe-
b58e-2b837bd9ba48
```

Sie können Ihre Ausgabe mithilfe des Arguments `certificate-statuses` filtern. Mit dem folgenden Befehl werden Zertifikate angezeigt, die den Status PENDING_VALIDATION haben:

```
aws acm list-certificates --certificate-statuses PENDING_VALIDATION
```

Sie können Ihre Ausgabe auch mithilfe des Arguments `includes` filtern. Der folgende Befehl zeigt Zertifikate an, die nach den folgenden Eigenschaften gefiltert werden. Die anzuzeigenden Zertifikate:

```
- Specify that the RSA algorithm and a 2048 bit key are used to generate key
 pairs.
- Contain a Key Usage extension that specifies that the certificates can be used
 to create digital signatures.
- Contain an Extended Key Usage extension that specifies that the certificates
 can be used for code signing.

aws acm list-certificates --max-items 10 --includes
 extendedKeyUsage=CODE_SIGNING,keyUsage=DIGITAL_SIGNATURE,keyTypes=RSA_2048
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that
 have a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest =
 ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft eine Liste aller Zertifikat-ARNs und den Domainnamen für jeden ARN ab. Das Cmdlet paginiert automatisch, um alle ARNs abzurufen. Um die Paginierung manuell zu steuern, verwenden Sie den MaxItem Parameter -, um zu steuern, wie viele Zertifikat-ARNs für jeden Serviceaufruf zurückgegeben werden, und den NextToken Parameter -, um den Startpunkt für jeden Aufruf anzugeben.

```
Get-ACMCertificateList
```

Ausgabe:

```
CertificateArn
   DomainName
--------------
   ----------
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Beispiel 2: Ruft eine Liste all Ihrer Zertifikat-ARNs ab, bei denen der Zertifikatsstatus mit den angegebenen Status übereinstimmt.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Beispiel 3: Dieses Beispiel gibt eine Liste aller Zertifikate in der Region us-east-1 zurück, die den Schlüsseltyp RSA_2048 und die erweiterte Schlüsselverwendung oder den Zweck CODE_SIGNING aufweisen. Die Werte für diese Filterparameter finden Sie im Referenzthema ListCertificates Filters API: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Ausgabe:

```
CertificateArn
   DomainName
```

```
--------------
   ----------
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-
af8d-2133d8f30zzz *.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-
a734-800430c80zzz nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-
bad559525zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-
e7ca-44c5-803e-24d9f2f36zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/
xxxxxxxx-1241-4b71-80b1-090305a62zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-
f94617c99zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-
e08ccc0eezzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-
a714-2d277d3eezzz *.route53docs.com
```

- Einzelheiten zur API finden Sie unter ListCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft eine Liste aller Zertifikat-ARNs und den Domainnamen für jeden ARN ab. Das Cmdlet paginiert automatisch, um alle ARNs abzurufen. Um die Paginierung manuell zu steuern, verwenden Sie den MaxItem Parameter -, um zu steuern, wie viele Zertifikat-ARNs für jeden Serviceaufruf zurückgegeben werden, und den NextToken Parameter -, um den Startpunkt für jeden Aufruf anzugeben.

```
Get-ACMCertificateList
```

Ausgabe:

```
CertificateArn
   DomainName
--------------
   ----------
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Beispiel 2: Ruft eine Liste all Ihrer Zertifikat-ARNs ab, bei denen der Zertifikatsstatus mit den angegebenen Status übereinstimmt.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Beispiel 3: Dieses Beispiel gibt eine Liste aller Zertifikate in der Region us-east-1 zurück, die den Schlüsseltyp RSA_2048 und die erweiterte Schlüsselverwendung oder den Zweck CODE_SIGNING aufweisen. Die Werte für diese Filterparameter finden Sie im Referenzthema ListCertificates Filters API: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Ausgabe:

```
CertificateArn
   DomainName
--------------
   ----------
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-
af8d-2133d8f30zzz *.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-
a734-800430c80zzz nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-
bad559525zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-
e7ca-44c5-803e-24d9f2f36zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/
xxxxxxxx-1241-4b71-80b1-090305a62zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-
f94617c99zzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-
e08ccc0eezzz
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-
a714-2d277d3eezzz *.route53docs.com
```

- Einzelheiten zur API finden Sie unter ListCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None,
 all
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If
 None,
                          all key usages are included.
        :param extended_key_usage: Filters the results to the specified extended
 key
```

```
                                    usages. If None, all extended key usages are
                                    included.
        :param key_types: Filters the results to the specified key types. If
None, all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates
```

- Einzelheiten zur API finden Sie ListCertificatesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_acm->listcertificates(
      iv_maxitems = iv_max_items
      it_certificatestatuses = it_statuses
      io_includes = io_includes
    ).
    MESSAGE 'Certificates listed successfully.' TYPE 'I'.
  CATCH /aws1/cx_acminvalidargsex.
    MESSAGE 'Invalid arguments provided.' TYPE 'I'.
  CATCH /aws1/cx_acmvalidationex.
    MESSAGE 'Validation error occurred.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListCertificatesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTagsForCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the tags for an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listTagsForCertificate(const Aws::String &certificateArn,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ListTagsForCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::ListTagsForCertificateOutcome outcome =
            acm_client.ListTagsForCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cout << "Error: ListTagsForCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Information about tags for "
                     "certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        Aws::Vector<Aws::ACM::Model::Tag> tags =
                result.GetTags();
```

```
        if (tags.size() > 0) {
            for (const Aws::ACM::Model::Tag &tag: tags) {
                std::cout << "Key:   " << tag.GetKey() << std::endl;
                std::cout << "Value: " << tag.GetValue()
                          << std::endl << std::endl;
            }
        }
        else {
            std::cout << "No tags found." << std::endl;
        }

        return true;
    }
}
```

- Einzelheiten zur API finden Sie ListTagsForCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie die Tags auf, die auf ein ACM-Zertifikat angewendet werden

Mit dem folgenden `list-tags-for-certificate`-Befehl werden die Tags aufgelistet, die auf ein Zertifikat in Ihrem Konto angewendet werden:

```
aws acm list-tags-for-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
```

Der vorhergehende Befehl erzeugt eine Ausgabe, die der folgenden ähnelt:

```
{
  "Tags": [
      {
          "Value": "Website",
          "Key": "Purpose"
      },
      {
          "Value": "Alice",
```

```
            "Key": "Admin"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListTagsForCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM)
 certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
        });
    }
}
```

- Einzelheiten zur API finden Sie ListTagsForCertificatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
certificate_arn)
            raise
        else:
            return tags
```

- Einzelheiten zur API finden Sie ListTagsForCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        DATA(lo_result) = lo_acm->listtagsforcertificate(
          iv_certificatearn = iv_certificate_arn
        ).
        ot_tags = lo_result->get_tags( ).
        MESSAGE 'Certificate tags retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTagsForCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RemoveTagsFromCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTagsFromCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Remove a tag from an ACM certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::removeTagsFromCertificate(const Aws::String &certificateArn,
                                            const Aws::String &tagKey,
                                            const Aws::String &tagValue,
                                            const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::Vector<Aws::ACM::Model::Tag> tags;

    Aws::ACM::Model::Tag tag;
    tag.SetKey(tagKey);

    tags.push_back(tag);

    Aws::ACM::Model::RemoveTagsFromCertificateRequest request;
    request.WithCertificateArn(certificateArn)
            .WithTags(tags);

    Aws::ACM::Model::RemoveTagsFromCertificateOutcome outcome =
            acmClient.RemoveTagsFromCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RemoveTagFromCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
        return false;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey << "' removed from "
                  << "certificate with ARN '" << certificateArn << "'." <<
 std::endl;

        return true;
    }
}
```

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So entfernen Sie ein Tag aus einem ACM-Zertifikat

Der folgende `remove-tags-from-certificate`-Befehl entfernt zwei Tags aus dem angegebenen Zertifikat. Verwenden Sie ein Leerzeichen, um mehrere Tags voneinander zu trennen:

```
aws acm remove-tags-from-certificate --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
 --tags Key=Admin,Value=Alice Key=Purpose,Value=Website
```

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
```

```
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from
 which to remove tags
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
 RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
```

```
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove_tags(self, certificate_arn, tags):
        """
        Removes tags from a certificate. If the value of a tag is specified, the
tag is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags),
certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
            )
            raise
```

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->removetagsfromcertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags removed from certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RenewCertificate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `RenewCertificate` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Renew an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::renewCertificate(const Aws::String &certificateArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RenewCertificateRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::RenewCertificateOutcome outcome =
            acmClient.RenewCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RenewCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Renewed certificate with ARN '"
                  << certificateArn << "'." << std::endl;

        return true;
    }
}
```

- Einzelheiten zur API finden Sie RenewCertificatein der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
```

```
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie RenewCertificatein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`RequestCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RequestCertificate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Request an AWS Certificate Manager (ACM) certificate.
/*!
  \param domainName: A fully qualified domain name.
  \param idempotencyToken: Customer chosen string for idempotency.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::requestCertificate(const Aws::String &domainName,
                                     const Aws::String &idempotencyToken,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RequestCertificateRequest request;
    request.WithDomainName(domainName)
            .WithIdempotencyToken(idempotencyToken);

    Aws::ACM::Model::RequestCertificateOutcome outcome =
            acmClient.RequestCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "RequestCertificate error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The newly requested certificate's "
                     "ARN is '" <<
                  outcome.GetResult().GetCertificateArn() <<
                  "'." << std::endl;
```

```
        return true;
    }
}
```

- Einzelheiten zur API finden Sie RequestCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fordern Sie ein neues ACM-Zertifikat an

Mit dem folgenden `request-certificate`-Befehl wird mithilfe der DNS-Validierung ein neues Zertifikat für die Domain www.example.com angefordert:

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS
```

Sie können ein Idempotenz-Token eingeben, um zwischen mehreren Aufrufen von `request-certificate` zu unterscheiden:

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS
 --idempotency-token 91adc45q
```

Sie können einen oder mehrere Subject Alternative Names angeben, um ein Zertifikat anzufordern, das mehr als eine Apex-Domain schützt.

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names www.example.net
```

Sie können einen alternativen Namen eingeben, der auch für den Zugriff auf Ihre Website verwendet werden kann:

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names www.example.com
```

Sie können ein Sternchen (*) als Platzhalter verwenden, um ein Zertifikat für mehrere Subdomains in derselben Domain zu erstellen:

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
idempotency-token 91adc45q --subject-alternative-names *.example.com
```

Sie können auch mehrere alternative Namen eingeben:

```
aws acm request-certificate --domain-name example.com --validation-method DNS --
subject-alternative-names b.example.com c.example.com d.example.com
```

Wenn Sie E-Mail für die Validierung verwenden, können Sie Optionen für die Domainvalidierung eingeben, um die Domain anzugeben, an die die Validierungs-E-Mail gesendet werden soll:

```
aws acm request-certificate --domain-name example.com --validation-
method EMAIL --subject-alternative-names www.example.com --domain-validation-
options DomainName=example.com,ValidationDomain=example.com
```

Mit dem folgenden Befehl wird die Protokollierung der Zertifikatstransparenz deaktiviert, wenn Sie ein neues Zertifikat anfordern:

```
aws acm request-certificate --domain-name www.example.com --validation-method DNS
 --options CertificateTransparencyLoggingPreference=DISABLED --idempotency-
token 184627
```

- Einzelheiten zur API finden Sie RequestCertificatein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RequestCert {

    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");

        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response =
 acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie RequestCertificatein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com"
```

Ausgabe:

```
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Beispiel 2: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Ausgabe:

```
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- Einzelheiten zur API finden Sie unter RequestCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com"
```

Ausgabe:

```
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Beispiel 2: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Ausgabe:

```
arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- Einzelheiten zur API finden Sie unter RequestCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being
 issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
```

```
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
when
                                   the email domain differs from the primary
domain of
                                   the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn
```

- Einzelheiten zur API finden Sie RequestCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       " iv_domain_name = 'example.com'
       " iv_validation_method = 'DNS' or 'EMAIL'
       DATA(lo_result) = lo_acm->requestcertificate(
         iv_domainname = iv_domain_name
         it_subjectalternativenames = COND #( WHEN it_alternate_domains IS NOT
INITIAL
                                                THEN it_alternate_domains )
         iv_validationmethod = iv_validation_method
       ).
       ov_certificate_arn = lo_result->get_certificatearn( ).
       MESSAGE 'Certificate requested successfully.' TYPE 'I'.
     CATCH /aws1/cx_acminvalidparameterex.
       MESSAGE 'Invalid parameter provided.' TYPE 'I'.
     CATCH /aws1/cx_acmlimitexceededex.
       MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
     CATCH /aws1/cx_acminvdomvationoptsex.
       MESSAGE 'Invalid domain validation options.' TYPE 'I'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie RequestCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ResendValidationEmail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ResendValidationEmail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Resend the email that requests domain ownership validation.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param domainName: A fully qualified domain name.
  \param validationDomain: The base validation domain that will act as the suffix
                           of the email addresses.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::resendValidationEmail(const Aws::String &certificateArn,
                                        const Aws::String &domainName,
                                        const Aws::String &validationDomain,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ResendValidationEmailRequest request;
    request.WithCertificateArn(certificateArn)
            .WithDomain(domainName)
            .WithValidationDomain(validationDomain);

    Aws::ACM::Model::ResendValidationEmailOutcome outcome =
            acmClient.ResendValidationEmail(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "ResendValidationEmail error: " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The validation email has been resent."
                  << std::endl;

        return true;
    }
}
```

- Einzelheiten zur API finden Sie ResendValidationEmailin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So senden Sie die Validierungs-E-Mail für Ihre ACM-Zertifikatsanfrage erneut

Der folgende `resend-validation-email`-Befehl weist die Amazon-Zertifizierungsstelle an, eine Validierungs-E-Mail an die entsprechenden Adressen zu senden:

```
aws acm resend-validation-email --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
 --domain www.example.com --validation-domain example.com
```

- Einzelheiten zur API finden Sie ResendValidationEmailin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Fordert an, dass die E-Mail zur Validierung der Eigentümerschaft der Domain „www.example.com“ gesendet wird. Wenn $ Ihrer Shell auf 'Medium' oder niedriger gesetzt ConfirmPreference ist, fordert das Cmdlet zur Bestätigung auf, bevor der Vorgang fortgesetzt wird. Fügen Sie den Schalter -Force hinzu, um Bestätigungsaufforderungen zu unterdrücken.

```
$params = @{
```

```
    CertificateArn="arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    Domain="www.example.com"
    ValidationDomain="example.com"
}
Send-ACMValidationEmail @params
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (ResendValidationEmailV4).

Tools für V5 PowerShell

Beispiel 1: Fordert an, dass die E-Mail zur Validierung der Eigentümerschaft der Domain „www.example.com“ gesendet wird. Wenn $ Ihrer Shell auf 'Medium' oder niedriger gesetzt ConfirmPreference ist, fordert das Cmdlet zur Bestätigung auf, bevor der Vorgang fortgesetzt wird. Fügen Sie den Schalter -Force hinzu, um Bestätigungsaufforderungen zu unterdrücken.

```
$params = @{
    CertificateArn="arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    Domain="www.example.com"
    ValidationDomain="example.com"
}
Send-ACMValidationEmail @params
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (ResendValidationEmailV5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
```

```
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def resend_validation_email(self, certificate_arn, domain,
validation_domain):
        """
        Request that validation email is sent again, for a certificate that was
        previously requested with email validation.

        :param certificate_arn: The ARN of the certificate.
        :param domain: The primary domain of the certificate.
        :param validation_domain: Alternate domain to use for determining email
                                  addresses to use for validation.
        """
        try:
            self.acm_client.resend_validation_email(
                CertificateArn=certificate_arn,
                Domain=domain,
                ValidationDomain=validation_domain,
            )
            logger.info(
                "Validation email resent to validation domain %s.",
validation_domain
            )
        except ClientError:
            logger.exception(
                "Couldn't resend validation email to %s.", validation_domain
            )
            raise
```

- Einzelheiten zur API finden Sie ResendValidationEmailin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        " iv_domain = 'example.com'
        " iv_validation_domain = 'example.com'
        lo_acm->resendvalidationemail(
          iv_certificatearn = iv_certificate_arn
          iv_domain = iv_domain
          iv_validationdomain = iv_validation_domain
        ).
        MESSAGE 'Validation email resent successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidstateex.
        MESSAGE 'Certificate is not in a valid state.' TYPE 'I'.
      CATCH /aws1/cx_acminvdomvationoptsex.
        MESSAGE 'Invalid domain validation options.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ResendValidationEmailin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateCertificateOptions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateCertificateOptions` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an AWS Certificate Manager (ACM) certificate option.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param loggingEnabled: Boolean specifying logging enabled.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::updateCertificateOption(const Aws::String &certificateArn,
                                          bool loggingEnabled,
                                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::UpdateCertificateOptionsRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::CertificateOptions options;

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(

 Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED);
    }
    else {
        options.SetCertificateTransparencyLoggingPreference(

 Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED);
    }

    request.SetOptions(options);

    Aws::ACM::Model::UpdateCertificateOptionsOutcome outcome =
```

```
            acmClient.UpdateCertificateOptions(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "UpdateCertificateOption error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The option '"
                  << (loggingEnabled ? "enabled" : "disabled") << "' has been set
 for "
                                                                  "the
 certificate with the ARN '"
                  << certificateArn << "'."
                  << std::endl;

        return true;
    }
}
```

- Einzelheiten zur API finden Sie UpdateCertificateOptionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktualisieren Sie die Zertifikatsoptionen

Mit dem folgenden `update-certificate-options`-Befehl wird die Protokollierung für Zertifikatstransparenz deaktiviert:

```
aws acm update-certificate-options --certificate-
arn arn:aws:acm:region:account:certificate/12345678-1234-1234-1234-123456789012
 --options CertificateTransparencyLoggingPreference=DISABLED
```

- Einzelheiten zur API finden Sie UpdateCertificateOptionsin der AWS CLI Befehlsreferenz.

Codebeispiele für API Gateway mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon API Gateway mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden AWS. Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Weitere -Quellen

- API Gateway Entwicklerhandbuch – Weitere Informationen zu API Gateway.
- API-Referenz zu API Gateway – Details zu allen verfügbaren Aktionen von API Gateway.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von API Gateway AWS SDKs
  - Aktionen für API Gateway mit AWS SDKs
    - Verwenden Sie CreateDeploymentmit einem AWS SDK oder CLI
    - Verwenden Sie CreateResourcemit einem AWS SDK oder CLI
    - Verwenden Sie CreateRestApimit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDeploymentmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteRestApimit einem AWS SDK oder CLI
    - Verwenden Sie GetBasePathMappingmit einem AWS SDK oder CLI
    - Verwenden Sie GetResourcesmit einem AWS SDK oder CLI
    - Verwenden Sie GetRestApismit einem AWS SDK oder CLI

    - Verwenden Sie ListBasePathMappingsmit einem AWS SDK
    - Verwenden Sie PutIntegrationmit einem AWS SDK oder CLI
    - Verwenden Sie PutIntegrationResponsemit einem AWS SDK oder CLI
    - Verwenden Sie PutMethodmit einem AWS SDK oder CLI
    - Verwenden Sie PutMethodResponsemit einem AWS SDK oder CLI
    - Verwenden Sie UpdateBasePathMappingmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von API Gateway AWS SDKs
  - Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
  - Leihbibliothek-REST-API erstellen
  - Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
  - Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
  - Erstellen einer Websocket-Chat-Anwendung mit API Gateway
  - Erstellen und implementieren Sie eine REST-API mit einem AWS SDK
  - Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- AWS Community-Beiträge für API Gateway
  - Erstellen und Testen einer Serverless-Anwendung

Grundlegende Beispiele für die Verwendung von API Gateway AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon API Gateway mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für API Gateway mit AWS SDKs
  - Verwenden Sie CreateDeploymentmit einem AWS SDK oder CLI
  - Verwenden Sie CreateResourcemit einem AWS SDK oder CLI
  - Verwenden Sie CreateRestApimit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDeploymentmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteRestApimit einem AWS SDK oder CLI
  - Verwenden Sie GetBasePathMappingmit einem AWS SDK oder CLI
  - Verwenden Sie GetResourcesmit einem AWS SDK oder CLI

- Verwenden Sie GetRestApismit einem AWS SDK oder CLI
- Verwenden Sie ListBasePathMappingsmit einem AWS SDK
- Verwenden Sie PutIntegrationmit einem AWS SDK oder CLI
- Verwenden Sie PutIntegrationResponsemit einem AWS SDK oder CLI
- Verwenden Sie PutMethodmit einem AWS SDK oder CLI
- Verwenden Sie PutMethodResponsemit einem AWS SDK oder CLI
- Verwenden Sie UpdateBasePathMappingmit einem AWS SDK oder CLI

Aktionen für API Gateway mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne API-Gateway-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die API Gateway API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von API Gateway AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon API Gateway.

Beispiele

- Verwenden Sie CreateDeploymentmit einem AWS SDK oder CLI
- Verwenden Sie CreateResourcemit einem AWS SDK oder CLI
- Verwenden Sie CreateRestApimit einem AWS SDK oder CLI
- Verwenden Sie DeleteDeploymentmit einem AWS SDK oder CLI
- Verwenden Sie DeleteRestApimit einem AWS SDK oder CLI
- Verwenden Sie GetBasePathMappingmit einem AWS SDK oder CLI
- Verwenden Sie GetResourcesmit einem AWS SDK oder CLI
- Verwenden Sie GetRestApismit einem AWS SDK oder CLI
- Verwenden Sie ListBasePathMappingsmit einem AWS SDK
- Verwenden Sie PutIntegrationmit einem AWS SDK oder CLI
- Verwenden Sie PutIntegrationResponsemit einem AWS SDK oder CLI
- Verwenden Sie PutMethodmit einem AWS SDK oder CLI

- Verwenden Sie PutMethodResponsemit einem AWS SDK oder CLI
- Verwenden Sie UpdateBasePathMappingmit einem AWS SDK oder CLI

Verwenden Sie **`CreateDeployment`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDeployment` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So stellen Sie die konfigurierten Ressourcen für eine API in einer neuen Stufe bereit

Befehl:

```
aws apigateway create-deployment --rest-api-id 1234123412 --stage-name dev --
stage-description 'Development Stage' --description 'First deployment to the dev
 stage'
```

So stellen Sie die konfigurierten Ressourcen für eine API in einer bestehenden Stufe bereit

Befehl:

```
aws apigateway create-deployment --rest-api-id 1234123412 --stage-name dev --
description 'Second deployment to the dev stage'
```

So stellen Sie die konfigurierten Ressourcen für eine API in einer bestehenden Stufe mit Stufenvariablen bereit

aws apigateway create-deployment --rest-api-id 1.234.123.412 --stage-name dev --description 'Third deployment to the dev stage' --variables key='value',otherKey='otherValue'

- Einzelheiten zur API finden Sie CreateDeploymentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

       try {
           CreateDeploymentRequest request = CreateDeploymentRequest.builder()
                   .restApiId(restApiId)
                   .description("Created using the AWS API Gateway Java API")
                   .stageName(stageName)
                   .build();

           CreateDeploymentResponse response =
apiGateway.createDeployment(request);
           System.out.println("The id of the deployment is " + response.id());
           return response.id();

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie CreateDeploymentin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from
 any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
```

```
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.
{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

- Einzelheiten zur API finden Sie CreateDeploymentin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->createdeployment(
```

```
          iv_restapiid = iv_rest_api_id
          iv_stagename = iv_stage_name
          iv_description = 'Deployment created by ABAP SDK' ).
        DATA(lv_deployment_id) = oo_result->get_id( ).
        MESSAGE 'Deployment created with ID: ' && lv_deployment_id TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDeploymentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateResource` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine Ressource in einer API

Befehl:

```
aws apigateway create-resource --rest-api-id 1234123412 --parent-id a1b2c3 --
path-part 'new-resource'
```

- Einzelheiten zur API finden Sie CreateResourcein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the
 parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
```

```
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id
```

- Einzelheiten zur API finden Sie CreateResourcein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->createresource(
          iv_restapiid = iv_rest_api_id
          iv_parentid = iv_parent_id
          iv_pathpart = iv_resource_path ).
        DATA(lv_resource_id) = oo_result->get_id( ).
        MESSAGE 'Resource created with ID: ' && lv_resource_id TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateRestApi`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRestApi` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine API

Befehl:

```
aws apigateway create-rest-api --name 'My First API' --description 'This is my
 first API'
```

So erstellen Sie eine doppelte API aus einer vorhandenen API

Befehl:

```
aws apigateway create-rest-api --name 'Copy of My First API' --description 'This
 is a copy of my first API' --clone-from 1234123412
```

- Einzelheiten zur API finden Sie CreateRestApiin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

       try {
           CreateRestApiRequest request = CreateRestApiRequest.builder()
                   .cloneFrom(restApiId)
                   .description("Created using the Gateway Java API")
                   .name(restApiName)
                   .build();

           CreateRestApiResponse response = apiGateway.createRestApi(request);
           System.out.println("The id of the new api is " + response.id());
           return response.id();

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie CreateRestApiin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root
 resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used
 in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
```

```
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- Einzelheiten zur API finden Sie CreateRestApiin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->createrestapi(
          iv_name = iv_api_name
          iv_description = 'Sample REST API created by ABAP SDK' ).
        DATA(lv_api_id) = oo_result->get_id( ).
        MESSAGE 'REST API created with ID: ' && lv_api_id TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
      CATCH /aws1/cx_agwunauthorizedex INTO DATA(lo_unauthorized).
        MESSAGE lo_unauthorized->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_unauthorized.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateRestApiin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteDeployment`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDeployment` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Bereitstellung in einer API

Befehl:

```
aws apigateway delete-deployment --rest-api-id 1234123412 --deployment-id a1b2c3
```

- Einzelheiten zur API finden Sie DeleteDeploymentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteSpecificDeployment(ApiGatewayClient apiGateway,
String restApiId, String deploymentId) {
```

```
        try {
            DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
                    .restApiId(restApiId)
                    .deploymentId(deploymentId)
                    .build();

            apiGateway.deleteDeployment(request);
            System.out.println("Deployment was deleted");

        } catch (ApiGatewayException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDeploymentin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteRestApi`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRestApi` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So löschen Sie eine API

Befehl:

```
aws apigateway delete-rest-api --rest-api-id 1234123412
```

- Einzelheiten zur API finden Sie DeleteRestApiin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {

    try {
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()
                .restApiId(restApiId)
                .build();

        apiGateway.deleteRestApi(request);
        System.out.println("The API was successfully deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Einzelheiten zur API finden Sie DeleteRestApiin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
```

```
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def delete_rest_api(self):
        """
        Deletes a REST API, including all of its resources and configuration.
        """
        try:
            self.apig_client.delete_rest_api(restApiId=self.api_id)
            logger.info("Deleted REST API %s.", self.api_id)
            self.api_id = None
        except ClientError:
            logger.exception("Couldn't delete REST API %s.", self.api_id)
            raise
```

- Einzelheiten zur API finden Sie DeleteRestApiin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_agw->deleterestapi(
      iv_restapiid = iv_rest_api_id ).
    MESSAGE 'REST API deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteRestApiin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetBasePathMapping`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBasePathMapping` verwendet wird.

CLI

AWS CLI

So rufen Sie die Basispfadzuordnung für einen benutzerdefinierten Domainnamen ab

Befehl:

```
aws apigateway get-base-path-mapping --domain-name subdomain.domain.tld --base-
path v1
```

Ausgabe:

```
{
    "basePath": "v1",
    "restApiId": "1234w4321e",
    "stage": "api"
}
```

- Einzelheiten zur API finden Sie GetBasePathMappingin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
```

```
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- Einzelheiten zur API finden Sie GetBasePathMappingin der AWS SDK für PHP API-Referenz.

Verwenden Sie **`GetResources`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetResources` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So rufen Sie eine Liste der Ressourcen für eine REST-API ab

Befehl:

```
aws apigateway get-resources --rest-api-id 1234123412
```

Ausgabe:

```
{
    "items": [
        {
            "path": "/resource/subresource",
            "resourceMethods": {
                "POST": {}
            },
            "id": "024ace",
            "pathPart": "subresource",
            "parentId": "ai5b02"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetResourcesin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root
 resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used
 in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
```

```
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- Einzelheiten zur API finden Sie GetResourcesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_agw->getresources(
      iv_restapiid = iv_rest_api_id ).
    DATA(lt_resources) = oo_result->get_items( ).
    DATA(lv_count) = lines( lt_resources ).
    MESSAGE 'Found ' && lv_count && ' resources' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetResourcesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetRestApis`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetRestApis` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So rufen Sie eine Liste der REST-APIs ab

Befehl:

```
aws apigateway get-rest-apis
```

Ausgabe:

```
{
    "items": [
        {
            "createdDate": 1438884790,
            "id": "12s44z21rb",
            "name": "My First API"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetRestApisin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def get_rest_api_id(self, api_name):
        """
        Gets the ID of a REST API from its name by searching the list of REST
 APIs
        for the current account. Because names need not be unique, this returns
 only
        the first API with the specified name.

        :param api_name: The name of the API to look up.
        :return: The ID of the specified API.
        """
        try:
            rest_api = None
            paginator = self.apig_client.get_paginator("get_rest_apis")
            for page in paginator.paginate():
```

```
                rest_api = next(
                    (item for item in page["items"] if item["name"] == api_name),
None
                )
                if rest_api is not None:
                    break
            self.api_id = rest_api["id"]
            logger.info("Found ID %s for API %s.", rest_api["id"], api_name)
        except ClientError:
            logger.exception("Couldn't find ID for API %s.", api_name)
            raise
        else:
            return rest_api["id"]
```

- Einzelheiten zur API finden Sie GetRestApisin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt die REST-APIs von Amazon API Gateway in der Region an.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
    let resp = client.get_rest_apis().send().await?;

    for api in resp.items() {
        println!("ID:          {}", api.id().unwrap_or_default());
        println!("Name:        {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version:     {}", api.version().unwrap_or_default());
        println!(
            "Created:     {}",
            api.created_date().unwrap().to_chrono_utc()?
        );
```

```
        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie GetRestApisin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->getrestapis( ).
        DATA(lt_apis) = oo_result->get_items( ).
        DATA(lv_count) = lines( lt_apis ).
        MESSAGE 'Found ' && lv_count && ' REST APIs' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetRestApisin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListBasePathMappings`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListBasePathMappings`.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
```

```
    }
}

function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- Einzelheiten zur API finden Sie ListBasePathMappingsin der AWS SDK für PHP API-Referenz.

Verwenden Sie **`PutIntegration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutIntegration` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine Mock-Integrationsanforderung

Befehl:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3
 --http-method GET --type MOCK --request-templates '{ "application/json":
 "{\"statusCode\": 200}" }'
```

So erstellen Sie eine HTTP-Integrationsanforderung

Befehl:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-id a1b2c3
 --http-method GET --type HTTP --integration-http-method GET --uri 'https://
domain.tld/path'
```

So erstellen Sie eine AWS Integrationsanfrage mit einem Lambda-Funktionsendpunkt

Befehl:

```
aws apigateway put-integration --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --type AWS --integration-http-method POST
 --uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:function_name/invocations'
```

- Einzelheiten zur API finden Sie PutIntegrationin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """
```

```
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
```

```
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
```

```
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Einzelheiten zur API finden Sie PutIntegrationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->putintegration(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_type = 'AWS_PROXY'
          iv_integrationhttpmethod = 'POST'
          iv_uri = iv_integration_uri ).
        MESSAGE 'Integration configured for method' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
```

```
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutIntegrationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutIntegrationResponse`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutIntegrationResponse` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine Integrationsantwort als Standardantwort mit einer definierten Zuweisungsvorlage

Befehl:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --status-code 200 --selection-pattern "" --response-
templates '{"application/json": "{\"json\": \"template\"}"}'
```

So erstellen Sie eine Integrationsantwort mit einer Regex von 400 und einem statisch definierten Header-Wert

Befehl:

```
aws apigateway put-integration-response --rest-api-id 1234123412 --resource-
id a1b2c3 --http-method GET --status-code 400 --selection-pattern 400 --response-
parameters '{"method.response.header.custom-header": "'"'"'custom-value'"'"'"}'
```

- Einzelheiten zur API finden Sie PutIntegrationResponsein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
```

```
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
```

```
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Einzelheiten zur API finden Sie PutIntegrationResponsein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_agw->putintegrationresponse(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_statuscode = '200' ).
    MESSAGE 'Integration response configured for status 200' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutIntegrationResponsein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutMethod`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutMethod` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine Methode für eine Ressource in einer API ohne Autorisierung, ohne API-Schlüssel und mit einem benutzerdefinierten Methodenanforderungsheader

Befehl:

```
aws apigateway put-method --rest-api-id 1234123412 --resource-id a1b2c3 --
http-method PUT --authorization-type "NONE" --no-api-key-required --request-
parameters "method.request.header.custom-header=false"
```

- Einzelheiten zur API finden Sie PutMethodin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
```

```
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
```

```
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
```

```
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Einzelheiten zur API finden Sie PutMethodin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->putmethod(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_authorizationtype = 'NONE' ).
        MESSAGE 'Method ' && iv_http_method && ' added to resource' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
```

```
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutMethodin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutMethodResponse`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutMethodResponse` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Bereitstellen einer REST-API

CLI

AWS CLI

So erstellen Sie eine Methodenantwort unter dem angegebenen Statuscode mit einem benutzerdefinierten Methodenantwortheader

Befehl:

```
aws apigateway put-method-response --rest-api-id 1234123412 --
resource-id a1b2c3 --http-method GET --status-code 400 --response-
parameters "method.response.header.custom-header=false"
```

- Einzelheiten zur API finden Sie PutMethodResponsein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
 integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
 with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
 POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
```

```
                        Gateway permission to use the specified action with the
                        service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
```

```
                requestTemplates={"application/json":
 json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- Einzelheiten zur API finden Sie PutMethodResponsein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_agw->putmethodresponse(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_statuscode = '200' ).
        MESSAGE 'Method response configured for status 200' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutMethodResponsein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateBasePathMapping`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateBasePathMapping` verwendet wird.

CLI

AWS CLI

So ändern Sie den Basispfad für einen benutzerdefinierten Domainnamen

Befehl:

```
aws apigateway update-base-path-mapping --domain-name api.domain.tld --base-
path prod --patch-operations op='replace',path='/basePath',value='v1'
```

Ausgabe:

```
{
    "basePath": "v1",
    "restApiId": "1234123412",
```

```
    "stage": "api"
}
```

- Einzelheiten zur API finden Sie UpdateBasePathMappingin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
```

```
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- Einzelheiten zur API finden Sie UpdateBasePathMappingin der AWS SDK für PHP API-Referenz.

Szenarien für die Verwendung von API Gateway AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in API Gateway mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von API Gateway oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Leihbibliothek-REST-API erstellen
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen einer Websocket-Chat-Anwendung mit API Gateway
- Erstellen und implementieren Sie eine REST-API mit einem AWS SDK
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die anhand fiktiver Daten ein System zur Nachverfolgung täglicher Fälle COVID-19 in den Vereinigte Staaten simuliert.

Python

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon AWS Lambda DynamoDB verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle von Infektionen COVID-19 in Vereinigte Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über API Gateway eingehen.

- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

Python

SDK für Python (Boto3)

Zeigt, wie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) API und AWS Chalice verwendet wird, um eine REST-API zu erstellen, die von einer Amazon Aurora Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldedaten verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.

- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Eine Lambda-Funktion erstellen und bereitstellen
- Erstellen Sie eine REST-API
- Lambda-Proxyintegration konfigurieren
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value
```

```
        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
```

```
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
```

```
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}
```

```
# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""
```

```
# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy

- CreateDeployment
- CreateFunction
- CreateResource
- CreateRestApi
- CreateRole
- DeleteFunction
- DeleteRestApi
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetResources
- GetRestApis
- PutIntegration
- PutMethod

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Websocket-Chat-Anwendung mit API Gateway

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon API Gateway V2 eine Websocket-API erstellen, die in Amazon DynamoDB integriert AWS Lambda werden kann.

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.

- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Erstellen und implementieren Sie eine REST-API mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine REST-API, die von API Gateway bereitgestellt wird.
- Fügen Sie der REST-API Ressourcen hinzu, um ein Benutzerprofil darzustellen.
- Fügen Sie Integrationsmethoden hinzu, damit die REST-API eine DynamoDB-Tabelle verwendet, um Benutzerdatenprofile zu speichern.
- Senden Sie HTTP-Anfragen an die REST-API, um Benutzerprofile hinzuzufügen und abzurufen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die API-Gateway-Operationen umschließt.

```
import argparse
import json
```

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root
 resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used
 in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise
```

```
        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the
parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
```

```
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The
integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate
REST
                                 elements, such as query parameters, to the
request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
```

```
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json":
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

```
    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from
 any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.
{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

Stellen Sie eine REST-API bereit und rufen Sie sie mit dem Requests-Paket auf.

```
def usage_demo(table_name, role_name, rest_api_name):
    """
    Demonstrates how to used API Gateway to create and deploy a REST API, and how
    to use the Requests package to call it.

    :param table_name: The name of the demo DynamoDB table.
    :param role_name: The name of the demo role that grants API Gateway
 permission to
                        call DynamoDB.
    :param rest_api_name: The name of the demo REST API created by the demo.
    """
    gateway = ApiGatewayToService(boto3.client("apigateway"))
    role = boto3.resource("iam").Role(role_name)

    print("Creating REST API in API Gateway.")
    gateway.create_rest_api(rest_api_name)

    print("Adding resources to the REST API.")
    profiles_id = gateway.add_rest_resource(gateway.root_id, "profiles")
    username_id = gateway.add_rest_resource(profiles_id, "{username}")

    # The DynamoDB service requires that all integration requests use POST.
    print("Adding integration methods to read and write profiles in Amazon
 DynamoDB.")
    gateway.add_integration_method(
        profiles_id,
        "GET",
        "dynamodb",
        "Scan",
        "POST",
        role.arn,
        {"TableName": table_name},
    )
    gateway.add_integration_method(
        profiles_id,
        "POST",
        "dynamodb",
        "PutItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Item": {
```

```
                "username": {"S": "$input.path('$.username')"},
                "name": {"S": "$input.path('$.name')"},
                "title": {"S": "$input.path('$.title')"},
            },
        },
    )
    gateway.add_integration_method(
        username_id,
        "GET",
        "dynamodb",
        "GetItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Key": {"username": {"S": "$method.request.path.username"}},
        },
    )

    stage = "test"
    print(f"Deploying the {stage} stage.")
    gateway.deploy_api(stage)

    profiles_url = gateway.api_url("profiles")
    print(
        f"Using the Requests package to post some people to the profiles REST API
at "
        f"{profiles_url}."
    )
    requests.post(
        profiles_url,
        json={"username": "will", "name": "William Shakespeare", "title":
"playwright"},
    )
    requests.post(
        profiles_url,
        json={
            "username": "ludwig",
            "name": "Ludwig van Beethoven",
            "title": "composer",
        },
    )
    requests.post(
        profiles_url,
```

```
        json={"username": "jane", "name": "Jane Austen", "title": "author"},
    )
    print("Getting the list of profiles from the REST API.")
    profiles = requests.get(profiles_url).json()
    pprint(profiles)
    print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/
jane).")
    jane = requests.get(f"{profiles_url}/jane").json()
    pprint(jane)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDeployment
  - CreateResource
  - CreateRestApi
  - DeleteRestApi
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutIntegrationResponse
  - PutMethod
  - PutMethodResponse

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt,

die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS Community-Beiträge für API Gateway

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden AWS. Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Beispiele

- Erstellen und Testen einer Serverless-Anwendung

Erstellen und Testen einer Serverless-Anwendung

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

.NET

SDK für .NET

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des .NET SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Go

SDK für Go V2

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Java

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Rust

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Codebeispiele für die API Gateway Management API mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie die Amazon API Gateway Management API mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch zur Management-API von API Gateway – Weitere Informationen zur Management-API von API Gateway.
- API-Referenz zum API Gateway Management – Details zu allen verfügbaren Management-API-Aktionen von API Gateway.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung der API Gateway Management API AWS SDKs
  - Aktionen für die API Gateway Management API unter Verwendung AWS SDKs
    - Verwenden Sie PostToConnectionmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung der API Gateway Management API AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Management API von Amazon API Gateway mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für die API Gateway Management API unter Verwendung AWS SDKs
  - Verwenden Sie PostToConnectionmit einem AWS SDK oder CLI

Aktionen für die API Gateway Management API unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne API Gateway Management API-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zum Management API von API Gateway.

Beispiele

- Verwenden Sie PostToConnectionmit einem AWS SDK oder CLI

Verwenden Sie **`PostToConnection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PostToConnection` verwendet wird.

CLI

AWS CLI

Um Daten an eine WebSocket Verbindung zu senden

Das folgende `post-to-connection` Beispiel sendet eine Nachricht an einen Client, der mit der angegebenen WebSocket API verbunden ist.

```
aws apigatewaymanagementapi post-to-connection \
    --connection-id L0SM9cOFvHcCIhw= \
    --data "Hello from API Gateway!" \
    --endpoint-url https://aabbccddee.execute-api.us-west-2.amazonaws.com/prod
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden von @connections-Befehlen in Ihrem Backend-Service im Entwicklerhandbuch für Amazon API Gateway.

- Einzelheiten zur API finden Sie PostToConnectionunter AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

    let endpoint_url = format!(
        "https://{api_id}.execute-api.{region}.amazonaws.com/{stage}",
        api_id = api_id,
        region = region,
        stage = stage
    );

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let api_management_config = config::Builder::from(&shared_config)
        .endpoint_url(endpoint_url)
        .build();
    let client = Client::from_conf(api_management_config);
```

- Einzelheiten zur API finden Sie PostToConnectionin der API-Referenz zum AWS SDK für Rust.

Codebeispiele für Application Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Application Auto Scaling mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Benutzerhandbuch zum Application Auto Scaling – Weitere Informationen zu Application Auto Scaling.
- API-Referenz zum Application Auto Scaling – Details zu allen verfügbaren Aktionen zum Application Auto Scaling.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Application Auto Scaling mit AWS SDKs
  - Aktionen für Application Auto Scaling mit AWS SDKs
    - Verwenden Sie DeleteScalingPolicymit einem AWS SDK oder CLI
    - Verwendung von DeleteScheduledAction mit einer CLI
    - Verwendung von DeregisterScalableTarget mit einer CLI
    - Verwendung von DescribeScalableTargets mit einer CLI
    - Verwendung von DescribeScalingActivities mit einer CLI
    - Verwenden Sie mit einem DescribeScalingPolicies AWS SDK oder CLI
    - Verwendung von DescribeScheduledActions mit einer CLI
    - Verwendung von PutScalingPolicy mit einer CLI
    - Verwendung von PutScheduledAction mit einer CLI
    - Verwenden Sie mit einem RegisterScalableTarget AWS SDK oder CLI

Grundlegende Beispiele für Application Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Application Auto Scaling mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Application Auto Scaling mit AWS SDKs
  - Verwenden Sie DeleteScalingPolicymit einem AWS SDK oder CLI
  - Verwendung von DeleteScheduledAction mit einer CLI

- Verwendung von DeregisterScalableTarget mit einer CLI
- Verwendung von DescribeScalableTargets mit einer CLI
- Verwendung von DescribeScalingActivities mit einer CLI
- Verwenden Sie mit einem DescribeScalingPolicies AWS SDK oder CLI
- Verwendung von DescribeScheduledActions mit einer CLI
- Verwendung von PutScalingPolicy mit einer CLI
- Verwendung von PutScheduledAction mit einer CLI
- Verwenden Sie mit einem RegisterScalableTarget AWS SDK oder CLI

Aktionen für Application Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Application Auto Scaling Scaling-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine komplette Liste finden Sie in der API-Referenz zum Application Auto Scaling.

Beispiele

- Verwenden Sie DeleteScalingPolicymit einem AWS SDK oder CLI
- Verwendung von DeleteScheduledAction mit einer CLI
- Verwendung von DeregisterScalableTarget mit einer CLI
- Verwendung von DescribeScalableTargets mit einer CLI
- Verwendung von DescribeScalingActivities mit einer CLI
- Verwenden Sie mit einem DescribeScalingPolicies AWS SDK oder CLI
- Verwendung von DescribeScheduledActions mit einer CLI
- Verwendung von PutScalingPolicy mit einer CLI
- Verwendung von PutScheduledAction mit einer CLI
- Verwenden Sie mit einem RegisterScalableTarget AWS SDK oder CLI

Verwenden Sie **`DeleteScalingPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteScalingPolicy` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Skalierungsrichtlinie

In diesem Beispiel wird eine Skalierungsrichtlinie für die Web-App des Amazon-ECS-Service gelöscht, die im Standard-Cluster ausgeführt wird.

Befehl:

```
aws application-autoscaling delete-scaling-policy --policy-name web-app-cpu-lt-25
 --scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-
app --service-namespace ecs
```

- Einzelheiten zur API finden Sie DeleteScalingPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReq
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
```

```
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
```

```
    public static void deregisterScalableTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
 DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- Einzelheiten zur API finden Sie DeleteScalingPolicyin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht die angegebene Skalierungsrichtlinie für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Einzelheiten zur API finden Sie unter DeleteScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht die angegebene Skalierungsrichtlinie für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-
out" -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- Einzelheiten zur API finden Sie unter DeleteScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteScheduledAction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteScheduledAction` verwendet wird.

CLI

AWS CLI

Löschen einer geplanten Aktion

Das folgende `delete-scheduled-action` Beispiel löscht die angegebene geplante Aktion aus der angegebenen Amazon AppStream 2.0-Flotte:

```
aws application-autoscaling delete-scheduled-action \
    --service-namespace appstream \
    --scalable-dimension appstream:fleet:DesiredCapacity \
    --resource-id fleet/sample-fleet \
```

```
    --scheduled-action-name my-recurring-action
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch zu Application Auto Scaling.

- Einzelheiten zur API finden Sie DeleteScheduledActionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht die angegebene geplante Aktion für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht die angegebene geplante Aktion für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterScalableTarget`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterScalableTarget` verwendet wird.

CLI

AWS CLI

So registrieren Sie ein skalierbares Ziel

In diesem Beispiel wird die Registrierung eines skalierbaren Ziels für einen Amazon ECS-Service mit dem Namen Web-App aufgehoben, der im Standard-Cluster ausgeführt wird.

Befehl:

```
aws application-autoscaling deregister-scalable-target --service-namespace ecs --
scalable-dimension ecs:service:DesiredCount --resource-id service/default/web-app
```

In diesem Beispiel wird die Registrierung eines skalierbaren Ziels für eine benutzerdefinierte Ressource aufgehoben. Die Datei custom-resource-id.txt enthält eine Zeichenfolge, die die Ressourcen-ID identifiziert. Bei einer benutzerdefinierten Ressource ist dies der Pfad zur benutzerdefinierten Ressource über Ihren Amazon-API-Gateway-Endpunkt.

Befehl:

```
aws application-autoscaling deregister-scalable-target --
service-namespace custom-resource --scalable-dimension custom-
resource:ResourceType:Property --resource-id file://~/custom-resource-id.txt
```

Inhalt der Datei custom-resource-id.txt:

```
https://example.execute-api.us-west-2.amazonaws.com/prod/
scalableTargetDimensions/1-23456789
```

- Einzelheiten zur API finden Sie DeregisterScalableTargetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet hebt die Registrierung eines skalierbaren Ziels von Application Auto Scaling auf. Durch das Aufheben der Registrierung eines skalierbaren Ziels werden die zugehörigen Skalierungsrichtlinien gelöscht.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeregisterScalableTarget AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet hebt die Registrierung eines skalierbaren Ziels von Application Auto Scaling auf. Durch das Aufheben der Registrierung eines skalierbaren Ziels werden die zugehörigen Skalierungsrichtlinien gelöscht.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeregisterScalableTarget AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeScalableTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalableTargets` verwendet wird.

CLI

AWS CLI

So beschreiben Sie skalierbare Ziele

Das folgende Beispiel für `describe-scalable-targets` beschreibt die skalierbaren Ziele für den `ecs`-Service-Namespace.

```
aws application-autoscaling describe-scalable-targets \
    --service-namespace ecs
```

Ausgabe:

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "ecs",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "MinCapacity": 1,
            "MaxCapacity": 10,
            "RoleARN": "arn:aws:iam::123456789012:role/
aws-service-role/ecs.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ECSService",
            "CreationTime": 1462558906.199,
            "SuspendedState": {
                "DynamicScalingOutSuspended": false,
                "ScheduledScalingSuspended": false,
```

```
                "DynamicScalingInSuspended": false
            },
            "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS -Services, die Sie mit Application Auto Scaling nutzen können im Benutzerhandbuch für Application Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScalableTargetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den skalierbaren Zielen von Auto Scaling für Anwendungen im angegebenen Namespace bereitgestellt.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Ausgabe:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Einzelheiten zur API finden Sie unter DescribeScalableTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den skalierbaren Zielen von Auto Scaling für Anwendungen im angegebenen Namespace bereitgestellt.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Ausgabe:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- Einzelheiten zur API finden Sie unter DescribeScalableTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeScalingActivities`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingActivities` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Skalierungsaktivitäten für den angegebenen Amazon-ECS-Service

Das folgende Beispiel für `describe-scaling-activities` beschreibt die Skalierungsaktivitäten für einen Amazon-ECS-Service mit dem Namen `web-app`, der im `default`-Cluster ausgeführt wird. Die Ausgabe zeigt eine Skalierungsaktivität, die durch eine Skalierungsrichtlinie initiiert wurde.

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace ecs \
    --resource-id service/default/web-app
```

Ausgabe:

```
{
```

```
    "ScalingActivities": [
        {
            "ScalableDimension": "ecs:service:DesiredCount",
            "Description": "Setting desired count to 1.",
            "ResourceId": "service/default/web-app",
            "ActivityId": "e6c5f7d1-dbbb-4a3f-89b2-51f33e766399",
            "StartTime": 1462575838.171,
            "ServiceNamespace": "ecs",
            "EndTime": 1462575872.111,
            "Cause": "monitor alarm web-app-cpu-lt-25 in state ALARM triggered
 policy web-app-cpu-lt-25",
            "StatusMessage": "Successfully set desired count to 1. Change
 successfully fulfilled by ecs.",
            "StatusCode": "Successful"
        }
    ]
}
```

Weitere Informationen finden Sie unter Skalierungsaktivitäten für Application Auto Scaling im Benutzerhandbuch für Application Auto Scaling.

Beispiel 2: So beschreiben Sie Skalierungsaktivitäten für die angegebene DynamoDB-Tabelle

Das folgende Beispiel für `describe-scaling-activities` beschreibt die Skalierungsaktivitäten für eine DynamoDB-Tabelle mit dem Namen `TestTable`. Die Ausgabe zeigt Skalierungsaktivitäten, die durch zwei verschiedene geplante Aktionen ausgelöst wurden.

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace dynamodb \
    --resource-id table/TestTable
```

Ausgabe:

```
{
    "ScalingActivities": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 10.",
            "ResourceId": "table/my-table",
            "ActivityId": "4d1308c0-bbcf-4514-a673-b0220ae38547",
            "StartTime": 1561574415.086,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574449.51,
```

```
            "Cause": "maximum capacity was set to 10",
            "StatusMessage": "Successfully set write capacity units to 10. Change
successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 5 and max capacity to 10",
            "ResourceId": "table/my-table",
            "ActivityId": "f2b7847b-721d-4e01-8ef0-0c8d3bacc1c7",
            "StartTime": 1561574414.644,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-second-scheduled-action was
triggered",
            "StatusMessage": "Successfully set min capacity to 5 and max capacity
to 10",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting write capacity units to 15.",
            "ResourceId": "table/my-table",
            "ActivityId": "d8ea4de6-9eaa-499f-b466-2cc5e681ba8b",
            "StartTime": 1561574108.904,
            "ServiceNamespace": "dynamodb",
            "EndTime": 1561574140.255,
            "Cause": "minimum capacity was set to 15",
            "StatusMessage": "Successfully set write capacity units to 15. Change
successfully fulfilled by dynamodb.",
            "StatusCode": "Successful"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Description": "Setting min capacity to 15 and max capacity to 20",
            "ResourceId": "table/my-table",
            "ActivityId": "3250fd06-6940-4e8e-bb1f-d494db7554d2",
            "StartTime": 1561574108.512,
            "ServiceNamespace": "dynamodb",
            "Cause": "scheduled action name my-first-scheduled-action was
triggered",
            "StatusMessage": "Successfully set min capacity to 15 and max
capacity to 20",
            "StatusCode": "Successful"
        }
```

```
    ]
}
```

Weitere Informationen finden Sie unter Skalierungsaktivitäten für Application Auto Scaling im Benutzerhandbuch für Application Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Liefert beschreibende Informationen zu den Skalierungsaktivitäten im angegebenen Service-Namespace aus den letzten sechs Wochen.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Ausgabe:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Einzelheiten zur API finden Sie unter DescribeScalingActivities AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Liefert beschreibende Informationen zu den Skalierungsaktivitäten im angegebenen Service-Namespace aus den letzten sechs Wochen.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Ausgabe:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in
 state ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- Einzelheiten zur API finden Sie unter DescribeScalingActivities AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeScalingPolicies`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingPolicies` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Skalierungsrichtlinien

Dieser Beispielbefehl beschreibt die Skalierungsrichtlinien für den ECS-Service-Namespace.

Befehl:

```
aws application-autoscaling describe-scaling-policies --service-namespace ecs
```

Ausgabe:

```
{
    "ScalingPolicies": [
        {
            "PolicyName": "web-app-cpu-gt-75",
            "ScalableDimension": "ecs:service:DesiredCount",
```

```
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462561899.23,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 60,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": 200,
                        "MetricIntervalLowerBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-gt-75",
            "PolicyType": "StepScaling",
            "Alarms": [
                {
                    "AlarmName": "web-app-cpu-gt-75",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-gt-75"
                }
            ],
            "ServiceNamespace": "ecs"
        },
        {
            "PolicyName": "web-app-cpu-lt-25",
            "ScalableDimension": "ecs:service:DesiredCount",
            "ResourceId": "service/default/web-app",
            "CreationTime": 1462562575.099,
            "StepScalingPolicyConfiguration": {
                "Cooldown": 1,
                "StepAdjustments": [
                    {
                        "ScalingAdjustment": -50,
                        "MetricIntervalUpperBound": 0.0
                    }
                ],
                "AdjustmentType": "PercentChangeInCapacity"
            },
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/web-app-cpu-lt-25",
            "PolicyType": "StepScaling",
```

```
            "Alarms": [
                {
                    "AlarmName": "web-app-cpu-lt-25",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:web-app-cpu-lt-25"
                }
            ],
            "ServiceNamespace": "ecs"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeScalingPoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet beschreibt die Richtlinien für Application Auto Scaling für den angegebenen Service-Namespace.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Ausgabe:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                            : 9/3/2019 2:48:15 AM
PolicyARN                               : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                          policyName/default-scale-out
PolicyName                              : default-scale-out
PolicyType                              : StepScaling
ResourceId                              : fleet/LabFleet
ScalableDimension                       : appstream:fleet:DesiredCapacity
ServiceNamespace                        : appstream
StepScalingPolicyConfiguration          :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

```
Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                                : default-scale-in
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Einzelheiten zur API finden Sie unter DescribeScalingPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet beschreibt die Richtlinien für Application Auto Scaling für den angegebenen Service-Namespace.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Ausgabe:

```
Alarms                                    : {Appstream2-LabFleet-default-scale-
out-Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                                : default-scale-out
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

```
Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                                : default-scale-in
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- Einzelheiten zur API finden Sie unter DescribeScalingPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());
```

```
    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeScalingPoliciesin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DescribeScheduledActions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScheduledActions` verwendet wird.

CLI

AWS CLI

So beschreiben Sie geplante Aktionen

Das folgende Beispiel für `describe-scheduled-actions` zeigt Details zu den geplanten Aktionen für den angegebenen Service-Namespace:

```
aws application-autoscaling describe-scheduled-actions \
    --service-namespace dynamodb
```

Ausgabe:

```
{
    "ScheduledActions": [
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:35:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571888.361,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-first-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 15,
                "MaxCapacity": 20
            },
            "ScheduledActionName": "my-first-scheduled-action",
```

```
            "ServiceNamespace": "dynamodb"
        },
        {
            "ScalableDimension": "dynamodb:table:WriteCapacityUnits",
            "Schedule": "at(2019-05-20T18:40:00)",
            "ResourceId": "table/my-table",
            "CreationTime": 1561571946.021,
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledAction:2d36aa3b-cdf9-4565-
b290-81db519b227d:resource/dynamodb/table/my-table:scheduledActionName/my-second-
scheduled-action",
            "ScalableTargetAction": {
                "MinCapacity": 5,
                "MaxCapacity": 10
            },
            "ScheduledActionName": "my-second-scheduled-action",
            "ServiceNamespace": "dynamodb"
        }
    ]
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch zu Application Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScheduledActionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die für die Auto-Scaling-Gruppe geplanten Aktionen auf, die noch nicht ausgeführt wurden oder deren Endzeit noch nicht erreicht ist.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Ausgabe:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
```

```
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Einzelheiten zur API finden Sie unter DescribeScheduledActions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die für die Auto-Scaling-Gruppe geplanten Aktionen auf, die noch nicht ausgeführt wurden oder deren Endzeit noch nicht erreicht ist.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Ausgabe:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- Einzelheiten zur API finden Sie unter DescribeScheduledActions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutScalingPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutScalingPolicy` verwendet wird.

CLI

AWS CLI

Beispiel 1: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung mit einer vordefinierten Metrikspezifikation an

Das folgende Beispiel für `put-scaling-policy` wendet eine Skalierungsrichtlinie zur Zielverfolgung mit einer vordefinierten Metrikspezifikation auf einen Amazon-ECS-Service mit dem Namen Web-App im Standard-Cluster an. Die Richtlinie hält die durchschnittliche CPU-Auslastung des Service bei 75 Prozent, mit Aufskalierungs- und Abskalierungs-Ruhephasen von 60 Sekunden. Die Ausgabe enthält die ARNs und Namen der beiden CloudWatch Alarme, die in Ihrem Namen erstellt wurden.

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cpu75-target-tracking-scaling-policy --policy-
type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

In diesem Beispiel wird davon ausgegangen, dass Sie im aktuellen Verzeichnis über eine config.json-Datei mit dem folgenden Inhalt verfügen:

```
{
     "TargetValue": 75.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ECSServiceAverageCPUUtilization"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

Ausgabe:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/cpu75-target-tracking-scaling-policy",
    "Alarms": [
        {
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-1b437334-d19b-4a63-a812-6c67aaf2910d"
        }
    ]
}
```

Beispiel 2: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung mit einer benutzerdefinierten Metrikspezifikation an

Das folgende Beispiel für `put-scaling-policy` wendet eine Skalierungsrichtlinie für die Ziel-Nachverfolgung mit einer benutzerdefinierten Metrikspezifikation auf einen Amazon-ECS-Service mit dem Namen Web-App im Standard-Cluster an. Die Richtlinie hält die durchschnittliche Auslastung des Service bei 75 Prozent, mit Aufskalierungs- und Abskalierungs-Ruhephasen von 60 Sekunden. Die Ausgabe enthält die ARNs und Namen der beiden CloudWatch Alarme, die in Ihrem Namen erstellt wurden.

```
aws application-autoscaling put-scaling-policy --service-namespace ecs \
--scalable-dimension ecs:service:DesiredCount \
--resource-id service/default/web-app \
--policy-name cms75-target-tracking-scaling-policy
--policy-type TargetTrackingScaling \
--target-tracking-scaling-policy-configuration file://config.json
```

In diesem Beispiel wird davon ausgegangen, dass Sie im aktuellen Verzeichnis über eine config.json-Datei mit dem folgenden Inhalt verfügen:

```
{
    "TargetValue":75.0,
    "CustomizedMetricSpecification":{
        "MetricName":"MyUtilizationMetric",
        "Namespace":"MyNamespace",
        "Dimensions": [
```

```
            {
                "Name":"MyOptionalMetricDimensionName",
                "Value":"MyOptionalMetricDimensionValue"
            }
        ],
        "Statistic":"Average",
        "Unit":"Percent"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60
}
```

Ausgabe:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-west-2:012345678910:scalingPolicy:
 8784a896-b2ba-47a1-b08c-27301cc499a1:resource/ecs/service/default/web-
app:policyName/cms75-target-tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmHigh-9bc77b56-0571-4276-ba0f-d4178882e0a0"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4",
            "AlarmName": "TargetTracking-service/default/web-app-
AlarmLow-9b6ad934-6d37-438e-9e05-02836ddcbdc4"
        }
    ]
}
```

Beispiel 3: So wenden Sie eine Skalierungsrichtlinie für die Ziel-Nachverfolgung nur für die horizontale Skalierung nach oben an

Das folgende Beispiel für `put-scaling-policy` wendet eine Skalierungsrichtlinie für die Ziel-Nachverfolgung auf einen Amazon-ECS-Service mit dem Namen `web-app` im Standard-Cluster an. Die Richtlinie wird verwendet, um den ECS-Service aufzuskalieren, wenn die

`RequestCountPerTarget`-Metrik aus dem Application Load Balancer den Schwellenwert überschreitet. Die Ausgabe enthält den ARN und den Namen des CloudWatch Alarms, der in Ihrem Namen erstellt wurde.

```
aws application-autoscaling put-scaling-policy \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --policy-name alb-scale-out-target-tracking-scaling-policy \
    --policy-type TargetTrackingScaling \
    --target-tracking-scaling-policy-configuration file://config.json
```

Inhalt von `config.json`:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
         "PredefinedMetricType": "ALBRequestCountPerTarget",
         "ResourceLabel": "app/EC2Co-EcsEl-1TKLTMITMM0EO/f37c06a68c1748aa/
targetgroup/EC2Co-Defau-LDNM7Q3ZH1ZN/6d4ea56ca2d6a18d"
     },
     "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60,
    "DisableScaleIn": true
}
```

Ausgabe:

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:6d8972f3-efc8-437c-92d1-6270f29a66e7:resource/
ecs/service/default/web-app:policyName/alb-scale-out-target-tracking-scaling-
policy",
    "Alarms": [
        {
            "AlarmName": "TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca",
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-service/default/web-app-AlarmHigh-
d4f0770c-b46e-434a-a60f-3b36d653feca"
        }
    ]
```

```
}
```

Weitere Informationen finden Sie in den Skalierungsrichtlinien für die Ziel-Nachverfolgung für Application Auto Scaling im AWS Benutzerhandbuch für Application Auto Scaling.

- Einzelheiten zur API finden Sie PutScalingPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine Richtlinie für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Ausgabe:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Einzelheiten zur API finden Sie unter PutScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine Richtlinie für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
```

```
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Ausgabe:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- Einzelheiten zur API finden Sie unter PutScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutScheduledAction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutScheduledAction` verwendet wird.

CLI

AWS CLI

So fügen Sie eine geplante Aktion zu einer DynamoDB-Tabelle hinzu

In diesem Beispiel wird eine geplante Aktion zu einer DynamoDB-Tabelle hinzugefügt, die aufgerufen wird TestTable , um nach einem wiederkehrenden Zeitplan zu skalieren. Gemäß dem angegebenen Zeitplan (täglich um 12:15 Uhr UTC) wird Application Auto Scaling auf den von angegebenen Wert skaliert MinCapacity, wenn die aktuelle Kapazität unter dem für angegebenen Wert liegt. MinCapacity

Befehl:

```
aws application-autoscaling put-scheduled-action --service-
namespace dynamodb --scheduled-action-name my-recurring-action --
schedule "cron(15 12 * * ? *)" --resource-id table/TestTable --
scalable-dimension dynamodb:table:WriteCapacityUnits --scalable-target-
action MinCapacity=6
```

Weitere Informationen finden Sie unter „Geplante Skalierung“ im Benutzerhandbuch für Application Auto Scaling.

- Einzelheiten zur API finden Sie PutScheduledActionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine geplante Aktion für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Einzelheiten zur API finden Sie unter PutScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine geplante Aktion für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- Einzelheiten zur API finden Sie unter PutScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`RegisterScalableTarget`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterScalableTarget` verwendet wird.

CLI

AWS CLI

Beispiel 1: So registrieren Sie einen ECS-Service als skalierbares Ziel

Im folgenden Beispiel für `register-scalable-target` wird ein Amazon-ECS-Service bei Application Auto Scaling registriert. Außerdem wird dem skalierbaren Ziel ein Tag mit dem Schlüsselnamen `environment` und dem Wert `production` hinzugefügt.

```
aws application-autoscaling register-scalable-target \
    --service-namespace ecs \
    --scalable-dimension ecs:service:DesiredCount \
    --resource-id service/default/web-app \
    --min-capacity 1 --max-capacity 10 \
    --tags environment=production
```

Ausgabe:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Beispiele für andere AWS Dienste und benutzerdefinierte Ressourcen finden Sie in den Themen unter AWS Dienste, die Sie mit Application Auto Scaling verwenden können im Application Auto Scaling Scaling-Benutzerhandbuch.

Beispiel 2: So setzen Sie die Skalierungsaktivitäten für ein skalierbares Ziel aus

Im folgenden Beispiel für `register-scalable-target` werden die Skalierungsaktivitäten für ein vorhandenes skalierbares Ziel ausgesetzt.

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=true,DynamicScalingOutSuspended=true,ScheduledScalingSusp
```

Ausgabe:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Weitere Informationen finden Sie unter Unterbrechung und Wiederaufnahme der Skalierung von Application Auto Scaling im Benutzerhandbuch für Application Auto Scaling.

Beispiel 3: So nehmen Sie die Skalierungsaktivitäten für ein skalierbares Ziel wieder auf

Mit dem folgenden Beispiel für `register-scalable-target` werden alle Skalierungsaktivitäten für ein vorhandenes skalierbares Ziel fortgesetzt.

```
aws application-autoscaling register-scalable-target \
    --service-namespace dynamodb \
    --scalable-dimension dynamodb:table:ReadCapacityUnits \
    --resource-id table/my-table \
    --suspended-
state DynamicScalingInSuspended=false,DynamicScalingOutSuspended=false,ScheduledScalingSu
```

Ausgabe:

```
{
    "ScalableTargetARN": "arn:aws:application-autoscaling:us-
west-2:123456789012:scalable-target/1234abcd56ab78cd901ef1234567890ab123"
}
```

Weitere Informationen finden Sie unter Unterbrechung und Wiederaufnahme der Skalierung von Application Auto Scaling im Benutzerhandbuch für Application Auto Scaling.

- Einzelheiten zur API finden Sie RegisterScalableTargetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExcep
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResp
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequ
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResp
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecificati
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetReque
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicy
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
permissions.
               policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table,
which is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
```

```
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());
```

```
        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
            // If policies exist, consider updating an existing policy instead of
creating a new one.
            System.out.println("Policy already exists. Consider updating it
instead.");
            List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
                System.out.println("Policy name:" +pol.policyName());
            }
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();

            PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling
policy for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " +
e.awsErrorDetails().errorMessage());
            }
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie RegisterScalableTargetin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet registriert oder aktualisiert ein skalierbares Ziel. Ein skalierbares Ziel ist eine Ressource, die dank Application Auto Scaling auf- bzw. abskaliert werden kann.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Einzelheiten zur API finden Sie unter RegisterScalableTarget AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet registriert oder aktualisiert ein skalierbares Ziel. Ein skalierbares Ziel ist eine Ressource, die dank Application Auto Scaling auf- bzw. abskaliert werden kann.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- Einzelheiten zur API finden Sie unter RegisterScalableTarget AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Application Recovery Controller mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Route 53 Application Recovery Controller mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Application Recovery Controller – Weitere Informationen zum Application Recovery Controller.
- API-Referenz für Application Recovery Controller – Details zu allen verfügbaren Aktionen des Application Recovery Controllers.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Application Recovery Controller AWS SDKs
  - Aktionen für die Verwendung von Application Recovery Controller AWS SDKs
    - Verwenden Sie GetRoutingControlStatemit einem AWS SDK
    - Verwenden Sie UpdateRoutingControlStatemit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Application Recovery Controller AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Route 53 Application Recovery Controller mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für die Verwendung von Application Recovery Controller AWS SDKs
  - Verwenden Sie GetRoutingControlStatemit einem AWS SDK
  - Verwenden Sie UpdateRoutingControlStatemit einem AWS SDK

Aktionen für die Verwendung von Application Recovery Controller AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne Application Recovery Controller-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Route 53 Application Recovery Controller.

Beispiele

- Verwenden Sie GetRoutingControlStatemit einem AWS SDK
- Verwenden Sie UpdateRoutingControlStatemit einem AWS SDK

Verwenden Sie **`GetRoutingControlState`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetRoutingControlState` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
```

```
        }
        return null;
    }
```

- Einzelheiten zur API finden Sie GetRoutingControlStatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
```

```
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn
            )
            return response
        except Exception as error:
            print(error)
            raise error
```

- Einzelheiten zur API finden Sie GetRoutingControlStatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
```

```
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.

          " Create session for this region
          lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

          " Create client with the specific endpoint
          lo_client = create_recovery_client(
            iv_endpoint = lv_url
            iv_region   = lv_region
            io_session  = lo_session ).

          " Try to get the routing control state
          oo_result = lo_client->getroutingcontrolstate(
            iv_routingcontrolarn = iv_routing_control_arn ).

          " If successful, return the result
```

```
          RETURN.

        CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
          " This endpoint is temporarily unavailable, try the next one
          lo_exception = lo_endpoint_ex.
          CONTINUE.

        CATCH /aws1/cx_r5vaccessdeniedex
              /aws1/cx_r5vinternalserverex
              /aws1/cx_r5vresourcenotfoundex
              /aws1/cx_r5vthrottlingex
              /aws1/cx_r5vvalidationex
              /aws1/cx_rt_generic INTO lo_exception.
          " For other errors, re-raise immediately
          RAISE EXCEPTION lo_exception.
      ENDTRY.
    ENDLOOP.

    " If we get here, all endpoints failed - re-raise the last exception
    IF lo_exception IS BOUND.
      RAISE EXCEPTION lo_exception.
    ENDIF.
```

- Einzelheiten zur API finden Sie GetRoutingControlStatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateRoutingControlState`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `UpdateRoutingControlState` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

 .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- Einzelheiten zur API finden Sie UpdateRoutingControlStatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to update the
 state for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
    :return: The routing control update response.
```

```
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.update_routing_control_state(
                RoutingControlArn=routing_control_arn,
                RoutingControlState=routing_control_state,
            )
            return response
        except Exception as error:
            print(error)
```

- Einzelheiten zur API finden Sie UpdateRoutingControlStatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.
```

```
    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.

          " Create session for this region
          lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

          " Create client with the specific endpoint
          lo_client = create_recovery_client(
            iv_endpoint = lv_url
            iv_region   = lv_region
            io_session  = lo_session ).

          " Try to update the routing control state
          oo_result = lo_client->updateroutingcontrolstate(
            iv_routingcontrolarn     = iv_routing_control_arn
            iv_routingcontrolstate   = iv_routing_control_state
            it_safetyrulestooverride = it_safety_rules_override ).

          " If successful, return the result
          RETURN.

        CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
```

```
          " This endpoint is temporarily unavailable, try the next one
          lo_exception = lo_endpoint_ex.
          CONTINUE.

        CATCH /aws1/cx_r5vaccessdeniedex
              /aws1/cx_r5vconflictexception
              /aws1/cx_r5vinternalserverex
              /aws1/cx_r5vresourcenotfoundex
              /aws1/cx_r5vthrottlingex
              /aws1/cx_r5vvalidationex
              /aws1/cx_rt_generic INTO lo_exception.
          " For other errors, re-raise immediately
          RAISE EXCEPTION lo_exception.
      ENDTRY.
    ENDLOOP.

    " If we get here, all endpoints failed - re-raise the last exception
    IF lo_exception IS BOUND.
      RAISE EXCEPTION lo_exception.
    ENDIF.
```

- Einzelheiten zur API finden Sie UpdateRoutingControlStatein der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Athena mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Athena mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Athena-Benutzerhandbuch — Weitere Informationen zu Athena.
- Athena-API-Referenz — Details zu allen verfügbaren Athena-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Athena AWS SDKs
  - Erste Schritte mit Query Analytics

Szenarien für die Verwendung von Athena AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Athena mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Athena oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Query Analytics

Erste Schritte mit Query Analytics

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen S3-Bucket für Abfrageergebnisse
- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Ausführen einer -Abfrage
- Benannte Abfragen erstellen und verwenden
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
```

```
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
 [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
```

```
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi
```

```
# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
 ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \
```

```
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
```

```
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
```

```
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^
\\\\(]+[\\\\(][^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
```

```
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
```

```
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
```

```
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID"
 \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
```

```
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
```

```
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi
```

```
    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
 | cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)
```

```
    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
```

```
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults
  - ListDatabases
  - ListNamedQueries
  - ListTableMetadata
  - Mb
  - Rb

- Rm
- StartQueryExecution

Codebeispiele für Audit Manager mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Audit Manager mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zu Audit Manager – Weitere Informationen zu Audit Manager.
- API-Referenz von Audit Manager – Details zu allen verfügbaren Aktionen von Audit Manager.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Audit Manager AWS SDKs
  - Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework aus einem AWS Config Konformitätspaket mit einem AWS SDK
  - Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework, das Security Hub CSPM-Steuerelemente enthält, mithilfe eines AWS SDK
  - Erstellen Sie einen Audit Manager Manager-Bewertungsbericht, der Beweise für einen Tag enthält, anhand eines AWS SDK

Szenarien für die Verwendung von Audit Manager AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Audit Manager mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Audit Manager oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework aus einem AWS Config Konformitätspaket mit einem AWS SDK
- Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework, das Security Hub CSPM-Steuerelemente enthält, mithilfe eines AWS SDK
- Erstellen Sie einen Audit Manager Manager-Bewertungsbericht, der Beweise für einen Tag enthält, anhand eines AWS SDK

Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework aus einem AWS Config Konformitätspaket mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Holen Sie sich eine Liste der AWS Config Konformitätspakete.
- Erstellen eines benutzerdefinierten Audit-Manager-Steuerelements für jede verwaltete Regel in einem Konformitätspaket
- Erstellen eines benutzerdefinierten Audit-Manager-Frameworks, das die Steuerelemente enthält

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print(
                "Number of conformance packs fetched: ",
                len(conformance_packs.get("ConformancePackDetails")),
            )
            print("Fetched the following conformance packs: ")
            all_cpack_names = {
                cp["ConformancePackName"]
                for cp in conformance_packs.get("ConformancePackDetails")
            }
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                "Provide ConformancePackName that you want to create a custom "
                "framework for: "
            )
            if cpack_name not in all_cpack_names:
                print(f"{cpack_name} is not in the list of conformance packs!")
                print(
                    "Provide a conformance pack name from the available list of "
                    "conformance packs."
                )
                raise Exception("Invalid conformance pack")
            print("-" * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
```

```
        """
        Create custom controls for all managed AWS Config rules in a conformance
pack.

        :param cpack_name: The name of the conformance pack to create controls
for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack =
self.config_client.describe_conformance_pack_compliance(
                ConformancePackName=cpack_name
            )
            print(
                "Number of rules in the conformance pack: ",
                len(rules_in_pack.get("ConformancePackRuleComplianceList")),
            )
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                print(f"\t{rule.get('ConfigRuleName')}")
            print("-" * 88)
            print(
                "Creating a custom control for each rule and a custom framework "
                "consisting of these rules in Audit Manager."
            )
            am_controls = []
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                config_rule = self.config_client.describe_config_rules(
                    ConfigRuleNames=[rule.get("ConfigRuleName")]
                )
                source_id = (
                    config_rule.get("ConfigRules")[0]
                    .get("Source", {})
                    .get("SourceIdentifier")
                )
                custom_control = self.auditmanager_client.create_control(
                    name="Config-" + rule.get("ConfigRuleName"),
                    controlMappingSources=[
                        {
                            "sourceName": "ConfigRule",
                            "sourceSetUpOption": "System_Controls_Mapping",
                            "sourceType": "AWS_Config",
                            "sourceKeyword": {
                                "keywordInputType": "SELECT_FROM_LIST",
                                "keywordValue": source_id,
```

```
                            },
                        }
                    ],
                ).get("control", {})
                am_controls.append({"id": custom_control.get("id")})
            print("Successfully created a control for each config rule.")
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom controls.")
            raise
        else:
            return am_controls

    def create_custom_framework(self, cpack_name, am_control_ids):
        """
        Create a custom Audit Manager framework from a selected AWS Config
 conformance
        pack.

        :param cpack_name: The name of the conformance pack to create a framework
 from.
        :param am_control_ids: The IDs of the custom controls created from the
                               conformance pack.
        """
        try:
            print("Creating custom framework...")
            custom_framework =
 self.auditmanager_client.create_assessment_framework(
                name="Config-Conformance-pack-" + cpack_name,
                controlSets=[{"name": cpack_name, "controls": am_control_ids}],
            )
            print(
                f"Successfully created the custom framework: ",
                f"{custom_framework.get('framework').get('name')}: ",
                f"{custom_framework.get('framework').get('id')}",
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
```

```
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print("-" * 88)
    print(
        "You can use this sample to select a conformance pack from AWS Config and
 "
        "use AWS Audit Manager to create a custom control for all the managed "
        "rules under the conformance pack. A custom framework is also created "
        "with these controls."
    )
    print("-" * 88)
    conf_pack = ConformancePack(boto3.client("config"),
 boto3.client("auditmanager"))
    cpack_name = conf_pack.get_conformance_pack()
    am_controls = conf_pack.create_custom_controls(cpack_name)
    conf_pack.create_custom_framework(cpack_name, am_controls)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateAssessmentFramework
  - CreateControl

Erstellen Sie ein benutzerdefiniertes Audit Manager Manager-Framework, das Security Hub CSPM-Steuerelemente enthält, mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie eine Liste aller Standardsteuerelemente ab, deren Datenquelle Security Hub CSPM ist.
- Erstellen eines benutzerdefinierten Audit-Manager-Frameworks, das die Steuerelemente enthält

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
        """
        print("-" * 88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", maxResults=100
                )
            else:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", nextToken=next_token, maxResults=100
                )
```

```
            print("Total controls found:",
 len(control_list.get("controlMetadataList")))
            for control in control_list.get("controlMetadataList"):
                control_details = self.auditmanager_client.get_control(
                    controlId=control.get("id")
                ).get("control", {})
                if "AWS Security Hub" in control_details.get("controlSources"):
                    sechub_control_list.append({"id": control_details.get("id")})
            next_token = control_list.get("nextToken")
            if not next_token:
                break
            page += 1
        print("Number of Security Hub controls found: ",
 len(sechub_control_list))
        return sechub_control_list

    def create_custom_framework(self, am_controls):
        """
        Create a custom framework with a list of controls.

        :param am_controls: The list of controls to include in the framework.
        """
        try:
            print("Creating custom framework...")
            custom_framework =
 self.auditmanager_client.create_assessment_framework(
                name="All Security Hub Controls Framework",
                controlSets=[{"name": "Security-Hub", "controls": am_controls}],
            )
            print(
                f"Successfully created the custom framework: "
                f"{custom_framework.get('framework').get('name')}: "
                f"{custom_framework.get('framework').get('id')}"
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print("-" * 88)
```

```
    print(" This script creates a custom framework with all Security Hub
 controls.")
    print("-" * 88)
    sechub = SecurityHub(boto3.client("auditmanager"))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateAssessmentFramework
  - GetControl
  - ListControls

Erstellen Sie einen Audit Manager Manager-Bewertungsbericht, der Beweise für einen Tag enthält, anhand eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie einen Audit-Manager-Bewertungsbericht mit eintägigen Nachweisen erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
```

```
import boto3
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print("-" * 40)
        try:
            assessment_id = input("Provide assessment id [uuid]: ").lower()
            try:
                assessment_uuid = uuid.UUID(assessment_id)
            except ValueError:
                logger.error("Assessment Id is not a valid UUID: %s",
 assessment_id)
                raise
            evidence_folder = input("Provide evidence date [yyyy-mm-dd]: ")
            try:
                evidence_date = dateutil.parser.parse(evidence_folder).date()
            except ValueError:
                logger.error("Invalid date : %s", evidence_folder)
                raise
            try:
                self.auditmanager_client.get_assessment(
                    assessmentId=str(assessment_uuid)
                )
            except ClientError:
                logger.exception("Couldn't get assessment %s.", assessment_uuid)
                raise
        except (ValueError, ClientError):
            return None, None
        else:
            return assessment_uuid, evidence_date

    def clear_staging(self, assessment_uuid, evidence_date):
        """
        Find all the evidence in the report and clear it.
        """
        next_token = None
```

```
        page = 1
        interested_folder_id_list = []
        while True:
            print(f"Page [{page}]")
            if next_token is None:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid), maxResults=1000
                    )
                )
            else:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid),
                        nextToken=next_token,
                        maxResults=1000,
                    )
                )
            folders = folder_list.get("evidenceFolders")
            print(f"Got {len(folders)} folders.")
            for folder in folders:
                folder_id = folder.get("id")
                if folder.get("name") == str(evidence_date):
                    interested_folder_id_list.append(folder_id)
                if folder.get("assessmentReportSelectionCount") == folder.get(
                    "totalEvidence"
                ):
                    print(
                        f"Removing folder from report selection :
{folder.get('name')} "
                        f"{folder_id} {folder.get('controlId')}"
                    )

self.auditmanager_client.disassociate_assessment_report_evidence_folder(
                        assessmentId=str(assessment_uuid),
evidenceFolderId=folder_id
                    )
                elif folder.get("assessmentReportSelectionCount") > 0:
                    # Get all evidence in the folder and
                    # add selected evidence in the selected_evidence_list.
                    evidence_list = (
                        self.auditmanager_client.get_evidence_by_evidence_folder(
                            assessmentId=str(assessment_uuid),
                            controlSetId=folder_id,
```

```
                            evidenceFolderId=folder_id,
                            maxResults=1000,
                        )
                    )
                    selected_evidence_list = []
                    for evidence in evidence_list.get("evidence"):
                        if evidence.get("assessmentReportSelection") == "Yes":
                            selected_evidence_list.append(evidence.get("id"))
                    print(
                        f"Removing evidence report selection :
{folder.get('name')} "
                        f"{len(selected_evidence_list)}"
                    )

self.auditmanager_client.batch_disassociate_assessment_report_evidence(
                        assessmentId=str(assessment_uuid),
                        evidenceFolderId=folder_id,
                        evidenceIds=selected_evidence_list,
                    )
            next_token = folder_list.get("nextToken")
            if not next_token:
                break
            page += 1
        return interested_folder_id_list

    def add_folder_to_staging(self, assessment_uuid, folder_id_list):
        print(f"Adding folders to report : {folder_id_list}")
        for folder in folder_id_list:
            self.auditmanager_client.associate_assessment_report_evidence_folder(
                assessmentId=str(assessment_uuid), evidenceFolderId=folder
            )

    def get_report(self, assessment_uuid):
        report = self.auditmanager_client.create_assessment_report(
            name="ReportViaScript",
            description="testing",
            assessmentId=str(assessment_uuid),
        )
        if self._is_report_generated(report.get("assessmentReport").get("id")):
            report_url = self.auditmanager_client.get_assessment_report_url(
                assessmentReportId=report.get("assessmentReport").get("id"),
                assessmentId=str(assessment_uuid),
            )
            print(report_url.get("preSignedUrl"))
```

```
            urllib.request.urlretrieve(
                report_url.get("preSignedUrl").get("link"),
                report_url.get("preSignedUrl").get("hyperlinkName"),
            )
            print(
                f"Report saved as
 {report_url.get('preSignedUrl').get('hyperlinkName')}."
            )
        else:
            print("Report generation did not finish in 15 minutes.")
            print(
                "Failed to download report. Go to the console and manually
 download "
                "the report."
            )

    def _is_report_generated(self, assessment_report_id):
        max_wait_time = 0
        while max_wait_time < 900:
            print(f"Checking status of the report {assessment_report_id}")
            report_list =
 self.auditmanager_client.list_assessment_reports(maxResults=1)
            if (
                report_list.get("assessmentReports")[0].get("id")
                == assessment_report_id
                and report_list.get("assessmentReports")[0].get("status") ==
 "COMPLETE"
            ):
                return True
            print("Sleeping for 5 seconds...")
            time.sleep(5)
            max_wait_time += 5


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager samples demo!")
    print("-" * 88)
    print(
        "This script creates an assessment report for an assessment with all the
 "
        "evidence collected on the provided date."
    )
    print("-" * 88)
```

```
    report = AuditReport(boto3.client("auditmanager"))
    assessment_uuid, evidence_date = report.get_input()
    if assessment_uuid is not None and evidence_date is not None:
        folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
        report.add_folder_to_staging(assessment_uuid, folder_id_list)
        report.get_report(assessment_uuid)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - AssociateAssessmentReportEvidenceFolder
  - BatchDisassociateAssessmentReportEvidence
  - CreateAssessmentReport
  - DisassociateAssessmentReportEvidenceFolder
  - GetAssessment
  - GetAssessmentReportUrl
  - GetEvidenceByEvidenceFolder
  - GetEvidenceFoldersByAssessment
  - ListAssessmentReports

Codebeispiele für Aurora mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Aurora mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Aurora Benutzerhandbuch – Weitere Informationen zu Aurora.
- Aurora-API-Referenz – Details zu allen verfügbaren Aurora-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Aurora AWS SDKs
  - Hello Aurora
  - Erlernen Sie die Grundlagen von Aurora mit einem AWS SDK
  - Aktionen für Aurora mit AWS SDKs
    - Verwenden Sie CreateDBCluster mit einem AWS SDK
    - Verwenden Sie CreateDBClusterParameterGroupmit einem AWS SDK
    - Verwenden Sie CreateDBClusterSnapshotmit einem AWS SDK
    - Verwenden Sie CreateDBInstance mit einem AWS SDK
    - Verwenden Sie DeleteDBCluster mit einem AWS SDK
    - Verwenden Sie DeleteDBClusterParameterGroupmit einem AWS SDK
    - Verwenden Sie DeleteDBInstance mit einem AWS SDK
    - Verwenden Sie DescribeDBClusterParameterGroupsmit einem AWS SDK
    - Verwenden Sie DescribeDBClusterParametersmit einem AWS SDK
    - Verwenden Sie DescribeDBClusterSnapshotsmit einem AWS SDK
    - Verwenden Sie DescribeDBClusters mit einem AWS SDK
    - Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK
    - Verwenden Sie DescribeDBInstances mit einem AWS SDK
    - Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
    - Verwenden Sie ModifyDBClusterParameterGroupmit einem AWS SDK
- Szenarien für die Verwendung von Aurora AWS SDKs

- Leihbibliothek-REST-API erstellen
- Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Grundlegende Beispiele für die Verwendung von Aurora AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Aurora mit AWS SDKs verwendet werden.

Beispiele

- Hello Aurora
- Erlernen Sie die Grundlagen von Aurora mit einem AWS SDK
- Aktionen für Aurora mit AWS SDKs
  - Verwenden Sie CreateDBCluster mit einem AWS SDK
  - Verwenden Sie CreateDBClusterParameterGroupmit einem AWS SDK
  - Verwenden Sie CreateDBClusterSnapshotmit einem AWS SDK
  - Verwenden Sie CreateDBInstance mit einem AWS SDK
  - Verwenden Sie DeleteDBCluster mit einem AWS SDK
  - Verwenden Sie DeleteDBClusterParameterGroupmit einem AWS SDK
  - Verwenden Sie DeleteDBInstance mit einem AWS SDK
  - Verwenden Sie DescribeDBClusterParameterGroupsmit einem AWS SDK
  - Verwenden Sie DescribeDBClusterParametersmit einem AWS SDK
  - Verwenden Sie DescribeDBClusterSnapshotsmit einem AWS SDK
  - Verwenden Sie DescribeDBClusters mit einem AWS SDK
  - Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK
  - Verwenden Sie DescribeDBInstances mit einem AWS SDK
  - Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
  - Verwenden Sie ModifyDBClusterParameterGroupmit einem AWS SDK

Hello Aurora

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Aurora beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
```

```
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_aurora.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());
```

```
        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
```

```
# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für Ruby - API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
```

```
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Informationen zu APIs finden Sie unter DescribeDBClusters in der API-Referenz zum AWS -SDK für Rust.

Erlernen Sie die Grundlagen von Aurora mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine benutzerdefinierte Aurora-DB-Cluster-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the
 DescribeDBClusterParametersAsync method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
    12. Create a database instance in the cluster using the CreateDBInstanceAsync
 method.
    13. Wait for the DB instance to be ready using the DescribeDBInstances
 method.
```

```
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

       Console.WriteLine(sepBar);
       Console.WriteLine(
```

```
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;
```

```
            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
```

```
                $"\t{i}. Family: {parameterGroupFamily.Key}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family
 by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
 new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
```

```
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
```

```
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
 group.");

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
 "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
```

```
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
 choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
 dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
 dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
 family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
```

```
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
 param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
  cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
  parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
  {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
  clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
  clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
```

```
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
```

```
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
```

```
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await
auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
```

```
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
            }
```

```
            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
```

```
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
```

```
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
```

```
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
```

```
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
```

```
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
```

```
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
```

```
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
```

```
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });
```

```
        return response.DBInstance;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
```

```
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
           parameterGroupFound = false;
       }
       else {
           std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   }

   if (!parameterGroupFound) {
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 2. Get available parameter group families for the specified engine.
       if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                engineVersions, client)) {
           return false;
       }

       std::cout << "Getting available parameter group families for " <<
DB_ENGINE
                 << "."
                 << std::endl;
       std::vector<Aws::String> families;
       for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
           Aws::String family = version.GetDBParameterGroupFamily();
           if (std::find(families.begin(), families.end(), family) ==
               families.end()) {
               families.push_back(family);
               std::cout << "  " << families.size() << ": " << family <<
std::endl;
           }
       }

       int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                           static_cast<int>(families.size()));
       dbParameterGroupFamily = families[choice - 1];
   }
   if (!parameterGroupFound) {
       // 3.  Create a DB cluster parameter group.
       Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
```

```
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
```

```
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

```
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
        std::cout << "Let's create a DB cluster." << std::endl;
        const Aws::String administratorName = askQuestion(
```

```
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                                1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;
```

```
        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
```

```
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
""",
                             client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;
```

```
        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
```

```
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
```

```
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." <<
std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
```

```
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
```

```
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
```

```
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

```
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

```
//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
```

```
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);
```

```
            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
            }

            if (clusterDeleting) {
```

```
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups

- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)
```

```
// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
```

```
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
```

```
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
```

```
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
```

```
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
 if err != nil {
  panic(err)
 }
 if dbInstance == nil {
  log.Println("Let's create a database instance in your DB cluster.")
  log.Println("First, choose a DB instance type:")
  instOpts, err := scenario.dbClusters.GetOrderableInstances(
   ctx, *cluster.Engine, *cluster.EngineVersion)
  if err != nil {
   panic(err)
  }
  var instChoices []string
  for _, opt := range instOpts {
   instChoices = append(instChoices, *opt.DBInstanceClass)
  }
  slices.Sort(instChoices)
  instChoices = slices.Compact(instChoices)
  instIndex := scenario.questioner.AskChoice(
   "Which DB instance class do you want to use?\n", instChoices)
  log.Println("Creating a database instance. This typically takes several
 minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
```

```
    ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
  *cluster.Engine,
    instChoices[instIndex])
   if err != nil {
    panic(err)
   }
   for *dbInstance.DBInstanceStatus != "available" {
    scenario.helper.Pause(30)
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
  *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Println("Instance data:")
  log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
  log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
  log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
  log.Printf("\tEngine: %v\n", *dbInstance.Engine)
  log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
  log.Println(strings.Repeat("-", 88))
  return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}
```

```
// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {
```

```
 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}
```

```
// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
```

```
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

```
// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

```
// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
```

```
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
```

```
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
```

```
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots

- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
```

```
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
 group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
 family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
 identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
 +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials\"\n";
```

```
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
 dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
```

```
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
```

```
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }
```

```
    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
```

```
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
               DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
               List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
               for (DBClusterSnapshot snapshot : snapshotList) {
                   snapshotReadyStr = snapshot.status();
                   if (snapshotReadyStr.contains("available")) {
                       snapshotReady = true;
```

```
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
```

```
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();
```

```
            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
```

```
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
```

```
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
```

```
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
 User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()
```

```
    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
 dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected
 engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
```

```
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
```

```
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
```

```
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
```

```
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
```

```
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
```

```
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.

- CreateDBCluster
- CreateDBClusterParameterGroup
- CreateDBClusterSnapshot
- CreateDBInstance
- DeleteDBCluster
- DeleteDBClusterParameterGroup
- DeleteDBInstance
- DescribeDBClusterParameterGroups
- DescribeDBClusterParameters
- DescribeDBClusterSnapshots
- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
```

```
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
engine and
        create a DB cluster parameter group that is compatible with a selected
engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
{parameter_group_name}."
        )
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
```

```
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.
```

```
        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
```

```
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
```

```
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
 how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)
```

```
    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
        Before the DB cluster parameter group can be deleted, all associated DB
instances and
        DB clusters must first be deleted.

        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
```

```
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
```

```
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
```

```
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
```

```
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

```
    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
```

```
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
```

```
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

```
    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Bibliothek, die die Szenarienpezifischen Funktionen für das Aurora-Szenario enthält.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
```

```
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
```

```
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
```

```
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }
```

```
        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
 Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
```

```
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
```

```
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
```

```
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
```

```
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
```

```
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
```

```
   // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
   pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
       let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
       let snapshot = self
           .rds
           .snapshot_cluster(id, format!("{id}_{name}").as_str())
           .await;
       match snapshot {
           Ok(output) => match output.db_cluster_snapshot {
               Some(snapshot) => Ok(snapshot),
               None => Err(ScenarioError::with("Missing Snapshot")),
           },
           Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
       }
   }

   pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
       let mut clean_up_errors: Vec<ScenarioError> = vec![];

       // Delete the instance. rds.DeleteDbInstance.
       let delete_db_instance = self
           .rds
           .delete_db_instance(
               self.db_instance_identifier
                   .as_deref()
                   .expect("instance identifier"),
           )
           .await;
       if let Err(err) = delete_db_instance {
           let identifier = self
               .db_instance_identifier
               .as_deref()
               .unwrap_or("Missing Instance Identifier");
           let message = format!("failed to delete db instance {identifier}");
           clean_up_errors.push(ScenarioError::new(message, &err));
       } else {
           // Wait for the instance to delete
           let waiter = Waiter::default();
           while waiter.sleep().await.is_ok() {
               let describe_db_instances =
self.rds.describe_db_instances().await;
```

```
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;
```

```
        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
 rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Tests für die Bibliothek, die Automocks für den RDS-Client-Wrapper nutzen.

```
use crate::rds::MockRdsImpl;
```

```
use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
```

```
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
```

```
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
```

```
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
```

```
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}
```

```
#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
```

```
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()
```

```
 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
```

```
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
```

```
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
```

```
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}
```

```
#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Eine Binärdatei zum vollständigen Ausführen des Szenarien, wobei Inquirer verwendet wird, damit der Benutzer einige Entscheidungen treffen kann.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }
```

```
    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
```

```
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
```

```
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
 'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
```

```
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
```

```
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
```

```
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
```

```
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Ein Wrapper für den Service Amazon RDS, der Automocking für Tests ermöglicht.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
```

```
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
```

```
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
```

```
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
```

```
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
```

```
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

Die Cargo.toml mit den Abhängigkeiten, die in diesem Szenario verwendet werden.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
```

```
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Aktionen für Aurora mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Aurora-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden können.

Diese Auszüge rufen die Aurora-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Aurora AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon-Aurora-API-Referenz.

Beispiele

- Verwenden Sie CreateDBCluster mit einem AWS SDK
- Verwenden Sie CreateDBClusterParameterGroupmit einem AWS SDK
- Verwenden Sie CreateDBClusterSnapshotmit einem AWS SDK
- Verwenden Sie CreateDBInstance mit einem AWS SDK
- Verwenden Sie DeleteDBCluster mit einem AWS SDK
- Verwenden Sie DeleteDBClusterParameterGroupmit einem AWS SDK
- Verwenden Sie DeleteDBInstance mit einem AWS SDK
- Verwenden Sie DescribeDBClusterParameterGroupsmit einem AWS SDK
- Verwenden Sie DescribeDBClusterParametersmit einem AWS SDK
- Verwenden Sie DescribeDBClusterSnapshotsmit einem AWS SDK
- Verwenden Sie DescribeDBClusters mit einem AWS SDK
- Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK
- Verwenden Sie DescribeDBInstances mit einem AWS SDK
- Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
- Verwenden Sie ModifyDBClusterParameterGroupmit einem AWS SDK

Verwenden Sie **`CreateDBCluster mit einem`** AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
```

```
        return response.DBCluster;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
```

```
                        << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK für C++ -API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
```

```
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createDBCluster(RdsClient rdsClient, String
 dbParameterGroupFamily, String dbName,
```

```
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
 CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
 rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
```

```
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
```

```
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
```

```
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
```

```
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }
```

```
        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
```

```
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
```

```
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
```

```
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
```

```
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });
```

```
    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter CreateDBCluster in der API-Referenz zum AWS - SDK für Rust.

Verwenden Sie **`CreateDBClusterParameterGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBClusterParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
```

```
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
```

```
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
```

```
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
```

```
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
```

```
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_output) = lo_rds->createdbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      iv_dbparametergroupfamily = iv_param_group_family
      iv_description = iv_description
    ).
    oo_result = lo_output->get_dbclusterparametergroup( ).
  CATCH /aws1/cx_rdsdbparmgralrexfault.
    " Re-raise exception - parameter group already exists
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbparmgralrexfault.
  CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
    " Re-raise exception - quota exceeded
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrquotaexcd00.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDBClusterParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateDBClusterSnapshot`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBClusterSnapshot` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
```

```
}
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
```

```
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
```

```
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }
```

```
            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
```

```
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
```

```
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
```

```
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Einzelheiten zur API finden Sie CreateDBClusterSnapshotin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`CreateDBInstance mit einem`** AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
```

```
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für .NET - API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
```

```
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
 "",
                             client);
            return false;
        }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .engine("aurora-mysql")
                   .dbInstanceClass(instanceClass)
                   .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
       return "";
   }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
 that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
 created.
        :param db_engine: The database engine of a database to create in the DB
 instance.
```

```
                            This must be compatible with the configured parameter
group
                            of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
```

```
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
```

```
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
```

```
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
```

```
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter CreateDBInstance in der API-Referenz zum AWS - SDK für Rust.

Verwenden Sie **`DeleteDBCluster mit`** einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK für C++ -API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
```

```
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
```

```
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DeleteDBCluster in der API-Referenz zum AWS -SDK für Rust.

Verwenden Sie **`DeleteDBClusterParameterGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBClusterParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
 groupName)
    {
```

```
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
```

```
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
```

```
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
```

```
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
```

```
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_rds->deletedbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
    ).
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    " Re-raise exception - invalid state
    RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDBClusterParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteDBInstance mit`** einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
```

```
  return err
 } else {
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
```

```
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
```

```
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
```

```
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
```

```
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
```

```
    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
```

```
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DeleteDBInstance in der API-Referenz zum AWS - SDK für Rust.

Verwenden Sie **`DescribeDBClusterParameterGroups`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBClusterParameterGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        else:
            return parameter_group
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_output) = lo_rds->describedbclusterparamgroups(
          iv_dbclusterparamgroupname = iv_param_group_name
        ).
        DATA(lt_param_groups) = lo_output->get_dbclusterparametergroups( ).
        IF lines( lt_param_groups ) > 0.
          oo_result = lt_param_groups[ 1 ].
        ENDIF.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDBClusterParameterGroupsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeDBClusterParameters`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBClusterParameters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
```

```
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
```

```
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
```

```
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA lv_marker TYPE /aws1/rdsstring VALUE ''.
    DATA lt_all_parameters TYPE /aws1/cl_rdsparameter=>tt_parameterslist.

    DO.
      DATA(lo_output) = lo_rds->describedbclusterparameters(
        iv_dbclusterparamgroupname = iv_param_group_name
        iv_source = iv_source
        iv_marker = lv_marker
      ).

      LOOP AT lo_output->get_parameters( ) INTO DATA(lo_param).
        IF iv_name_prefix IS INITIAL OR
           lo_param->get_parametername( ) CP |{ iv_name_prefix }*|.
          APPEND lo_param TO lt_all_parameters.
        ENDIF.
      ENDLOOP.

      lv_marker = lo_output->get_marker( ).
      IF lv_marker IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_parameters = lt_all_parameters.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDBClusterParametersin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeDBClusterSnapshots`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBClusterSnapshots` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Return a list of DB snapshots for a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>List of DB snapshots.</returns>
   public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
   {
       var results = new List<DBClusterSnapshot>();

       DescribeDBClusterSnapshotsResponse response;
       DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
       {
           DBClusterIdentifier = dbClusterIdentifier
       };
       // Get the full list if there are multiple pages.
       do
       {
```

```
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
```

```
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .build();

           while (!snapshotReady) {
```

```
                DescribeDbClusterSnapshotsResponse response =
 rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
 response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
```

```
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Einzelheiten zur API finden Sie DescribeDBClusterSnapshotsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie **`DescribeDBClusters`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBClusters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
```

```
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
```

```
    err = nil
   } else {
    log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
   }
   return nil, err
  } else {
   return &output.DBClusters[0], err
  }
 }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeDbClusterParameters(RdsClient rdsClient, String
 dbCLusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
 DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }
```

```
            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
```

```
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
```

```
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DescribeDBClusters in der API-Referenz zum AWS - SDK für Rust.

Verwenden Sie **`DescribeDBEngineVersions`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBEngineVersions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                       engine versions to those that are
 compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
```

```
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");
```

```
        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeDBInstances`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
```

```
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
 dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
```

```
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Waits until the database instance is available.
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
```

```
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
```

```
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
```

```
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
```

```
                    ErrorKind::Other,
                    "describe db clusters error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
Describe Cluster
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DescribeDBInstances in der API-Referenz zum AWS -SDK für Rust.

Verwenden Sie **`DescribeOrderableDBInstanceOptions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeOrderableDBInstanceOptions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
```

```
        var paginateInstanceOptions =
 _amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
```

```
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für C ++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet die DB-Engine-Versionen auf, die eine bestimmte DB-Instance-Klasse in einer AWS-Region unterstützen.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Beispiel 2: Dieses Beispiel listet die DB-Instance-Klassen auf, die von einer bestimmten DB-Engine-Version in einer AWS-Region unterstützt werden.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Einzelheiten zur API finden Sie unter DescribeOrderableDBInstanceOptions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet die DB-Engine-Versionen auf, die eine bestimmte DB-Instance-Klasse in einer AWS-Region unterstützen.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Beispiel 2: Dieses Beispiel listet die DB-Instance-Klassen auf, die von einer bestimmten DB-Engine-Version in einer AWS-Region unterstützt werden.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Einzelheiten zur API finden Sie unter DescribeOrderableDBInstanceOptions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
           .rds
           .describe_orderable_db_instance_options(
               DB_ENGINE,
               self.engine_version
                   .as_ref()
                   .expect("engine version for db instance options")
                   .as_str(),
           )
           .await;

       describe_orderable_db_instance_options_items
           .map(|options| {
               options
                   .iter()
                   .filter(|o| o.storage_type() == Some("aurora"))
                   .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                   .collect::<Vec<String>>()
           })
           .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
   }

   pub async fn describe_orderable_db_instance_options(
       &self,
       engine: &str,
       engine_version: &str,
   ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
```

```
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
```

```
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;
```

```
    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
 TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ModifyDBClusterParameterGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ModifyDBClusterParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
```

```
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();
```

```
            List<DBClusterParameterGroup> groups =
 rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
 group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
 group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
```

```
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
```

```
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
```

```
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rds->modifydbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      it_parameters = it_update_parameters
    ).
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    " Re-raise exception - parameter group not found
    RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    " Re-raise exception - invalid state
    RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
ENDTRY.
```

- Einzelheiten zur API finden Sie ModifyDBClusterParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Aurora AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Aurora mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Aurora oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Leihbibliothek-REST-API erstellen

- Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

Python

SDK für Python (Boto3)

Zeigt, wie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) API und AWS Chalice verwendet wird, um eine REST-API zu erstellen, die von einer Amazon Aurora Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldedaten verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für .NET In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend entwickelt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript (v3) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank verfolgt und Berichte mithilfe von

Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PHP-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in einer Amazon Aurora Aurora-Serverless-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.

- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldedaten enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Codebeispiele für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon EC2 Auto Scaling mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Auto Scaling-Benutzerhandbuch – Weitere Informationen zu Auto Scaling.
- API-Referenz zum Auto Scaling – Details zu allen verfügbaren Auto-Scaling-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Auto Scaling mit AWS SDKs
  - Hallo, Auto Scaling
  - Erlernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK
  - Aktionen für Auto Scaling mit AWS SDKs
    - Verwendung von AttachInstances mit einer CLI
    - Verwenden Sie mit einem AttachLoadBalancerTargetGroups AWS SDK oder CLI
    - Verwendung von AttachLoadBalancers mit einer CLI
    - Verwendung von CompleteLifecycleAction mit einer CLI
    - Verwenden Sie mit einem CreateAutoScalingGroup AWS SDK oder CLI
    - Verwendung von CreateLaunchConfiguration mit einer CLI
    - Verwendung von CreateOrUpdateTags mit einer CLI
    - Verwenden Sie mit einem DeleteAutoScalingGroup AWS SDK oder CLI
    - Verwendung von DeleteLaunchConfiguration mit einer CLI
    - Verwendung von DeleteLifecycleHook mit einer CLI
    - Verwendung von DeleteNotificationConfiguration mit einer CLI
    - Verwendung von DeletePolicy mit einer CLI
    - Verwendung von DeleteScheduledAction mit einer CLI
    - Verwendung von DeleteTags mit einer CLI
    - Verwendung von DescribeAccountLimits mit einer CLI
    - Verwendung von DescribeAdjustmentTypes mit einer CLI
    - Verwenden Sie mit einem DescribeAutoScalingGroups AWS SDK oder CLI
    - Verwenden Sie DescribeAutoScalingInstancesmit einem AWS SDK oder CLI
    - Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI
    - Verwendung von DescribeLaunchConfigurations mit einer CLI
    - Verwendung von DescribeLifecycleHookTypes mit einer CLI
    - Verwendung von DescribeLifecycleHooks mit einer CLI
    - Verwendung von DescribeLoadBalancers mit einer CLI
    - Verwendung von DescribeMetricCollectionTypes mit einer CLI

    - Verwendung von DescribeNotificationConfigurations mit einer CLI

- Verwendung von DescribePolicies mit einer CLI
- Verwenden Sie mit einem DescribeScalingActivities AWS SDK oder CLI
- Verwendung von DescribeScalingProcessTypes mit einer CLI
- Verwendung von DescribeScheduledActions mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTerminationPolicyTypes mit einer CLI
- Verwendung von DetachInstances mit einer CLI
- Verwendung von DetachLoadBalancers mit einer CLI
- Verwenden Sie mit einem DisableMetricsCollection AWS SDK oder CLI
- Verwenden Sie EnableMetricsCollectionmit einem AWS SDK oder CLI
- Verwendung von EnterStandby mit einer CLI
- Verwendung von ExecutePolicy mit einer CLI
- Verwendung von ExitStandby mit einer CLI
- Verwendung von PutLifecycleHook mit einer CLI
- Verwendung von PutNotificationConfiguration mit einer CLI
- Verwendung von PutScalingPolicy mit einer CLI
- Verwendung von PutScheduledUpdateGroupAction mit einer CLI
- Verwendung von RecordLifecycleActionHeartbeat mit einer CLI
- Verwendung von ResumeProcesses mit einer CLI
- Verwenden Sie mit einem SetDesiredCapacity AWS SDK oder CLI
- Verwendung von SetInstanceHealth mit einer CLI
- Verwendung von SetInstanceProtection mit einer CLI
- Verwendung von SuspendProcesses mit einer CLI
- Verwenden Sie mit einem TerminateInstanceInAutoScalingGroup AWS SDK oder CLI
- Verwenden Sie UpdateAutoScalingGroupmit einem AWS SDK oder CLI

- Szenarien für Auto Scaling mit AWS SDKs
  - Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK

Grundlegende Beispiele für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon EC2 Auto Scaling mit AWS SDKs verwendet werden.

Beispiele

- Hallo, Auto Scaling
- Erlernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK
- Aktionen für Auto Scaling mit AWS SDKs
  - Verwendung von AttachInstances mit einer CLI
  - Verwenden Sie mit einem AttachLoadBalancerTargetGroups AWS SDK oder CLI
  - Verwendung von AttachLoadBalancers mit einer CLI
  - Verwendung von CompleteLifecycleAction mit einer CLI
  - Verwenden Sie mit einem CreateAutoScalingGroup AWS SDK oder CLI
  - Verwendung von CreateLaunchConfiguration mit einer CLI
  - Verwendung von CreateOrUpdateTags mit einer CLI
  - Verwenden Sie mit einem DeleteAutoScalingGroup AWS SDK oder CLI
  - Verwendung von DeleteLaunchConfiguration mit einer CLI
  - Verwendung von DeleteLifecycleHook mit einer CLI
  - Verwendung von DeleteNotificationConfiguration mit einer CLI
  - Verwendung von DeletePolicy mit einer CLI
  - Verwendung von DeleteScheduledAction mit einer CLI
  - Verwendung von DeleteTags mit einer CLI
  - Verwendung von DescribeAccountLimits mit einer CLI
  - Verwendung von DescribeAdjustmentTypes mit einer CLI
  - Verwenden Sie mit einem DescribeAutoScalingGroups AWS SDK oder CLI
  - Verwenden Sie DescribeAutoScalingInstancesmit einem AWS SDK oder CLI
  - Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI
  - Verwendung von DescribeLaunchConfigurations mit einer CLI
  - Verwendung von DescribeLifecycleHookTypes mit einer CLI
  - Verwendung von DescribeLifecycleHooks mit einer CLI
  - Verwendung von DescribeLoadBalancers mit einer CLI

- Verwendung von DescribeMetricCollectionTypes mit einer CLI
- Verwendung von DescribeNotificationConfigurations mit einer CLI
- Verwendung von DescribePolicies mit einer CLI
- Verwenden Sie mit einem DescribeScalingActivities AWS SDK oder CLI
- Verwendung von DescribeScalingProcessTypes mit einer CLI
- Verwendung von DescribeScheduledActions mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTerminationPolicyTypes mit einer CLI
- Verwendung von DetachInstances mit einer CLI
- Verwendung von DetachLoadBalancers mit einer CLI
- Verwenden Sie mit einem DisableMetricsCollection AWS SDK oder CLI
- Verwenden Sie EnableMetricsCollectionmit einem AWS SDK oder CLI
- Verwendung von EnterStandby mit einer CLI
- Verwendung von ExecutePolicy mit einer CLI
- Verwendung von ExitStandby mit einer CLI
- Verwendung von PutLifecycleHook mit einer CLI
- Verwendung von PutNotificationConfiguration mit einer CLI
- Verwendung von PutScalingPolicy mit einer CLI
- Verwendung von PutScheduledUpdateGroupAction mit einer CLI
- Verwendung von RecordLifecycleActionHeartbeat mit einer CLI
- Verwendung von ResumeProcesses mit einer CLI
- Verwenden Sie mit einem SetDesiredCapacity AWS SDK oder CLI
- Verwendung von SetInstanceHealth mit einer CLI
- Verwendung von SetInstanceProtection mit einer CLI
- Verwendung von SuspendProcesses mit einer CLI
- Verwenden Sie mit einem TerminateInstanceInAutoScalingGroup AWS SDK oder CLI
- Verwenden Sie UpdateAutoScalingGroupmit einem AWS SDK oder CLI

Hallo, Auto Scaling

Grundlagen 675

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Auto Scaling beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        if (response.AutoScalingGroups == null ||
 response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
            return;
        }
        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {
```

```
 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_autoscaling.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
```

```
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
```

```
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsunter AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function helloService()
    {
```

```
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
```

```
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
```

```
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der API-Referenz zum AWS SDK für Rust.

Erlernen Sie die Grundlagen von Auto Scaling mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen einer Amazon-EC2-Auto-Scaling-Gruppe mit einer Startvorlage und Availability Zones sowie Abrufen von Informationen zu laufenden Instances
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.

- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann Ressourcen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when
deleting the
        // launch template at the end of the application.
```

```
        var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);

        // Confirm that the template was created by asking for a description of
it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones[0].ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();
```

```
        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);

        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups.Any())
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
```

```
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities.Any())
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        if (metrics.Any())
        {
            Console.WriteLine($"Metrics collected for {groupName}:");
            metrics.ForEach(metric =>
            {
                Console.Write($"Metric name: {metric.MetricName}\t");
                Console.WriteLine($"Namespace: {metric.Namespace}");
            });
        }

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        if (dataPoints.Any())
```

```
        {
            Console.WriteLine("Details for the metrics collected:");
            dataPoints.ForEach(detail => { Console.WriteLine(detail); });
        }

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
```

```
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
 operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
```

```
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
 Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the
 group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
 for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
 have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
```

```
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
```

```
    }
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Auto Scaling, Amazon EC2 und CloudWatch Aktionen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
```

```
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
```

```
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
 response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
 response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
```

```
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
```

```
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groups = new List<AutoScalingGroup>();
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        if (response.AutoScalingGroups != null)
        {
            groups = response.AutoScalingGroups;
        }

        return groups;
    }


    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };
```

```
        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };
```

```
        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
```

```
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
```

```
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }
```

```
    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }
```

```
    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
 launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}
```

```
namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
    public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
    {
        var metrics = new List<Metric>();
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
```

```
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);
        if (response.Metrics != null)
        {
            metrics = response.Metrics;
        }
        return metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var dataPoints = new List<Datapoint>();
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
        if (response.Datapoints != null)
```

```
        {
            dataPoints = response.Datapoints;
        }

        return dataPoints;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
```

```
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." <<
 std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
```

```
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following
Availability Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
```

```
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
              "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
```

```
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
              << "to replace the terminated EC2 instance."
              << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
```

```
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                            1,

static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
```

```
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());
```

```
        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }
```

```
    return cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                                const
 Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
 instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
```

```
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." <<
std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
&details: instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
 Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
 {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
                Aws::String("Delete the Auto Scaling group '") + groupName +
                "'  (y/n)?"))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                    autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                        << "The minimum size and desired capacity of the Auto
 Scaling group "
```

```
                        << "was set to zero before terminating the instances."
                        << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                          << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }

        Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
        if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                               autoScalingClient)) {
            if (!autoScalingGroups.empty()) {
                Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                        autoScalingGroups[0].GetInstances());
                for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                    request.SetInstanceId(instanceID);
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =

autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                     request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
            }

            std::cout
```

```
                    << "Waiting for the EC2 instances to terminate before
deleting the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was
deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                         const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
```

```
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_WEST_2)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
 scenario.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("First, we will create a launch template using a
CloudFormation script");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " +
e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
        }
    }
```

```
   public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
       // Step 1: Create new launch template version
       String newAmiId = "ami-0025f0db847eb6254";
       RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
               .imageId(newAmiId)
               .build();

       CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
               .launchTemplateName(launchTemplateName)
               .versionDescription("Updated with valid AMI")
               .sourceVersion("1")
               .launchTemplateData(launchTemplateData)
               .build();

       CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
       int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

       // Step 2: Modify default version
       ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
               .launchTemplateName(launchTemplateName)
               .defaultVersion(String.valueOf(newVersionNumber))
               .build();

       ec2.modifyLaunchTemplate(modifyRequest);
       System.out.println("Updated launch template to version " +
newVersionNumber + " with AMI " + newAmiId);
   }


   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                   .autoScalingGroupName(groupName)
                   .maxRecords(10)
                   .build();

           DescribeScalingActivitiesResponse response = autoScalingClient
```

```
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
                                              Ec2Client ec2Client,
                                              String groupName,
                                              String launchTemplateName,
                                              String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest =
DescribeSubnetsRequest.builder()
```

```
                    .filters(Filter.builder().name("vpc-
id").values(vpcId).build())
                    .build();

            DescribeSubnetsResponse subnetResponse =
 ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
 first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
 CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC
 ID
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
                    waiter.waitUntilGroupExists(groupsRequest);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (Ec2Exception | AutoScalingException e) {
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();
```

```
            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
       try {
           DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
           System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
           System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
                   .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .build();

           WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                   .waitUntilGroupInService(groupsRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
in the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of
the groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
```

```
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
```

```
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
```

```
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
```

```
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}
```

```
suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateAutoScalingGroup

- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;
```

```
    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an
 Auto Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you
 created in step 0.
```

```
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
```

```
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the
 group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups:
 {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
```

```
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that
 have occurred for the group so far.
         */
```

```
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']}
 recorded:\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
```

```
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }
```

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
 None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
 instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
 new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
 specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
```

```
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
```

```
        print(f"Metrics enabled for {group_name}.")
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
```

```
    print(f"Here's the state of {group_name}:")
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities),
group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)

    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
```

```
                metric_data = svc_helper.get_metric_statistics(
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")
```

```
    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Amazon EC2 und CloudWatch Aktionen.

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
 instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
```

```
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID
%s.",
                template_name,
```

```
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:

self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
```

```
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
```

```
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
```

```
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
 list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
 else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
 instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
```

```
            f"Here are the details of the instance{'s' if len(instances) > 1 else
 ''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
```

```
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
```

```
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
 0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
 AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
```

```
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
```

```
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
 the group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
 instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
```

```
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
 instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
```

```
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
 a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
```

```
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
```

```
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
```

```
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
```

```
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
 scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
        //   * Recommended: InstanceType='t1.micro',
 ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
```

```
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
```

```
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };
```

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
```

```
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting:
{}",
                                    group.status().unwrap_or("unknown error")
                                )));
```

```
                                return Err(errors);
                            }
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing
after three rounds",
            )])
        }
    }

    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });
```

```
        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

```
        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
```

```
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
 {
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
```

```
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
```

```
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
```

```
            .min_size(0)
            .desired_capacity(0)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
```

```
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Auto Scaling Scaling-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Auto-Scaling-API auf und sind Codeausschnitte aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Auto Scaling mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon EC2 Auto Scaling.

Beispiele

- Verwendung von AttachInstances mit einer CLI
- Verwenden Sie mit einem AttachLoadBalancerTargetGroups AWS SDK oder CLI
- Verwendung von AttachLoadBalancers mit einer CLI
- Verwendung von CompleteLifecycleAction mit einer CLI
- Verwenden Sie mit einem CreateAutoScalingGroup AWS SDK oder CLI
- Verwendung von CreateLaunchConfiguration mit einer CLI
- Verwendung von CreateOrUpdateTags mit einer CLI
- Verwenden Sie mit einem DeleteAutoScalingGroup AWS SDK oder CLI
- Verwendung von DeleteLaunchConfiguration mit einer CLI
- Verwendung von DeleteLifecycleHook mit einer CLI
- Verwendung von DeleteNotificationConfiguration mit einer CLI
- Verwendung von DeletePolicy mit einer CLI
- Verwendung von DeleteScheduledAction mit einer CLI
- Verwendung von DeleteTags mit einer CLI
- Verwendung von DescribeAccountLimits mit einer CLI
- Verwendung von DescribeAdjustmentTypes mit einer CLI
- Verwenden Sie mit einem DescribeAutoScalingGroups AWS SDK oder CLI
- Verwenden Sie DescribeAutoScalingInstancesmit einem AWS SDK oder CLI
- Verwendung von DescribeAutoScalingNotificationTypes mit einer CLI
- Verwendung von DescribeLaunchConfigurations mit einer CLI
- Verwendung von DescribeLifecycleHookTypes mit einer CLI
- Verwendung von DescribeLifecycleHooks mit einer CLI
- Verwendung von DescribeLoadBalancers mit einer CLI
- Verwendung von DescribeMetricCollectionTypes mit einer CLI
- Verwendung von DescribeNotificationConfigurations mit einer CLI
- Verwendung von DescribePolicies mit einer CLI
- Verwenden Sie mit einem DescribeScalingActivities AWS SDK oder CLI

- Verwendung von DescribeScalingProcessTypes mit einer CLI
- Verwendung von DescribeScheduledActions mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTerminationPolicyTypes mit einer CLI
- Verwendung von DetachInstances mit einer CLI
- Verwendung von DetachLoadBalancers mit einer CLI
- Verwenden Sie mit einem DisableMetricsCollection AWS SDK oder CLI
- Verwenden Sie EnableMetricsCollectionmit einem AWS SDK oder CLI
- Verwendung von EnterStandby mit einer CLI
- Verwendung von ExecutePolicy mit einer CLI
- Verwendung von ExitStandby mit einer CLI
- Verwendung von PutLifecycleHook mit einer CLI
- Verwendung von PutNotificationConfiguration mit einer CLI
- Verwendung von PutScalingPolicy mit einer CLI
- Verwendung von PutScheduledUpdateGroupAction mit einer CLI
- Verwendung von RecordLifecycleActionHeartbeat mit einer CLI
- Verwendung von ResumeProcesses mit einer CLI
- Verwenden Sie mit einem SetDesiredCapacity AWS SDK oder CLI
- Verwendung von SetInstanceHealth mit einer CLI
- Verwendung von SetInstanceProtection mit einer CLI
- Verwendung von SuspendProcesses mit einer CLI
- Verwenden Sie mit einem TerminateInstanceInAutoScalingGroup AWS SDK oder CLI
- Verwenden Sie UpdateAutoScalingGroupmit einem AWS SDK oder CLI

Verwendung von **`AttachInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachInstances` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Instance hinzu

In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe angehängt.

```
aws autoscaling attach-instances \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie AttachInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe hinzugefügt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto-Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe hinzugefügt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto-Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`AttachLoadBalancerTargetGroups`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachLoadBalancerTargetGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- Einzelheiten zur API finden Sie AttachLoadBalancerTargetGroupsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Zielgruppe hinzu

In diesem Beispiel wird die angegebene Zielgruppe zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Elastic Load Balancing und Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie AttachLoadBalancerTargetGroupsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
```

- Einzelheiten zur API finden Sie AttachLoadBalancerTargetGroupsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie AttachLoadBalancerTargetGroupsin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`AttachLoadBalancers`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachLoadBalancers` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe einen Classic Load Balancer hinzu

In diesem Beispiel wird der angegebene Classic Load Balancer der spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling attach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Elastic Load Balancing und Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie AttachLoadBalancersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer an die spezifizierte Auto-Scaling-Gruppe hinzugefügt.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer an die spezifizierte Auto-Scaling-Gruppe hinzugefügt.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CompleteLifecycleAction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CompleteLifecycleAction` verwendet wird.

CLI

AWS CLI

So schließen Sie eine Lebenszyklusaktion ab

In diesem Beispiel wird Amazon EC2 Auto Scaling darüber informiert, dass die angegebene Lebenszyklusaktion abgeschlossen ist, sodass das Starten oder Beenden der Instance abgeschlossen werden kann.

```
aws autoscaling complete-lifecycle-action \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-result CONTINUE \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie CompleteLifecycleActionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter CompleteLifecycleAction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter CompleteLifecycleAction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateAutoScalingGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
        {
            availabilityZone,
        };

    var request = new CreateAutoScalingGroupRequest
```

```
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```

```
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Auto-Scaling-Gruppe

Im folgenden Beispiel für `create-auto-scaling-group` wird eine Auto-Scaling-Gruppe in Subnetzen in mehreren Availability Zones innerhalb einer Region erstellt. Die Instances werden mit der Standardversion der angegebenen Startvorlage gestartet. Bitte beachten Sie, dass für die meisten anderen Einstellungen, wie z. B. die Beendigungsrichtlinien und die Konfiguration der Zustandsprüfung, die Standardeinstellungen verwendet werden.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Auto-Scaling-Gruppen im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 2: So fügen Sie einen Application Load Balancer, einen Network Load Balancer oder einen Gateway Load Balancer hinzu

Dieses Beispiel gibt den ARN einer Zielgruppe für einen Load Balancer an, der den erwarteten Datenverkehr unterstützt. Der Typ der Zustandsprüfung gibt `ELB` an, sodass die Auto-Scaling-Gruppe eine Instance ersetzt, wenn Elastic Load Balancing diese als fehlerhaft meldet. Der Befehl definiert außerdem eine Wartefrist für die Zustandsprüfung von `600` Sekunden. Die Wartefrist trägt dazu bei, ein vorzeitiges Beenden neu gestarteter Instances zu verhindern.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/943f017f100becff \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --min-size 1 \
```

```
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Elastic Load Balancing und Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So geben Sie eine Platzierungsgruppe an und verwenden die neueste Version der Startvorlage

Dieses Beispiel startet Instances in einer Platzierungsgruppe innerhalb einer einzelnen Availability Zone. Dies kann für Gruppen mit niedriger Latenz und HPC-Workloads nützlich sein. Außerdem werden in diesem Beispiel die minimale Größe, die maximale Größe und die gewünschte Kapazität der Gruppe angegeben.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \
    --min-size 1 \
    --max-size 5 \
    --desired-capacity 3 \
    --placement-group my-placement-group \
    --vpc-zone-identifier "subnet-6194ea3b"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Platzierungsgruppen im Benutzerhandbuch für Amazon EC2 für Linux-Instances.

Beispiel 4: So geben Sie eine Auto-Scaling-Gruppe mit einer einzelnen Instance an und verwenden eine bestimmte Version der Startvorlage

Dieses Beispiel erstellt eine Auto-Scaling-Gruppe mit einer minimalen und maximalen Kapazität von 1, um sicherzustellen, dass eine Instance ausgeführt wird. Der Befehl gibt außerdem Version 1 einer Startvorlage an, in der die ID einer vorhandenen ENI angegeben ist. Wenn Sie eine Startvorlage verwenden, die eine vorhandene ENI für eth0 angibt, müssen Sie für die Auto-Scaling-Gruppe eine Availability Zone angeben, die mit der Netzwerkschnittstelle übereinstimmt, ohne in der Anforderung zusätzlich eine Subnetz-ID anzugeben.

```
aws autoscaling create-auto-scaling-group \
```

```
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
 \
    --min-size 1 \
    --max-size 1 \
    --availability-zones us-west-2a
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Auto-Scaling-Gruppen im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 5: So geben Sie eine andere Beendigungsrichtlinie an

In diesem Beispiel wird eine Auto-Scaling-Gruppe mithilfe einer Startkonfiguration erstellt und die Beendigungsrichtlinie so festgelegt, dass die ältesten Instances zuerst beendet werden. Der Befehl weist außerdem der Gruppe und ihren Instances ein Tag mit dem Schlüssel `Role` und dem Wert `WebServer` zu.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-configuration-name my-lc \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "OldestInstance" \
    --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Arbeiten mit Beendigungsrichtlinien für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So geben Sie einen Start-Lebenszyklus-Hook an

Das folgende Beispiel erstellt eine Auto-Scaling-Gruppe mit einem Lebenszyklus-Hook, der eine benutzerdefinierte Aktion beim Starten einer Instance unterstützt.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Inhalt der `config.json`-Datei:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-launch-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-
queue",
        "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
        "NotificationMetadata": "SQS message metadata",
        "HeartbeatTimeout": 4800,
        "DefaultResult": "ABANDON"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": [{
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "test",
        "Key": "environment"
    }]
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 7: So geben Sie einen Beendigungs-Lebenszyklus-Hook an

Das folgende Beispiel erstellt eine Auto-Scaling-Gruppe mit einem Lebenszyklus-Hook, der eine benutzerdefinierte Aktion beim Beenden einer Instance unterstützt.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Inhalt von `config.json`:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-termination-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "HeartbeatTimeout": 120,
        "DefaultResult": "CONTINUE"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067"
    ],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 8: So geben Sie eine benutzerdefinierte Beendigungsrichtlinie an

In diesem Beispiel wird eine Auto-Scaling-Gruppe erstellt, die eine benutzerdefinierte Lambda-Funktionsrichtlinie für Beendigungen angibt, die Amazon EC2 Auto Scaling mitteilt, welche Instances sicher beim Abskalieren beendet werden können.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "arn:aws:lambda:us-
west-2:123456789012:function:HelloFunction:prod" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen einer benutzerdefinierten Beendigungsrichtlinie mit Lambda im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
<vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
```

```
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createAutoScalingGroup(
```

```
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Auto-Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßige gewünschte Kapazität entspricht der minimalen Größe. Daher startet diese Auto-Scaling-Gruppe zwei Instances – jeweils eine in jeder der angegebenen beiden Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Einzelheiten zur API finden Sie unter CreateAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Auto-Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßige gewünschte Kapazität entspricht der minimalen Größe. Daher startet diese Auto-Scaling-Gruppe zwei Instances – jeweils eine in jeder der angegebenen beiden Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- Einzelheiten zur API finden Sie unter CreateAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
```

```
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the
configuration of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
 exists. "
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch
 configurations. "
                    "Consider deleting unused resources or request a limit
 increase. "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
 {
    client
        .create_auto_scaling_group()
```

```
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_launch_template TYPE REF TO /aws1/cl_asclaunchtemplatespec.

    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: iv_launch_template_name = 'my-launch-template'
    " Example: iv_min_size = 1
    " Example: iv_max_size = 3
    " Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)

    TRY.
        " Create launch template specification
        lo_launch_template = NEW /aws1/cl_asclaunchtemplatespec(
          iv_launchtemplatename = iv_launch_template_name
          iv_version = '$Default' ).

        " Create the Auto Scaling group
```

```
        " Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
        IF iv_vpc_zone_identifier IS NOT INITIAL.
          " VPC-based deployment - use subnet IDs
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            iv_vpczoneidentifier = iv_vpc_zone_identifier
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ELSE.
          " EC2-Classic or default VPC - use availability zones
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            it_availabilityzones = it_group_zones
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ENDIF.

        " Wait for the group to be created (simplified - in production use proper
 polling)
        WAIT UP TO 10 SECONDS.

        MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

      CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
        RAISE EXCEPTION lo_already_exists.
      CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
        RAISE EXCEPTION lo_limit_exceeded.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`CreateLaunchConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLaunchConfiguration` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Startkonfiguration

In diesem Beispiel wird eine einfache Startkonfiguration erstellt.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen einer Startkonfiguration im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So erstellen Sie eine Startkonfiguration mit einer Sicherheitsgruppe, einem key pair und einem Bootstrapping-Skript

In diesem Beispiel wird eine Startkonfiguration mit einer Sicherheitsgruppe, einem key pair und einem Bootstrapping-Skript erstellt, die in den Benutzerdaten enthalten sind.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --security-groups sg-eb2af88example \
    --key-name my-key-pair \
    --user-data file://myuserdata.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen einer Startkonfiguration im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So erstellen Sie eine Startkonfiguration mit einer IAM-Rolle

In diesem Beispiel wird eine Startkonfiguration mit dem Instance-Profilnamen einer IAM-Rolle erstellt.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --iam-instance-profile my-autoscaling-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter IAM-Rolle für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 4: So erstellen Sie eine Startkonfiguration mit aktivierter detaillierter Überwachung

In diesem Beispiel wird eine Startkonfiguration mit aktivierter detaillierter EC2-Überwachung erstellt, an die EC2-Metriken innerhalb von 1 Minute gesendet werden CloudWatch .

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --instance-monitoring Enabled=true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Konfigurieren der Überwachung für Auto-Scaling-Instances im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 5: So erstellen Sie eine Startkonfiguration, die Spot Instances startet

In diesem Beispiel wird eine Startkonfiguration erstellt, die Spot Instances als einzige Kaufoption verwendet.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --spot-price "0.50"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Anfordern von Spot Instances im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So erstellen Sie eine Startkonfiguration unter Verwendung einer EC2-Instance

In diesem Beispiel wird eine Startkonfiguration erstellt, die auf den Attributen einer vorhandenen Instance basiert. Es überschreibt die Tenancy für die Platzierung und ob eine öffentliche IP-Adresse festgelegt wird, indem die Optionen `--placement-tenancy` und `--no-associate-public-ip-address` einbezogen werden.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc-from-instance \
    --instance-id i-0123a456700123456 \
    --instance-type m5.large \
    --no-associate-public-ip-address \
    --placement-tenancy dedicated
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen einer Startkonfiguration mithilfe einer EC2-Instance im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 7: So erstellen Sie eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Amazon EBS-Volume

In diesem Beispiel wird eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Amazon-EBSgp3-Volume mit dem Gerätenamen `/dev/sdh` und einer Volume-Größe von 20 erstellt.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":
{"VolumeSize":20,"VolumeType":"gp3"}}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter EBS in der API-Referenz für Amazon EC2 Auto Scaling.

Informationen zur Syntax für das Angeben von JSON-formatted Parameterwerten finden Sie unter Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

Beispiel 8: So erstellen Sie eine Startkonfiguration mit einer Blockgerät-Zuweisung für ein Instance-Speicher-Volume

In diesem Beispiel wird eine Startkonfiguration mit `ephemeral1` als Instance-Speicher-Volume mit dem Gerätenamen `/dev/sdc` erstellt.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/
sdc","VirtualName":"ephemeral1"}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie BlockDeviceMappingin der Amazon EC2 Auto Scaling API-Referenz.

Informationen zur Syntax für das Angeben von JSON-formatted Parameterwerten finden Sie unter Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

Beispiel 9: So erstellen Sie eine Startkonfiguration und verhindern, dass ein Blockgerät beim Start eine Verbindung herstellt

In diesem Beispiel wird eine Startkonfiguration erstellt, die das Blockgerät unterdrückt, das in der Blockgerät-Zuweisung des AMI angegeben ist (z. B. `/dev/sdf`).

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie BlockDeviceMappingin der Amazon EC2 Auto Scaling API-Referenz.

Informationen zur Syntax für das Angeben von JSON-formatted Parameterwerten finden Sie unter Verwenden von Anführungszeichen mit Zeichenfolgen in der AWS CLI im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateLaunchConfiguration AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto-Scaling-Gruppen mit dieser Startkonfiguration gestarteten EC2-Instances verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- Einzelheiten zur API finden Sie unter CreateLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto-Scaling-Gruppen mit dieser Startkonfiguration gestarteten EC2-Instances verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- Einzelheiten zur API finden Sie unter CreateLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateOrUpdateTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOrUpdateTags` verwendet wird.

CLI

AWS CLI

So erstellen oder ändern Sie Tags für eine vorhandene Auto-Scaling-Gruppe

In diesem Beispiel werden der angegebenen Auto-Scaling-Gruppe zwei Tags hinzugefügt.

```
aws autoscaling create-or-update-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-
asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Markieren von Auto Scaling-Gruppen und Instances im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie CreateOrUpdateTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebenen Auto-Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'my'TagValue. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto-Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Einzelheiten zur API finden Sie unter CreateOrUpdateTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Auto-Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist TagValue 'my'. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto-Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- Einzelheiten zur API finden Sie unter CreateOrUpdateTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteAutoScalingGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
```

```
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
```

```
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };
```

```
        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Auto Scaling group '" << groupName << "' was
 deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So löschen Sie die angegebene Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen der Auto-Scaling-Infrastruktur im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So erzwingen Sie das Löschen der angegebenen Auto-Scaling-Gruppe

Löschen Sie die Auto-Scaling-Gruppe, ohne darauf zu warten, dass die Instances in der Gruppe beendet werden, indem Sie die `--force-delete`-Option verwenden.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --force-delete
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen der Auto-Scaling-Infrastruktur im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
```

```
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht, sofern keine Instances ausgeführt werden. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht und alle ausgeführten Instances, die sie enthält, werden beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Einzelheiten zur API finden Sie unter DeleteAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht, sofern keine Instances ausgeführt werden. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht und alle ausgeführten Instances, die sie enthält, werden beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- Einzelheiten zur API finden Sie unter DeleteAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"
```

```
        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
```

```
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
 Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
```

```
}
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " Example: iv_group_name = 'my-auto-scaling-group'

   TRY.
       ao_asc->deleteautoscalinggroup(
         iv_autoscalinggroupname = iv_group_name ).

       " Wait for the group to be deleted (simplified - in production use proper
polling)
       WAIT UP TO 10 SECONDS.

       MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

     CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
       RAISE EXCEPTION lo_activity_in_progress.
     CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).
       RAISE EXCEPTION lo_resource_in_use.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteLaunchConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLaunchConfiguration` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Startkonfiguration

In diesem Beispiel wird die angegebene Startkonfiguration gelöscht.

```
aws autoscaling delete-launch-configuration \
    --launch-configuration-name my-launch-config
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen der Auto-Scaling-Infrastruktur im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DeleteLaunchConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, sofern sie nicht einer Auto-Scaling-Gruppe zugewiesen ist. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Einzelheiten zur API finden Sie unter DeleteLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, sofern sie nicht einer Auto-Scaling-Gruppe zugewiesen ist. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- Einzelheiten zur API finden Sie unter DeleteLaunchConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteLifecycleHook`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLifecycleHook` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Lebenszyklus-Hook

In diesem Beispiel wird der angegebene Lebenszyklus-Hook gelöscht.

```
aws autoscaling delete-lifecycle-hook \
    --lifecycle-hook-name my-lifecycle-hook \
    --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteLifecycleHookin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Einzelheiten zur API finden Sie unter DeleteLifecycleHook AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- Einzelheiten zur API finden Sie unter DeleteLifecycleHook AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteNotificationConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteNotificationConfiguration` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Auto-Scaling-Benachrichtigung

In diesem Beispiel wird die angegebene Benachrichtigung aus der spezifizierten Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen der Benachrichtigungskonfiguration im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DeleteNotificationConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Einzelheiten zur API finden Sie unter DeleteNotificationConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- Einzelheiten zur API finden Sie unter DeleteNotificationConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeletePolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePolicy` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Skalierungsrichtlinie

In diesem Beispiel wird die angegebene Scaling-Richtlinie gelöscht.

```
aws autoscaling delete-policy \
    --auto-scaling-group-name my-asg \
```

```
    --policy-name alb1000-target-tracking-scaling-policy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeletePolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteScheduledAction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteScheduledAction` verwendet wird.

CLI

AWS CLI

So löschen Sie eine geplante Aktion aus einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene geplante Aktion aus der spezifizierten Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-scheduled-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteScheduledActionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Einzelheiten zur API finden Sie unter DeleteScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- Einzelheiten zur API finden Sie unter DeleteScheduledAction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTags` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Tag aus einer Auto-Scaling-Gruppe

In diesem Beispiel wird das angegebene Tag aus der angegebenen Auto-Scaling-Gruppe gelöscht.

```
aws autoscaling delete-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Markieren von Auto Scaling-Gruppen und Instances im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DeleteTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Tag der spezifizierten Auto-Scaling-Gruppe entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Beispiel 3: Bei Powershell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag der spezifizierten Auto-Scaling-Gruppe entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Beispiel 3: Bei Powershell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAccountLimits`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAccountLimits` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Limits Ihres Kontos bei Amazon EC2 Auto Scaling

In diesem Beispiel werden die Amazon EC2 Auto Scaling Scaling-Limits für Ihr AWS Konto beschrieben.

```
aws autoscaling describe-account-limits
```

Ausgabe:

```
{
    "NumberOfLaunchConfigurations": 5,
    "MaxNumberOfLaunchConfigurations": 100,
    "NumberOfAutoScalingGroups": 3,
    "MaxNumberOfAutoScalingGroups": 20
}
```

Weitere Informationen finden Sie unter Servicekontingente für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeAccountLimitsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Auto Scaling Scaling-Ressourcenlimits für Ihr AWS Konto beschrieben.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Auto Scaling Scaling-Ressourcenlimits für Ihr AWS Konto beschrieben.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAdjustmentTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAdjustmentTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen der Skalierungsanpassung

In diesem Beispiel werden die verfügbaren Anpassungstypen beschrieben.

```
aws autoscaling describe-adjustment-types
```

Ausgabe:

```
{
    "AdjustmentTypes": [
        {
            "AdjustmentType": "ChangeInCapacity"
        },
        {
            "AdjustmentType": "ExactCapacity"
        },
        {
```

```
            "AdjustmentType": "PercentChangeInCapacity"
        }
    ]
}
```

Weitere Informationen finden Sie unter Skalierungsanpassungstypen im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeAdjustmentTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Anpassungstypen beschrieben, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Einzelheiten zur API finden Sie unter DescribeAdjustmentTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Anpassungstypen beschrieben, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type
```

```
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- Einzelheiten zur API finden Sie unter DescribeAdjustmentTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeAutoScalingGroups`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingGroups` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
```

```
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die angegebene Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-names my-asg
```

Ausgabe:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-1234567890abcde12"
            },
            "MinSize": 0,
            "MaxSize": 1,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-06905f55584de02da",
                    "InstanceType": "t2.micro",
                    "AvailabilityZone": "us-west-2a",
```

```
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "ProtectedFromScaleIn": false,
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcde12"
                    }
                }
            ],
            "CreatedTime": "2023-10-28T02:39:22.152Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":"arn",
            "TrafficSources": []
        }
    ]
}
```

Beispiel 2: So beschreiben Sie die ersten 100 angegebenen Auto-Scaling-Gruppen

In diesem Beispiel werden die angegebenen Auto-Scaling-Gruppen beschrieben. Sie können bis zu 100 Gruppennamen angeben.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 100 \
    --auto-scaling-group-names "group1" "group2" "group3" "group4"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 3: So beschreiben Sie eine Auto-Scaling-Gruppe in der angegebenen Region

In diesem Beispiel werden die Auto-Scaling-Gruppen in der angegebenen Region beschrieben, wobei maximal 75 Gruppen berücksichtigt werden.

```
aws autoscaling describe-auto-scaling-groups \
```

```
    --max-items 75 \
    --region us-east-1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 4: So beschreiben Sie die angegebene Anzahl von Auto-Scaling-Gruppen

Um eine bestimmte Anzahl von Auto-Scaling-Gruppen zurückzugeben, verwenden Sie die `--max-items`-Option.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Gruppen. Um die zusätzlichen Gruppen abzurufen, verwenden Sie den Wert dieses Felds mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```
aws autoscaling describe-auto-scaling-groups \
    --starting-token Z3M3LMPEXAMPLE
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 5: So beschreiben Sie Auto-Scaling-Gruppen, die Startkonfigurationen verwenden

Dieses Beispiel verwendet die `--query`-Option, um Auto-Scaling-Gruppen zu beschreiben, die Startkonfigurationen verwenden.

```
aws autoscaling describe-auto-scaling-groups \
    --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

Ausgabe:

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
```

```
        "LaunchConfigurationName": "my-lc",
        "MinSize": 0,
        "MaxSize": 1,
        "DesiredCapacity": 1,
        "DefaultCooldown": 300,
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
        ],
        "LoadBalancerNames": [],
        "TargetGroupARNs": [],
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": 0,
        "Instances": [
            {
                "InstanceId": "i-088c57934a6449037",
                "InstanceType": "t2.micro",
                "AvailabilityZone": "us-west-2c",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc",
                "ProtectedFromScaleIn": false
            }
        ],
        "CreatedTime": "2023-10-28T02:39:22.152Z",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN":"arn",
        "TrafficSources": []
    }
]
```

Weitere Informationen finden Sie unter AWS CLI-Ausgabe filtern im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto-Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup
```

Beispiel 6: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-Kaufoptionen“ auf „Startvorlage beachten“ eingestellt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate kann über die Eigenschaft "MixedInstancesPolicy.LaunchTemplate" darauf zugegriffen werden.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto-Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup
```

Beispiel 6: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-

Kaufoptionen“ auf „Startvorlage beachten“ eingestellt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate kann über die Eigenschaft "MixedInstancesPolicy.LaunchTemplate" darauf zugegriffen werden.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.
```

```
        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
    DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        " Build group names parameter
        CREATE OBJECT lo_group_name
          EXPORTING
            iv_value = iv_group_name.
        APPEND lo_group_name TO lt_group_names.

        " Describe the Auto Scaling group
        DATA(lo_output) = ao_asc->describeautoscalinggroups(
          it_autoscalinggroupnames = lt_group_names ).

        " Return the first (and only) group in the result
        DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
        IF lines( lt_groups ) > 0.
          READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
          oo_output = lo_group.
        ENDIF.

        MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeAutoScalingInstances`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
```

```
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
 Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
 &instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine oder mehrere Instances

In diesem Beispiel wird die angegebene Instance beschrieben.

```
aws autoscaling describe-auto-scaling-instances \
    --instance-ids i-06905f55584de02da
```

Ausgabe:

```
{
    "AutoScalingInstances": [
        {
            "InstanceId": "i-06905f55584de02da",
            "InstanceType": "t2.micro",
            "AutoScalingGroupName": "my-asg",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "HEALTHY",
            "ProtectedFromScaleIn": false,
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-1234567890abcde12",
                "LaunchTemplateName": "my-launch-template",
                "Version": "1"
            }
        }
    ]
}
```

Beispiel 2: So beschreiben Sie eine oder mehrere Instances

Dieses Beispiel verwendet die `--max-items`-Option, um anzugeben, wie viele Instances mit diesem Aufruf zurückgegeben werden sollen.

```
aws autoscaling describe-auto-scaling-instances \
    --max-items 1
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Instances. Um die zusätzlichen Instances abzurufen, verwenden Sie den Wert dieses Felds mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```
aws autoscaling describe-auto-scaling-instances \
    --starting-token Z3M3LMPEXAMPLE
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 3: So beschreiben Sie Instances, die Startkonfigurationen verwenden

In diesem Beispiel wird die `--query`-Option verwendet, um Instances zu beschreiben, die Startkonfigurationen verwenden.

```
aws autoscaling describe-auto-scaling-instances \
    --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

Ausgabe:

```
[
    {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AutoScalingGroupName": "my-asg",
        "AvailabilityZone": "us-west-2c",
        "LifecycleState": "InService",
        "HealthStatus": "HEALTHY",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
    }
]
```

Weitere Informationen finden Sie unter AWS CLI-Ausgabe filtern im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
```

```
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

 .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
 response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
 instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
```

```
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die IDs Ihrer Auto-Scaling-Instances aufgeführt.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Instance beschrieben.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die IDs Ihrer Auto-Scaling-Instances aufgeführt.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Instance beschrieben.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
 Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
```

```
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
```

```
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: it_instance_ids contains a list of instance IDs

    TRY.
        DATA(lo_output) = ao_asc->describeautoscalinginstances(
          it_instanceids = it_instance_ids ).

        ot_output = lo_output->get_autoscalinginstances( ).
```

```
        MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeAutoScalingNotificationTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutoScalingNotificationTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Benachrichtigungstypen

In diesem Beispiel werden die verfügbaren Benachrichtigungstypen beschrieben.

```
aws autoscaling describe-auto-scaling-notification-types
```

Ausgabe:

```
{
    "AutoScalingNotificationTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCH",
        "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
        "autoscaling:EC2_INSTANCE_TERMINATE",
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
        "autoscaling:TEST_NOTIFICATION"
    ]
}
```

Weitere Informationen finden Sie unter Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeAutoScalingNotificationTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Benachrichtigungstypen aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingNotificationTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Benachrichtigungstypen aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- Einzelheiten zur API finden Sie unter DescribeAutoScalingNotificationTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLaunchConfigurations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLaunchConfigurations` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die angegebene Startkonfiguration

In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
aws autoscaling describe-launch-configurations \
    --launch-configuration-names my-launch-config
```

Ausgabe:

```
{
    "LaunchConfigurations": [
        {
            "LaunchConfigurationName": "my-launch-config",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
            "ImageId": "ami-0528a5175983e7f28",
            "KeyName": "my-key-pair-uswest2",
            "SecurityGroups": [
                "sg-05eaec502fcdadc2e"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "",
            "InstanceType": "t2.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "SnapshotId": "snap-06c1606ba5ca274b1",
```

```
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "DeleteOnTermination": true,
                        "Encrypted": false
                    }
                }
            ],
            "InstanceMonitoring": {
                "Enabled": true
            },
            "CreatedTime": "2020-10-28T02:39:22.321Z",
            "EbsOptimized": false,
            "AssociatePublicIpAddress": true,
            "MetadataOptions": {
                "HttpTokens": "required",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "disabled"
            }
        }
    ]
}
```

Beispiel 2: So beschreiben Sie eine bestimmte Anzahl von Startkonfigurationen

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Startkonfigurationen zurückzugeben.

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, stehen weitere Startkonfigurationen zur Verfügung. Um die zusätzlichen Startkonfigurationen abzurufen, verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt.

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- Einzelheiten zur API finden Sie DescribeLaunchConfigurationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Ausgabe:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
```

```
SecurityGroups                : {sg-67ef0308}
SpotPrice                     :
UserData                      :
```

Beispiel 3: In diesem Beispiel werden die beiden angegebenen Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: In diesem Beispiel werden alle Ihre Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeLaunchConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Ausgabe:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
AssociatePublicIpAddress      : True
BlockDeviceMappings           : {/dev/xvda}
```

```
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

Beispiel 3: In diesem Beispiel werden die beiden angegebenen Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: In diesem Beispiel werden alle Ihre Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeLaunchConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLifecycleHookTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLifecycleHookTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen von Lebenszyklus-Hooks

In diesem Beispiel werden die verfügbaren Typen von Lebenszyklus-Hooks beschrieben.

```
aws autoscaling describe-lifecycle-hook-types
```

Ausgabe:

```
{
    "LifecycleHookTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCHING",
        "autoscaling:EC2_INSTANCE_TERMINATING"
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLifecycleHookTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Typen von Lebenszyklus-Hooks aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHookTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Typen von Lebenszyklus-Hooks aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHookTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLifecycleHooks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLifecycleHooks` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Lebenszyklus-Hooks

In diesem Beispiel werden die Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-lifecycle-hooks \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "LifecycleHooks": [
        {
            "GlobalTimeout": 3000,
            "HeartbeatTimeout": 30,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-hook",
            "DefaultResult": "ABANDON",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"
        },
        {
            "GlobalTimeout": 6000,
            "HeartbeatTimeout": 60,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-termination-hook",
            "DefaultResult": "CONTINUE",
```

```
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLifecycleHooksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Beispiel 2: In diesem Beispiel werden alle Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lebenszyklus-Hooks für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHooks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Beispiel 2: In diesem Beispiel werden alle Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lebenszyklus-Hooks für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- Einzelheiten zur API finden Sie unter DescribeLifecycleHooks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancers`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancers` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Classic Load Balancers für eine Auto-Scaling-Gruppe

In diesem Beispiel werden die Classic Load Balancers für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-load-balancers \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "LoadBalancers": [
        {
            "State": "Added",
            "LoadBalancerName": "my-load-balancer"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Load Balancers für die angegebene Auto-Scaling-Gruppebeschrieben.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Load Balancers für die angegebene Auto-Scaling-Gruppebeschrieben.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMetricCollectionTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMetricCollectionTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Arten der Metrikerfassung

In diesem Beispiel werden die verfügbaren Arten der Metrikerfassung beschrieben.

```
aws autoscaling describe-metric-collection-types
```

Ausgabe:

```
{
    "Metrics": [
        {
            "Metric": "GroupMinSize"
        },
        {
            "Metric": "GroupMaxSize"
        },
        {
            "Metric": "GroupDesiredCapacity"
        },
        {
            "Metric": "GroupInServiceInstances"
```

```
        },
        {
            "Metric": "GroupInServiceCapacity"
        },
        {
            "Metric": "GroupPendingInstances"
        },
        {
            "Metric": "GroupPendingCapacity"
        },
        {
            "Metric": "GroupTerminatingInstances"
        },
        {
            "Metric": "GroupTerminatingCapacity"
        },
        {
            "Metric": "GroupStandbyInstances"
        },
        {
            "Metric": "GroupStandbyCapacity"
        },
        {
            "Metric": "GroupTotalInstances"
        },
        {
            "Metric": "GroupTotalCapacity"
        }
    ],
    "Granularities": [
        {
            "Granularity": "1Minute"
        }
    ]
}
```

Weitere Informationen finden Sie unter Auto-Scaling-Gruppen-Metriken im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeMetricCollectionTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----------
1Minute
```

- Einzelheiten zur API finden Sie unter DescribeMetricCollectionTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----------
1Minute
```

- Einzelheiten zur API finden Sie unter DescribeMetricCollectionTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeNotificationConfigurations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNotificationConfigurations` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die Benachrichtigungskonfigurationen einer angegebenen Gruppe

In diesem Beispiel werden die Benachrichtigungskonfigurationen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

Weitere Informationen finden Sie unter Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 1: So beschreiben Sie eine bestimmte Anzahl von Benachrichtigungskonfigurationen

Verwenden Sie den `max-items`-Parameter, um eine bestimmte Anzahl von Benachrichtigungskonfigurationen zurückzugeben.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-auto-scaling-group \
    --max-items 1
```

Ausgabe:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
```

```
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Benachrichtigungskonfigurationen. Verwenden Sie den Wert dieses Feldes mit dem `starting-token`-Parameter in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Benachrichtigungskonfigurationen zu erhalten.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeNotificationConfigurationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die der angegebenen Auto Scaling-Gruppe zugewiesen sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
```

```
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die allen Ihren Auto-Scaling-Gruppen zugewiesen sind.

```
Get-ASNotificationConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeNotificationConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die der angegebenen Auto Scaling-Gruppe zugewiesen sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die allen Ihren Auto-Scaling-Gruppen zugewiesen sind.

```
Get-ASNotificationConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeNotificationConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribePolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePolicies` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie die Skalierungsrichtlinien einer angegebenen Gruppe

In diesem Beispiel wird die Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "alb1000-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
```

```
                    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
                },
                "TargetValue": 1000.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        },
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "cpu40-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 40.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Dynamische Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie die Skalierungsrichtlinien eines bestimmten Namens

Verwenden Sie die `--policy-names`-Option, um bestimmte Skalierungsrichtlinien zurückzugeben.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --policy-names cpu40-target-tracking-scaling-policy
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Dynamische Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie eine Reihe von Skalierungsrichtlinien

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Richtlinien zurückzugeben.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Wenn die Ausgabe ein `NextToken`-Feld enthält, verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf, um die zusätzlichen Richtlinien abzurufen.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-
token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter Dynamische Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribePoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Ausgabe:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Beispiel 2: In diesem Beispiel werden die angegebenen Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASPolicy
```

- Einzelheiten zur API finden Sie unter DescribePolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Ausgabe:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Beispiel 2: In diesem Beispiel werden die angegebenen Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASPolicy
```

- Einzelheiten zur API finden Sie unter DescribePolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeScalingActivities`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingActivities` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
```

```
        }
        return activities;
    }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
```

```
                allActivities.insert(allActivities.end(), activities.begin(),
 activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Skalierungsaktivitäten für die angegebene Gruppe

In diesem Beispiel werden die Skalierungsaktivitäten für die angegebene Auto-Scaling-Gruppe beschrieben.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "Activities": [
```

```
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verifizieren einer Skalierungsaktivität für eine Auto-Scaling-Gruppe im Benutzerhandbuch fzu ür Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie die Skalierungsaktivitäten für eine gelöschte Gruppe

Fügen Sie die `--include-deleted-groups`-Option hinzu, um Skalierungsaktivitäten nach dem Löschen der Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg \
    --include-deleted-groups
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "Description": "Launching a new EC2 instance.  Status Reason: Your
 Spot request price of 0.001 is lower than the minimum required Spot request
 fulfillment price of 0.0031. Launching EC2 instance failed.",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of
 AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
```

```
 capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
 response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than
 the minimum required Spot request fulfillment price of 0.0031. Launching EC2
 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        }
    ]
}
```

Weitere Informationen finden Sie unter Fehlerbehebung bei Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie eine bestimmte Anzahl von Skalierungsaktivitäten

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Aktivitäten zurückzugeben.

```
aws autoscaling describe-scaling-activities \
    --max-items 1
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
```

```
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Aktivitäten. Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Aktivitäten abzurufen.

```
aws autoscaling describe-scaling-activities \
    --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter Verifizieren einer Skalierungsaktivität für eine Auto-Scaling-Gruppe im Benutzerhandbuch fzu ür Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                   .autoScalingGroupName(groupName)
```

```
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
 activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function describeScalingActivities($autoScalingGroupName)
    {
        return $this->autoScalingClient->describeScalingActivities([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Skalierungsaktivität beschrieben.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScalingActivity
```

- Einzelheiten zur API finden Sie unter DescribeScalingActivities AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
```

```
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Skalierungsaktivität beschrieben.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScalingActivity
```

- Einzelheiten zur API finden Sie unter DescribeScalingActivities AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client
```

```
    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
Any]]:
        """
        Gets information about scaling activities for the group. Scaling
activities
        are things like instances stopping or starting in response to user
requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling
activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
```

```
            raise
        else:
            return activities
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
```

```
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    DATA(lo_output) = ao_asc->describescalingactivities(
      iv_autoscalinggroupname = iv_group_name ).

    ot_output = lo_output->get_activities( ).

    MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeScalingProcessTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScalingProcessTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Prozesstypen

In diesem Beispiel werden die verfügbaren Prozesstypen beschrieben.

```
aws autoscaling describe-scaling-process-types
```

Ausgabe:

```
{
    "Processes": [
        {
            "ProcessName": "AZRebalance"
        },
        {
            "ProcessName": "AddToLoadBalancer"
        },
        {
            "ProcessName": "AlarmNotification"
        },
        {
            "ProcessName": "HealthCheck"
        },
        {
            "ProcessName": "InstanceRefresh"
        },
        {
            "ProcessName": "Launch"
        },
        {
            "ProcessName": "ReplaceUnhealthy"
        },
        {
            "ProcessName": "ScheduledActions"
        },
        {
            "ProcessName": "Terminate"
        }
    ]
}
```

Weitere Informationen erhalten Sie unter Aus- und Fortsetzen von Skalierungsprozessen im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScalingProcessTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Prozesstypen aufgelistet, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Einzelheiten zur API finden Sie unter DescribeScalingProcessTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Prozesstypen aufgelistet, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
```

```
ReplaceUnhealthy
ScheduledActions
Terminate
```

- Einzelheiten zur API finden Sie unter DescribeScalingProcessTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeScheduledActions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScheduledActions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle geplanten Aktionen

In diesem Beispiel werden alle Ihre geplanten Aktionen beschrieben.

```
aws autoscaling describe-scheduled-actions
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie geplante Aktionen für die angegebene Gruppe

Verwenden Sie die `--auto-scaling-group-name`-Option, um die geplanten Aktionen für eine bestimmte Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie die angegebene geplante Aktion

Verwenden Sie die `--scheduled-action-names`-Option, um eine bestimmte geplante Aktion zu beschreiben.

```
aws autoscaling describe-scheduled-actions \
    --scheduled-action-names my-recurring-action
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 4: So beschreiben Sie geplante Aktionen mit einer bestimmten Startzeit

Verwenden Sie die `--start-time`-Option, um die geplanten Aktionen zu beschreiben, die zu einer bestimmten Zeit beginnen.

```
aws autoscaling describe-scheduled-actions \
    --start-time "2023-12-01T04:00:00Z"
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
```

```
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 5: So beschreiben Sie geplante Aktionen, die zu einem bestimmten Zeitpunkt enden

Verwenden Sie die `--end-time`-Option, um die geplanten Aktionen zu beschreiben, die zu einem bestimmten Zeitpunkt enden.

```
aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
```

```
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 6: So beschreiben Sie eine bestimmte Anzahl von geplanten Aktionen

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von geplanten Aktionen zurückzugeben.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

Ausgabe:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere geplante Aktionen. Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Aktivitäten abzurufen.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeScheduledActionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die geplanten Skalierungsaktionen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Beispiel 2: In diesem Beispiel werden die angegebenen geplanten Skalierungsaktionen beschrieben.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Beispiel 3: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- Einzelheiten zur API finden Sie unter DescribeScheduledActions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die geplanten Skalierungsaktionen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
```

```
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Beispiel 2: In diesem Beispiel werden die angegebenen geplanten Skalierungsaktionen beschrieben.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Beispiel 3: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- Einzelheiten zur API finden Sie unter DescribeScheduledActions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

So beschreiben Sie alle Tags

In diesem Beispiel werden alle Ihre Tags beschrieben.

```
aws autoscaling describe-tags
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "Research",
            "Key": "Dept"
        },
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "WebServer",
            "Key": "Role"
        }
    ]
}
```

Weitere Informationen finden Sie unter Markieren von Auto Scaling-Gruppen und Instances im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 2: So beschreiben Sie Tags für eine bestimmte Gruppe

Verwenden Sie die `--filters`-Option, um Tags für eine bestimmte Auto-Scaling-Gruppe zu beschreiben.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

Weitere Informationen finden Sie unter Markieren von Auto Scaling-Gruppen und Instances im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 3: So beschreiben Sie die angegebene Anzahl von Tags

Verwenden Sie die `--max-items`-Option, um eine bestimmte Anzahl von Tags zurückzugeben.

```
aws autoscaling describe-tags \
    --max-items 1
```

Wenn die Ausgabe ein `NextToken`-Feld enthält, gibt es weitere Tags. Verwenden Sie den Wert dieses Feldes mit der `--starting-token`-Option in einem nachfolgenden Aufruf wie folgt, um die zusätzlichen Tags zu erhalten.

```
aws autoscaling describe-tags \
    --filters Name=auto-scaling-group,Values=my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

Weitere Informationen finden Sie unter Markieren von Auto Scaling-Gruppen und Instances im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“ beschrieben. Die möglichen Werte für den Filternamen sind „auto-scaling-group“, „key“, „value“ und „propagate-at-launch“. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Filter-Parameter zu erstellen.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Beispiel 3: In diesem Beispiel werden alle Tags für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASTag
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“ beschrieben. Die möglichen Werte für den Filternamen sind „auto-scaling-group“, „key“, „value“ und „propagate-at-launch“. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Filter-Parameter zu erstellen.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Beispiel 3: In diesem Beispiel werden alle Tags für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASTag
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTerminationPolicyTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTerminationPolicyTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die verfügbaren Typen von Beendigungsrichtlinien

In diesem Beispiel werden die verfügbaren Arten von Beendigungsrichtlinien beschrieben.

```
aws autoscaling describe-termination-policy-types
```

Ausgabe:

```
{
    "TerminationPolicyTypes": [
        "AllocationStrategy",
        "ClosestToNextInstanceHour",
        "Default",
```

```
        "NewestInstance",
        "OldestInstance",
        "OldestLaunchConfiguration",
        "OldestLaunchTemplate"
    ]
}
```

Weitere Informationen finden Sie unter Steuern, welche Auto Scaling-Instances bei der horizontalen Skalierung nach unten beendet werden im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DescribeTerminationPolicyTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Beendigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeTerminationPolicyTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Beendigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- Einzelheiten zur API finden Sie unter DescribeTerminationPolicyTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachInstances` verwendet wird.

CLI

AWS CLI

So trennen Sie eine Instance von einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt.

```
aws autoscaling detach-instances \
    --instance-ids i-030017cfa84b20135 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
            "AutoScalingGroupName": "my-asg",
            "Description": "Detaching EC2 instance: i-030017cfa84b20135",
            "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was
 detached in response to a user request, shrinking the capacity from 2 to 1.",
            "StartTime": "2020-04-12T15:02:16.179Z",
            "StatusCode": "InProgress",
            "Progress": 50,
```

```
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DetachInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter DetachInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter DetachInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachLoadBalancers`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachLoadBalancers` verwendet wird.

CLI

AWS CLI

So trennen Sie einen Classic Load Balancer von einer Auto-Scaling-Gruppe

In diesem Beispiel wird der angegebene Classic Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
aws autoscaling detach-load-balancers \
    --load-balancer-names my-load-balancer \
```

```
    --auto-scaling-group-name my-asg
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Anhängen eines Load Balancers an Ihre Auto-Scaling-Gruppe im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie DetachLoadBalancersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DisableMetricsCollection`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableMetricsCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
            autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So deaktivieren Sie die Erfassung von Metriken für Auto-Scaling-Gruppen

In diesem Beispiel wird die Erfassung der `GroupDesiredCapacity`-Metrik für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die spezifizierten Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DisableMetricsCollection AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die spezifizierten Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter DisableMetricsCollection AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.
```

```
        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
'{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        ao_asc->disablemetricscollection(
          iv_autoscalinggroupname = iv_group_name ).

        MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.
```

```
    CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
      RAISE EXCEPTION lo_contention.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableMetricsCollection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableMetricsCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
        {
            "GroupMaxSize",
```

```
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
```

```
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So aktivieren Sie die Erfassung von Auto-Scaling-Gruppen

In diesem Beispiel wird die Datenerfassung für die angegebene Auto-Scaling-Gruppe aktiviert.

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

Beispiel 2: So sammeln Sie Daten für die angegebene Metrik einer Auto-Scaling-Gruppe

Verwenden Sie die `--metrics`-Option, um Daten für eine bestimmte Metrik zu sammeln.

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity --granularity "1Minute"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter CloudWatch Monitoring-Metriken für Ihre Auto Scaling Scaling-Gruppen und -Instances im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: In diesem Beispiel wird eine Überwachung aller Metriken der angegebenen Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Einzelheiten zur API finden Sie unter EnableMetricsCollection AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: In diesem Beispiel wird eine Überwachung aller Metriken der angegebenen Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- Einzelheiten zur API finden Sie unter EnableMetricsCollection AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
 Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
 activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
 collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
 Granularity="1Minute"
```

```
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
                logger.error(
                    f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: iv_group_name = 'my-auto-scaling-group'
```

```
    " Example: it_metrics contains list of metrics like 'GroupMinSize',
 'GroupMaxSize', etc.

    TRY.
        ao_asc->enablemetricscollection(
          iv_autoscalinggroupname = iv_group_name
          it_metrics = it_metrics
          iv_granularity = '1Minute' ).

        MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`EnterStandby`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnterStandby` verwendet wird.

CLI

AWS CLI

So verschieben Sie Instances in den Standby-Modus

In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt. Das ist nützlich, um eine Instance zu aktualisieren oder Fehler zu beheben, die sich derzeit im Service befindet.

```
aws autoscaling enter-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
 moved to standby in response to a user request, shrinking the capacity from 1 to
 0.",
            "StartTime": "2020-10-31T20:31:00.949Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

Weitere Informationen finden Sie unter Instance-Lebenszyklus in Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie EnterStandbyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
```

```
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter EnterStandby AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter EnterStandby AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ExecutePolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ExecutePolicy` verwendet wird.

CLI

AWS CLI

So führen Sie eine Skalierungsrichtlinie aus

In diesem Beispiel wird die Skalierungsrichtlinie mit dem Namen `my-step-scale-out-policy` für die angegebene Auto-Scaling-Gruppe ausgeführt.

```
aws autoscaling execute-policy \
    --auto-scaling-group-name my-asg \
    --policy-name  my-step-scale-out-policy \
    --metric-value 95 \
    --breach-threshold 80
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Schrittweise und einfache Skalierungsrichtlinien im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie ExecutePolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel führt die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe aus.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe ausgeführt, nachdem die Wartezeit abgelaufen ist.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter ExecutePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel führt die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe aus.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe ausgeführt, nachdem die Wartezeit abgelaufen ist.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter ExecutePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ExitStandby`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ExitStandby` verwendet wird.

CLI

AWS CLI

So verschieben Sie Instances aus dem Standby-Modus

In diesem Beispiel wird die angegebenen Instance aus dem Standby-Modus verschoben.

```
aws autoscaling exit-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-061c63c5eb45f0416",
```

```
            "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
 moved out of standby in response to a user request, increasing the capacity from
 0 to 1.",
            "StartTime": "2020-10-31T20:32:50.222Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

Weitere Informationen finden Sie unter Vorübergehendes Entfernen von Instances aus einer Auto-Scaling-Gruppe im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie ExitStandbyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance aus dem Standby-Modus verschoben.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter ExitStandby AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance aus dem Standby-Modus verschoben.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter ExitStandby AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutLifecycleHook`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutLifecycleHook` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Lebenszyklus-Hook

In diesem Beispiel wird ein Lebenszyklus-Hook erstellt, der bei allen neu gestarteten Instances mit einem Timeout von 4 800 Sekunden aufgerufen wird. Dies ist nützlich, um die Instanzen im

Wartezustand zu halten, bis die Benutzerdatenskripts abgeschlossen sind, oder um eine AWS Lambda-Funktion mit aufzurufen. EventBridge

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --heartbeat-timeout 4800
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn ein Lebenszyklus-Hook mit demselben Namen bereits vorhanden ist, wird er durch den neuen Lebenszyklus-Hook überschrieben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So senden Sie eine E-Mail-Nachricht in Amazon SNS, die Sie über den Statuswechsel einer Instance informiert

In diesem Beispiel wird ein Lebenszyklus-Hook mit dem Amazon-SNS-Thema und der IAM-Rolle erstellt, die zum Empfangen von Benachrichtigungen beim Instance-Start verwendet werden sollen.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So veröffentlichen Sie eine Nachricht in einer Amazon-SQS-Warteschlange

In diesem Beispiel wird ein Lebenszyklus-Hook erstellt, der eine Nachricht mit Metadaten in der angegebenen Amazon-SQS-Warteschlange veröffentlicht.

```
aws autoscaling put-lifecycle-hook \
```

```
    --auto-scaling-group name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \
    --role-arn arn:aws:iam::123456789012:role/my-notification-role \
    --notification-metadata "SQS message metadata"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie PutLifecycleHookin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Einzelheiten zur API finden Sie unter PutLifecycleHook AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- Einzelheiten zur API finden Sie unter PutLifecycleHook AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutNotificationConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutNotificationConfiguration` verwendet wird.

CLI

AWS CLI

So fügen Sie eine neue Benachrichtigung hinzu

In diesem Beispiel wird die angegebene Benachrichtigung der spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling put-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --notification-type autoscaling:TEST_NOTIFICATION
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Abrufen von Amazon-SNS-Benachrichtigungen über Skalierungen einer Auto-Scaling-Gruppe im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie PutNotificationConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das spezifizierte SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Einzelheiten zur API finden Sie unter PutNotificationConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das spezifizierte SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- Einzelheiten zur API finden Sie unter PutNotificationConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutScalingPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutScalingPolicy` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Auto-Scaling-Gruppe eine Richtlinie zur Zielverfolgung hinzu

Im folgenden Beispiel für `put-scaling-policy` wird eine Skalierungsrichtlinie zur Zielverfolgung auf die angegebene Auto-Scaling-Gruppe angewendet. Die Ausgabe enthält die

ARNs und Namen der beiden CloudWatch Alarme, die in Ihrem Namen erstellt wurden. Wenn bereits eine Skalierungsrichtlinie mit demselben Namen vorhanden ist, wird diese durch die neue Skalierungsrichtlinie überschrieben.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

Inhalt von `config.json`:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
          "PredefinedMetricType": "ALBRequestCountPerTarget",
          "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
     }
}
```

Ausgabe:

```
{
     "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
     "Alarms": [
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
             "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
         },
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
             "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
         }
     ]
 }
```

Weitere Beispiele finden Sie unter Beispiel für Skalierungsrichtlinien für die AWS Befehlszeilenschnittstelle (AWS CLI) im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutScalingPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der spezifizierten Auto-Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter interpretiert wird. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Einzelheiten zur API finden Sie unter PutScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der spezifizierten Auto-Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter interpretiert wird. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- Einzelheiten zur API finden Sie unter PutScalingPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutScheduledUpdateGroupAction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutScheduledUpdateGroupAction` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie einer Auto-Scaling-Gruppe eine geplante Aktion hinzu

In diesem Beispiel wird die angegebene geplante Aktion der angegebenen Auto-Scaling-Gruppe hinzugefügt.

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action \
    --start-time "2023-05-12T08:00:00Z" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn eine geplante Aktion mit demselben Namen bereits vorhanden ist, wird sie durch die neue geplante Aktion überschrieben.

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So geben Sie einen wiederkehrenden Zeitplan an

In diesem Beispiel wird eine geplante Aktion erstellt, die nach einem wiederkehrenden Zeitplan skaliert wird und jedes Jahr am 1. Januar, Juni und Dezember um 00:30 Uhr ausgeführt wird.

```
aws autoscaling put-scheduled-update-group-action \
```

```
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action \
    --recurrence "30 0 1 1,6,12 *" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Wenn eine geplante Aktion mit demselben Namen bereits vorhanden ist, wird sie durch die neue geplante Aktion überschrieben.

Weitere Informationen finden Sie unter Geplante Skalierung im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie PutScheduledUpdateGroupActionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Einzelheiten zur API finden Sie unter PutScheduledUpdateGroupAction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- Einzelheiten zur API finden Sie unter PutScheduledUpdateGroupAction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RecordLifecycleActionHeartbeat`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RecordLifecycleActionHeartbeat` verwendet wird.

CLI

AWS CLI

So zeichnen Sie den Heartbeat einer Lebenszyklusaktion auf

Dieses Beispiel zeichnet den Heartbeat für eine Lebenszyklusaktion auf, um die Instance im Status „Ausstehend“ zu halten.

```
aws autoscaling record-lifecycle-action-heartbeat \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Lebenszyklus-Hooks für Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie RecordLifecycleActionHeartbeatin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instance im Status „Ausstehend“, bis die benutzerdefinierte Aktion abgeschlossen ist.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter RecordLifecycleActionHeartbeat AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instance im Status „Ausstehend“, bis die benutzerdefinierte Aktion abgeschlossen ist.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- Einzelheiten zur API finden Sie unter RecordLifecycleActionHeartbeat AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResumeProcesses`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResumeProcesses` verwendet wird.

CLI

AWS CLI

So setzen Sie einen ausgesetzten Prozess fort

In diesem Beispiel wird der angegebene ausgesetzte Skalierungsprozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
aws autoscaling resume-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen erhalten Sie unter Aus- und Fortsetzen von Skalierungsprozessen im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie ResumeProcessesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle ausgesetzten Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter ResumeProcesses AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle ausgesetzten Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter ResumeProcesses AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`SetDesiredCapacity`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetDesiredCapacity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
            autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So geben Sie die gewünschte Kapazität für die Auto-Scaling-Gruppe an

In diesem Beispiel wird die gewünschte Kapazität für die angegebene Auto-Scaling-Gruppe angegeben.

```
aws autoscaling set-desired-capacity \
    --auto-scaling-group-name my-asg \
    --desired-capacity 2 \
    --honor-cooldown
```

Wenn dieser Befehl erfolgreich war, kehrt er zum Prompt zurück.

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
```

```
            'AutoScalingGroupName' => $autoScalingGroupName,
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben und gewartet, bis die Ruhephase abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter SetDesiredCapacity AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben und gewartet, bis die Ruhephase abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- Einzelheiten zur API finden Sie unter SetDesiredCapacity AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
 keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
```

```
                f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
```

```
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: iv_capacity = 2

    TRY.
        ao_asc->setdesiredcapacity(
          iv_autoscalinggroupname = iv_group_name
          iv_desiredcapacity = iv_capacity
          iv_honorcooldown = abap_false ).

        MESSAGE 'Desired capacity set successfully' TYPE 'I'.

      CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
```

```
        RAISE EXCEPTION lo_activity_in_progress.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`SetInstanceHealth`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetInstanceHealth` verwendet wird.

CLI

AWS CLI

So geben Sie den Zustand einer Instance an

In diesem Beispiel wird der Zustand der angegebenen Instance auf `Unhealthy` gesetzt.

```
aws autoscaling set-instance-health \
    --instance-id i-061c63c5eb45f0416 \
    --health-status Unhealthy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetInstanceHealthin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerhaft“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instance.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerfrei“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Einzelheiten zur API finden Sie unter SetInstanceHealth AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerhaft“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instance.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerfrei“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- Einzelheiten zur API finden Sie unter SetInstanceHealth AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetInstanceProtection`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetInstanceProtection` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie die Instance-Schutzeinstellung für eine Instance

In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
aws autoscaling set-instance-protection \
```

```
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg --protected-from-scale-in
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So ändern Sie die Instance-Schutzeinstellung für eine Instance

In diesem Beispiel wird der Instance-Schutz für die angegebene Instance deaktiviert.

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --no-protected-from-scale-in
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetInstanceProtectionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instance-Schutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Einzelheiten zur API finden Sie unter SetInstanceProtection AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instance-Schutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- Einzelheiten zur API finden Sie unter SetInstanceProtection AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SuspendProcesses`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SuspendProcesses` verwendet wird.

CLI

AWS CLI

So setzen Sie Auto-Scaling-Prozesse aus

In diesem Beispiel wird der angegebene Skalierungsprozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
aws autoscaling suspend-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen erhalten Sie unter Aus- und Fortsetzen von Skalierungsprozessen im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie SuspendProcessesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter SuspendProcesses AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- Einzelheiten zur API finden Sie unter SuspendProcesses AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`TerminateInstanceInAutoScalingGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TerminateInstanceInAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beenden Sie eine Instance in einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Instance in der angegebenen Auto-Scaling-Gruppe beendet, ohne dass die Größe der Gruppe aktualisiert wird. Amazon EC2 Auto Scaling startet eine Ersatz-Instance, nachdem die angegebene Instance beendet wurde.

```
aws autoscaling terminate-instance-in-auto-scaling-group \
    --instance-id i-061c63c5eb45f0416 \
    --no-should-decrement-desired-capacity
```

Ausgabe:

```
{
    "Activities": [
        {
            "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
            "Cause": "",
            "StartTime": "2020-10-31T20:34:25.680Z",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                   .instanceId(instanceId)
                   .shouldDecrementDesiredCapacity(false)
                   .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance " + instanceId);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' =>
 $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
```

```
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
 $attempts < 5) {
                error_log("Cannot terminate an instance while it is still
 pending. Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto-Scaling-Gruppe reduziert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
```

```
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling-Gruppe zu reduzieren. Auto Scaling startet eine neue Ersatz-Instance.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter TerminateInstanceInAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto-Scaling-Gruppe reduziert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling-Gruppe zu reduzieren. Auto Scaling startet eine neue Ersatz-Instance.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- Einzelheiten zur API finden Sie unter TerminateInstanceInAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
 capacity
                                  of the group. When passing True for this
 parameter,
                                  you can stop an instance without having a
 replacement
                                  instance start when the desired capacity
 threshold is
                                  crossed.
        :return: A dictionary containing details of the scaling activity that
 occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
```

```
            response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
    }
}

async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
```

```
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
    DATA(lo_output) = ao_asc->terminateinstinautoscgroup(
      iv_instanceid = iv_instance_id
      iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

    oo_output = lo_output->get_activity( ).

    MESSAGE 'Instance terminated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateAutoScalingGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAutoScalingGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
```

```
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie die Größenbeschränkungen einer Auto-Scaling-Gruppe

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe mit einer Mindestgröße von 2 und einer Höchstgröße von 10 aktualisiert.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --min-size 2 \
    --max-size 10
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Festlegen der Größenbeschränkungen für Ihre Auto-Scaling-Gruppe im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 2: So fügen Sie Zustandsprüfungen für Elastic Load Balancing hinzu und legen fest, welche Availability Zones und Subnetze verwendet werden sollen

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen für Elastic Load Balancing-hinzuzufügen. Dieser Befehl aktualisiert auch den Wert von `--vpc-zone-identifier` mit einer Liste von Subnetz-IDs in mehreren Availability Zones.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --health-check-type ELB \
    --health-check-grace-period 600 \
```

```
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Elastic Load Balancing und Amazon EC2 Auto Scaling im Benutzerhandbuch für Amazon EC2 Auto Scaling.

Beispiel 3: So aktualisieren Sie die Platzierungsgruppe und die Beendigungsrichtlinie

In diesem Beispiel werden die zu verwendende Platzierungsgruppe und die Beendigungsrichtlinie aktualisiert.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --placement-group my-placement-group \
    --termination-policies "OldestInstance"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Auto-Scaling-Gruppen im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 4: So verwenden Sie die neueste Version der Startvorlage

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um die neueste Version der angegebenen Startvorlage zu verwenden.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Startvorlagen im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 5: So verwenden Sie eine bestimmte Version der Startvorlage

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um eine bestimmte Version einer Startvorlage anstelle der neuesten oder der Standardversion zu verwenden.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
```

```
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Startvorlagen im Benutzerhandbuch zu Amazon EC2 Auto Scaling.

Beispiel 6: So definieren Sie eine Richtlinie für gemischte Instances und aktivieren den Kapazitätsausgleich

In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um eine Richtlinie für gemischte Instances zu verwenden, gleichzeitig wird die Kapazitätsanpassung aktiviert. Mit dieser Struktur können Sie Gruppen mit Spot und On-Demand Kapazitäten angeben und unterschiedliche Startvorlagen für unterschiedliche Architekturen verwenden.

```
aws autoscaling update-auto-scaling-group \
    --cli-input-json file://~/config.json
```

Inhalt von `config.json`:

```
{
    "AutoScalingGroupName": "my-asg",
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c6g.large",
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template-for-arm",
                        "Version": "$Latest"
                    }
                },
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
```

```
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    }
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Auto-Scaling-Gruppen mit mehreren Instance-Typen und Kaufoptionen im Benutzerhandbuch für Amazon EC2 Auto Scaling.

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
```

```
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
 " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
```

```
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
        if (array_key_exists('MaxSize', $args)) {
```

```
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die minimale und maximale Größe der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die standardmäßige Ruhephase der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen mit Elastic Load Balancing zu verwenden.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Einzelheiten zur API finden Sie unter UpdateAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die minimale und maximale Größe der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die standardmäßige Ruhephase der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen mit Elastic Load Balancing zu verwenden.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- Einzelheiten zur API finden Sie unter UpdateAutoScalingGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
```

```
                    "The Auto Scaling group '%s' is currently in use and cannot
 be modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
 to update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
 Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
```

```
}
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_max_size = 5

TRY.
    ao_asc->updateautoscalinggroup(
      iv_autoscalinggroupname = iv_group_name
      iv_maxsize = iv_max_size
      iv_minsize = iv_min_size ).

    MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für Auto Scaling mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Auto Scaling mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Auto Scaling oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK

Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
  Scaling.
    /// The launch template specifies a Bash script in its user data field that
  runs after
    /// the instance is started. This script installs the Python packages and
  starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
  run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
  calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                     "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
```

```
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);
```

```
            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =
```

```
 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
```

```
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
```

```
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
```

```
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
```

```
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
```

```
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();
```

```
            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();
```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
```

```
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
```

```
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
```

```
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
      type: "confirm",
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
```

```
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
```

```
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
```

```
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
```

```
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
```

```
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
```

```
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
```

```
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
```

```
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
```

```
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
```

```
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
```

```
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
```

```
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
```

```
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
```

```
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
```

```
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
```

```
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
```

```
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
```

```
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
```

```
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
```

```
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
```

```
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
```

```
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
```

```
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
```

```
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
```

```
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
```

```
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
```

```
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
```

```
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```

```
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
```

```
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
```

```
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
```

```
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.

- AttachLoadBalancerTargetGroups
- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")
```

```
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
```

```
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
```

```
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
```

```
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
```

```
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
```

```
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.
```

```
        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
```

```
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
```

```
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
```

```
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
```

```
            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
```

```
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
```

```
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
```

```
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
```

```
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
```

```
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")
```

```
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
```

```
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
```

```
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
```

```
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
```

```
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
 found. "
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
```

```
        Initializes the RecommendationService class with the necessary
parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
```

```
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
```

```
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Codebeispiele für AWS Batch verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Batch mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS Batch Benutzerhandbuch — Weitere Informationen zu AWS Batch.
- AWS Batch API-Referenz — Details zu allen verfügbaren AWS Batch Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Batch verwenden AWS SDKs
  - Hallo AWS Batch
  - Lernen Sie die Grundlagen von AWS Batch mit einem AWS SDK
  - Aktionen für AWS Batch verwenden AWS SDKs
    - Verwenden Sie mit CreateComputeEnvironmenteinem AWS SDK oder CLI
    - Verwenden Sie CreateJobQueuemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteComputeEnvironmentmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteJobQueuemit einem AWS SDK oder CLI
    - Verwenden Sie DeregisterJobDefinitionmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeComputeEnvironmentsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeJobQueuesmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeJobsmit einem AWS SDK oder CLI
    - Verwenden Sie ListJobsPaginatormit einem AWS SDK
    - Verwenden Sie RegisterJobDefinitionmit einem AWS SDK oder CLI
    - Verwenden Sie SubmitJobmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateComputeEnvironmentmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateJobQueuemit einem AWS SDK oder CLI

- Szenarien für AWS Batch verwenden AWS SDKs
  - Erste Schritte mit Batch-Verarbeitung und serverlosen Containern

Grundlegende Beispiele für AWS Batch verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Batch mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS Batch
- Lernen Sie die Grundlagen von AWS Batch mit einem AWS SDK
- Aktionen für AWS Batch verwenden AWS SDKs
  - Verwenden Sie mit CreateComputeEnvironmenteinem AWS SDK oder CLI
  - Verwenden Sie CreateJobQueuemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteComputeEnvironmentmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteJobQueuemit einem AWS SDK oder CLI
  - Verwenden Sie DeregisterJobDefinitionmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeComputeEnvironmentsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeJobQueuesmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListJobsPaginatormit einem AWS SDK
  - Verwenden Sie RegisterJobDefinitionmit einem AWS SDK oder CLI
  - Verwenden Sie SubmitJobmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateComputeEnvironmentmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateJobQueuemit einem AWS SDK oder CLI

Hallo AWS Batch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Batch beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or
 empty");
        }
```

```
        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
{
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return batchClient;
    }
}
```

- Einzelheiten zur API finden Sie JobsPaginator in der Liste in der AWS SDK for Java 2.x API-Referenz.

Lernen Sie die Grundlagen von AWS Batch mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine AWS Batch Rechenumgebung.
- Überprüfen des Status der Datenverarbeitungsumgebung
- Richten Sie eine AWS Batch Job-Warteschlange und eine Jobdefinition ein.
- Registrieren einer Auftragsdefinition
- Reichen Sie einen AWS Batch Job ein.
- Abrufen einer Liste der für die Auftragswarteschlange relevanten Aufträge
- Überprüfen des Auftragsstatus
- AWS Batch Ressourcen löschen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Batch Funktionen demonstriert werden.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from
 Amazon ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/
ecr
 *
 */
```

```
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-
text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that
 dynamically provisions the required compute
            resources for batch computing workloads. The Java V2
 `BatchAsyncClient` allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute
 environment, job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.
```

```
            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
 execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
 describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

 batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
 ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
 ex.getMessage());
                }
                countdown(2);

 batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
```

```
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon
ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch
jobs.
```

```
            After creating a compute environment, you can define job queues and
job definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to
easily configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" +
computeEnvironmentName + "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause !=
null ? cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +"
Compute Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the
execution of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
```

```
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and
so on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
```

```
                    On which platform/CPU architecture combination did you build
the Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
```

```
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}",
job.jobId(), job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job:
{}", rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
```

```
            }

            logger.info("Second, we will disable and then delete the Job
Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);
```

```
            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: "
+ ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }


batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");

            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
```

```
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
```

```
                            .orElseThrow(() -> new RuntimeException("Default VPC
not found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build(),
                                            Filter.builder()
                                                    .name("default-for-az")
                                                    .values("true")
                                                    .build())
                                    .build())
                            .thenApply(DescribeSubnetsResponse::subnets)
                            .thenApply(subnets -> subnets.stream()
                                    .findFirst()
                                    .map(Subnet::subnetId)
                                    .orElseThrow(() -> new RuntimeException("No
default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture =
defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("group-name")
                                                    .values("default")
                                                    .build(),
                                            Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build())
                                    .build())

.thenApply(DescribeSecurityGroupsResponse::securityGroups)
                            .thenApply(securityGroups -> securityGroups.stream()
                                    .findFirst()
                                    .map(SecurityGroup::groupId)
                                    .orElseThrow(() -> new RuntimeException("No
default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
```

```
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

Eine Wrapper-Klasse für AWS Batch SDK-Methoden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import
 software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import
 software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import
 software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
```

```
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import
 software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import
 software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;
```

```
   private static final Logger logger =
LoggerFactory.getLogger(BatchActions.class);

   private static BatchAsyncClient getAsyncClient() {
       if (batchClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryPolicy(RetryPolicy.builder()
                   .numRetries(3)
                   .build())
               .build();

           batchClient = BatchAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return batchClient;
   }

   /**
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to
create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute
environment
    * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute
environment has been created
```

```
    * @throws BatchException if there is an error creating the compute
environment
    * @throws RuntimeException if there is an unexpected error during the
operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
           .serviceRole(batchIAMRole)
           .build();

       CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
              throw new RuntimeException(errorMessage, ex);
           }
       });

       return response;
   }

   public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
       DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .build();
```

```
        return
getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute
environment, or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });
```

```
        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }

    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
     * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
     * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
     */
    public CompletableFuture<String> createJobQueueAsync(String jobQueueName,
String computeEnvironmentName) {
        if (jobQueueName == null || jobQueueName.isEmpty()) {
            throw new IllegalArgumentException("Job queue name cannot be null or
empty");
        }
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
        }

        CreateJobQueueRequest request = CreateJobQueueRequest.builder()
            .jobQueueName(jobQueueName)
            .priority(1)
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });
```

```
        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }

    /**
     * Asynchronously lists the jobs in the specified job queue with the given
job status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or
empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
{
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
```

```
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution
role
    *                        that provides permissions for the containers in
the job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes
exceptionally with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration =
NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
               .executionRoleArn(executionRoleARN)
               .resourceRequirements(
                       Arrays.asList(
                               ResourceRequirement.builder()
                                       .type(ResourceType.VCPU)
                                       .value("1")
                                       .build(),
                               ResourceRequirement.builder()
                                       .type(ResourceType.MEMORY)
                                       .value("2048")
                                       .build()
                       )
               )
               .networkConfiguration(networkConfiguration)
              .runtimePlatform(b -> b
                       .cpuArchitecture(cpuArch)
                       .operatingSystemFamily("LINUX"))
               .build();

       RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
               .jobDefinitionName(jobDefinitionName)
               .type(JobDefinitionType.CONTAINER)
               .containerProperties(containerProperties)
```

```
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has
been deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }
```

```
    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue
update operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to
delete.
     * @return A CompletableFuture that represents the asynchronous deletion of
the job queue.
     *         The future completes when the job queue has been successfully
deleted or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally
with the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();
```

```
        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }

    /**
     * Asynchronously describes the job queue associated with the specified
compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find
the associated job queue for
     * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
     * @throws RuntimeException if the job queue description fails
     */
    public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
        DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
            .build();

        CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
        return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
            if (describeJobQueuesResponse != null) {
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name =
getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the
compute environment: " + jobQueueARN);
                        }
```

```
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to
disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
```

```
                throw new RuntimeException("Failed to disable compute
environment: " + ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of
the submitted job
     */
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName,
String jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
```

```
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
     * @return a {@link CompletableFuture} that completes when the job queue is
disabled
     */
    public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
        AtomicBoolean isDisabled = new AtomicBoolean(false);
        return CompletableFuture.runAsync(() -> {
            while (!isDisabled.get()) {
                DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                    .jobQueues(jobQueueArn)
                    .build();

                CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
                responseFuture.whenComplete((describeResponse, ex) -> {
                    if (describeResponse != null) {
                        for (JobQueueDetail jobQueue :
describeResponse.jobQueues()) {
                            if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                                isDisabled.set(true);
                                break;
                            }
                        }
```

```
                    } else {
                        throw new RuntimeException("Error describing job queues",
ex);
                    }
                }).join();

                if (!isDisabled.get()) {
                    try {
                        logger.info("Waiting for job queue to be disabled...");
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                        throw new RuntimeException("Thread interrupted while
waiting for job queue to be disabled", e);
                    }
                }
            }
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to
be disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
```

```
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new
RuntimeException("Failed to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn =
response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

Aktionen für AWS Batch verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Batch Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS Batch API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS Batch verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Batch -API-Referenz.

Beispiele

- Verwenden Sie mit CreateComputeEnvironmenteinem AWS SDK oder CLI
- Verwenden Sie CreateJobQueuemit einem AWS SDK oder CLI
- Verwenden Sie DeleteComputeEnvironmentmit einem AWS SDK oder CLI
- Verwenden Sie DeleteJobQueuemit einem AWS SDK oder CLI

- Verwenden Sie DeregisterJobDefinitionmit einem AWS SDK oder CLI
- Verwenden Sie DescribeComputeEnvironmentsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeJobQueuesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListJobsPaginatormit einem AWS SDK
- Verwenden Sie RegisterJobDefinitionmit einem AWS SDK oder CLI
- Verwenden Sie SubmitJobmit einem AWS SDK oder CLI
- Verwenden Sie UpdateComputeEnvironmentmit einem AWS SDK oder CLI
- Verwenden Sie UpdateJobQueuemit einem AWS SDK oder CLI

Verwenden Sie mit **`CreateComputeEnvironment`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateComputeEnvironment` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

Um eine verwaltete Rechenumgebung mit On-Demand Instanzen zu erstellen

In diesem Beispiel wird eine verwaltete Datenverarbeitungsumgebung mit bestimmten C4-Instance-Typen erstellt, die bei Bedarf gestartet werden. Die Rechenumgebung wird aufgerufen C4OnDemand.

Befehl:

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
C4OnDemand.json
```

JSON-Dateiformat:

```
{
  "computeEnvironmentName": "C4OnDemand",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "type": "EC2",
    "minvCpus": 0,
    "maxvCpus": 128,
    "desiredvCpus": 48,
    "instanceTypes": [
      "c4.large",
      "c4.xlarge",
      "c4.2xlarge",
      "c4.4xlarge",
      "c4.8xlarge"
    ],
    "subnets": [
      "subnet-220c0e0a",
      "subnet-1a95556d",
      "subnet-978f6dce"
    ],
    "securityGroupIds": [
      "sg-cf5093b2"
    ],
    "ec2KeyPair": "id_rsa",
    "instanceRole": "ecsInstanceRole",
    "tags": {
      "Name": "Batch Instance - C4OnDemand"
    }
  },
  "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

Ausgabe:

```
{
    "computeEnvironmentName": "C4OnDemand",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/C4OnDemand"
}
```

So erstellen Sie eine verwaltete Datenverarbeitungsumgebung mit Spot Instances

In diesem Beispiel wird eine verwaltete Rechenumgebung mit dem Instance-Typ M4 erstellt, die gestartet wird, wenn der Spot-Angebotspreis 20% des On-Demand Preises für den Instance-Typ erreicht oder darunter liegt. Die Datenverarbeitungsumgebung wird als M4Spot bezeichnet.

Befehl:

```
aws batch create-compute-environment --cli-input-json file://<path_to_json_file>/
M4Spot.json
```

JSON-Dateiformat:

```
{
  "computeEnvironmentName": "M4Spot",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "type": "SPOT",
    "spotIamFleetRole": "arn:aws:iam::012345678910:role/aws-ec2-spot-fleet-role",
    "minvCpus": 0,
    "maxvCpus": 128,
    "desiredvCpus": 4,
    "instanceTypes": [
      "m4"
    ],
    "bidPercentage": 20,
    "subnets": [
      "subnet-220c0e0a",
      "subnet-1a95556d",
      "subnet-978f6dce"
    ],
    "securityGroupIds": [
      "sg-cf5093b2"
    ],
    "ec2KeyPair": "id_rsa",
    "instanceRole": "ecsInstanceRole",
    "tags": {
      "Name": "Batch Instance - M4Spot"
    }
  },
  "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole"
}
```

Ausgabe:

```
{
    "computeEnvironmentName": "M4Spot",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/M4Spot"
}
```

- Einzelheiten zur API finden Sie CreateComputeEnvironmentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to
create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute
environment
     * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute
environment has been created
     * @throws BatchException if there is an error creating the compute
environment
     * @throws RuntimeException if there is an unexpected error during the
operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
```

```
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }
```

- Einzelheiten zur API finden Sie CreateComputeEnvironmentin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateJobQueue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateJobQueue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So erstellen Sie eine Auftragswarteschlange mit niedriger Priorität mit einer einzigen Datenverarbeitungsumgebung

In diesem Beispiel wird eine Job-Warteschlange namens erstellt LowPriority , die die M4Spot-Rechenumgebung verwendet.

Befehl:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/
LowPriority.json
```

JSON-Dateiformat:

```
{
  "jobQueueName": "LowPriority",
  "state": "ENABLED",
  "priority": 10,
  "computeEnvironmentOrder": [
    {
      "order": 1,
      "computeEnvironment": "M4Spot"
    }
  ]
}
```

Ausgabe:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/LowPriority",
    "jobQueueName": "LowPriority"
}
```

So erstellen Sie eine Auftragswarteschlange mit hoher Priorität mit zwei Datenverarbeitungsumgebungen

In diesem Beispiel wird eine Jobwarteschlange mit dem Namen erstellt HighPriority , die die C4OnDemand Rechenumgebung mit der Reihenfolge 1 und die M4Spot-Computerumgebung

mit der Reihenfolge 2 verwendet. Der Scheduler versucht zuerst, Jobs in der C4OnDemand Rechenumgebung zu platzieren.

Befehl:

```
aws batch create-job-queue --cli-input-json file://<path_to_json_file>/
HighPriority.json
```

JSON-Dateiformat:

```
{
  "jobQueueName": "HighPriority",
  "state": "ENABLED",
  "priority": 1,
  "computeEnvironmentOrder": [
    {
      "order": 1,
      "computeEnvironment": "C4OnDemand"
    },
    {
      "order": 2,
      "computeEnvironment": "M4Spot"
    }
  ]
}
```

Ausgabe:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/HighPriority",
    "jobQueueName": "HighPriority"
}
```

- Einzelheiten zur API finden Sie CreateJobQueuein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName,
String computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
       response.whenComplete((resp, ex) -> {
```

```
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }
```

- Einzelheiten zur API finden Sie CreateJobQueuein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteComputeEnvironment`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteComputeEnvironment` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So löschen Sie eine Datenverarbeitungsumgebung

In diesem Beispiel wird die P2OnDemand Rechenumgebung gelöscht.

Befehl:

```
aws batch delete-compute-environment --compute-environment P2OnDemand
```

- Einzelheiten zur API finden Sie DeleteComputeEnvironmentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
       DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .build();

       return
getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
           .whenComplete((response, ex) -> {
               if (ex != null) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof BatchException) {
                       throw new RuntimeException(cause);
                   } else {
                       throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                   }
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteComputeEnvironmentin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteJobQueue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteJobQueue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So löschen Sie eine Auftragswarteschlange

In diesem Beispiel wird die GPGPU-Auftragswarteschlange gelöscht.

Befehl:

```
aws batch delete-job-queue --job-queue GPGPU
```

- Einzelheiten zur API finden Sie DeleteJobQueuein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to
delete.
     * @return A CompletableFuture that represents the asynchronous deletion of
the job queue.
     *         The future completes when the job queue has been successfully
deleted or if an error occurs.
```

```
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally
with the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteJobQueuein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeregisterJobDefinition`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterJobDefinition` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So heben Sie die Registrierung einer Auftragsdefinition auf

In diesem Beispiel wird die Registrierung einer Auftragsdefinition namens „sleep10“ aufgehoben.

Befehl:

```
aws batch deregister-job-definition --job-definition sleep10
```

- Einzelheiten zur API finden Sie DeregisterJobDefinitionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deregisters a job definition asynchronously.
    *
    * @param jobdefinition the name of the job definition to be deregistered
    * @return a CompletableFuture that completes when the job definition has
been deregistered
    * or an exception has occurred
    */
   public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
       DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
           .jobDefinition(jobDefinition)
           .build();

       CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Einzelheiten zur API finden Sie DeregisterJobDefinitionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeComputeEnvironments`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeComputeEnvironments` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So beschreiben Sie eine Datenverarbeitungsumgebung

Dieses Beispiel beschreibt die P2OnDemand Rechenumgebung.

Befehl:

```
aws batch describe-compute-environments --compute-environments P2OnDemand
```

Ausgabe:

```
{
    "computeEnvironments": [
        {
            "status": "VALID",
            "serviceRole": "arn:aws:iam::012345678910:role/AWSBatchServiceRole",
            "computeEnvironmentArn": "arn:aws:batch:us-
east-1:012345678910:compute-environment/P2OnDemand",
            "computeResources": {
                "subnets": [
                    "subnet-220c0e0a",
                    "subnet-1a95556d",
                    "subnet-978f6dce"
```

```
                ],
                "tags": {
                    "Name": "Batch Instance - P2OnDemand"
                },
                "desiredvCpus": 48,
                "minvCpus": 0,
                "instanceTypes": [
                    "p2"
                ],
                "securityGroupIds": [
                    "sg-cf5093b2"
                ],
                "instanceRole": "ecsInstanceRole",
                "maxvCpus": 128,
                "type": "EC2",
                "ec2KeyPair": "id_rsa"
            },
            "statusReason": "ComputeEnvironment Healthy",
            "ecsClusterArn": "arn:aws:ecs:us-east-1:012345678910:cluster/
P2OnDemand_Batch_2c06f29d-d1fe-3a49-879d-42394c86effc",
            "state": "ENABLED",
            "computeEnvironmentName": "P2OnDemand",
            "type": "MANAGED"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeComputeEnvironmentsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute
environment, or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot
be null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- Einzelheiten zur API finden Sie DescribeComputeEnvironmentsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeJobQueues`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeJobQueues` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So beschreiben Sie eine Auftragswarteschlange

Dieses Beispiel beschreibt die HighPriority Job-Warteschlange.

Befehl:

```
aws batch describe-job-queues --job-queues HighPriority
```

Ausgabe:

```
{
    "jobQueues": [
        {
            "status": "VALID",
            "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/
HighPriority",
            "computeEnvironmentOrder": [
                {
                    "computeEnvironment": "arn:aws:batch:us-
east-1:012345678910:compute-environment/C4OnDemand",
                    "order": 1
                }
            ],
            "statusReason": "JobQueue Healthy",
            "priority": 1,
            "state": "ENABLED",
            "jobQueueName": "HighPriority"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeJobQueuesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the job queue associated with the specified
compute environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find
the associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name =
getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
```

```
                            logger.info("Job queue ARN associated with the
compute environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- Einzelheiten zur API finden Sie DescribeJobQueuesin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeJobs` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So beschreiben Sie einen Auftrag

Im folgenden Beispiel für `describe-jobs` wird ein Auftrag mit der angegebenen Auftrags-ID beschrieben.

```
aws batch describe-jobs \
    --jobs bcf0b186-a532-4122-842e-2ccab8d54efb
```

Ausgabe:

```
{
    "jobs": [
        {
            "status": "SUBMITTED",
            "container": {
                "mountPoints": [],
                "image": "busybox",
                "environment": [],
                "vcpus": 1,
                "command": [
                    "sleep",
                    "60"
                ],
                "volumes": [],
                "memory": 128,
                "ulimits": []
            },
            "parameters": {},
            "jobDefinition": "arn:aws:batch:us-east-1:012345678910:job-
definition/sleep60:1",
            "jobQueue": "arn:aws:batch:us-east-1:012345678910:job-queue/
HighPriority",
            "jobId": "bcf0b186-a532-4122-842e-2ccab8d54efb",
            "dependsOn": [],
            "jobName": "example",
            "createdAt": 1480483387803
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeJobsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- Einzelheiten zur API finden Sie DescribeJobsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListJobsPaginator`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListJobsPaginator`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously lists the jobs in the specified job queue with the given
job status.
    *
    * @param jobQueue the name of the job queue to list jobs from
    * @return a List<JobSummary> that contains the jobs that succeeded
    */
   public List<JobSummary> listJobsAsync(String jobQueue) {
       if (jobQueue == null || jobQueue.isEmpty()) {
           throw new IllegalArgumentException("Job queue cannot be null or
empty");
       }

       ListJobsRequest listJobsRequest = ListJobsRequest.builder()
           .jobQueue(jobQueue)
           .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
           .build();

       List<JobSummary> jobSummaries = new ArrayList<>();
       ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
       CompletableFuture<Void> future = listJobsPaginator.subscribe(response ->
{
           jobSummaries.addAll(response.jobSummaryList());
       });
       future.join();
```

```
        return jobSummaries;
    }
```

- Einzelheiten zur API finden Sie ListJobsPaginatorin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`RegisterJobDefinition`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterJobDefinition` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So registrieren Sie eine Auftragsdefinition

In diesem Beispiel wird eine Auftragsdefinition für einen einfachen Container-Auftrag registriert.

Befehl:

```
aws batch register-job-definition --job-definition-name sleep30 --type container
 --container-properties '{ "image": "busybox", "vcpus": 1, "memory": 128,
 "command": [ "sleep", "30"]}'
```

Ausgabe:

```
{
    "jobDefinitionArn": "arn:aws:batch:us-east-1:012345678910:job-definition/
sleep30:1",
    "jobDefinitionName": "sleep30",
    "revision": 1
}
```

- Einzelheiten zur API finden Sie RegisterJobDefinitionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Registers a new job definition asynchronously in AWS Batch.
    * <p>
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution
role
    *                         that provides permissions for the containers in
the job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes
exceptionally with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration =
NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
```

```
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- Einzelheiten zur API finden Sie RegisterJobDefinitionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`SubmitJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SubmitJob` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So senden Sie einen Auftrag

In diesem Beispiel wird ein einfacher Container-Job namens example an die HighPriority Auftragswarteschlange gesendet.

Befehl:

```
aws batch submit-job --job-name example --job-queue HighPriority  --job-
definition sleep60
```

Ausgabe:

```
{
    "jobName": "example",
    "jobId": "876da822-4198-45f2-a252-6cea32512ea8"
}
```

- Einzelheiten zur API finden Sie SubmitJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of
the submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName,
String jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture.thenApply(SubmitJobResponse::jobId);
   }
```

- Einzelheiten zur API finden Sie SubmitJobin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UpdateComputeEnvironment`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateComputeEnvironment` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So aktualisieren Sie eine Datenverarbeitungsumgebung

In diesem Beispiel wird die P2OnDemand Rechenumgebung deaktiviert, sodass sie gelöscht werden kann.

Befehl:

```
aws batch update-compute-environment --compute-environment P2OnDemand --
state DISABLED
```

Ausgabe:

```
{
    "computeEnvironmentName": "P2OnDemand",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-
environment/P2OnDemand"
}
```

- Einzelheiten zur API finden Sie UpdateComputeEnvironmentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Disables the specified compute environment asynchronously.
    *
    * @param computeEnvironmentName the name of the compute environment to
disable
    * @return a CompletableFuture that completes when the compute environment is
disabled
    */
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute
environment: " + ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- Einzelheiten zur API finden Sie UpdateComputeEnvironmentin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UpdateJobQueue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateJobQueue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Stapelverarbeitung und serverlosen Containern

CLI

AWS CLI

So aktualisieren Sie eine Auftragswarteschlange

In diesem Beispiel wird eine Auftragswarteschlange deaktiviert, sodass sie gelöscht werden kann.

Befehl:

```
aws batch update-job-queue --job-queue GPGPU --state DISABLED
```

Ausgabe:

```
{
    "jobQueueArn": "arn:aws:batch:us-east-1:012345678910:job-queue/GPGPU",
    "jobQueueName": "GPGPU"
}
```

- Einzelheiten zur API finden Sie UpdateJobQueuein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue
update operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }
```

- Einzelheiten zur API finden Sie UpdateJobQueuein der AWS SDK for Java 2.x API-Referenz.

Szenarien für AWS Batch verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS Batch mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS Batch oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Batch-Verarbeitung und serverlosen Containern

Erste Schritte mit Batch-Verarbeitung und serverlosen Containern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Ausführungsrolle
- Erstellen Sie eine Rechenumgebung
- Erstellen Sie eine Job-Warteschlange
- Erstellen Sie eine Auftragsdefinition
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Batch Fargate Getting Started Script - Security Hardened Version
# This script demonstrates creating AWS Batch resources with Fargate
 orchestration
#

set -euo pipefail  # Exit on any error, undefined variables, and pipe failures

# Configuration
SCRIPT_NAME="batch-fargate-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
RANDOM_SUFFIX=$(openssl rand -hex 6)
COMPUTE_ENV_NAME="batch-fargate-compute-${RANDOM_SUFFIX}"
JOB_QUEUE_NAME="batch-fargate-queue-${RANDOM_SUFFIX}"
JOB_DEF_NAME="batch-fargate-jobdef-${RANDOM_SUFFIX}"
```

```
JOB_NAME="batch-hello-world-${RANDOM_SUFFIX}"
ROLE_NAME="BatchEcsTaskExecutionRole-${RANDOM_SUFFIX}"
TRUST_POLICY_FILE="batch-trust-policy-${RANDOM_SUFFIX}.json"

# Security: Set restrictive umask
umask 0077

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Logging function with sanitization
log() {
    local message="${1//[$'\t\r\n']/}"
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] ${message}" | tee -a "${LOG_FILE}"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources created so far..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error ${LINENO}' ERR

# Validate AWS credentials
validate_aws_credentials() {
    if ! aws sts get-caller-identity &>/dev/null; then
        log "ERROR: AWS credentials are not configured or invalid"
        exit 1
    fi
}

# Function to wait for resource to be ready
wait_for_compute_env() {
    local env_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for compute environment ${env_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
```

```
        local status
        status=$(aws batch describe-compute-environments \
            --compute-environments "${env_name}" \
            --query 'computeEnvironments[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "VALID" ]; then
            log "Compute environment ${env_name} is ready"
            return 0
        elif [ "${status}" = "INVALID" ] || [ "${status}" = "NOT_FOUND" ]; then
            log "ERROR: Compute environment ${env_name} failed to create
 properly"
            return 1
        fi

        log "Compute environment status: ${status}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for compute environment ${env_name}"
    return 1
}

# Function to wait for job queue to be ready
wait_for_job_queue() {
    local queue_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for job queue ${queue_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local state
        state=$(aws batch describe-job-queues \
            --job-queues "${queue_name}" \
            --query 'jobQueues[0].state' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${state}" = "ENABLED" ]; then
            log "Job queue ${queue_name} is ready"
            return 0
        elif [ "${state}" = "DISABLED" ] || [ "${state}" = "NOT_FOUND" ]; then
            log "ERROR: Job queue ${queue_name} failed to create properly"
```

```
            return 1
        fi

        log "Job queue state: ${state}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job queue ${queue_name}"
    return 1
}

# Function to wait for job completion
wait_for_job() {
    local job_id="${1}"
    local max_attempts=120
    local attempt=0

    log "Waiting for job ${job_id} to complete..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-jobs \
            --jobs "${job_id}" \
            --query 'jobs[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "SUCCEEDED" ]; then
            log "Job ${job_id} completed successfully"
            return 0
        elif [ "${status}" = "FAILED" ]; then
            log "ERROR: Job ${job_id} failed"
            return 1
        fi

        log "Job status: ${status}. Waiting 30 seconds..."
        sleep 30
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job ${job_id}"
    return 1
}
```

```
# Function to wait for resource state before deletion
wait_for_resource_state() {
    local resource_type="${1}"
    local resource_name="${2}"
    local expected_state="${3}"
    local max_attempts=30
    local attempt=0

    log "Waiting for ${resource_type} ${resource_name} to reach state:
 ${expected_state}"

    while [ ${attempt} -lt ${max_attempts} ]; do
        local current_state=""

        case "${resource_type}" in
            "JOB_QUEUE")
                current_state=$(aws batch describe-job-queues \
                    --job-queues "${resource_name}" \
                    --query 'jobQueues[0].state' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            "COMPUTE_ENV")
                current_state=$(aws batch describe-compute-environments \
                    --compute-environments "${resource_name}" \
                    --query 'computeEnvironments[0].status' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            *)
                log "WARNING: Unknown resource type: ${resource_type}"
                return 1
                ;;
        esac

        if [ "${current_state}" = "${expected_state}" ]; then
            log "${resource_type} ${resource_name} is now in state:
 ${expected_state}"
            return 0
        fi

        log "${resource_type} ${resource_name} state: ${current_state} (waiting
 for ${expected_state})"
        sleep 10
        ((attempt++))
    done
```

```
    log "WARNING: ${resource_type} ${resource_name} did not reach expected state
 after ${max_attempts} attempts"
    return 1
}

# Cleanup function
cleanup_resources() {
    log "Starting cleanup of created resources..."

    # Clean up in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local resource="${CREATED_RESOURCES[i]}"
        local resource_type
        local resource_name

        resource_type=$(echo "${resource}" | cut -d: -f1)
        resource_name=$(echo "${resource}" | cut -d: -f2-)

        log "Cleaning up ${resource_type}: ${resource_name}"

        case "${resource_type}" in
            "JOB_QUEUE")
                aws batch update-job-queue --job-queue "${resource_name}" --state
 DISABLED 2>/dev/null || true
                wait_for_resource_state "JOB_QUEUE" "${resource_name}" "DISABLED"
 || true
                aws batch delete-job-queue --job-queue "${resource_name}" 2>/dev/
null || true
                ;;
            "COMPUTE_ENV")
                aws batch update-compute-environment --compute-environment
 "${resource_name}" --state DISABLED 2>/dev/null || true
                wait_for_resource_state "COMPUTE_ENV" "${resource_name}"
 "DISABLED" || true
                aws batch delete-compute-environment --compute-environment
 "${resource_name}" 2>/dev/null || true
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name
 "${resource_name}" --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" 2>/dev/null || true
                aws iam delete-role --role-name "${resource_name}" 2>/dev/null ||
 true
```

```
                ;;
            "FILE")
                rm -f "${resource_name}" 2>/dev/null || true
                ;;
            *)
                log "WARNING: Unknown resource type for cleanup:
 ${resource_type}"
                ;;
        esac
    done

    log "Cleanup completed"
}

# Validate input parameters
validate_inputs() {
    if [ -z "${ACCOUNT_ID:-}" ]; then
        log "ERROR: ACCOUNT_ID is not set"
        return 1
    fi

    if [ -z "${DEFAULT_VPC:-}" ]; then
        log "ERROR: DEFAULT_VPC is not set"
        return 1
    fi

    if [ -z "${SUBNETS:-}" ]; then
        log "ERROR: SUBNETS is not set"
        return 1
    fi
}

# Validate container image format
validate_container_image() {
    local image="${1}"

    # Check if image contains any shell metacharacters that could be dangerous
    if [[ "${image}" =~ [';$`|&<>()[]{}\\'] ]]; then
        log "ERROR: Container image contains potentially dangerous characters:
 ${image}"
        return 1
    fi

    # Basic ECR/Docker image format validation
```

```
    if ! [[ "${image}" =~ ^[a-zA-Z0-9._/:-]+$ ]]; then
        log "ERROR: Container image format is invalid: ${image}"
        return 1
    fi

    return 0
}

# Main script execution
main() {
    log "Starting AWS Batch Fargate tutorial script - Security Hardened Version"
    log "Log file: ${LOG_FILE}"

    # Validate AWS credentials
    validate_aws_credentials

    # Get AWS account ID
    log "Getting AWS account ID..."
    local ACCOUNT_ID
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    if [ -z "${ACCOUNT_ID}" ] || [ "${ACCOUNT_ID}" = "None" ]; then
        log "ERROR: Could not retrieve AWS account ID"
        exit 1
    fi
    log "Account ID: ${ACCOUNT_ID}"

    # Get default VPC and subnets
    log "Getting default VPC and subnets..."
    local DEFAULT_VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs \
        --filters "Name=is-default,Values=true" \
        --query 'Vpcs[0].VpcId' \
        --output text)

    if [ "${DEFAULT_VPC}" = "None" ] || [ "${DEFAULT_VPC}" = "null" ] || [ -z
 "${DEFAULT_VPC}" ]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi

    log "Default VPC: ${DEFAULT_VPC}"

    # Get subnets in the default VPC
    local SUBNETS
```

```
    SUBNETS=$(aws ec2 describe-subnets \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" \
        --query 'Subnets[*].SubnetId' \
        --output text)

    if [ -z "${SUBNETS}" ]; then
        log "ERROR: No subnets found in default VPC"
        exit 1
    fi

    # Convert tab/space-separated subnets to JSON array format
    local SUBNET_ARRAY
    SUBNET_ARRAY=$(echo "${SUBNETS}" | tr '\t ' '\n' | sed 's/^/"/;s/$/"/' |
 paste -sd ',' -)
    log "Subnets: ${SUBNETS}"
    log "Subnet array: [${SUBNET_ARRAY}]"

    # Get default security group for the VPC
    local DEFAULT_SG
    DEFAULT_SG=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" "Name=group-
name,Values=default" \
        --query 'SecurityGroups[0].GroupId' \
        --output text)

    if [ "${DEFAULT_SG}" = "None" ] || [ "${DEFAULT_SG}" = "null" ] || [ -z
 "${DEFAULT_SG}" ]; then
        log "ERROR: No default security group found in VPC"
        exit 1
    fi

    log "Default security group: ${DEFAULT_SG}"

    # Step 1: Create IAM execution role
    log "Step 1: Creating IAM execution role..."

    # Create trust policy document with proper escaping
    cat > "${TRUST_POLICY_FILE}" << 'EOFPOLICY'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOFPOLICY
    CREATED_RESOURCES+=("FILE:${TRUST_POLICY_FILE}")

    # Validate trust policy file before using it
    if ! jq empty "${TRUST_POLICY_FILE}" 2>/dev/null; then
        log "ERROR: Trust policy file is not valid JSON"
        exit 1
    fi

    # Create the role
    aws iam create-role \
        --role-name "${ROLE_NAME}" \
        --assume-role-policy-document "file://${TRUST_POLICY_FILE}"
    aws iam tag-role --role-name "${ROLE_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=getting-started-batch-fargate
    CREATED_RESOURCES+=("IAM_ROLE:${ROLE_NAME}")

    # Attach policy
    aws iam attach-role-policy \
        --role-name "${ROLE_NAME}" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"

    log "IAM role created: ${ROLE_NAME}"

    # Wait for IAM role propagation
    log "Waiting for IAM role propagation (15 seconds)..."
    sleep 15

    # Step 2: Create compute environment
    log "Step 2: Creating Fargate compute environment..."

    aws batch create-compute-environment \
        --compute-environment-name "${COMPUTE_ENV_NAME}" \
        --type MANAGED \
        --state ENABLED \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-resources "{
```

```
            \"type\": \"FARGATE\",
            \"maxvCpus\": 256,
            \"subnets\": [${SUBNET_ARRAY}],
            \"securityGroupIds\": [\"${DEFAULT_SG}\"]
        }"
    CREATED_RESOURCES+=("COMPUTE_ENV:${COMPUTE_ENV_NAME}")

    # Wait for compute environment to be ready
    if ! wait_for_compute_env "${COMPUTE_ENV_NAME}"; then
        log "ERROR: Compute environment failed to reach VALID state"
        exit 1
    fi

    # Step 3: Create job queue
    log "Step 3: Creating job queue..."

    aws batch create-job-queue \
        --job-queue-name "${JOB_QUEUE_NAME}" \
        --state ENABLED \
        --priority 900 \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-environment-order "order=1,computeEnvironment=
${COMPUTE_ENV_NAME}"
    CREATED_RESOURCES+=("JOB_QUEUE:${JOB_QUEUE_NAME}")

    # Wait for job queue to be ready
    if ! wait_for_job_queue "${JOB_QUEUE_NAME}"; then
        log "ERROR: Job queue failed to reach ENABLED state"
        exit 1
    fi

    # Step 4: Create job definition
    log "Step 4: Creating job definition..."

    local CONTAINER_IMAGE="busybox:latest"
    validate_container_image "${CONTAINER_IMAGE}"

    aws batch register-job-definition \
        --job-definition-name "${JOB_DEF_NAME}" \
        --type container \
        --platform-capabilities FARGATE \
        --container-properties "{
            \"image\": \"${CONTAINER_IMAGE}\",
            \"resourceRequirements\": [
```

```
                {\"type\": \"VCPU\", \"value\": \"0.25\"},
                {\"type\": \"MEMORY\", \"value\": \"512\"}
            ],
            \"command\": [\"echo\", \"hello world\"],
            \"networkConfiguration\": {
                \"assignPublicIp\": \"DISABLED\"
            },
            \"executionRoleArn\": \"arn:aws:iam::${ACCOUNT_ID}:role/
${ROLE_NAME}\"
        }"

    log "Job definition created: ${JOB_DEF_NAME}"

    # Step 5: Submit job
    log "Step 5: Submitting job..."

    local JOB_ID
    JOB_ID=$(aws batch submit-job \
        --job-name "${JOB_NAME}" \
        --job-queue "${JOB_QUEUE_NAME}" \
        --job-definition "${JOB_DEF_NAME}" \
        --query 'jobId' \
        --output text)

    if [ -z "${JOB_ID}" ] || [ "${JOB_ID}" = "None" ]; then
        log "ERROR: Failed to submit job"
        exit 1
    fi

    log "Job submitted with ID: ${JOB_ID}"

    # Step 6: Wait for job completion and view output
    log "Step 6: Waiting for job completion..."
    if ! wait_for_job "${JOB_ID}"; then
        log "ERROR: Job failed or timed out"
        exit 1
    fi

    # Get log stream name
    log "Getting job logs..."
    local LOG_STREAM
    LOG_STREAM=$(aws batch describe-jobs \
        --jobs "${JOB_ID}" \
```

```
        --query 'jobs[0].attempts[0].taskProperties.containers[0].logStreamName'
 \
        --output text)

    if [ "${LOG_STREAM}" != "None" ] && [ "${LOG_STREAM}" != "null" ] && [ -n
 "${LOG_STREAM}" ]; then
        log "Log stream: ${LOG_STREAM}"
        log "Job output:"
        aws logs get-log-events \
            --log-group-name "/aws/batch/job" \
            --log-stream-name "${LOG_STREAM}" \
            --query 'events[*].message' \
            --output text 2>/dev/null | tee -a "${LOG_FILE}" || true
    else
        log "No log stream available for job"
    fi

    log "Tutorial completed successfully!"

    # Show created resources
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${resource}"
    done
    echo ""
    echo "============================================"
    echo "CLEANUP"
    echo "============================================"

    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateComputeEnvironment

- CreateJobQueue
- DeleteComputeEnvironment
- DeleteJobQueue
- DescribeComputeEnvironments
- DescribeJobQueues
- DescribeJobs
- RegisterJobDefinition
- SubmitJob
- UpdateComputeEnvironment
- UpdateJobQueue

Codebeispiele für Amazon Bedrock mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Bedrock mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Bedrock-Benutzerhandbuch – Weitere Informationen zu Amazon Bedrock.
- API-Referenz zu Amazon Bedrock – Details zu allen verfügbaren Aktionen von Amazon Bedrock.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Bedrock AWS SDKs
  - Hello Amazon Bedrock

- Aktionen für die Verwendung von Amazon Bedrock AWS SDKs
  - Verwenden Sie GetFoundationModelmit einem AWS SDK
  - Verwenden Sie mit ListFoundationModelseinem AWS SDK
- Szenarien für die Verwendung von Amazon Bedrock AWS SDKs
  - Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Grundlegende Beispiele für die Verwendung von Amazon Bedrock AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Bedrock mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Bedrock
- Aktionen für die Verwendung von Amazon Bedrock AWS SDKs
  - Verwenden Sie GetFoundationModelmit einem AWS SDK
  - Verwenden Sie mit ListFoundationModelseinem AWS SDK

Hello Amazon Bedrock

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Bedrock.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.Bedrock;
using Amazon.Bedrock.Model;
```

```
namespace BedrockActions;

/// <summary>
/// This example shows how to list foundation models.
/// </summary>
internal class HelloBedrock
{
    /// <summary>
    /// Main method to call the ListFoundationModelsAsync method.
    /// </summary>
    /// <param name="args"> The command line arguments. </param>
    static async Task Main(string[] args)
    {
        // Specify a region endpoint where Amazon Bedrock is available. For
 a list of supported region see https://docs.aws.amazon.com/bedrock/latest/
userguide/what-is-bedrock.html#bedrock-regions
        AmazonBedrockClient bedrockClient = new(RegionEndpoint.USWest2);

        await ListFoundationModelsAsync(bedrockClient);

    }


    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
 bedrockClient)
    {
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
 ListFoundationModelsRequest()
            {
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
```

```
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }


    /// <summary>
    /// Write the foundation model summary to console.
    /// </summary>
    /// <param name="foundationModel"> The foundation model summary to write to
 console. </param>
    private static void WriteToConsole(FoundationModelSummary foundationModel)
    {
        Console.WriteLine($"{foundationModel.ModelId}, Customization:
 {string.Join(", ", foundationModel.CustomizationsSupported)}, Stream:
 {foundationModel.ResponseStreamingSupported}, Input: {string.Join(",
 ", foundationModel.InputModalities)}, Output: {string.Join(", ",
 foundationModel.OutputModalities)}");
    }
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrock"
)

const region = "us-east-1"

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock client and
// list the available foundation models in your account and the chosen region.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(region))
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 bedrockClient := bedrock.NewFromConfig(sdkConfig)
 result, err := bedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})
 if err != nil {
  fmt.Printf("Couldn't list foundation models. Here's why: %v\n", err)
  return
```

```
 }
 if len(result.ModelSummaries) == 0 {
  fmt.Println("There are no foundation models.")
 }
 for _, modelSummary := range result.ModelSummaries {
  fmt.Println(*modelSummary.ModelId)
 }
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

const REGION = "us-east-1";
const client = new BedrockClient({ region: REGION });

export const main = async () => {
  const command = new ListFoundationModelsCommand({});

  const response = await client.send(command);
  const models = response.modelSummaries;

  console.log("Listing the available Bedrock foundation models:");
```

```
  for (const model of models) {
    console.log("=".repeat(42));
    console.log(` Model: ${model.modelId}`);
    console.log("-".repeat(42));
    console.log(` Name: ${model.modelName}`);
    console.log(` Provider: ${model.providerName}`);
    console.log(` Model ARN: ${model.modelArn}`);
    console.log(` Input modalities: ${model.inputModalities}`);
    console.log(` Output modalities: ${model.outputModalities}`);
    console.log(` Supported customizations: ${model.customizationsSupported}`);
    console.log(` Supported inference types: ${model.inferenceTypesSupported}`);
    console.log(` Lifecycle status: ${model.modelLifecycle.status}`);
    console.log(`${"=".repeat(42)}\n`);
  }

  const active = models.filter(
    (m) => m.modelLifecycle.status === "ACTIVE",
  ).length;
  const legacy = models.filter(
    (m) => m.modelLifecycle.status === "LEGACY",
  ).length;

  console.log(
    `There are ${active} active and ${legacy} legacy foundation models in
 ${REGION}.`,
  );

  return response;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Lists the available Amazon Bedrock models.
"""
import logging
import json
import boto3


from botocore.exceptions import ClientError


logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def list_foundation_models(bedrock_client):
    """
    Gets a list of available Amazon Bedrock foundation models.

    :return: The list of available bedrock foundation models.
    """

    try:
        response = bedrock_client.list_foundation_models()
        models = response["modelSummaries"]
        logger.info("Got %s foundation models.", len(models))
        return models

    except ClientError:
        logger.error("Couldn't list foundation models.")
        raise
```

```
def main():
    """Entry point for the example. Uses the AWS SDK for Python (Boto3)
    to create an Amazon Bedrock client. Then lists the available Bedrock models
    in the region set in the callers profile and credentials.
    """

    bedrock_client = boto3.client(service_name="bedrock")

    fm_models = list_foundation_models(bedrock_client)
    for model in fm_models:
        print(f"Model: {model['modelName']}")
        print(json.dumps(model, indent=2))
        print("---------------------------\n")

    logger.info("Done.")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ListFoundationModelsin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import Foundation
```

```
import AWSBedrock

struct ExampleCommand: ParsableCommand {
    static var configuration = CommandConfiguration(
        commandName: "ListFoundationModels",
        abstract: """
        This example demonstrates how to retrieve a list of the available
        foundation models from Amazon Bedrock.
        """,
        discussion: """
        """
    )

    /// Construct a string listing the specified modalities.
    ///
    /// - Parameter modalities: An array of the modalities to list.
    ///
    /// - Returns: A string with a human-readable list of modalities.
    func buildModalityList(modalities: [BedrockClientTypes.ModelModality]?) ->
 String {
        var first = true
        var str = ""

        if modalities == nil {
            return "<none>"
        }

        for modality in modalities! {
            if !first {
                str += ", "
            }
            first = false
            str += modality.rawValue
        }

        return str
    }

    /// Construct a string listing the specified customizations.
    ///
    /// - Parameter customizations: An array of the customizations to list.
    ///
    /// - Returns: A string listing the customizations.
```

```
    func buildCustomizationList(customizations:
 [BedrockClientTypes.ModelCustomization]?) -> String {
        var first = true
        var str = ""

        if customizations == nil {
            return "<none>"
        }

        for customization in customizations! {
            if !first {
                str += ", "
            }
            first = false
            str += customization.rawValue
        }

        return str
    }

    /// Construct a string listing the specified inferences.
    ///
    /// - Parameter inferences: An array of inferences to list.
    ///
    /// - Returns: A string listing the specified inferences.
    func buildInferenceList(inferences: [BedrockClientTypes.InferenceType]?) ->
 String {
        var first = true
        var str = ""

        if inferences == nil {
            return "<none>"
        }

        for inference in inferences! {
            if !first {
                str += ", "
            }
            first = false
            str += inference.rawValue
        }

        return str
    }
```

```
    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region:
 "us-east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
        }

        print("\(summaries.count) models available.")
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der API-Referenz zum AWS SDK für Swift.

Aktionen für die Verwendung von Amazon Bedrock AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Bedrock-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Bedrock-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Bedrock AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Bedrock.

Beispiele

- Verwenden Sie GetFoundationModelmit einem AWS SDK
- Verwenden Sie mit ListFoundationModelseinem AWS SDK

Verwenden Sie **`GetFoundationModel`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetFoundationModel` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erfahren Sie mehr über ein Basismodell mithilfe des synchronen Amazon-Bedrock-Clients.

```
   /**
    * Get details about an Amazon Bedrock foundation model.
    *
    * @param bedrockClient   The service client for accessing Amazon Bedrock.
    * @param modelIdentifier The model identifier.
    * @return An object containing the foundation model's details.
    */
   public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
       try {
           GetFoundationModelResponse response =
bedrockClient.getFoundationModel(
                   r -> r.modelIdentifier(modelIdentifier)
           );

           FoundationModelDetails model = response.modelDetails();

           System.out.println(" Model ID:                     " +
model.modelId());
           System.out.println(" Model ARN:                    " +
model.modelArn());
           System.out.println(" Model Name:                   " +
model.modelName());
           System.out.println(" Provider Name:                " +
model.providerName());
           System.out.println(" Lifecycle status:             " +
model.modelLifecycle().statusAsString());
           System.out.println(" Input modalities:             " +
model.inputModalities());
           System.out.println(" Output modalities:            " +
model.outputModalities());
```

```
            System.out.println(" Supported customizations:     " +
model.customizationsSupported());
            System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ValidationException e) {
            throw new IllegalArgumentException(e.getMessage());
        } catch (SdkException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Erfahren Sie mehr über ein Basismodell mithilfe des asynchronen Amazon-Bedrock-Clients.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The async service client for accessing Amazon
Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
        try {
            CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = future.get().modelDetails();

            System.out.println(" Model ID:                      " +
model.modelId());
            System.out.println(" Model ARN:                     " +
model.modelArn());
```

```
            System.out.println(" Model Name:                    " +
model.modelName());
            System.out.println(" Provider Name:                 " +
model.providerName());
            System.out.println(" Lifecycle status:              " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:              " +
model.inputModalities());
            System.out.println(" Output modalities:             " +
model.outputModalities());
            System.out.println(" Supported customizations:      " +
model.customizationsSupported());
            System.out.println(" Supported inference types:     " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

            return model;

        } catch (ExecutionException e) {
            if (e.getMessage().contains("ValidationException")) {
                throw new IllegalArgumentException(e.getMessage());
            } else {
                System.err.println(e.getMessage());
                throw new RuntimeException(e);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- Einzelheiten zur API finden Sie GetFoundationModelin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Details über ein Basismodell ab.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  GetFoundationModelCommand,
} from "@aws-sdk/client-bedrock";

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @return {FoundationModelDetails} - The list of available bedrock foundation
 models.
 */
export const getFoundationModel = async () => {
  const client = new BedrockClient();

  const command = new GetFoundationModelCommand({
    modelIdentifier: "amazon.titan-embed-text-v1",
  });

  const response = await client.send(command);

  return response.modelDetails;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const model = await getFoundationModel();
  console.log(model);
}
```

- Einzelheiten zur API finden Sie GetFoundationModelin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Details über ein Basismodell ab.

```
    def get_foundation_model(self, model_identifier):
        """
        Get details about an Amazon Bedrock foundation model.

        :return: The foundation model's details.
        """

        try:
            return self.bedrock_client.get_foundation_model(
                modelIdentifier=model_identifier
            )["modelDetails"]
        except ClientError:
            logger.error(
                f"Couldn't get foundation models details for {model_identifier}"
            )
            raise
```

- Einzelheiten zur API finden Sie GetFoundationModelin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListFoundationModels`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListFoundationModels` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Bedrock-Basismodelle.

```
   /// <summary>
   /// List foundation models.
   /// </summary>
   /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
   private static async Task ListFoundationModelsAsync(AmazonBedrockClient
bedrockClient)
   {
       Console.WriteLine("List foundation models with no filter.");

       try
       {
           var response = await bedrockClient.ListFoundationModelsAsync(new
ListFoundationModelsRequest()
           {
           });

           if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
           {
               foreach (var fm in response.ModelSummaries)
               {
                   WriteToConsole(fm);
               }
           }
           else
           {
               Console.WriteLine("Something wrong happened");
```

```
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Bedrock-Basismodelle.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/bedrock"
 "github.com/aws/aws-sdk-go-v2/service/bedrock/types"
)

// FoundationModelWrapper encapsulates Amazon Bedrock actions used in the
 examples.
// It contains a Bedrock service client that is used to perform foundation model
 actions.
type FoundationModelWrapper struct {
 BedrockClient *bedrock.Client
}
```

```
// ListPolicies lists Bedrock foundation models that you can use.
func (wrapper FoundationModelWrapper) ListFoundationModels(ctx context.Context)
 ([]types.FoundationModelSummary, error) {

 var models []types.FoundationModelSummary

 result, err := wrapper.BedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})

 if err != nil {
  log.Printf("Couldn't list foundation models. Here's why: %v\n", err)
 } else {
  models = result.ModelSummaries
 }
 return models, err
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die verfügbaren Amazon-Bedrock-Basismodelle mithilfe des synchronen Amazon-Bedrock-Clients auf.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The service client for accessing Amazon Bedrock.
```

```
     * @return A list of objects containing the foundation models' details
     */
    public static List<FoundationModelSummary> listFoundationModels(BedrockClient
 bedrockClient) {

        try {
            ListFoundationModelsResponse response =
 bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = response.modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
 region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (SdkClientException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

Listen Sie die verfügbaren Amazon-Bedrock-Basismodelle mithilfe des Amazon-Bedrock-Clients auf.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The async service client for accessing Amazon
 Bedrock.
     * @return A list of objects containing the foundation models' details
     */
```

```
   public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
       try {
           CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

           List<FoundationModelSummary> models = future.get().modelSummaries();

           if (models.isEmpty()) {
               System.out.println("No available foundation models in " +
region.toString());
           } else {
               for (FoundationModelSummary model : models) {
                   System.out.println("Model ID: " + model.modelId());
                   System.out.println("Provider: " + model.providerName());
                   System.out.println("Name:     " + model.modelName());
                   System.out.println();
               }
           }

           return models;

       } catch (InterruptedException e) {
           Thread.currentThread().interrupt();
           System.err.println(e.getMessage());
           throw new RuntimeException(e);
       } catch (ExecutionException e) {
           System.err.println(e.getMessage());
           throw new RuntimeException(e);
       }
   }
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die verfügbaren Basismodelle auf.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

/**
 * List the available Amazon Bedrock foundation models.
 *
 * @return {FoundationModelSummary[]} - The list of available bedrock foundation
 models.
 */
export const listFoundationModels = async () => {
  const client = new BedrockClient();

  const input = {
    // byProvider: 'STRING_VALUE',
    // byCustomizationType: 'FINE_TUNING' || 'CONTINUED_PRE_TRAINING',
    // byOutputModality: 'TEXT' || 'IMAGE' || 'EMBEDDING',
    // byInferenceType: 'ON_DEMAND' || 'PROVISIONED',
  };

  const command = new ListFoundationModelsCommand(input);

  const response = await client.send(command);

  return response.modelSummaries;
};
```

```
// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const models = await listFoundationModels();
  console.log(models);
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("==========================================")
            println(" Model ID: ${model.modelId}")
            println("------------------------------------------")
            println(" Name: ${model.modelName}")
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations:
 ${model.customizationsSupported}")
            println(" Supported inference types:
 ${model.inferenceTypesSupported}")
            println("------------------------------------------\n")
        }
        return response.modelSummaries
```

```
    }
}
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
    public function listFoundationModels()
    {
        $bedrockClient = new BedrockClient([
            'region' => 'us-west-2',
            'profile' => 'default'
        ]);
        $response = $bedrockClient->listFoundationModels();
        return $response['modelSummaries'];
    }
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
    def list_foundation_models(self):
        """
        List the available Amazon Bedrock foundation models.

        :return: The list of available bedrock foundation models.
        """

        try:
            response = self.bedrock_client.list_foundation_models()
            models = response["modelSummaries"]
            logger.info("Got %s foundation models.", len(models))
            return models

        except ClientError:
            logger.error("Couldn't list foundation models.")
            raise
```

- Einzelheiten zur API finden Sie ListFoundationModelsin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSBedrock

        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region:
 "us-east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
```

```
        }
```

- Einzelheiten zur API finden Sie ListFoundationModelsin der API-Referenz zum AWS SDK für Swift.

Szenarien für die Verwendung von Amazon Bedrock AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Bedrock mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Bedrock oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

Python

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.

- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finden Sie das aktuelle Repository mit den meisten Trends GitHub und fassen Sie es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step Functions

Codebeispiele für Amazon Bedrock Runtime mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Bedrock Runtime mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zu Amazon Bedrock Runtime – Weitere Informationen zu Amazon Bedrock Runtime.
- API-Referenz zu Amazon Bedrock Runtime – Details zu allen verfügbaren Aktionen von Amazon Bedrock Runtime.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Bedrock Runtime AWS SDKs
  - Hallo Amazon Bedrock Runtime
- Szenarien für die Verwendung von Amazon Bedrock Runtime AWS SDKs
  - Erstellen Sie eine Beispielanwendung, die Spielplätze für die Interaktion mit Amazon Bedrock Foundation-Modellen anbietet, indem Sie eine AWS SDK
  - Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK
  - Generieren Sie Videos aus Textanfragen mit Amazon Bedrock und Nova-Reel
  - Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions
  - Ein Anwendungsbeispiel, das veranschaulicht, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können
- Amazon Nova für Amazon Bedrock Runtime
  - Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock
  - Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
  - Senden und Verarbeiten eines Dokuments mit Amazon Nova in Amazon Bedrock
  - Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können
- Amazon Nova Canvas für Amazon Bedrock Runtime
  - Aufrufen von Amazon Nova Canvas in Amazon Bedrock, um ein Bild zu generieren
- Amazon Nova Reel für Amazon Bedrock Runtime

- Verwenden von Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren
- Amazon Titan Image Generator für Amazon Bedrock Runtime
  - Aufrufen von Amazon Titan Image in Amazon Bedrock, um ein Bild zu generieren
- Amazon Titan Text für Amazon Bedrock Runtime
  - Aufrufen von Modellen von Amazon Titan Text in Amazon Bedrock mithilfe der Invoke-Model-API
- Amazon Titan Text Embeddings für Amazon Bedrock Runtime
  - Aufrufen von Amazon Titan Text Embeddings in Amazon Bedrock
- Anthropic Claude für Amazon Bedrock Runtime
  - Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock
  - Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
  - Senden und Verarbeiten eines Dokuments mit Anthropic Claude in Amazon Bedrock
  - Verwenden der Argumentationsfunktion von Anthropic Claude 3.7 Sonnet in Amazon Bedrock
  - Verwenden der Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock
  - Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können
- Cohere Command für Amazon Bedrock Runtime
  - Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock
  - Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
  - Senden und Verarbeiten eines Dokuments mit Cohere-Command-Modellen in Amazon Bedrock
  - Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können
- DeepSeek für Amazon Bedrock Runtime
  - Senden und Verarbeiten eines Dokuments mit DeepSeek auf Amazon Bedrock
- Meta Lama für Amazon Bedrock Runtime
  - Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock
  - Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
  - Senden und Verarbeiten eines Dokuments mit Llama in Amazon Bedrock
- Mistral AI für Amazon Bedrock Runtime

- Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Mistral-Modellen in Amazon Bedrock
- Stabiler Image-Core für Amazon Bedrock Runtime
  - Rufen Sie Stability.ai Stable Image Core auf Amazon Bedrock auf, um ein Bild zu generieren

Grundlegende Beispiele für die Verwendung von Amazon Bedrock Runtime AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Bedrock Runtime mit AWS SDKs verwendet werden.

Beispiele

- Hallo Amazon Bedrock Runtime

Hallo Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon Bedrock Runtime beginnen können.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;
```

```
/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the
 request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
```

```
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;

/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);
```

```
        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the
 request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "flag"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock Runtime client
// and invokes Anthropic Claude using the Converse API.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {

 region := flag.String("region", "us-east-1", "The AWS region")
 flag.Parse()

 fmt.Printf("Using AWS region: %s\n", *region)

 ctx := context.Background()
```

```
sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(*region))
if err != nil {
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}

client := bedrockruntime.NewFromConfig(sdkConfig)

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
modelId := "global.anthropic.claude-haiku-4-5-20251001-v1:0"

// Start a conversation with the user message.
prompt := "Hello. In a short paragraph, explain what you can do."

message := types.Message{
 Content: []types.ContentBlock{
  &types.ContentBlockMemberText{Value: prompt},
 },
 Role: types.ConversationRoleUser,
}

fmt.Printf("Model: %s\n", modelId)
fmt.Printf("Prompt: %s\n\n", prompt)

result, err := client.Converse(ctx, &bedrockruntime.ConverseInput{
 ModelId:  aws.String(modelId),
 Messages: []types.Message{message},
})

if err != nil {
 log.Fatalf("ERROR: Can't invoke '%s'. Reason: %v\n", modelId, err)
}

// Extract and print the response text.
responseMessage, ok := result.Output.(*types.ConverseOutputMemberMessage)
if !ok {
 log.Fatal("ERROR: Unexpected output type from Converse API")
}
responseContentBlock := responseMessage.Value.Content[0]
text, ok := responseContentBlock.(*types.ContentBlockMemberText)
```

```
 if !ok {
  log.Fatal("ERROR: Unexpected content block type from Converse API")
 }
 fmt.Printf("Response: %s\n", text.Value)
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Send a prompt to Amazon Bedrock using the Converse API.

const AWS_REGION = "us-east-1";

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
const MODEL_ID = "global.anthropic.claude-haiku-4-5-20251001-v1:0";
const PROMPT = "Hi. In a short paragraph, explain what you can do.";

const hello = async () => {
  console.log("=".repeat(35));
  console.log("Welcome to the Amazon Bedrock demo!");
  console.log("=".repeat(35));
```

```
  console.log("Model: Anthropic Claude Haiku");
  console.log(`Prompt: ${PROMPT}\n`);
  console.log("Invoking model...\n");

  // Create a new Bedrock Runtime client instance.
  const client = new BedrockRuntimeClient({ region: AWS_REGION });

  // Create the command with the model ID, the user message, and a basic
 configuration.
  const command = new ConverseCommand({
    modelId: MODEL_ID,
    messages: [
      {
        role: "user",
        content: [{ text: PROMPT }],
      },
    ],
  });

  // Send the command to the model and wait for the response.
  try {
    const response = await client.send(command);

    // Extract and print the response text.
    const responseText = response.output.message.content[0].text;
    console.log(`Response: ${responseText}`);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "BedrockRuntimeServiceException"
    ) {
      console.error(
        `ERROR: Can't invoke '${MODEL_ID}'. Reason: ${caught.message}`,
      );
      throw caught;
    }
    throw caught;
  }
};

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await hello();
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Szenarien für die Verwendung von Amazon Bedrock Runtime AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Bedrock Runtime mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Bedrock Runtime oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Beispielanwendung, die Spielplätze für die Interaktion mit Amazon Bedrock Foundation-Modellen anbietet, indem Sie eine AWS SDK
- Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK
- Generieren Sie Videos aus Textanfragen mit Amazon Bedrock und Nova-Reel
- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions
- Ein Anwendungsbeispiel, das veranschaulicht, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Erstellen Sie eine Beispielanwendung, die Spielplätze für die Interaktion mit Amazon Bedrock Foundation-Modellen anbietet, indem Sie eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Playgrounds erstellen können, um über verschiedene Modalitäten mit Amazon-Bedrock-Basismodellen zu interagieren.

.NET

SDK für .NET

.NET Basismodell (FM) Playground ist eine .NET MAUI Blazor-Beispielanwendung, die zeigt, wie Amazon Bedrock aus C#-Code verwendet werden kann. Dieses Beispiel zeigt, wie .NET- und C#-Entwickler Amazon Bedrock verwenden können, um generative AI-enabled Anwendungen zu erstellen. Sie können die Amazon-Bedrock-Basismodelle mithilfe der folgenden vier Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Voice-Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Java

SDK für Java 2.x

Der Java Foundation Model (FM) Playground ist eine Spring-Boot-Beispielanwendung, die zeigt, wie Amazon Bedrock mit Java verwendet wird. Dieses Beispiel zeigt, wie Java-Entwickler Amazon Bedrock verwenden können, um generative AI-enabled Anwendungen zu erstellen. Sie können die Amazon-Bedrock-Basismodelle in den folgenden drei Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHubProjekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Python

SDK für Python (Boto3)

Der Python Foundation Model (FM) Playground ist eine Python/FastAPI Beispielanwendung, die zeigt, wie Amazon Bedrock mit Python verwendet wird. Dieses Beispiel zeigt, wie Python-Entwickler Amazon Bedrock verwenden können, um generative AI-enabled Anwendungen zu erstellen. Sie können die Amazon-Bedrock-Basismodelle in den folgenden drei Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHubProjekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines verwalteten Prompts.
- Erstellen einer Version des Prompts.
- Aufrufen des Prompts mithilfe der Version.
- Bereinigen der Ressourcen (optional).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines verwalteten Prompts.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
```

```
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of
songs: {{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
```

```
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise
```

```
def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
```

```
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Generieren Sie Videos aus Textanfragen mit Amazon Bedrock und Nova-Reel

Das folgende Codebeispiel zeigt, wie eine Spring Boot-App mithilfe von Amazon Bedrock und dem Modell Videos aus Textanfragen generiert. Nova-Reel

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie Videos aus Textanfragen mit Amazon Bedrock und. Nova-Reel

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
                        .build())
                .build();

        try (BedrockRuntimeAsyncClient bedrockClient =
 getBedrockRuntimeAsyncClient()) {
            CompletableFuture<StartAsyncInvokeResponse>
 startAsyncInvokeResponseCompletableFuture =
 bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
 startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
 startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
```

```
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());

            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
```

```
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetAsyncInvoke
  - StartAsyncInvoke

Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

Python

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finden Sie das aktuelle Repository mit den meisten Trends GitHub und fassen Sie es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step Functions

Ein Anwendungsbeispiel, das veranschaulicht, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Die folgenden Codebeispiele zeigen, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen

zwischen der KI und der Außenwelt zu vermitteln. Es wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell verwendet, damit dieses auf Grundlage der Eingaben des Benutzenden Wetterinformationen in Echtzeit bereitstellen kann.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
```

```
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve
 current weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and
 refer to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up
 information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your
 purpose.
        - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in
 Seattle?";
```

```
    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content =
new List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);
```

```
            // Repeat the loop until the user decides to exit the application
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
 message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
 SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
 return the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
 system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
 necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
 Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
 allowed.</param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
 modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
 infinite loop
```

```
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();
```

```
        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }
```

```
    /// <summary>
    /// Invokes the specified tool with the given payload and returns the tool's
response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input
data.</param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
_weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
new { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
weather info request:")
    {
```

```
        var userInput = default_prompt;
        if (_interactive)
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of
a city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?
```

```
        To exit the program, simply type 'x' and press Enter.

        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
```

```
        Console.WriteLine(message);
        Console.WriteLine();
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-
schema/reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();
```

```
        toolSpecification.Name = "Weather_Tool";
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the
 location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";
```

```
        try
        {
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data:
 {Message}", e.Message);
            throw;
        }
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
```

```
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
```

```
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
```

```
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for
 user-specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before
 each step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
 ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
 Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
 Bedrock.
     */
```

```
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
```

```
    */
   private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
       if (maxRecursion <= 0) {
           // Stop the process, the number of recursive calls could indicate an
infinite loop
           System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
       }

       // Append the model's response to the ongoing conversation
       conversation.add(modelResponse.output().message());

       String modelResponseVal = modelResponse.stopReasonAsString();
       if (modelResponseVal.compareTo("tool_use") == 0) {
           // If the stop reason is "tool_use", forward everything to the tool
use handler
           handleToolUse(modelResponse.output(), conversation, maxRecursion -
1);
       }

       if (modelResponseVal.compareTo("end_turn") == 0) {
           // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
           if (!interactive) {
               defaultPrompt = "x";
           }
       }
   }

   /**
    * Handles the use of a tool by the model in a conversation.
    *
    * @param modelResponse the response from the model, which may include a tool
use request
    * @param conversation  the current conversation, which will be updated with
the tool use results
    * @param maxRecursion  the maximum number of recursive calls allowed to
handle the model's response
    */
   private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
```

```
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new
ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);
```

```
        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + "
with input: " + inputData.get("latitude").toString() + ", " +
inputData.get("longitude").toString() + "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
```

```
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse
sendConversationToBedrockwithSpec(List<Message> conversation, ToolSpecification
toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
```

```
            return getUserInput("\tPlease enter your weather info request, e.g.,
 the name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages,
 have a look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger =
 LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-
schema/reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
 longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
```

```
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on
 its WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
```

```
            HttpResponse<String> response = httpClient.send(request,
HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson,
new TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
```

```
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for
 the converse request
     * @param systemPrompt the system prompt to be included in the converse
 request
     * @param conversation a list of messages representing the conversation
 history
     * @param toolSpec     the specification of the tool to be used in the
 converse request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
```

```
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
 ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: "
 + ex.getMessage(), ex);
        }
    }
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/* Before running this JavaScript code example, set up your development
 environment, including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
```

```
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer
 to it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each
 step.\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information.
 \n" +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
```

```
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before
 sending the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with
 Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
// param "messages" - The conversation history including the next message to
 send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
```

```
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
```

```
// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
```

```
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
```

```
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or
 coordinates." +
    "Weather information will be provided using a custom Tool and open-meteo
 API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
```

```
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look
 at:\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
```

```
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
```

```
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
```

```
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()
```

```
    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
```

```
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
```

```
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
```

```
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
 True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Szenario und die Logik für die Demo. Dies orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
```

```
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }
```

```
        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
```

```
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
```

```
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
```

```
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Dienstprogramme zum Drucken der Nachrichteninhaltsblöcke.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError>
 {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock,
 ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
```

```
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84
 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
```

```
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}

#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
```

```
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Rust.

Amazon Nova für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Amazon Nova in Amazon Bedrock
- Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock

Die folgenden Codebeispiele zeigen, wie eine Textnachricht mithilfe der Converse-API von Bedrock an Amazon Nova gesendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// Use the Converse API to send a text message to Amazon Nova.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
```

```
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

Senden Sie mithilfe der Converse-API von Bedrock und einer Tool-Konfiguration eine Konversation mit Nachrichten an Amazon Nova.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;
```

```
    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
```

```
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
func (wrapper ConverseWrapper) ConverseNova(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "amazon.nova-lite-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock und dem asynchronen Java-Client eine Textnachricht an Amazon Nova.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";
```

```
        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
the user
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
length
                                .temperature(0.5F)  // Using temperature for
randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
client.converse(request);
            return asyncResponse.thenApply(
                    response ->
response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
```

```
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
```

```
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
 the user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
 length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
```

```
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).text();

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converse();
        System.out.println(response);
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure and send a request
// - Process the response

import {
```

```
  BedrockRuntimeClient,
  ConversationRole,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use:
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and
 text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one line.";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
```

```
  },
};

// Step 5: Send and process the request
// - Send the request to the model
// - Extract and return the generated text from the response
try {
  const response = await client.send(new ConverseCommand(request));
  console.log(response.output.message.content[0].text);
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  throw error;
}
```

Senden Sie mithilfe der Converse-API von Bedrock und einer Tool-Konfiguration eine Konversation mit Nachrichten an Amazon Nova.

```
// This example demonstrates how to send a conversation of messages to Amazon
 Nova using Bedrock's Converse API with a tool configuration.
// It shows how to:
// - 1. Set up the Amazon Bedrock runtime client
// - 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response (model ID, user input, system prompt, and the tool
 spec)
// - 3. Send the request to Amazon Bedrock, and returns the response.
// - 4. Add the tool response to the conversation, and send it back to Amazon
 Bedrock.
// - 5. Publish the response.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client

// Credentials will be automatically loaded from the environment
const bedRockRuntimeClient = new BedrockRuntimeClient({
  region: "us-east-1",
});
```

```
// Step 2. Define the parameters required enable Amazon Bedrock to use a tool
 when formulating its response.

// The Bedrock Model ID.
const modelId = "amazon.nova-lite-v1:0";

// The system prompt to help Amazon Bedrock craft it's response.
const system_prompt = [
  {
    text:
      "You are a music expert that provides the most popular song played on a
 radio station, using only the\n" +
      "the top_song tool, which he call sign for the radio station for which you
 want the most popular song. " +
      "Example calls signs are WZPZ and WKRP. \n" +
      "- Only use the top_song tool. Never guess or make up information. \n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
      "- Only respond to queries about the most popular song played on a radio
 station\n" +
      "Remind off-topic users of your purpose. \n" +
      "- Never claim to search online, access external data, or use tools besides
 the top_song tool.\n",
  },
];
//  The user's question.
const message = [
  {
    role: "user",
    content: [{ text: "What is the most popular song on WZPZ?" }],
  },
];
// The tool specification. In this case, it uses an example schema for
// a tool that gets the most popular song played on a radio station.
const tool_config = {
  tools: [
    {
      toolSpec: {
        name: "top_song",
        description: "Get the most popular song played on a radio station.",
        inputSchema: {
          json: {
            type: "object",
            properties: {
```

```
              sign: {
                type: "string",
                description:
                  "The call sign for the radio station for which you want the
 most popular song. Example calls signs are WZPZ and WKRP.",
              },
            },
            required: ["sign"],
          },
        },
      },
    },
  ],
};

// Helper function to return the song and artist from top_song tool.
async function get_top_song(call_sign) {
  try {
    if (call_sign === "WZPZ") {
      const song = "Elemental Hotel";
      const artist = "8 Storey Hike";
      return { song, artist };
    }
  } catch (error) {
    console.log(`${error.message}`);
  }
}

// 3. Send the request to Amazon Bedrock, and returns the response.
export async function SendConversationtoBedrock(
  modelId,
  message,
  system_prompt,
  tool_config,
) {
  try {
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: message,
        system: system_prompt,
        toolConfig: tool_config,
      }),
    );
```

```
    if (response.stopReason === "tool_use") {
      const toolResultFinal = [];
      try {
        const output_message = response.output.message;
        message.push(output_message);
        const toolRequests = output_message.content;
        const toolMessage = toolRequests[0].text;
        console.log(toolMessage.replace(/<[^>]+>/g, ""));
        for (const toolRequest of toolRequests) {
          if (Object.hasOwn(toolRequest, "toolUse")) {
            const toolUse = toolRequest.toolUse;
            const sign = toolUse.input.sign;
            const toolUseID = toolUse.toolUseId;
            console.log(
              `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
            );
            if (toolUse.name === "top_song") {
              const toolResult = [];
              try {
                const top_song = await get_top_song(toolUse.input.sign).then(
                  (top_song) => top_song,
                );
                const toolResult = {
                  toolResult: {
                    toolUseId: toolUseID,
                    content: [
                      {
                        json: { song: top_song.song, artist: top_song.artist },
                      },
                    ],
                  },
                };
                toolResultFinal.push(toolResult);
              } catch (err) {
                const toolResult = {
                  toolUseId: toolUseID,
                  content: [{ json: { text: err.message } }],
                  status: "error",
                };
              }
            }
          }
        }
        const toolResultMessage = {
```

```
          role: "user",
          content: toolResultFinal,
        };
        // Step 4. Add the tool response to the conversation, and send it back to
 Amazon Bedrock.

        message.push(toolResultMessage);
        await SendConversationtoBedrock(
          modelId,
          message,
          system_prompt,
          tool_config,
        );
      } catch (caught) {
        console.error(`${caught.message}`);
        throw caught;
      }
    }

    // 4. Publish the response.
    if (response.stopReason === "end_turn") {
      const finalMessage = response.output.message.content[0].text;
      const messageToPrint = finalMessage.replace(/<[^>]+>/g);
      console.log(messageToPrint.replace(/<[^>]+>/g));
      return messageToPrint;
    }
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        `${caught.name} - Model not ready, please wait and try again.`,
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        `${caught.name} - Error occurred while sending Converse request`,
      );
      throw caught;
    }
  }
}
await SendConversationtoBedrock(modelId, message, system_prompt, tool_config);
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one
 line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
use Aws\Exception\AwsException;
use RuntimeException;

class Converse
{
    public function converse(): string
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
```

```
            // Send the message to the model, using a basic inference
 configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
}

$demo = new Converse();
echo $demo->converse();
```

- Details zur API finden Sie unter Converse in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
# Use the Conversation API to send a text message to Amazon Nova.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Amazon Nova.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-micro-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
```

```
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Swift.

Aufrufen von Amazon Nova in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova senden und den Antwortstream in Echtzeit verarbeiten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Amazon Nova
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
```

```
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
```

```
/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text
 responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy,
 speed, and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from
 the user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
```

```
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response
length
                                .temperature(0.5F)  // Using temperature for
randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of
each chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s",
modelId, err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// This example demonstrates how to use the Amazon Nova foundation models
// to generate streaming text responses.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure a streaming request
// - Process the streaming response

import {
  BedrockRuntimeClient,
  ConversationRole,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";
```

```
// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and
 text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one paragraph";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the streaming request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
  },
};

// Step 5: Send and process the streaming request
```

```
// - Send the request to the model
// - Process each chunk of the streaming response
try {
  const response = await client.send(new ConverseStreamCommand(request));

  for await (const chunk of response.stream) {
    if (chunk.contentBlockDelta) {
      // Print each text chunk as it arrives
      process.stdout.write(chunk.contentBlockDelta.delta?.text || "");
    }
  }
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  process.exitCode = 1;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message
```

```
/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }
```

```
        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Amazon Nova Text
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )
```

```
    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Einzelheiten zur API finden Sie ConverseStreamin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Amazon Nova and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)
```

```
    // Set the model ID.
    let modelId = "amazon.nova-lite-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nNova Lite:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Swift.

Senden und Verarbeiten eines Dokuments mit Amazon Nova in Amazon Bedrock

In dem folgenden Codebeispiele wird gezeigt, wie ein Dokument mit Amazon Nova in Amazon Bedrock gesendet und verarbeitet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Amazon Nova in Amazon Bedrock.

```
# Send and process a document with Amazon Nova on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()
```

```
# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Die folgenden Codebeispiele zeigen, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Es wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell verwendet, damit dieses auf Grundlage der Eingaben des Benutzenden Wetterinformationen in Echtzeit bereitstellen kann.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
```

```
    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve
 current weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and
 refer to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up
 information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your
 purpose.
        - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
```

```
    ;

    private static string default_prompt = "What is the weather like in
Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
```

```
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content =
new List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
modelResponse, List<Message> conversation, int maxRecursion)
    {
```

```
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool
use handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls
allowed.</param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
List<Message> conversation, int maxRecursion)
```

```
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);
```

```
        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
    /// Invokes the specified tool with the given payload and returns the tool's
response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input
data.</param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
_weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
new { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
```

```
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of
a city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
```

```
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.

        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
```

```
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-
schema/reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
```

```
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the
 location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
```

```
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data:
 {Message}", e.Message);
            throw;
        }
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
```

```
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/>
 class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
```

```
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse
 request.");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for
 user-specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before
 each step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
```

```
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
 ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
 Scenario is complete.");
```

```
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has
returned its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }
```

```
   /**
    * Processes the response from the model and updates the conversation
accordingly.
    *
    * @param modelResponse the response from the model
    * @param conversation  the ongoing conversation
    * @param maxRecursion  the maximum number of recursions allowed
    */
   private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
       if (maxRecursion <= 0) {
           // Stop the process, the number of recursive calls could indicate an
infinite loop
           System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
       }

       // Append the model's response to the ongoing conversation
       conversation.add(modelResponse.output().message());

       String modelResponseVal = modelResponse.stopReasonAsString();
       if (modelResponseVal.compareTo("tool_use") == 0) {
           // If the stop reason is "tool_use", forward everything to the tool
use handler
           handleToolUse(modelResponse.output(), conversation, maxRecursion -
1);
       }

       if (modelResponseVal.compareTo("end_turn") == 0) {
           // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
           if (!interactive) {
               defaultPrompt = "x";
           }
       }
   }

   /**
    * Handles the use of a tool by the model in a conversation.
    *
    * @param modelResponse the response from the model, which may include a tool
use request
```

```
    * @param conversation  the current conversation, which will be updated with
the tool use results
    * @param maxRecursion  the maximum number of recursive calls allowed to
handle the model's response
    */
   private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
       List<ContentBlock> toolResults = new ArrayList<>();

       // The model's response can consist of multiple content blocks
       for (ContentBlock contentBlock : modelResponse.message().content()) {
           if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
               // If the content block contains text, print it to the console
               PrintModelResponse(contentBlock.text());
           }

           if (contentBlock.toolUse() != null) {
               ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

               // Add the tool use ID and the tool's response to the list of
results
               List<ToolResultContentBlock> contentBlockList = new
ArrayList<>();
               ToolResultContentBlock block = ToolResultContentBlock.builder()
                       .json(toolResponse.getContent())
                       .build();
               contentBlockList.add(block);

               ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                       .toolUseId(toolResponse.getToolUseId())
                       .content(contentBlockList)
                       .build();

               ContentBlock contentBlock1 = ContentBlock.builder()
                       .toolResult(toolResultBlock)
                       .build();

               toolResults.add(contentBlock1);
           }
       }

       // Embed the tool results in a new user message
       Message message = Message.builder()
               .role(ConversationRole.USER)
```

```
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
```

```
        System.out.println("Invoking tool: " + toolName + "
with input: " + inputData.get("latitude").toString() + ", " +
inputData.get("longitude").toString() + "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse
sendConversationToBedrockwithSpec(List<Message> conversation, ToolSpecification
toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and
return the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
```

```
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., 
the name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We 
hope you
                learned something new, or got some inspiration for your own apps 
today!
```

```
                For more Bedrock examples in different programming languages,
 have a look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger =
 LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-
schema/reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
 longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
```

```
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on
its WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
```

```
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson,
 new TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
```

```
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for
 the converse request
     * @param systemPrompt the system prompt to be included in the converse
 request
     * @param conversation a list of messages representing the conversation
 history
     * @param toolSpec     the specification of the tool to be used in the
 converse request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
```

```
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: "
+ ex.getMessage(), ex);
        }
    }
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/* Before running this JavaScript code example, set up your development
 environment, including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer
 to it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
```

```
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each
 step.\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information.
 \n" +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before
 sending the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with
 Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
```

```
// param "messages" - The conversation history including the next message to
 send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
```

```
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
```

```
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
```

```
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or
 coordinates." +
    "Weather information will be provided using a custom Tool and open-meteo
 API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
```

```
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
```

```
    "For more Bedrock examples in different programming languages, have a look
 at:\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Amazon Nova Canvas für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Amazon Nova Canvas in Amazon Bedrock, um ein Bild zu generieren

Aufrufen von Amazon Nova Canvas in Amazon Bedrock, um ein Bild zu generieren

Die folgenden Codebeispiele zeigen, wie Sie Amazon Nova Canvas in Amazon Bedrock aufrufen, um ein Bild zu generieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas.

using System;
using System.IO;
using System.Text.Json;
using System.Text.Json.Nodes;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);
```

```
// Set the model ID.
var modelId = "amazon.nova-canvas-v1:0";

// Define the image generation prompt for the model.
var prompt = "A stylized picture of a cute old steampunk robot.";

// Create a random seed between 0 and 858,993,459
int seed = new Random().Next(0, 858993460);

//Format the request payload using the model's native structure.
var nativeRequest = JsonSerializer.Serialize(new
{
    taskType = "TEXT_IMAGE",
    textToImageParams = new
    {
        text = prompt
    },
    imageGenerationConfig = new
    {
        seed,
        quality = "standard",
        width = 512,
        height = 512,
        numberOfImages = 1
    }
});

// Create a request with the model ID and the model's native request payload.
var request = new InvokeModelRequest()
{
    ModelId = modelId,
    Body = new MemoryStream(System.Text.Encoding.UTF8.GetBytes(nativeRequest)),
    ContentType = "application/json"
};

try
{
    // Send the request to the Bedrock Runtime and wait for the response.
    var response = await client.InvokeModelAsync(request);

    // Decode the response body.
    var modelResponse = await JsonNode.ParseAsync(response.Body);

    // Extract the image data.
```

```
    var base64Image = modelResponse["images"]?[0].ToString() ?? "";

    // Save the image in a local folder
    string savedPath = AmazonNovaCanvas.InvokeModel.SaveBase64Image(base64Image);
    Console.WriteLine($"Image saved to: {savedPath}");
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;
```

```
/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following
 structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
```

```
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/
images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
import {
  BedrockRuntimeClient,
  InvokeModelCommand,
} from "@aws-sdk/client-bedrock-runtime";
import { saveImage } from "../../utils/image-creation.js";
import { fileURLToPath } from "node:url";

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 *
 * @returns {Promise<string>} Base64-encoded image data
 */
export const invokeModel = async () => {
  // Step 1: Create the Amazon Bedrock runtime client
  // Credentials will be automatically loaded from the environment
  const client = new BedrockRuntimeClient({ region: "us-east-1" });

  // Step 2: Specify which model to use
  // For the latest available models, see:
```

```
  // https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
  const modelId = "amazon.nova-canvas-v1:0";

  // Step 3: Configure the request payload
  // First, set the main parameters:
  // - prompt: Text description of the image to generate
  // - seed: Random number for reproducible generation (0 to 858,993,459)
  const prompt = "A stylized picture of a cute old steampunk robot";
  const seed = Math.floor(Math.random() * 858993460);

  // Then, create the payload using the following structure:
  // - taskType: TEXT_IMAGE (specifies text-to-image generation)
  // - textToImageParams: Contains the text prompt
  // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
  // For a list of available request parameters, see:
  // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
  const payload = {
    taskType: "TEXT_IMAGE",
    textToImageParams: {
      text: prompt,
    },
    imageGenerationConfig: {
      seed,
      quality: "standard",
    },
  };

  // Step 4: Send and process the request
  // - Embed the payload in a request object
  // - Send the request to the model
  // - Extract and return the generated image data from the response
  try {
    const request = {
      modelId,
      body: JSON.stringify(payload),
    };
    const response = await client.send(new InvokeModelCommand(request));

    const decodedResponseBody = new TextDecoder().decode(response.body);
    // The response includes an array of base64-encoded PNG images
    /** @type {{images: string[]}} */
    const responseBody = JSON.parse(decodedResponseBody);
```

```
    return responseBody.images[0]; // Base64-encoded image data
  } catch (error) {
    console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
    throw error;
  }
};

// If run directly, execute the example and save the generated image
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("Generating image. This may take a few seconds...");
  invokeModel()
    .then(async (imageData) => {
      const imagePath = await saveImage(imageData, "nova-canvas");
      // Example path: javascriptv3/example_code/bedrock-runtime/output/nova-
canvas/image-01.png
      console.log(`Image saved to: ${imagePath}`);
    })
    .catch((error) => {
      console.error("Execution failed:", error);
      process.exitCode = 1;
    });
}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
# Use the native inference API to create an image with Amazon Nova Canvas

import base64
import json
```

```
import os
import random

import boto3

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID.
model_id = "amazon.nova-canvas-v1:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed between 0 and 858,993,459
seed = random.randint(0, 858993460)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "seed": seed,
        "quality": "standard",
        "height": 512,
        "width": 512,
        "numberOfImages": 1,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
```

```
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"nova_canvas_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"nova_canvas_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Einzelheiten zur API finden Sie InvokeModelin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas

import AWSBedrockRuntime
import AWSSDKIdentity
import Foundation

struct NovaImageOutput: Decodable {
    let images: [Data]
}

func generateImage(_ textPrompt: String) async throws {
    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
```

```
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    config.awsCredentialIdentityResolver = try SSOAWSCredentialIdentityResolver()

    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-canvas-v1:0"

    //  Format the request payload using the model's native structure.
    let input = InvokeModelInput(
        accept: "application/json",
        body: """
            {
                "textToImageParams": {
                    "text": "\(textPrompt)"
                },
                "taskType": "TEXT_IMAGE",
                "imageGenerationConfig": {
                    "seed": 42,
                    "quality": "standard",
                    "width": 512,
                    "height": 512,
                    "numberOfImages": 1
                }
            }
            """.data(using: .utf8),
        modelId: modelId
    )

    // Invoke the model with the request.
    let response = try await client.invokeModel(input: input)

    // Decode the response body.
    let output = try JSONDecoder().decode(NovaImageOutput.self, from:
response.body!)

    // Extract the image data.
    guard let data = output.images.first else {
        print("No image data found")
        return
    }

    // Save the generated image to a local folder.
    let fileURL = URL.documentsDirectory.appending(path: "nova_canvas.png")
```

```
    print(fileURL)
    try data.write(to: fileURL)
    print("Image is saved at \(fileURL)")
}
```

- Einzelheiten zur API finden Sie InvokeModelin der API-Referenz zum AWS SDK für Swift.

Amazon Nova Reel für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Verwenden von Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren

Verwenden von Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren

Die folgenden Codebeispiele zeigen, wie Amazon Nova Reel zur Generierung eines Videos anhand eines Textprompts verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren.

```
"""
This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.

It shows how to:
- Set up the Amazon Bedrock runtime client
```

```
- Configure a text-to-video request
- Submit an asynchronous job for video generation
- Poll for job completion status
- Access the generated video from S3
"""

import random
import time

import boto3

# Replace with your own S3 bucket to store the generated video
# Format: s3://your-bucket-name
OUTPUT_S3_URI = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAME"


def start_text_to_video_generation_job(bedrock_runtime, prompt, output_s3_uri):
    """
    Starts an asynchronous text-to-video generation job using Amazon Nova Reel.

    :param bedrock_runtime: The Bedrock runtime client
    :param prompt: The text description of the video to generate
    :param output_s3_uri: S3 URI where the generated video will be stored

    :return: The invocation ARN of the async job
    """
    # Specify the model ID for text-to-video generation
    model_id = "amazon.nova-reel-v1:0"

    # Generate a random seed between 0 and 2,147,483,646
    # This helps ensure unique video generation results
    seed = random.randint(0, 2147483646)

    # Configure the video generation request with additional parameters
    model_input = {
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": {"text": prompt},
        "videoGenerationConfig": {
            "fps": 24,
            "durationSeconds": 6,
            "dimension": "1280x720",
            "seed": seed,
        },
    }
```

```
    # Specify the S3 location for the output video
    output_config = {"s3OutputDataConfig": {"s3Uri": output_s3_uri}}

    # Invoke the model asynchronously
    response = bedrock_runtime.start_async_invoke(
        modelId=model_id, modelInput=model_input, outputDataConfig=output_config
    )

    invocation_arn = response["invocationArn"]

    return invocation_arn


def query_job_status(bedrock_runtime, invocation_arn):
    """
    Queries the status of an asynchronous video generation job.

    :param bedrock_runtime: The Bedrock runtime client
    :param invocation_arn: The ARN of the async invocation to check

    :return: The runtime response containing the job status and details
    """
    return bedrock_runtime.get_async_invoke(invocationArn=invocation_arn)


def main():
    """
    Main function that demonstrates the complete workflow for generating
    a video from a text prompt using Amazon Nova Reel.
    """
    # Create a Bedrock Runtime client
    # Note: Credentials will be loaded from the environment or AWS CLI config
    bedrock_runtime = boto3.client("bedrock-runtime", region_name="us-east-1")

    # Configure the text prompt and output location
    prompt = "Closeup of a cute old steampunk robot. Camera zoom in."

    # Verify the S3 URI has been set to a valid bucket
    if "REPLACE-WITH-YOUR-S3-BUCKET-NAME" in OUTPUT_S3_URI:
        print("ERROR: You must replace the OUTPUT_S3_URI with your own S3 bucket
 URI")
        return
```

```
    print("Submitting video generation job...")
    invocation_arn = start_text_to_video_generation_job(
        bedrock_runtime, prompt, OUTPUT_S3_URI
    )
    print(f"Job started with invocation ARN: {invocation_arn}")

    # Poll for job completion
    while True:
        print("\nPolling job status...")
        job = query_job_status(bedrock_runtime, invocation_arn)
        status = job["status"]

        if status == "Completed":
            bucket_uri = job["outputDataConfig"]["s3OutputDataConfig"]["s3Uri"]
            print(f"\nSuccess! The video is available at: {bucket_uri}/
output.mp4")
            break
        elif status == "Failed":
            print(
                f"\nVideo generation failed: {job.get('failureMessage', 'Unknown
 error')}"
            )
            break
        else:
            print("In progress. Waiting 15 seconds...")
            time.sleep(15)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - GetAsyncInvoke
  - StartAsyncInvoke

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren.

```
// This example demonstrates how to use Amazon Nova Reel to generate a video from
 a text prompt.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Configure a text-to-video request
// - Submit an asynchronous job for video generation
// - Poll for job completion status
// - Access the generated video from S3

import AWSBedrockRuntime
import Foundation
import Smithy

func startTextToVideoGenerationJob(
    bedrockRuntimeClient: BedrockRuntimeClient, prompt: String, outputS3Uri:
 String
) async throws -> String? {
    // Specify the model ID for text-to-video generation
    let modelId = "amazon.nova-reel-v1:0"

    // Configure the video generation request with additional parameters
    let modelInputSource: [String: Any] = [
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": [
            "text": "\(prompt)"
        ],
        "videoGenerationConfig": [
            "durationSeconds": 6,
            "fps": 24,
            "dimension": "1280x720",
        ],
```

```
    ]

    let modelInput = try Document.make(from: modelInputSource)

    let input = StartAsyncInvokeInput(
        modelId: modelId,
        modelInput: modelInput,
        outputDataConfig: .s3outputdataconfig(
            BedrockRuntimeClientTypes.AsyncInvokeS3OutputDataConfig(
                s3Uri: outputS3Uri
            )
        )
    )

    // Invoke the model asynchronously
    let output = try await bedrockRuntimeClient.startAsyncInvoke(input: input)
    return output.invocationArn
}

func queryJobStatus(
    bedrockRuntimeClient: BedrockRuntimeClient,
    invocationArn: String?
) async throws -> GetAsyncInvokeOutput {
    try await bedrockRuntimeClient.getAsyncInvoke(
        input: GetAsyncInvokeInput(invocationArn: invocationArn))
}

func main() async throws {
    // Create a Bedrock Runtime client
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Specify the S3 location for the output video
    let bucket = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAM"

    print("Submitting video generation job...")
    let invocationArn = try await startTextToVideoGenerationJob(
        bedrockRuntimeClient: client,
        prompt: "A pomegranate juice in a railway station",
        outputS3Uri: bucket
    )
    print("Job started with invocation ARN: \(String(describing:invocationArn))")
```

```
    // Poll for job completion
    var status: BedrockRuntimeClientTypes.AsyncInvokeStatus?
    var isReady = false
    var hasFailed = false

    while !isReady && !hasFailed {
        print("\nPolling job status...")
        status = try await queryJobStatus(
            bedrockRuntimeClient: client, invocationArn: invocationArn
        ).status
        switch status {
        case .completed:
            isReady = true
            print("Video is ready\nCheck S3 bucket: \(bucket)")
        case .failed:
            hasFailed = true
            print("Something went wrong")
        case .inProgress:
            print("Job is in progress...")
            try await Task.sleep(nanoseconds: 15 * 1_000_000_000)  // 15 seconds
        default:
            isReady = true
        }
    }
}

do {
    try await main()
} catch {
    print("An error occurred: \(error)")
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - GetAsyncInvoke
  - StartAsyncInvoke

Amazon Titan Image Generator für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Amazon Titan Image in Amazon Bedrock, um ein Bild zu generieren

Aufrufen von Amazon Titan Image in Amazon Bedrock, um ein Bild zu generieren

Die folgenden Codebeispiele zeigen, wie Sie Amazon Titan Image in Amazon Bedrock aufrufen, um ein Bild zu generieren.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
import (
 "context"
 "encoding/json"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}
```

```
type TitanImageRequest struct {
 TaskType              string                `json:"taskType"`
 TextToImageParams     TextToImageParams     `json:"textToImageParams"`
 ImageGenerationConfig ImageGenerationConfig `json:"imageGenerationConfig"`
}
type TextToImageParams struct {
 Text string `json:"text"`
}
type ImageGenerationConfig struct {
 NumberOfImages int     `json:"numberOfImages"`
 Quality        string  `json:"quality"`
 CfgScale       float64 `json:"cfgScale"`
 Height         int     `json:"height"`
 Width          int     `json:"width"`
 Seed           int64   `json:"seed"`
}

type TitanImageResponse struct {
 Images []string `json:"images"`
}

// Invokes the Titan Image model to create an image using the input provided
// in the request body.
func (wrapper InvokeModelWrapper) InvokeTitanImage(ctx context.Context, prompt
 string, seed int64) (string, error) {
 modelId := "amazon.titan-image-generator-v2:0"

 body, err := json.Marshal(TitanImageRequest{
  TaskType: "TEXT_IMAGE",
  TextToImageParams: TextToImageParams{
   Text: prompt,
  },
  ImageGenerationConfig: ImageGenerationConfig{
   NumberOfImages: 1,
   Quality:        "standard",
   CfgScale:       8.0,
   Height:         512,
   Width:          512,
   Seed:           seed,
  },
 })
```

```
 if err != nil {
  log.Fatal("failed to marshal", err)
 }

 output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
 &bedrockruntime.InvokeModelInput{
  ModelId:     aws.String(modelId),
  ContentType: aws.String("application/json"),
  Body:        body,
 })

 if err != nil {
  ProcessError(err, modelId)
 }

 var response TitanImageResponse
 if err := json.Unmarshal(output.Body, &response); err != nil {
  log.Fatal("failed to unmarshal", err)
 }

 base64ImageData := response.Images[0]

 return base64ImageData, nil

}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G2.
        var modelId = "amazon.titan-image-generator-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";

        // Define the prompt for the image generation.
```

```
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max.
 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
# Use the native inference API to create an image with Amazon Titan Image
 Generator

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Image Generator G1.
model_id = "amazon.titan-image-generator-v2:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 2147483647)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "numberOfImages": 1,
        "quality": "standard",
```

```
        "cfgScale": 8.0,
        "height": 512,
        "width": 512,
        "seed": seed,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"titan_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"titan_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Einzelheiten zur API finden Sie InvokeModelin AWS SDK for Python (Boto3) API Reference.

Amazon Titan Text für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Modellen von Amazon Titan Text in Amazon Bedrock mithilfe der Invoke-Model-API

Aufrufen von Modellen von Amazon Titan Text in Amazon Bedrock mithilfe der Invoke-Model-API

Das folgende Codebeispiel zeigt, wie mit der Invoke-Model-API eine Textnachricht an Amazon Titan Text gesendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden der API zum Aufrufen eines Modells zum Senden einer Textnachricht.

```
# Use the native inference API to send a text message to Amazon Titan Text.

import boto3
import json

from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Premier.
model_id = "amazon.titan-text-premier-v1:0"

# Define the prompt for the model.
prompt = "Describe the purpose of a 'hello world' program in one line."

# Format the request payload using the model's native structure.
native_request = {
    "inputText": prompt,
    "textGenerationConfig": {
        "maxTokenCount": 512,
```

```
        "temperature": 0.5,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

try:
    # Invoke the model with the request.
    response = client.invoke_model(modelId=model_id, body=request)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract and print the response text.
response_text = model_response["results"][0]["outputText"]
print(response_text)
```

- Einzelheiten zur API finden Sie InvokeModelin AWS SDK for Python (Boto3) API Reference.

Amazon Titan Text Embeddings für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Amazon Titan Text Embeddings in Amazon Bedrock

Aufrufen von Amazon Titan Text Embeddings in Amazon Bedrock

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Beginnen Sie mit der Erstellung Ihrer ersten Einbettung.
- Erstellen Sie Einbettungen, indem Sie die Anzahl der Dimensionen und die Normalisierung konfigurieren (nur V2).

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Ihre erste Einbettung mit Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";
```

```
        // The text to convert into an embedding.
        var inputText = "Please recommend books with a theme similar to the movie
 'Inception'.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
 inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

Rufen Sie Titan Text Embeddings V2 auf, indem Sie die Anzahl der Dimensionen und die Normalisierung konfigurieren.

```
   /**
    * Invoke Amazon Titan Text Embeddings V2 with additional inference
parameters.
    *
    * @param inputText  - The text to convert to an embedding.
    * @param dimensions - The number of dimensions the output embeddings should
have.
    *                   Values accepted by the model: 256, 512, 1024.
    * @param normalize  - A flag indicating whether or not to normalize the
output embeddings.
    * @return The {@link JSONObject} representing the model's response.
    */
   public static JSONObject invokeModel(String inputText, int dimensions,
boolean normalize) {

       // Create a Bedrock Runtime client in the AWS Region of your choice.
       var client = BedrockRuntimeClient.builder()
               .region(Region.US_WEST_2)
               .build();

       // Set the model ID, e.g., Titan Embed Text v2.0.
       var modelId = "amazon.titan-embed-text-v2:0";

       // Create the request for the model.
       var nativeRequest = """
               {
                   "inputText": "%s",
                   "dimensions": %d,
                   "normalize": %b
               }
               """.formatted(inputText, dimensions, normalize);

       // Encode and send the request.
       var response = client.invokeModel(request -> {
           request.body(SdkBytes.fromUtf8String(nativeRequest));
           request.modelId(modelId);
       });

       // Decode the model's response.
       var modelResponse = new JSONObject(response.body().asUtf8String());

       // Extract and print the generated embedding and the input text token
count.
       var embedding = modelResponse.getJSONArray("embedding");
```

```
        var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
        System.out.println("Embedding: " + embedding);
        System.out.println("\nInput token count: " + inputTokenCount);

        // Return the model's native response.
        return modelResponse;
    }
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Ihre erste Einbettung mit Amazon Titan Text Embeddings.

```
# Generate and print an embedding with Amazon Titan Text Embeddings V2.

import boto3
import json

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Embeddings V2.
model_id = "amazon.titan-embed-text-v2:0"

# The text to convert to an embedding.
input_text = "Please recommend books with a theme similar to the movie
 'Inception'."

# Create the request for the model.
native_request = {"inputText": input_text}

# Convert the native request to JSON.
```

```
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the model's native response body.
model_response = json.loads(response["body"].read())

# Extract and print the generated embedding and the input text token count.
embedding = model_response["embedding"]
input_token_count = model_response["inputTextTokenCount"]

print("\nYour input:")
print(input_text)
print(f"Number of input tokens: {input_token_count}")
print(f"Size of the generated embedding: {len(embedding)}")
print("Embedding:")
print(embedding)
```

- Einzelheiten zur API finden Sie InvokeModelin AWS SDK for Python (Boto3) API Reference.

Anthropic Claude für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Anthropic Claude in Amazon Bedrock
- Verwenden der Argumentationsfunktion von Anthropic Claude 3.7 Sonnet in Amazon Bedrock
- Verwenden der Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock
- Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Anthropic Claude senden können.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
```

```
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
import (
 "context"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// ConverseWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke Bedrock.
type ConverseWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



func (wrapper ConverseWrapper) ConverseClaude(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "anthropic.claude-3-haiku-20240307-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil

}
```

- Details zur API finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().getFirst().text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
```

```
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().getFirst().text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
```

```
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
# Use the Conversation API to send a text message to Anthropic Claude.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
```

```
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
```

```
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    match response {
        Ok(output) => {
            let text = get_converse_output_text(output)?;
            println!("{}", text);
            Ok(())
        }
        Err(e) => Err(e
            .as_service_error()
            .map(BedrockConverseError::from)
            .unwrap_or_else(|| BedrockConverseError("Unknown service
 error".into()))),
    }
}

fn get_converse_output_text(output: ConverseOutput) -> Result<String,
 BedrockConverseError> {
    let text = output
        .output()
        .ok_or("no output")?
        .as_message()
        .map_err(|_| "output not a message")?
        .content()
        .first()
        .ok_or("no content in message")?
        .as_text()
        .map_err(|_| "content is not text")?
```

```
        .to_string();
    Ok(text)
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    operation::converse::{ConverseError, ConverseOutput},
    types::{ContentBlock, ConversationRole, Message},
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in
 one line.";

#[derive(Debug)]
struct BedrockConverseError(String);
impl std::fmt::Display for BedrockConverseError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseError {}
impl From<&str> for BedrockConverseError {
    fn from(value: &str) -> Self {
        BedrockConverseError(value.to_string())
    }
}
impl From<&ConverseError> for BedrockConverseError {
    fn from(value: &ConverseError) -> Self {
        BedrockConverseError::from(match value {
            ConverseError::ModelTimeoutException(_) => "Model took too long",
            ConverseError::ModelNotReadyException(_) => "Model is not ready",
            _ => "Unknown",
        })
    }
```

```
}
```

- Details zu API finden Sie unter Converse in der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Anthropic Claude.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
```

```
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Swift.

Aufrufen von Anthropic Claude in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Anthropic Claude senden und den Antwortstrom in Echtzeit verarbeiten können.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
```

```
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
 the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});
```

```
try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
# Use the Conversation API to send a text message to Anthropic Claude
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Einzelheiten zur API finden Sie ConverseStreamin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Bedrock-API eine Textnachricht an Anthropic Claude und streamen Sie Antwort-Token. ConverseStream

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseStreamError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse_stream()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    let mut stream = match response {
        Ok(output) => Ok(output.stream),
        Err(e) => Err(BedrockConverseStreamError::from(
            e.as_service_error().unwrap(),
        )),
    }?;
```

```
    loop {
        let token = stream.recv().await;
        match token {
            Ok(Some(text)) => {
                let next = get_converse_output_text(text)?;
                print!("{}", next);
                Ok(())
            }
            Ok(None) => break,
            Err(e) => Err(e
                .as_service_error()
                .map(BedrockConverseStreamError::from)
                .unwrap_or(BedrockConverseStreamError(
                    "Unknown error receiving stream".into(),
                ))),
        }?
    }

    println!();

    Ok(())
}

fn get_converse_output_text(
    output: ConverseStreamOutputType,
) -> Result<String, BedrockConverseStreamError> {
    Ok(match output {
        ConverseStreamOutputType::ContentBlockDelta(event) => match event.delta()
 {
            Some(delta) => delta.as_text().cloned().unwrap_or_else(|_|
 "".into()),
            None => "".into(),
        },
        _ => "".into(),
    })
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
```

```
    error::ProvideErrorMetadata,
    operation::converse_stream::ConverseStreamError,
    types::{
        error::ConverseStreamOutputError, ContentBlock, ConversationRole,
        ConverseStreamOutput as ConverseStreamOutputType, Message,
    },
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in
 one line.";

#[derive(Debug)]
struct BedrockConverseStreamError(String);
impl std::fmt::Display for BedrockConverseStreamError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseStreamError {}
impl From<&str> for BedrockConverseStreamError {
    fn from(value: &str) -> Self {
        BedrockConverseStreamError(value.into())
    }
}

impl From<&ConverseStreamError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamError) -> Self {
        BedrockConverseStreamError(
            match value {
                ConverseStreamError::ModelTimeoutException(_) => "Model took too
 long",
                ConverseStreamError::ModelNotReadyException(_) => "Model is not
 ready",
                _ => "Unknown",
            }
            .into(),
        )
    }
```

```
}

impl From<&ConverseStreamOutputError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamOutputError) -> Self {
        match value {
            ConverseStreamOutputError::ValidationException(ve) =>
 BedrockConverseStreamError(
                ve.message().unwrap_or("Unknown ValidationException").into(),
            ),
            ConverseStreamOutputError::ThrottlingException(te) =>
 BedrockConverseStreamError(
                te.message().unwrap_or("Unknown ThrottlingException").into(),
            ),
            value => BedrockConverseStreamError(
                value
                    .message()
                    .unwrap_or("Unknown StreamOutput exception")
                    .into(),
            ),
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Anthropic Claude and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }
```

```
    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nAnthropic Claude:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Swift.

Senden und Verarbeiten eines Dokuments mit Anthropic Claude in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie ein Dokument mit Anthropic Claude in Amazon Bedrock gesendet und verarbeitet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und Verarbeiten eines Dokuments mit Anthropic Claude in Amazon Bedrock.

```
# Send and process a document with Anthropic Claude on Amazon Bedrock.
```

```
import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Verwenden der Argumentationsfunktion von Anthropic Claude 3.7 Sonnet in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie die Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock verwenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Argumentationsfunktion von Anthropic Claude 3.7 Sonnet mit dem asynchronen Bedrock-Runtime-Client.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
```

```
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
```

```
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and
 response text
                        for (ContentBlock block : content) {
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

Verwenden Sie die Argumentationsfunktion von Anthropic Claude 3.7 Sonnet mit dem synchronen Bedrock-Runtime-Client.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
```

```
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, text);

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Verwenden der Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie die Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock verwenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Argumentationsfunktion Fähigkeit von Anthropic Claude 3.7 Sonnet, um Streaming-Textantworten zu generieren.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
```

```
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new
AtomicReference<>();

        // Set up the stream handler to processes chunks of the response as they
arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of
each chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
```

```
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s",
modelId, err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Die folgenden Codebeispiele zeigen, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Es wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell verwendet, damit dieses auf Grundlage der Eingaben des Benutzenden Wetterinformationen in Echtzeit bereitstellen kann.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
```

```
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
```

```
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()
```

```
    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
```

```
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
```

```
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
```

```
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
 True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Szenario und die Logik für die Demo. Dies orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
```

```
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }
```

```
        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
```

```
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
```

```
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
```

```
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Dienstprogramme zum Drucken der Nachrichteninhaltsblöcke.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError>
 {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock,
 ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
```

```
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84
 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
```

```
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}

#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
```

```
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Rust.

Cohere Command für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Cohere-Command-Modellen in Amazon Bedrock
- Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command senden können.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
```

```
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
```

```
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
```

```
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
```

```
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
# Use the Conversation API to send a text message to Cohere Command.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Aufrufen von Cohere Command in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command senden und den Antwortstream in Echtzeit verarbeiten.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);
```

```
// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
```

```
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Cohere Command
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Einzelheiten zur API finden Sie ConverseStreamin AWS SDK for Python (Boto3) API Reference.

Senden und Verarbeiten eines Dokuments mit Cohere-Command-Modellen in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie man ein Dokument mit Cohere-Command-Modellen in Amazon Bedrock sendet und verarbeitet.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Cohere-Command-Modellen in Amazon Bedrock.

```
# Send and process a document with Cohere Command models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Command R+.
model_id = "cohere.command-r-plus-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
```

```
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Eine Demonstration zur Verwendung eines Tools, die zeigt, wie KI-Modelle in Amazon Bedrock mit einem benutzerdefinierten Tool oder einer API verbunden werden können

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer

externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and
 a weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-
inference.html
```

```
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.
If the user provides coordinates, infer the approximate location and refer to it
 in your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """
```

```
    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has
returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
```

```
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message
to send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and
return the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try
again."
            )
```

```
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool
use handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text,
and finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation
is sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use
request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the
tool
```

```
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of
results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
```

```
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException
```

```
def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based
 on its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    Returns the weather data or an error message if the request fails.
```

```
    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather":
True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

DeepSeek für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Senden und Verarbeiten eines Dokuments mit DeepSeek auf Amazon Bedrock

Senden und Verarbeiten eines Dokuments mit DeepSeek auf Amazon Bedrock

Das folgende Codebeispiel zeigt, wie ein Dokument mit DeepSeek auf Amazon Bedrock gesendet und verarbeitet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit DeepSeek auf Amazon Bedrock.

```
# Send and process a document with DeepSeek on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. DeepSeek-R1
model_id = "us.deepseek.r1-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
```

```
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 2000, "temperature": 0.3},
    )

    # Extract and print the reasoning and response text.
    reasoning, response_text = "", ""
    for item in response["output"]["message"]["content"]:
        for key, value in item.items():
            if key == "reasoningContent":
                reasoning = value["reasoningText"]["text"]
            elif key == "text":
                response_text = value

    print(f"\nReasoning:\n{reasoning}")
    print(f"\nResponse:\n{response_text}")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Meta Lama für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Llama in Amazon Bedrock

Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama senden können.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Converse API to send a text message to Meta Llama.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
```

```
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
```

```
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie eine Textnachricht an Meta Llama, indem Sie die Converse-API von Bedrock mit dem asynchronen Java-Client verwenden.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
```

```
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
# Use the Conversation API to send a text message to Meta Llama.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Meta Llama.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
```

```
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Swift.

Aufrufen von Meta Llama in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama senden und den Antwortstrom in Echtzeit verarbeiten.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
```

```
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
 the handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});
```

```
try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Meta Llama
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Einzelheiten zur API finden Sie ConverseStreamin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text
 message
// to Meta Llama and print the response stream.

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
```

```
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nMeta Llama:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der API-Referenz zum AWS SDK für Swift.

Senden und Verarbeiten eines Dokuments mit Llama in Amazon Bedrock

Das folgende Codebeispiel zeigt, wie ein Dokument mit Llama in Amazon Bedrock gesendet und verarbeitet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Llama in Amazon Bedrock.

```
# Send and process a document with Llama on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Llama 3.1 8B Instruct.
model_id = "us.meta.llama3-1-8b-instruct-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
```

```
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Mistral AI für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock
- Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream
- Senden und Verarbeiten eines Dokuments mit Mistral-Modellen in Amazon Bedrock

Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock

Die folgenden Codebeispiele zeigen, wie eine Textnachricht mit der Converse-API von Bedrock an Mistral gesendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
```

```
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
```

```
            var responseText =
 response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }

    }

    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
```

```
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
# Use the Conversation API to send a text message to Mistral.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)
```

```
except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

Aufrufen von Mistral in Amazon Bedrock mithilfe der Converse-API von Bedrock mit Antwortstream

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral senden und den Antwortstream in Echtzeit verarbeiten.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);
```

```
// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach
the handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
```

```
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Einzelheiten zur API finden Sie ConverseStreamin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Mistral
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Einzelheiten zur API finden Sie ConverseStreamin AWS SDK for Python (Boto3) API Reference.

Senden und Verarbeiten eines Dokuments mit Mistral-Modellen in Amazon Bedrock

Das folgende Code-Beispiel zeigt, wie Sie ein Dokument mit Mistral-Modellen in Amazon Bedrock senden und verarbeiten.

Python

SDK für Python (Boto3)

> **Note**
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Mistral-Modellen in Amazon Bedrock.

```
# Send and process a document with Mistral models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
```

```
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv,
 and txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Stabiler Image-Core für Amazon Bedrock Runtime

Die folgenden Codebeispiele zeigen, wie Amazon Bedrock Runtime mit AWS SDKs verwendet wird.

Beispiele

- Rufen Sie Stability.ai Stable Image Core auf Amazon Bedrock auf, um ein Bild zu generieren

Rufen Sie Stability.ai Stable Image Core auf Amazon Bedrock auf, um ein Bild zu generieren

Die folgenden Codebeispiele zeigen, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Bild zu generieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Image mit Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to
 run an inference based on the provided input.
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable
 Image Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string
 prompt, int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
```

```
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code "
 + response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für .NET API-Referenz.

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Image mit Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to
 run an inference based on the provided input.
        /// </summary>
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable
 Image Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string
 prompt, int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
```

```
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code "
 + response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
// Create an image with Stability AI Stable Image Core.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Stable Image Core.
        var modelId = "stability.stable-image-core-v1:1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response
 fields at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-stable-image-core-text-image-request-response.html
        var nativeRequestTemplate = """
                {
                    "prompt": "{{prompt}}",
                    "aspect_ratio": "1:1",
                    "seed": {{seed}},
                    "output_format": "png"
                }""";
```

```
        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random seed for the image generation (max. 4,294,967,294).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/images/0")
                    .queryFrom(responseBody)
                    .toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
```

```
}
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
    public function invokeStableDiffusion(string $prompt, int $seed = 0, string
 $aspect_ratio = '1:1')
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available parameters of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-image-core-v1:1';
            $body = [
                'prompt' => $prompt,
                'aspect_ratio' => $aspect_ratio,
                'seed' => $seed,
                'output_format' => 'png',
            ];

            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
```

```
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- Einzelheiten zur API finden Sie InvokeModelin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
# Use the native inference API to create an image with Stability AI Stable Image
 Core

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-west-2")

# Set the model ID, e.g., Stable Image Core.
model_id = "stability.stable-image-core-v1:1"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."
```

```
# Generate a random seed.
seed = random.randint(0, 4294967295)

# Format the request payload using the model's native structure.
native_request = {
    "prompt": prompt,
    "aspect_ratio": "1:1",
    "seed": seed,
    "output_format": "png",
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"stability_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"stability_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- Einzelheiten zur API finden Sie InvokeModelin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
    "Stable Image Core Input Parameters should be in a format like this:
*   {
*     "prompt": "Draw a dolphin with a mustache, photorealistic",
*     "aspect_ratio": "1:1",
*     "seed": 0,
*     "output_format": "png"
*   }
    DATA: BEGIN OF ls_input,
            prompt        TYPE /aws1/rt_shape_string,
            aspect_ratio  TYPE /aws1/rt_shape_string,
            seed          TYPE /aws1/rt_shape_integer,
            output_format TYPE /aws1/rt_shape_string,
          END OF ls_input.

    ls_input-prompt = iv_prompt.
    ls_input-aspect_ratio = '1:1'.
    ls_input-seed = 0. "or better, choose a random integer.
    ls_input-output_format = 'png'.

    DATA(lv_json) = /ui2/cl_json=>serialize(
      data = ls_input
                pretty_name   = /ui2/cl_json=>pretty_mode-low_case ).

    TRY.
        DATA(lo_response) = lo_bdr->invokemodel(
          iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
          iv_modelid = 'stability.stable-image-core-v1:1'
          iv_accept = 'application/json'
          iv_contenttype = 'application/json' ).

        "Stable Image Core Result Format:
```

```
*       {
*         "seeds": ["0"],
*         "finish_reasons": [null],
*         "images": ["iVBORw0KGgoAAAANSUhEUgAAAgAAA...."]
*       }
        DATA: BEGIN OF ls_response,
                images TYPE STANDARD TABLE OF /aws1/rt_shape_string,
              END OF ls_response.

        /ui2/cl_json=>deserialize(
          EXPORTING jsonx = lo_response->get_body( )
                    pretty_name = /ui2/cl_json=>pretty_mode-camel_case
          CHANGING  data  = ls_response ).
        IF ls_response-images IS NOT INITIAL.
          DATA(lv_image) =
 cl_http_utility=>if_http_utility~decode_x_base64( ls_response-images[ 1 ] ).
        ENDIF.
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

Rufen Sie das Stability.ai Stable Image Core Foundation-Modell auf, um Bilder mit dem L2-High-Level-Client zu generieren.

```
    TRY.
        DATA(lo_bdr_l2_sd) = /aws1/
cl_bdr_l2_factory=>create_stable_diffusion_xl_1( lo_bdr ).
        " iv_prompt contains a prompt like 'Show me a picture of a unicorn
 reading an enterprise financial report'.
        DATA(lv_image) = lo_bdr_l2_sd->text_to_image( iv_prompt ).
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

- Einzelheiten zur API finden Sie InvokeModelin der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Amazon Bedrock Agents mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Bedrock Agents mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zu Amazon-Bedrock-Agenten – Weitere Informationen zu Amazon-Bedrock-Agenten.
- API-Referenz zu Amazon-Bedrock-Agenten – Details zu allen verfügbaren Aktionen von Amazon-Bedrock-Agenten.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Bedrock Agents AWS SDKs
  - Hallo Amazon-Bedrock-Agenten
  - Aktionen für Amazon Bedrock-Agenten, die AWS SDKs
    - Verwenden Sie CreateAgentmit einem AWS SDK
    - Verwenden Sie mit CreateAgentActionGroupeinem AWS SDK
    - Verwenden Sie mit CreateAgentAliaseinem AWS SDK
    - Verwenden Sie mit CreateFloweinem AWS SDK
    - Verwenden Sie mit CreateFlowAliaseinem AWS SDK
    - Verwenden Sie mit CreateFlowVersioneinem AWS SDK

- Verwenden Sie mit CreateKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit CreatePrompteinem AWS SDK
- Verwenden Sie mit CreatePromptVersioneinem AWS SDK
- Verwenden Sie mit DeleteAgenteinem AWS SDK
- Verwenden Sie mit DeleteAgentAliaseinem AWS SDK
- Verwenden Sie mit DeleteFloweinem AWS SDK
- Verwenden Sie mit DeleteFlowAliaseinem AWS SDK
- Verwenden Sie mit DeleteFlowVersioneinem AWS SDK
- Verwenden Sie mit DeleteKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit DeletePrompteinem AWS SDK
- Verwenden Sie mit GetAgenteinem AWS SDK
- Verwenden Sie mit GetFloweinem AWS SDK
- Verwenden Sie mit GetFlowVersioneinem AWS SDK
- Verwenden Sie mit GetKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit GetPrompteinem AWS SDK
- Verwenden Sie mit ListAgentActionGroupseinem AWS SDK
- Verwenden Sie mit ListAgentKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListAgentseinem AWS SDK
- Verwenden Sie mit ListFlowAliaseseinem AWS SDK
- Verwenden Sie mit ListFlowVersionseinem AWS SDK
- Verwenden Sie mit ListFlowseinem AWS SDK
- Verwenden Sie mit ListKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListPromptseinem AWS SDK
- Verwenden Sie mit PrepareAgenteinem AWS SDK
- Verwenden Sie mit PrepareFloweinem AWS SDK
- Verwenden Sie mit UpdateFloweinem AWS SDK
- Verwenden Sie mit UpdateFlowAliaseinem AWS SDK
- Verwenden Sie mit UpdateKnowledgeBaseeinem AWS SDK

- Szenarien für die Verwendung von Amazon Bedrock Agents AWS SDKs

- Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK
- Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK
- Ein durchgängiges Beispiel, das zeigt, wie Amazon Bedrock Agents mit einem erstellt und aufgerufen werden AWS SDK
- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Grundlegende Beispiele für die Verwendung von Amazon Bedrock Agents AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Agenten für Amazon Bedrock mit AWS SDKs verwendet werden.

Beispiele

- Hallo Amazon-Bedrock-Agenten
- Aktionen für Amazon Bedrock-Agenten, die AWS SDKs
  - Verwenden Sie CreateAgentmit einem AWS SDK
  - Verwenden Sie mit CreateAgentActionGroupeinem AWS SDK
  - Verwenden Sie mit CreateAgentAliaseinem AWS SDK
  - Verwenden Sie mit CreateFloweinem AWS SDK
  - Verwenden Sie mit CreateFlowAliaseinem AWS SDK
  - Verwenden Sie mit CreateFlowVersioneinem AWS SDK
  - Verwenden Sie mit CreateKnowledgeBaseeinem AWS SDK
  - Verwenden Sie mit CreatePrompteinem AWS SDK
  - Verwenden Sie mit CreatePromptVersioneinem AWS SDK
  - Verwenden Sie mit DeleteAgenteinem AWS SDK
  - Verwenden Sie mit DeleteAgentAliaseinem AWS SDK
  - Verwenden Sie mit DeleteFloweinem AWS SDK
  - Verwenden Sie mit DeleteFlowAliaseinem AWS SDK
  - Verwenden Sie mit DeleteFlowVersioneinem AWS SDK

- Verwenden Sie mit DeleteKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit DeletePrompteinem AWS SDK
- Verwenden Sie mit GetAgenteinem AWS SDK
- Verwenden Sie mit GetFloweinem AWS SDK
- Verwenden Sie mit GetFlowVersioneinem AWS SDK
- Verwenden Sie mit GetKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit GetPrompteinem AWS SDK
- Verwenden Sie mit ListAgentActionGroupseinem AWS SDK
- Verwenden Sie mit ListAgentKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListAgentseinem AWS SDK
- Verwenden Sie mit ListFlowAliaseseinem AWS SDK
- Verwenden Sie mit ListFlowVersionseinem AWS SDK
- Verwenden Sie mit ListFlowseinem AWS SDK
- Verwenden Sie mit ListKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListPromptseinem AWS SDK
- Verwenden Sie mit PrepareAgenteinem AWS SDK
- Verwenden Sie mit PrepareFloweinem AWS SDK
- Verwenden Sie mit UpdateFloweinem AWS SDK
- Verwenden Sie mit UpdateFlowAliaseinem AWS SDK
- Verwenden Sie mit UpdateKnowledgeBaseeinem AWS SDK

Hallo Amazon-Bedrock-Agenten

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Agenten für Amazon Bedrock.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  GetAgentCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";

/**
 * @typedef {Object} AgentSummary
 */

/**
 * A simple scenario to demonstrate basic setup and interaction with the Bedrock
 Agents Client.
 *
 * This function first initializes the Amazon Bedrock Agents client for a
 specific region.
 * It then retrieves a list of existing agents using the streamlined paginator
 approach.
 * For each agent found, it retrieves detailed information using a command
 object.
 *
 * Demonstrates:
 * - Use of the Bedrock Agents client to initialize and communicate with the AWS
 service.
 * - Listing resources in a paginated response pattern.
 * - Accessing an individual resource using a command object.
 *
 * @returns {Promise<void>} A promise that resolves when the function has
 completed execution.
 */
export const main = async () => {
  const region = "us-east-1";

  console.log("=".repeat(68));

  console.log(`Initializing Amazon Bedrock Agents client for ${region}...`);
  const client = new BedrockAgentClient({ region });

  console.log("Retrieving the list of existing agents...");
  const paginatorConfig = { client };
```

```
  const pages = paginateListAgents(paginatorConfig, {});

  /** @type {AgentSummary[]} */
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  console.log(`Found ${agentSummaries.length} agents in ${region}.`);

  if (agentSummaries.length > 0) {
    for (const agentSummary of agentSummaries) {
      const agentId = agentSummary.agentId;
      console.log("=".repeat(68));
      console.log(`Retrieving agent with ID: ${agentId}:`);
      console.log("-".repeat(68));

      const command = new GetAgentCommand({ agentId });
      const response = await client.send(command);
      const agent = response.agent;

      console.log(` Name: ${agent.agentName}`);
      console.log(` Status: ${agent.agentStatus}`);
      console.log(` ARN: ${agent.agentArn}`);
      console.log(` Foundation model: ${agent.foundationModel}`);
    }
  }
  console.log("=".repeat(68));
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - GetAgent
  - ListAgents

Aktionen für Amazon Bedrock-Agenten, die AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Bedrock Agents-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die API der Agenten für Amazon Bedrock auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Bedrock Agents AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz der Agenten für Amazon Bedrock.

Beispiele

- Verwenden Sie CreateAgentmit einem AWS SDK
- Verwenden Sie mit CreateAgentActionGroupeinem AWS SDK
- Verwenden Sie mit CreateAgentAliaseinem AWS SDK
- Verwenden Sie mit CreateFloweinem AWS SDK
- Verwenden Sie mit CreateFlowAliaseinem AWS SDK
- Verwenden Sie mit CreateFlowVersioneinem AWS SDK
- Verwenden Sie mit CreateKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit CreatePrompteinem AWS SDK
- Verwenden Sie mit CreatePromptVersioneinem AWS SDK
- Verwenden Sie mit DeleteAgenteinem AWS SDK
- Verwenden Sie mit DeleteAgentAliaseinem AWS SDK
- Verwenden Sie mit DeleteFloweinem AWS SDK
- Verwenden Sie mit DeleteFlowAliaseinem AWS SDK
- Verwenden Sie mit DeleteFlowVersioneinem AWS SDK
- Verwenden Sie mit DeleteKnowledgeBaseeinem AWS SDK
- Verwenden Sie mit DeletePrompteinem AWS SDK
- Verwenden Sie mit GetAgenteinem AWS SDK
- Verwenden Sie mit GetFloweinem AWS SDK
- Verwenden Sie mit GetFlowVersioneinem AWS SDK
- Verwenden Sie mit GetKnowledgeBaseeinem AWS SDK

- Verwenden Sie mit GetPromptseinem AWS SDK
- Verwenden Sie mit ListAgentActionGroupseinem AWS SDK
- Verwenden Sie mit ListAgentKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListAgentseinem AWS SDK
- Verwenden Sie mit ListFlowAliaseseinem AWS SDK
- Verwenden Sie mit ListFlowVersionseinem AWS SDK
- Verwenden Sie mit ListFlowseinem AWS SDK
- Verwenden Sie mit ListKnowledgeBaseseinem AWS SDK
- Verwenden Sie mit ListPromptseinem AWS SDK
- Verwenden Sie mit PrepareAgenteinem AWS SDK
- Verwenden Sie mit PrepareFloweinem AWS SDK
- Verwenden Sie mit UpdateFloweinem AWS SDK
- Verwenden Sie mit UpdateFlowAliaseinem AWS SDK
- Verwenden Sie mit UpdateKnowledgeBaseeinem AWS SDK

Verwenden Sie **`CreateAgent`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateAgent` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  CreateAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Creates an Amazon Bedrock Agent.
 *
 * @param {string} agentName - A name for the agent that you create.
 * @param {string} foundationModel - The foundation model to be used by the agent
 you create.
 * @param {string} agentResourceRoleArn - The ARN of the IAM role with
 permissions required by the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing details of the created agent.
 */
export const createAgent = async (
  agentName,
  foundationModel,
  agentResourceRoleArn,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  const command = new CreateAgentCommand({
    agentName,
    foundationModel,
    agentResourceRoleArn,
  });
  const response = await client.send(command);

  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentName and accountId, and roleName with a
 unique name for the new agent,
```

```
  // the id of your AWS account, and the name of an existing execution role that
 the agent can use inside your account.
  // For foundationModel, specify the desired model. Ensure to remove the
 brackets '[]' before adding your data.

  // A string (max 100 chars) that can include letters, numbers, dashes '-', and
 underscores '_'.
  const agentName = "[your-bedrock-agent-name]";

  // Your AWS account id.
  const accountId = "[123456789012]";

  // The name of the agent's execution role. It must be prefixed by
 `AmazonBedrockExecutionRoleForAgents_`.
  const roleName = "[AmazonBedrockExecutionRoleForAgents_your-role-name]";

  // The ARN for the agent's execution role.
  // Follow the ARN format: 'arn:aws:iam::account-id:role/role-name'
  const roleArn = `arn:aws:iam::${accountId}:role/${roleName}`;

  // Specify the model for the agent. Change if a different model is preferred.
  const foundationModel = "anthropic.claude-v2";

  // Check for unresolved placeholders in agentName and roleArn.
  checkForPlaceholders([agentName, roleArn]);

  console.log("Creating a new agent...");

  const agent = await createAgent(agentName, foundationModel, roleArn);
  console.log(agent);
}
```

- Einzelheiten zur API finden Sie CreateAgentin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Agenten.

```
    def create_agent(self, agent_name, foundation_model, role_arn, instruction):
        """
        Creates an agent that orchestrates interactions between foundation
models,
        data sources, software applications, user conversations, and APIs to
carry
        out tasks to help customers.

        :param agent_name: A name for the agent.
        :param foundation_model: The foundation model to be used for
orchestration by the agent.
        :param role_arn: The ARN of the IAM role with permissions needed by the
agent.
        :param instruction: Instructions that tell the agent what it should do
and how it should
                            interact with users.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            response = self.client.create_agent(
                agentName=agent_name,
                foundationModel=foundation_model,
                agentResourceRoleArn=role_arn,
                instruction=instruction,
            )
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent. Here's why: {e}")
            raise
        else:
            return response["agent"]
```

- Einzelheiten zur API finden Sie CreateAgentin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateAgentActionGroup`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateAgentActionGroup`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Agentenaktionsgruppe.

```
    def create_agent_action_group(
            self, name, description, agent_id, agent_version, function_arn,
api_schema
    ):
        """
        Creates an action group for an agent. An action group defines a set of
actions that an
        agent should carry out for the customer.

        :param name: The name to give the action group.
        :param description: The description of the action group.
        :param agent_id: The unique identifier of the agent for which to create
the action group.
        :param agent_version: The version of the agent for which to create the
action group.
```

```
        :param function_arn: The ARN of the Lambda function containing the
business logic that is
                                carried out upon invoking the action.
        :param api_schema: Contains the OpenAPI schema for the action group.
        :return: Details about the action group that was created.
        """
        try:
            response = self.client.create_agent_action_group(
                actionGroupName=name,
                description=description,
                agentId=agent_id,
                agentVersion=agent_version,
                actionGroupExecutor={"lambda": function_arn},
                apiSchema={"payload": api_schema},
            )
            agent_action_group = response["agentActionGroup"]
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent action group. Here's why:
{e}")
            raise
        else:
            return agent_action_group
```

- Einzelheiten zur API finden Sie CreateAgentActionGroupin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateAgentAlias`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateAgentAlias`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Alias für einen Agenten.

```
    def create_agent_alias(self, name, agent_id):
        """
        Creates an alias of an agent that can be used to deploy the agent.

        :param name: The name of the alias.
        :param agent_id: The unique identifier of the agent.
        :return: Details about the alias that was created.
        """
        try:
            response = self.client.create_agent_alias(
                agentAliasName=name, agentId=agent_id
            )
            agent_alias = response["agentAlias"]
        except ClientError as e:
            logger.error(f"Couldn't create agent alias. {e}")
            raise
        else:
            return agent_alias
```

- Einzelheiten zur API finden Sie CreateAgentAliasin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateFlow`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateFlow`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Amazon-Bedrock-Flow.

```
def create_flow(client, flow_name, flow_description, role_arn, flow_def):
    """
    Creates an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_name (str): The name for the new flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: The response from CreateFlow.
    """
    try:

        logger.info("Creating flow: %s.", flow_name)

        response = client.create_flow(
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully created flow: %s. ID: %s",
                    flow_name,
                    {response['id']})
```

```
        return response

    except ClientError as e:
        logger.exception("Client error creating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error creating flow: %s", {str(e)})
        raise
```

- Einzelheiten zur API finden Sie CreateFlowin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateFlowAlias`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateFlowAlias`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Alias für einen Amazon-Bedrock-Flow.

```
def create_flow_alias(client, flow_id, flow_version, name, description):
    """
    Creates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
```

```
    Returns:
        str: The ID for the flow alias.
    """

    try:
        logger.info("Creating flow alias for flow: %s.", flow_id)

        response = client.create_flow_alias(
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully created flow alias for %s.", flow_id)

        return response['id']

    except ClientError as e:
        logging.exception("Client error creating alias for flow: %s - %s",
                flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating alias for flow : %s - %s",
                flow_id, str(e))
        raise
```

- Einzelheiten zur API finden Sie CreateFlowAliasin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateFlowVersion`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateFlowVersion`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Version eines Amazon-Bedrock-Flows.

```
def create_flow_version(client, flow_id, description):
    """
    Creates a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        description (str) : A description for the flow.

    Returns:
        str: The version for the flow.
    """
    try:

        logger.info("Creating flow version for flow: %s.", flow_id)

        # Call CreateFlowVersion operation
        response = client.create_flow_version(
            flowIdentifier=flow_id,
            description=description
        )

        logging.info("Successfully created flow version %s for flow %s.",
            response['version'], flow_id)

        return response['version']

    except ClientError as e:
        logging.exception("Client error creating flow: %s", str(e))
        raise
    except Exception as e:
```

```
        logging.exception("Unexpected error creating flow : %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie CreateFlowVersionin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateKnowledgeBase`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateKnowledgeBase`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Amazon-Bedrock-Wissensdatenbank.

```
def create_knowledge_base(bedrock_agent_client, name, role_arn,
 description=None):
    """
    Creates a new knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        name (str): The name of the knowledge base.
        role_arn (str): The ARN of the IAM role that the knowledge base assumes
 to access resources.
        description (str, optional): A description of the knowledge base.

    Returns:
        dict: The details of the created knowledge base.
    """
    try:
        kwargs = {
            "name": name,
            "roleArn": role_arn,
```

```
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            },
            "storageConfiguration": {
                "type": "OPENSEARCH_SERVERLESS",
                # Note: You will need to create an OpenSearch Serverless
 collection first and replace this ARN
                # with your actual collection ARN from the OpenSearch console. If
 you use the console instead,
                # you can use the quick-create flow to have Knowledge Bases
 create the collection for you.
                "opensearchServerlessConfiguration": {
                    "collectionArn": "arn:aws:aoss:us-
east-1::123456789012:collection/abcdefgh12345678defgh",
                        "fieldMapping": {
                        "metadataField": "metadata",
                        "textField": "text",
                        "vectorField": "vector"
                        },
                    "vectorIndexName": "test-uuid"
                    },
                },
            "clientToken": "test-client-token-" + str(uuid.uuid4())
        }

        if description:
            kwargs["description"] = description

        response = bedrock_agent_client.create_knowledge_base(**kwargs)

        logger.info("Created knowledge base with ID: %s",
 response["knowledgeBase"]["knowledgeBaseId"])
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't create knowledge base. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
```

```
        raise
```

- Einzelheiten zur API finden Sie CreateKnowledgeBasein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreatePrompt`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreatePrompt`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines verwalteten Prompts

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines von Amazon Bedrock verwalteten Prompts.

```
def create_prompt(client, prompt_name, prompt_description, prompt_template,
 model_id=None):
    """
    Creates an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_name (str): The name for the new prompt.
    prompt_description (str): The description for the new prompt.
    prompt_template (str): The template for the prompt.
    model_id (str, optional): The model ID to associate with the prompt.

    Returns:
        dict: The response from CreatePrompt.
```

```
    """
    try:
        logger.info("Creating prompt: %s.", prompt_name)

        # Create a variant with the template
        variant = {
            "name": "default",
            "templateType": "TEXT",
            "templateConfiguration": {
                "text": {
                    "text": prompt_template,
                    "inputVariables": []
                }
            }
        }

        # Extract input variables from the template
        # Look for patterns like {{variable_name}}

        variables = re.findall(r'{{(.*?)}}', prompt_template)
        for var in variables:
            variant["templateConfiguration"]["text"]
["inputVariables"].append({"name": var.strip()})

        # Add model ID if provided
        if model_id:
            variant["modelId"] = model_id

        # Create the prompt with the variant
        create_params = {
            'name': prompt_name,
            'description': prompt_description,
            'variants': [variant]
        }

        response = client.create_prompt(**create_params)

        logger.info("Successfully created prompt: %s. ID: %s",
                    prompt_name,
                    response['id'])

        return response

    except ClientError as e:
```

```
        logger.exception("Client error creating prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie CreatePromptin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreatePromptVersion`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreatePromptVersion`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines verwalteten Prompts

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Version eines von Amazon Bedrock verwalteten Prompts.

```
def create_prompt_version(client, prompt_id, description=None):
    """
    Creates a version of an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt to create a version for.
    description (str, optional): A description for the version.
```

```
    Returns:
        dict: The response from CreatePromptVersion.
    """
    try:
        logger.info("Creating version for prompt ID: %s.", prompt_id)

        create_params = {
            'promptIdentifier': prompt_id
        }

        if description:
            create_params['description'] = description

        response = client.create_prompt_version(**create_params)

        logger.info("Successfully created prompt version: %s",
response['version'])
        logger.info("Prompt version ARN: %s", response['arn'])

        return response


    except ClientError as e:
        logger.exception("Client error creating prompt version: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt version: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie CreatePromptVersionin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteAgent`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteAgent` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  DeleteAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Deletes an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent to delete.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-
agent").DeleteAgentCommandOutput>} An object containing the agent id, the status,
 and some additional metadata.
 */
export const deleteAgent = (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });
  const command = new DeleteAgentCommand({ agentId });
  return client.send(command);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets (`[]`) before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";
```

```
  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Deleting agent with ID ${agentId}...`);

  const response = await deleteAgent(agentId);
  console.log(response);
}
```

- Einzelheiten zur API finden Sie DeleteAgentin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Agenten.

```
    def delete_agent(self, agent_id):
        """
        Deletes an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
 raises an exception.
        """

        try:
            response = self.client.delete_agent(
                agentId=agent_id, skipResourceInUseCheck=False
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent. {e}")
            raise
        else:
```

```
            return response
```

- Einzelheiten zur API finden Sie DeleteAgentin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteAgentAlias`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteAgentAlias`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen des Alias eines Agenten.

```
    def delete_agent_alias(self, agent_id, agent_alias_id):
        """
        Deletes an alias of an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent that the alias
belongs to.
        :param agent_alias_id: The unique identifier of the alias to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """

        try:
            response = self.client.delete_agent_alias(
```

```
                agentId=agent_id, agentAliasId=agent_alias_id
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent alias. {e}")
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DeleteAgentAliasin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteFlow`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteFlow`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Amazon-Bedrock-Flows.

```
def delete_flow(client, flow_id):
    """
    Deletes an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
```

```
    flow_id (str): The identifier of the flow that you want to delete.

    Returns:
        dict: The response from the DeleteFLow operation.
    """
    try:

        logger.info("Deleting flow ID: %s.",
                    flow_id)

        # Call DeleteFlow operation
        response = client.delete_flow(
            flowIdentifier=flow_id,
            skipResourceInUseCheck=True
        )

        logger.info("Finished deleting flow ID: %s", flow_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error deleting flow: %s", {str(e)})
        raise
```

- Einzelheiten zur API finden Sie DeleteFlowin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteFlowAlias`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteFlowAlias`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Alias für einen Amazon-Bedrock-Flow.

```
def delete_flow_alias(client, flow_id, flow_alias_id):
    """
    Deletes an Amazon Bedrock flow alias.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from the call to DetectFLowAlias
    """
    try:

        logger.info("Deleting flow alias %s for flow: %s.", flow_alias_id,
 flow_id)

        # Delete the flow alias.
        response = client.delete_flow_alias(
            aliasIdentifier=flow_alias_id,
            flowIdentifier=flow_id
        )

        logging.info("Successfully deleted flow version for %s.", flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie DeleteFlowAliasin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteFlowVersion`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteFlowVersion`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen einer Version eines Amazon-Bedrock-Flows.

```
def delete_flow_version(client, flow_id, flow_version):
    """
    Deletes a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from DeleteFlowVersion.
    """
    try:

        logger.info("Deleting flow version %s for flow: %s.",flow_version,
 flow_id)
```

```
        # Call DeleteFlowVersion operation
        response = client.delete_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully deleted flow version %s for %s.",
                flow_version,
                flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s ", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie DeleteFlowVersionin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteKnowledgeBase`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteKnowledgeBase`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-Bedrock-Wissensdatenbank.

```
def delete_knowledge_base(bedrock_agent_client, knowledge_base_id):
```

```
    """
    Deletes a knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to delete.

    Returns:
        bool: True if the deletion was successful.
    """
    try:
        bedrock_agent_client.delete_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Deleted knowledge base: %s", knowledge_base_id)
        return True
    except ClientError as err:
        logger.error(
            "Couldn't delete knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Einzelheiten zur API finden Sie DeleteKnowledgeBasein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeletePrompt`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeletePrompt`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines verwalteten Prompts

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines von Amazon Bedrock verwalteten Prompts.

```
def delete_prompt(client, prompt_id):
    """
    Deletes an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to delete.

    Returns:
        dict: The response from the DeletePrompt operation.
    """
    try:
        logger.info("Deleting prompt ID: %s.", prompt_id)

        response = client.delete_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Finished deleting prompt ID: %s", prompt_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error deleting prompt: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie DeletePromptin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetAgent`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetAgent` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  GetAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves the details of an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing the agent details.
```

```
 */
export const getAgent = async (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const command = new GetAgentCommand({ agentId });
  const response = await client.send(command);
  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets '[]' before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Retrieving agent with ID ${agentId}...`);

  const agent = await getAgent(agentId);
  console.log(agent);
}
```

- Einzelheiten zur API finden Sie GetAgentin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Agenten.

```
    def get_agent(self, agent_id, log_error=True):
```

```
        """
        Gets information about an agent.

        :param agent_id: The unique identifier of the agent.
        :param log_error: Whether to log any errors that occur when getting the
agent.
                          If True, errors will be logged to the logger. If False,
errors
                          will still be raised, but not logged.
        :return: The information about the requested agent.
        """

        try:
            response = self.client.get_agent(agentId=agent_id)
            agent = response["agent"]
        except ClientError as e:
            if log_error:
                logger.error(f"Couldn't get agent {agent_id}. {e}")
            raise
        else:
            return agent
```

- Einzelheiten zur API finden Sie GetAgentin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetFlow`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetFlow`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Amazon-Bedrock-Flows.

```
def get_flow(client, flow_id):
    """
    Gets an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to get.

    Returns:
        dict: The response from the GetFlow operation.
    """
    try:

        logger.info("Getting flow ID: %s.",
                    flow_id)

        # Call GetFlow operation.
        response = client.get_flow(
            flowIdentifier=flow_id
        )

        logger.info("Retrieved flow ID: %s. Name: %s", flow_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting flow: %s", {str(e)})
        raise

    except Exception as e:
```

```
        logger.exception("Unexepcted error getting flow: %s", {str(e)})
        raise
```

- Einzelheiten zur API finden Sie GetFlowin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetFlowVersion`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetFlowVersion`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen einer Version eines Amazon-Bedrock-Flows.

```
def get_flow_version(client, flow_id, flow_version):
    """
    Gets information about a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        flow_version (str): The flow version of the flow.

    Returns:
        dict: The response from the call to GetFlowVersion.
    """
    try:
```

```
        logger.info("Deleting flow version for flow: %s.", flow_id)

        # Call GetFlowVersion operation
        response = client.get_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully got flow version %s information for flow %s.",
                     flow_version,
                     flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error getting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error getting flow version: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie GetFlowVersionin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetKnowledgeBase`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetKnowledgeBase`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Amazon-Bedrock-Wissensdatenbank ab.

```
def get_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Gets details about a specific knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base.

    Returns:
        dict: The details of the knowledge base.
    """
    try:
        response = bedrock_agent_client.get_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Retrieved knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]
    except ClientError as err:
        logger.error(
            "Couldn't get knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Einzelheiten zur API finden Sie GetKnowledgeBasein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetPrompt`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetPrompt`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines von Amazon Bedrock verwalteten Prompts.

```
def get_prompt(client, prompt_id):
    """
    Gets an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to get.

    Returns:
        dict: The response from the GetPrompt operation.
    """
    try:
        logger.info("Getting prompt ID: %s.", prompt_id)

        response = client.get_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Retrieved prompt ID: %s. Name: %s",
                    prompt_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error getting prompt: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie GetPromptin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListAgentActionGroups`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListAgentActionGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aktionsgruppen für einen Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  ListAgentActionGroupsCommand,
  paginateListAgentActionGroups,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves a list of Action Groups of an agent utilizing the paginator
 function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
```

```
 * a straightforward way to handle paginated results inside a `for await...of`
 loop.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithPaginator = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  // Create a paginator configuration
  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const params = { agentId, agentVersion };

  const pages = paginateListAgentActionGroups(paginatorConfig, params);

  // Paginate until there are no more results
  const actionGroupSummaries = [];
  for await (const page of pages) {
    actionGroupSummaries.push(...page.actionGroupSummaries);
  }

  return actionGroupSummaries;
};

/**
 * Retrieves a list of Action Groups of an agent utilizing the
 ListAgentActionGroupsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the
 client and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentActionGroupsWithPaginator()` example below.
 *
```

```
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithCommandObject = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const actionGroupSummaries = [];
  do {
    const command = new ListAgentActionGroupsCommand({
      agentId,
      agentVersion,
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{actionGroupSummaries: ActionGroupSummary[], nextToken?: string}}
 */
    const response = await client.send(command);

    for (const actionGroup of response.actionGroupSummaries || []) {
      actionGroupSummaries.push(actionGroup);
    }

    nextToken = response.nextToken;
  } while (nextToken);

  return actionGroupSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId and agentVersion with an existing
 agent's id and version.
  // Ensure to remove the brackets '[]' before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";
```

```
  // A string either containing `DRAFT` or a number with 1-5 digits (e.g., '123'
 or 'DRAFT').
  const agentVersion = "[DRAFT]";

  // Check for unresolved placeholders in agentId and agentVersion.
  checkForPlaceholders([agentId, agentVersion]);

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using ListAgentActionGroupsCommand:",
  );

  for (const actionGroup of await listAgentActionGroupsWithCommandObject(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using the paginateListAgents function:",
  );
  for (const actionGroup of await listAgentActionGroupsWithPaginator(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }
}
```

- Einzelheiten zur API finden Sie ListAgentActionGroupsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aktionsgruppen für einen Agenten.

```
    def list_agent_action_groups(self, agent_id, agent_version):
        """
        List the action groups for a version of an Amazon Bedrock Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of action group summaries for the version of the agent.
        """

        try:
            action_groups = []

            paginator = self.client.get_paginator("list_agent_action_groups")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                action_groups.extend(page["actionGroupSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list action groups. {e}")
            raise
        else:
            return action_groups
```

- Einzelheiten zur API finden Sie ListAgentActionGroupsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListAgentKnowledgeBases`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListAgentKnowledgeBases`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der einem Agenten zugeordneten Wissensdatenbanken.

```
    def list_agent_knowledge_bases(self, agent_id, agent_version):
        """
        List the knowledge bases associated with a version of an Amazon Bedrock
Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of knowledge base summaries for the version of the
agent.
        """

        try:
            knowledge_bases = []

            paginator = self.client.get_paginator("list_agent_knowledge_bases")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                knowledge_bases.extend(page["agentKnowledgeBaseSummaries"])
```

```
        except ClientError as e:
            logger.error(f"Couldn't list knowledge bases. {e}")
            raise
        else:
            return knowledge_bases
```

- Einzelheiten zur API finden Sie ListAgentKnowledgeBasesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListAgents`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListAgents` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Agenten, die zu einem Konto gehören.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  ListAgentsCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";
```

```
/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the paginator
 function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of`
 loop.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithPaginator = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const pages = paginateListAgents(paginatorConfig, {});

  // Paginate until there are no more results
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  return agentSummaries;
};

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the
 ListAgentsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the
 client and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentsWithPaginator()` example below.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
```

```
 */
export const listAgentsWithCommandObject = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const agentSummaries = [];
  do {
    const command = new ListAgentsCommand({
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{agentSummaries: AgentSummary[], nextToken?: string}} */
    const paginatedResponse = await client.send(command);

    agentSummaries.push(...(paginatedResponse.agentSummaries || []));

    nextToken = paginatedResponse.nextToken;
  } while (nextToken);

  return agentSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("=".repeat(68));
  console.log("Listing agents using ListAgentsCommand:");
  for (const agent of await listAgentsWithCommandObject()) {
    console.log(agent);
  }

  console.log("=".repeat(68));
  console.log("Listing agents using the paginateListAgents function:");
  for (const agent of await listAgentsWithPaginator()) {
    console.log(agent);
  }
}
```

- Einzelheiten zur API finden Sie ListAgentsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Agenten, die zu einem Konto gehören.

```
    def list_agents(self):
        """
        List the available Amazon Bedrock Agents.

        :return: The list of available bedrock agents.
        """

        try:
            all_agents = []

            paginator = self.client.get_paginator("list_agents")
            for page in paginator.paginate(PaginationConfig={"PageSize": 10}):
                all_agents.extend(page["agentSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list agents. {e}")
            raise
        else:
            return all_agents
```

- Einzelheiten zur API finden Sie ListAgentsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListFlowAliases`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListFlowAliases`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aliasse für einen Amazon-Bedrock-Flow.

```
def list_flow_aliases(client, flow_id):
    """
    Lists the aliases of an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowAliases.
    """
    try:

        finished = False

        logger.info("Listing flow aliases for flow: %s.", flow_id)

        print(f"Aliases for flow: {flow_id}")

        response = client.list_flow_aliases(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            for alias in response['flowAliasSummaries']:
                print(f"Alias Name: {alias['name']}")
                print(f"ID: {alias['id']}")
                print(f"Description: {alias.get('description', 'No
 description')}\n")
```

```
                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_aliases(maxResults=10,
                                                        nextToken=next_token)
                else:
                    finished = True

        logging.info("Successfully listed flow aliases for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow aliases: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow aliases: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie ListFlowAliasesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListFlowVersions`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListFlowVersions`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Versionen eines Amazon-Bedrock-Flows.

```
def list_flow_versions(client, flow_id):
    """
    Lists the versions of an Amazon Bedrock flow.
```

```
    Args:
        client: Amazon bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowVersions.
    """
    try:

        finished = False

        logger.info("Listing flow versions for flow: %s.", flow_id)

        response = client.list_flow_versions(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            print(f"Versions for flow:{flow_id}")
            for version in response['flowVersionSummaries']:
                print(f"Version: {version['version']}")
                print(f"Status: {version['status']}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_versions(maxResults=10,
                                                         nextToken=next_token)
                else:
                    finished = True


        logging.info("Successfully listed flow versions for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie ListFlowVersionsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListFlows`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListFlows`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Amazon-Bedrock-Flows.

```
def list_flows(client):
    """
    Lists versions of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        Nothing.
    """
    try:
        finished = False

        logger.info("Listing flows:")

        response = client.list_flows(maxResults=10)

        while finished is False:

            for flow in response['flowSummaries']:
```

```
                print(f"ID: {flow['id']}")
                print(f"Name: {flow['name']}")
                print(
                    f"Description: {flow.get('description', 'No description')}")
                print(f"Latest version: {flow['version']}")
                print(f"Status: {flow['status']}\n")

            if 'nextToken' in response:
                next_token = response['nextToken']
                response = client.list_flows(maxResults=10,
                                             nextToken=next_token)
            else:
                finished = True

        logging.info("Successfully listed flows.")


    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie ListFlowsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListKnowledgeBases`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListKnowledgeBases`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet die Amazon-Bedrock-Wissensdatenbanken auf.

```
def list_knowledge_bases(bedrock_agent_client, max_results=None):
    """
    Lists the knowledge bases in your AWS account.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        max_results (int, optional): The maximum number of knowledge bases to
 return.

    Returns:
        list: A list of knowledge base details.
    """
    try:
        kwargs = {}
        if max_results is not None:
            kwargs["maxResults"] = max_results

        # Initialize an empty list to store all knowledge bases
        all_knowledge_bases = []

        # Use paginator to handle pagination automatically
        paginator = bedrock_agent_client.get_paginator('list_knowledge_bases')
        page_iterator = paginator.paginate(**kwargs)

        # Iterate through each page of results
        for page in page_iterator:
            all_knowledge_bases.extend(page.get('knowledgeBaseSummaries', []))

        logger.info("Found %s knowledge bases.", len(all_knowledge_bases))
        return all_knowledge_bases
    except ClientError as err:
        logger.error(
            "Couldn't list knowledge bases. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Einzelheiten zur API finden Sie ListKnowledgeBasesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListPrompts`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListPrompts`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der von Amazon Bedrock verwalteten Prompts.

```
def list_prompts(client, max_results=10):
    """
    Lists Amazon Bedrock managed prompts.

    Args:
        client: Amazon Bedrock Agent boto3 client.
        max_results (int): Maximum number of results to return per page.

    Returns:
        list: A list of prompt summaries.
    """
    try:
        logger.info("Listing prompts:")

        # Create a paginator for the list_prompts operation
        paginator = client.get_paginator('list_prompts')

        # Create the pagination parameters
        pagination_config = {
            'maxResults': max_results
        }

        # Initialize an empty list to store all prompts
        all_prompts = []

        # Iterate through all pages
```

```
        for page in paginator.paginate(**pagination_config):
            all_prompts.extend(page.get('promptSummaries', []))

        logger.info("Successfully listed %s prompts.", len(all_prompts))
        return all_prompts

    except ClientError as e:
        logger.exception("Client error listing prompts: %s", str(e))
        raise
    except Exception as e:
        logger.exception("Unexpected error listing prompts: %s", str(e))
        raise
```

- Einzelheiten zur API finden Sie ListPromptsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`PrepareAgent`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PrepareAgent`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Agenten

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Vorbereiten eines Agenten auf interne Tests.

```
    def prepare_agent(self, agent_id):
        """
```

```
        Creates a DRAFT version of the agent that can be used for internal
testing.

        :param agent_id: The unique identifier of the agent to prepare.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            prepared_agent_details = self.client.prepare_agent(agentId=agent_id)
        except ClientError as e:
            logger.error(f"Couldn't prepare agent. {e}")
            raise
        else:
            return prepared_agent_details
```

- Einzelheiten zur API finden Sie PrepareAgentin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`PrepareFlow`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PrepareFlow`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Aufrufen eines Flows

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Vorbereiten eines Amazon-Bedrock-Flows.

```
def prepare_flow(client, flow_id):
    """
    Prepares an Amazon Bedrock Flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to prepare.

    Returns:
        str: The status of the flow preparation
    """
    try:

        # Prepare the flow.
        logger.info("Preparing flow ID: %s",
                    flow_id)

        response = client.prepare_flow(
            flowIdentifier=flow_id
        )

        status = response.get('status')

        while status == "Preparing":
            logger.info("Preparing flow ID: %s. Status %s",
                        flow_id, status)

            sleep(5)
            response = client.get_flow(
                flowIdentifier=flow_id
            )
            status = response.get('status')
            print(f"Flow Status: {status}")

        if status == "Prepared":
            logger.info("Finished preparing flow ID: %s. Status %s",
                        flow_id, status)
        else:
            logger.warning("flow ID: %s not prepared. Status %s",
                           flow_id, status)

        return status
```

```
    except ClientError as e:
        logger.exception("Client error preparing flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error preparing flow: %s", {str(e)})
        raise
```

- Einzelheiten zur API finden Sie PrepareFlowin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UpdateFlow`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateFlow`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren eines Amazon-Bedrock-Flows.

```
def update_flow(client, flow_id, flow_name, flow_description, role_arn,
 flow_def):
    """
    Updates an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
    flow_id (str): The ID for the flow that you want to update.
    flow_name (str): The name for the flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: Flow information if successful.
    """
```

```
    try:

        logger.info("Updating flow: %s.", flow_id)

        response = client.update_flow(
            flowIdentifier=flow_id,
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully updated flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response

    except ClientError as e:
        logger.exception("Client error updating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error updating flow: %s", {str(e)})
        raise
```

- Einzelheiten zur API finden Sie UpdateFlowin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UpdateFlowAlias`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateFlowAlias`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren des Alias eines Amazon-Bedrock-Flows.

```
def update_flow_alias(client, flow_id, alias_id, flow_version, name,
 description):
    """
    Updates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The response from UpdateFlowAlias.
    """

    try:
        logger.info("Updating flow alias %s for flow: %s.", alias_id, flow_id)

        response = client.update_flow_alias(
            aliasIdentifier=alias_id,
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully updated flow alias %s for %s.", alias_id,
 flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error updating alias %s for flow: %s - %s",
                alias_id, flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error updating alias %s for flow : %s -
 %s",
                alias_id, flow_id, str(e))
        raise
```

- Einzelheiten zur API finden Sie UpdateFlowAliasin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UpdateKnowledgeBase`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateKnowledgeBase`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren einer Amazon-Bedrock-Wissensdatenbank.

```
def update_knowledge_base(bedrock_agent_client, knowledge_base_id, name=None,
 description=None, role_arn=None):
    """
    Updates an existing knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to update.
        name (str, optional): The new name for the knowledge base.
        description (str, optional): The new description for the knowledge base.
        role_arn (str, optional): The new IAM role ARN for the knowledge base.

    Returns:
        dict: The details of the updated knowledge base.
    """
    try:
        kwargs = {
            "knowledgeBaseId": knowledge_base_id,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
```

```
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            }
        }

        if name:
            kwargs["name"] = name
        if description:
            kwargs["description"] = description
        if role_arn:
            kwargs["roleArn"] = role_arn

        response = bedrock_agent_client.update_knowledge_base(**kwargs)

        logger.info("Updated knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't update knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Einzelheiten zur API finden Sie UpdateKnowledgeBasein AWS SDK for Python (Boto3) API Reference.

Szenarien für die Verwendung von Amazon Bedrock Agents AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Bedrock Agents mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Agenten für Amazon Bedrock oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK
- Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK
- Ein durchgängiges Beispiel, das zeigt, wie Amazon Bedrock Agents mit einem erstellt und aufgerufen werden AWS SDK
- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Flow.
- Erstellen Sie den Flow.
- Stellen Sie den vollständig konfigurierten Flow bereit.
- Rufen Sie den Flow mit vom Benutzer bereitgestellten Prompts auf.
- Löschen aller erstellten Ressourcen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellt eine Musik-Playlist basierend auf dem vom Benutzenden angegebenen Genre und der Anzahl der Songs.

```
from datetime import datetime
import logging
```

```
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
```

```
    Creates a prompt node configuration for a Bedrock flow that generates music
playlists.

    The prompt node defines an inline prompt template that creates a music
playlist based on
    a specified genre and number of songs. The prompt uses two variables that are
mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting
of the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
```

```
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
```

```
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
```

```
        )

    # Then, create a connection between the output of the prompt node and the
 input of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
```

```
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
```

```
                                        f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
```

```
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
```

```
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
```

```
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])
```

```
    return {
        "flow_status": flow_status,
        "output": output

    }




def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """


    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)
```

```
        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """



    # Trust relationship policy - allows Amazon Bedrock service to assume this
 role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
```

```
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']
```

```
    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """



    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
```

```
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """




    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias

- GetFlowVersion
- InvokeFlow
- PrepareFlow

Ein umfassendes Beispiel, das zeigt, wie Sie verwaltete Amazon Bedrock-Eingabeaufforderungen mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines verwalteten Prompts.
- Erstellen einer Version des Prompts.
- Aufrufen des Prompts mithilfe der Version.
- Bereinigen der Ressourcen (optional).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines verwalteten Prompts.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
```

```
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of
 songs: {{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
```

```
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
```

```
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
```

```
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Ein durchgängiges Beispiel, das zeigt, wie Amazon Bedrock Agents mit einem erstellt und aufgerufen werden AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Agenten.
- Erstellen einer Lambda-Funktion, die die Funktionen des Agenten implementiert.
- Erstellen einer Lambda-Funktion, die die Funktionen des Agenten implementiert.
- Erstellen einer Aktionsgruppe, die den Agenten mit der Lambda-Funktion verbindet.
- Bereitstellen des vollständig konfigurierten Agenten.
- Aufrufen des Agenten mit vom Benutzer bereitgestellten Prompts.
- Löschen aller erstellten Ressourcen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines Agenten.

```
REGION = "us-east-1"
ROLE_POLICY_NAME = "agent_permissions"


class BedrockAgentScenarioWrapper:
    """Runs a scenario that shows how to get started using Amazon Bedrock
 Agents."""

    def __init__(
            self, bedrock_agent_client, runtime_client, lambda_client,
 iam_resource, postfix
    ):
        self.iam_resource = iam_resource
        self.lambda_client = lambda_client
```

```
        self.bedrock_agent_runtime_client = runtime_client
        self.postfix = postfix

        self.bedrock_wrapper = BedrockAgentWrapper(bedrock_agent_client)

        self.agent = None
        self.agent_alias = None
        self.agent_role = None
        self.prepared_agent_details = None
        self.lambda_role = None
        self.lambda_function = None

    def run_scenario(self):
        print("=" * 88)
        print("Welcome to the Amazon Bedrock Agents demo.")
        print("=" * 88)

        # Query input from user
        print("Let's start with creating an agent:")
        print("-" * 40)
        name, foundation_model = self._request_name_and_model_from_user()
        print("-" * 40)

        # Create an execution role for the agent
        self.agent_role = self._create_agent_role(foundation_model)

        # Create the agent
        self.agent = self._create_agent(name, foundation_model)

        # Prepare a DRAFT version of the agent
        self.prepared_agent_details = self._prepare_agent()

        # Create the agent's Lambda function
        self.lambda_function = self._create_lambda_function()

        # Configure permissions for the agent to invoke the Lambda function
        self._allow_agent_to_invoke_function()
        self._let_function_accept_invocations_from_agent()

        # Create an action group to connect the agent with the Lambda function
        self._create_agent_action_group()

        # If the agent has been modified or any components have been added,
prepare the agent again
```

```
        components = [self._get_agent()]
        components += self._get_agent_action_groups()
        components += self._get_agent_knowledge_bases()

        latest_update = max(component["updatedAt"] for component in components)
        if latest_update > self.prepared_agent_details["preparedAt"]:
            self.prepared_agent_details = self._prepare_agent()

        # Create an agent alias
        self.agent_alias = self._create_agent_alias()

        # Test the agent
        self._chat_with_agent(self.agent_alias)

        print("=" * 88)
        print("Thanks for running the demo!\n")

        if q.ask("Do you want to delete the created resources? [y/N] ",
q.is_yesno):
            self._delete_resources()
            print("=" * 88)
            print(
                "All demo resources have been deleted. Thanks again for running
the demo!"
            )
        else:
            self._list_resources()
            print("=" * 88)
            print("Thanks again for running the demo!")

    def _request_name_and_model_from_user(self):
        existing_agent_names = [
            agent["agentName"] for agent in self.bedrock_wrapper.list_agents()
        ]

        while True:
            name = q.ask("Enter an agent name: ", self.is_valid_agent_name)
            if name.lower() not in [n.lower() for n in existing_agent_names]:
                break
            print(
                f"Agent {name} conflicts with an existing agent. Please use a
different name."
            )
```

```
        models = ["anthropic.claude-instant-v1", "anthropic.claude-v2"]
        model_id = models[
            q.choose("Which foundation model would you like to use? ", models)
        ]

        return name, model_id

    def _create_agent_role(self, model_id):
        role_name = f"AmazonBedrockExecutionRoleForAgents_{self.postfix}"
        model_arn = f"arn:aws:bedrock:{REGION}::foundation-model/{model_id}*"

        print("Creating an an execution role for the agent...")

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service":
"bedrock.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )

            role.Policy(ROLE_POLICY_NAME).put(
                PolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "bedrock:InvokeModel",
                                "Resource": model_arn,
                            }
                        ],
                    }
                )
```

```
            )
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        return role

    def _create_agent(self, name, model_id):
        print("Creating the agent...")

        instruction = """
            You are a friendly chat bot. You have access to a function called
that returns
            information about the current date and time. When responding with
date or time,
            please make sure to add the timezone UTC.
            """
        agent = self.bedrock_wrapper.create_agent(
            agent_name=name,
            foundation_model=model_id,
            instruction=instruction,
            role_arn=self.agent_role.arn,
        )
        self._wait_for_agent_status(agent["agentId"], "NOT_PREPARED")

        return agent

    def _prepare_agent(self):
        print("Preparing the agent...")

        agent_id = self.agent["agentId"]
        prepared_agent_details = self.bedrock_wrapper.prepare_agent(agent_id)
        self._wait_for_agent_status(agent_id, "PREPARED")

        return prepared_agent_details

    def _create_lambda_function(self):
        print("Creating the Lambda function...")

        function_name = f"AmazonBedrockExampleFunction_{self.postfix}"

        self.lambda_role = self._create_lambda_role()

        try:
```

```
            deployment_package = self._create_deployment_package(function_name)

            lambda_function = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="Lambda function for Amazon Bedrock example",
                Runtime="python3.11",
                Role=self.lambda_role.arn,
                Handler=f"{function_name}.lambda_handler",
                Code={"ZipFile": deployment_package},
                Publish=True,
            )

            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)

        except ClientError as e:
            logger.error(
                f"Couldn't create Lambda function {function_name}. Here's why:
{e}"
            )
            raise

        return lambda_function

    def _create_lambda_role(self):
        print("Creating an execution role for the Lambda function...")

        role_name = f"AmazonBedrockExecutionRoleForLambda_{self.postfix}"

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service": "lambda.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
```

```
            )
            role.attach_policy(
                PolicyArn="arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
            )
            print(f"Created role {role_name}")
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        print("Waiting for the execution role to be fully propagated...")
        wait(10)

        return role

    def _allow_agent_to_invoke_function(self):
        policy = self.iam_resource.RolePolicy(
            self.agent_role.role_name, ROLE_POLICY_NAME
        )
        doc = policy.policy_document
        doc["Statement"].append(
            {
                "Effect": "Allow",
                "Action": "lambda:InvokeFunction",
                "Resource": self.lambda_function["FunctionArn"],
            }
        )

 self.agent_role.Policy(ROLE_POLICY_NAME).put(PolicyDocument=json.dumps(doc))

    def _let_function_accept_invocations_from_agent(self):
        try:
            self.lambda_client.add_permission(
                FunctionName=self.lambda_function["FunctionName"],
                SourceArn=self.agent["agentArn"],
                StatementId="BedrockAccess",
                Action="lambda:InvokeFunction",
                Principal="bedrock.amazonaws.com",
            )
        except ClientError as e:
            logger.error(
                f"Couldn't grant Bedrock permission to invoke the Lambda
 function. Here's why: {e}"
            )
```

```
            raise

    def _create_agent_action_group(self):
        print("Creating an action group for the agent...")

        try:
            with open("./scenario_resources/api_schema.yaml") as file:
                self.bedrock_wrapper.create_agent_action_group(
                    name="current_date_and_time",
                    description="Gets the current date and time.",
                    agent_id=self.agent["agentId"],
                    agent_version=self.prepared_agent_details["agentVersion"],
                    function_arn=self.lambda_function["FunctionArn"],
                    api_schema=json.dumps(yaml.safe_load(file)),
                )
        except ClientError as e:
            logger.error(f"Couldn't create agent action group. Here's why: {e}")
            raise

    def _get_agent(self):
        return self.bedrock_wrapper.get_agent(self.agent["agentId"])

    def _get_agent_action_groups(self):
        return self.bedrock_wrapper.list_agent_action_groups(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _get_agent_knowledge_bases(self):
        return self.bedrock_wrapper.list_agent_knowledge_bases(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _create_agent_alias(self):
        print("Creating an agent alias...")

        agent_alias_name = "test_agent_alias"
        agent_alias = self.bedrock_wrapper.create_agent_alias(
            agent_alias_name, self.agent["agentId"]
        )

        self._wait_for_agent_status(self.agent["agentId"], "PREPARED")

        return agent_alias
```

```
    def _wait_for_agent_status(self, agent_id, status):
        while self.bedrock_wrapper.get_agent(agent_id)["agentStatus"] != status:
            wait(2)

    def _chat_with_agent(self, agent_alias):
        print("-" * 88)
        print("The agent is ready to chat.")
        print("Try asking for the date or time. Type 'exit' to quit.")

        # Create a unique session ID for the conversation
        session_id = uuid.uuid4().hex

        while True:
            prompt = q.ask("Prompt: ", q.non_empty)

            if prompt == "exit":
                break

            response = asyncio.run(self._invoke_agent(agent_alias, prompt,
 session_id))

            print(f"Agent: {response}")

    async def _invoke_agent(self, agent_alias, prompt, session_id):
        response = self.bedrock_agent_runtime_client.invoke_agent(
            agentId=self.agent["agentId"],
            agentAliasId=agent_alias["agentAliasId"],
            sessionId=session_id,
            inputText=prompt,
        )

        completion = ""

        for event in response.get("completion"):
            chunk = event["chunk"]
            completion += chunk["bytes"].decode()

        return completion

    def _delete_resources(self):
        if self.agent:
            agent_id = self.agent["agentId"]

            if self.agent_alias:
```

```
                agent_alias_id = self.agent_alias["agentAliasId"]
                print("Deleting agent alias...")
                self.bedrock_wrapper.delete_agent_alias(agent_id, agent_alias_id)

            print("Deleting agent...")
            agent_status = self.bedrock_wrapper.delete_agent(agent_id)
["agentStatus"]
            while agent_status == "DELETING":
                wait(5)
                try:
                    agent_status = self.bedrock_wrapper.get_agent(
                        agent_id, log_error=False
                    )["agentStatus"]
                except ClientError as err:
                    if err.response["Error"]["Code"] ==
 "ResourceNotFoundException":
                        agent_status = "DELETED"

        if self.lambda_function:
            name = self.lambda_function["FunctionName"]
            print(f"Deleting function '{name}'...")
            self.lambda_client.delete_function(FunctionName=name)

        if self.agent_role:
            print(f"Deleting role '{self.agent_role.role_name}'...")
            self.agent_role.Policy(ROLE_POLICY_NAME).delete()
            self.agent_role.delete()

        if self.lambda_role:
            print(f"Deleting role '{self.lambda_role.role_name}'...")
            for policy in self.lambda_role.attached_policies.all():
                policy.detach_role(RoleName=self.lambda_role.role_name)
            self.lambda_role.delete()

    def _list_resources(self):
        print("-" * 40)
        print(f"Here is the list of created resources in '{REGION}'.")
        print("Make sure you delete them once you're done to avoid unnecessary
 costs.")
        if self.agent:
            print(f"Bedrock Agent:   {self.agent['agentName']}")
        if self.lambda_function:
            print(f"Lambda function: {self.lambda_function['FunctionName']}")
        if self.agent_role:
```

```
            print(f"IAM role:        {self.agent_role.role_name}")
        if self.lambda_role:
            print(f"IAM role:        {self.lambda_role.role_name}")

    @staticmethod
    def is_valid_agent_name(answer):
        valid_regex = r"^[a-zA-Z0-9_-]{1,100}$"
        return (
            answer
            if answer and len(answer) <= 100 and re.match(valid_regex, answer)
            else None,
            "I need a name for the agent, please. Valid characters are a-z, A-Z,
 0-9, _ (underscore) and - (hyphen).",
        )

    @staticmethod
    def _create_deployment_package(function_name):
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(
                "./scenario_resources/lambda_function.py", f"{function_name}.py"
            )
        buffer.seek(0)
        return buffer.read()


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    postfix = "".join(
        random.choice(string.ascii_lowercase + "0123456789") for _ in range(8)
    )
    scenario = BedrockAgentScenarioWrapper(
        bedrock_agent_client=boto3.client(
            service_name="bedrock-agent", region_name=REGION
        ),
        runtime_client=boto3.client(
            service_name="bedrock-agent-runtime", region_name=REGION
        ),
        lambda_client=boto3.client(service_name="lambda", region_name=REGION),
        iam_resource=boto3.resource("iam"),
        postfix=postfix,
    )
    try:
```

```
        scenario.run_scenario()
    except Exception as e:
        logging.exception(f"Something went wrong with the demo. Here's what:
{e}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAgent
  - CreateAgentActionGroup
  - CreateAgentAlias
  - DeleteAgent
  - DeleteAgentAlias
  - GetAgent
  - ListAgentActionGroups
  - ListAgentKnowledgeBases
  - ListAgents
  - PrepareAgent

Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

Python

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.

- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finden Sie das aktuelle Repository mit den meisten Trends GitHub und fassen Sie es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step Functions

Codebeispiele für Amazon Bedrock Agents Runtime mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Bedrock Agents Runtime mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zur Amazon-Bedrock-Agentenlaufzeit – Weitere Informationen zur Amazon-Bedrock-Agentenlaufzeit.
- API-Referenz zur Amazon-Bedrock-Agentenlaufzeit – Details zu allen verfügbaren Aktionen zur Amazon-Bedrock-Agentenlaufzeit.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Bedrock Agents Runtime AWS SDKs
  - Konversation mit einem Amazon-Bedrock-Flow
  - Aktionen für Amazon Bedrock Agents Runtime mit AWS SDKs
    - Verwenden Sie InvokeAgentmit einem AWS SDK
    - Verwenden Sie InvokeFlowmit einem AWS SDK
- Szenarien für die Verwendung von Amazon Bedrock Agents Runtime AWS SDKs
  - Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK
  - Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Grundlegende Beispiele für die Verwendung von Amazon Bedrock Agents Runtime AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Runtime der Agenten für Amazon Bedrock mit AWS SDKs verwendet werden.

Beispiele

- Konversation mit einem Amazon-Bedrock-Flow
- Aktionen für Amazon Bedrock Agents Runtime mit AWS SDKs
  - Verwenden Sie InvokeAgentmit einem AWS SDK
  - Verwenden Sie InvokeFlowmit einem AWS SDK

Konversation mit einem Amazon-Bedrock-Flow

Das folgende Codebeispiel zeigt, wie Sie InvokeFlow mit einem Amazon Bedrock-Flow kommunizieren können, der einen Agentenknoten enthält.

Weitere Informationen finden Sie unter Konversation mit einem Amazon-Bedrock-Flow.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Shows how to run an Amazon Bedrock flow with InvokeFlow and handle muli-turn
 interaction
for a single conversation.
For more information, see https://docs.aws.amazon.com/bedrock/latest/userguide/
flows-multi-turn-invocation.html.

"""
```

```
import logging
import boto3
import botocore

import botocore.exceptions

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def invoke_flow(client, flow_id, flow_alias_id, input_data, execution_id):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
        execution_id: Execution ID for continuing a flow. Use the value None on
 first run.

    Returns:
        Dict containing flow_complete status, input_required info, and
 execution_id
    """

    response = None
    request_params = None

    if execution_id is None:
        # Don't pass execution ID for first run.
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }
    else:
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "executionId": execution_id,
            "inputs": [input_data],
```

```
            "enableTrace": True
        }

    response = client.invoke_flow(**request_params)

    if "executionId" not in request_params:
        execution_id = response['executionId']

    input_required = None
    flow_status = ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Check if more input us needed from user.
        elif 'flowMultiTurnInputRequestEvent' in event:
            input_required = event

        # Print the model output.
        elif 'flowOutputEvent' in event:
            print(event['flowOutputEvent']['content']['document'])

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "input_required": input_required,
        "execution_id": execution_id
    }


def converse_with_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Run a conversation with the supplied flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
```

```
        flow_alias_id: The alias ID of the flow.

    """

    flow_execution_id = None
    finished = False

    # Get the intial prompt from the user.
    user_input = input("Enter input: ")

    # Use prompt to create input data.
    flow_input_data = {
        "content": {
            "document": user_input
        },
        "nodeName": "FlowInputNode",
        "nodeOutputName": "document"
    }

    try:
        while not finished:
            # Invoke the flow until successfully finished.

            result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data,
 flow_execution_id)

            status = result['flow_status']
            flow_execution_id = result['execution_id']
            more_input = result['input_required']
            if status == "INPUT_REQUIRED":
                # The flow needs more information from the user.
                logger.info("The flow %s requires more input", flow_id)
                user_input = input(
                    more_input['flowMultiTurnInputRequestEvent']['content']
['document'] + ": ")
                flow_input_data = {
                    "content": {
                        "document": user_input
                    },
                    "nodeName": more_input['flowMultiTurnInputRequestEvent']
['nodeName'],
                    "nodeInputName": "agentInputText"
```

```
                }
            elif status == "SUCCESS":
                # The flow completed successfully.
                finished = True
                logger.info("The flow %s successfully completed.", flow_id)

    except botocore.exceptions.ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})


def main():
    """
    Main entry point for the script.
    """

    # Replace these with your actual flow ID and flow alias ID.
    FLOW_ID = 'YOUR_FLOW_ID'
    FLOW_ALIAS_ID = 'YOUR_FLOW_ALIAS_ID'

    logger.info("Starting conversation with FLOW: %s ID: %s",
                FLOW_ID, FLOW_ALIAS_ID)

    # Get the Bedrock agent runtime client.
    session = boto3.Session(profile_name='default')
    bedrock_agent_client = session.client('bedrock-agent-runtime')

    # Start the conversation.
    converse_with_flow(bedrock_agent_client, FLOW_ID, FLOW_ALIAS_ID)

    logger.info("Conversation with FLOW: %s ID: %s finished",
                FLOW_ID, FLOW_ALIAS_ID)


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie InvokeFlowin AWS SDK for Python (Boto3) API Reference.

Aktionen für Amazon Bedrock Agents Runtime mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Bedrock Agents Runtime-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden können.

Diese Ausschnitte rufen die API für Runtime der Agenten für Amazon Bedrock auf und stammen aus größeren Programmen, die im entsprechenden Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Bedrock Agents Runtime AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Runtime der Agenten für Amazon Bedrock.

Beispiele

- Verwenden Sie InvokeAgentmit einem AWS SDK
- Verwenden Sie InvokeFlowmit einem AWS SDK

Verwenden Sie **`InvokeAgent`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `InvokeAgent` verwendet wird.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  BedrockAgentRuntimeClient,
  InvokeAgentCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";
```

```
/**
 * @typedef {Object} ResponseBody
 * @property {string} completion
 */

/**
 * Invokes a Bedrock agent to run an inference using the input
 * provided in the request body.
 *
 * @param {string} prompt - The prompt that you want the Agent to complete.
 * @param {string} sessionId - An arbitrary identifier for the session.
 */
export const invokeBedrockAgent = async (prompt, sessionId) => {
  const client = new BedrockAgentRuntimeClient({ region: "us-east-1" });
  // const client = new BedrockAgentRuntimeClient({
  //   region: "us-east-1",
  //   credentials: {
  //     accessKeyId: "accessKeyId", // permission to invoke agent
  //     secretAccessKey: "accessKeySecret",
  //   },
  // });

  const agentId = "AJBHXXILZN";
  const agentAliasId = "AVKP1ITZAA";

  const command = new InvokeAgentCommand({
    agentId,
    agentAliasId,
    sessionId,
    inputText: prompt,
  });

  try {
    let completion = "";
    const response = await client.send(command);

    if (response.completion === undefined) {
      throw new Error("Completion is undefined");
    }

    for await (const chunkEvent of response.completion) {
      const chunk = chunkEvent.chunk;
      console.log(chunk);
```

```
      const decodedResponse = new TextDecoder("utf-8").decode(chunk.bytes);
      completion += decodedResponse;
    }

    return { sessionId: sessionId, completion };
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const result = await invokeBedrockAgent("I need help.", "123");
  console.log(result);
}
```

- Einzelheiten zur API finden Sie InvokeAgentin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aufrufen eines Agenten.

```
    def invoke_agent(self, agent_id, agent_alias_id, session_id, prompt):
        """
        Sends a prompt for the agent to process and respond to.

        :param agent_id: The unique identifier of the agent to use.
        :param agent_alias_id: The alias of the agent to use.
        :param session_id: The unique identifier of the session. Use the same
 value across requests
                            to continue the same conversation.
        :param prompt: The prompt that you want Claude to complete.
```

```
        :return: Inference response from the model.
        """

        try:
            # Note: The execution time depends on the foundation model,
complexity of the agent,
            # and the length of the prompt. In some cases, it can take up to a
minute or more to
            # generate a response.
            response = self.agents_runtime_client.invoke_agent(
                agentId=agent_id,
                agentAliasId=agent_alias_id,
                sessionId=session_id,
                inputText=prompt,
            )

            completion = ""

            for event in response.get("completion"):
                chunk = event["chunk"]
                completion = completion + chunk["bytes"].decode()

        except ClientError as e:
            logger.error(f"Couldn't invoke agent. {e}")
            raise

        return completion
```

- Einzelheiten zur API finden Sie InvokeAgentin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_config::{BehaviorVersion, SdkConfig};
use aws_sdk_bedrockagentruntime::{
    self as bedrockagentruntime,
    types::{error::ResponseStreamError, ResponseStream},
};
#[allow(unused_imports)]
use mockall::automock;

const BEDROCK_AGENT_ID: &str = "AJBHXXILZN";
const BEDROCK_AGENT_ALIAS_ID: &str = "AVKP1ITZAA";
const BEDROCK_AGENT_REGION: &str = "us-east-1";

#[cfg(not(test))]
pub use EventReceiverImpl as EventReceiver;
#[cfg(test)]
pub use MockEventReceiverImpl as EventReceiver;

pub struct EventReceiverImpl {
    inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
        ResponseStream,
        ResponseStreamError,
    >,
}

#[cfg_attr(test, automock)]
impl EventReceiverImpl {
    #[allow(dead_code)]
    pub fn new(
        inner:
 aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
            ResponseStream,
            ResponseStreamError,
        >,
    ) -> Self {
        Self { inner }
    }

    pub async fn recv(
        &mut self,
    ) -> Result<
        Option<ResponseStream>,
        aws_sdk_bedrockagentruntime::error::SdkError<
```

```
            ResponseStreamError,
            aws_smithy_types::event_stream::RawMessage,
        >,
    > {
        self.inner.recv().await
    }
}

#[tokio::main]
async fn main() -> Result<(), Box<bedrockagentruntime::Error>> {
    let result = invoke_bedrock_agent("I need help.".to_string(),
 "123".to_string()).await?;
    println!("{}", result);
    Ok(())
}

async fn invoke_bedrock_agent(
    prompt: String,
    session_id: String,
) -> Result<String, bedrockagentruntime::Error> {
    let sdk_config: SdkConfig = aws_config::defaults(BehaviorVersion::latest())
        .region(BEDROCK_AGENT_REGION)
        .load()
        .await;
    let bedrock_client = bedrockagentruntime::Client::new(&sdk_config);

    let command_builder = bedrock_client
        .invoke_agent()
        .agent_id(BEDROCK_AGENT_ID)
        .agent_alias_id(BEDROCK_AGENT_ALIAS_ID)
        .session_id(session_id)
        .input_text(prompt);

    let response = command_builder.send().await?;

    let response_stream = response.completion;

    let event_receiver = EventReceiver::new(response_stream);

    process_agent_response_stream(event_receiver).await
}

async fn process_agent_response_stream(
    mut event_receiver: EventReceiver,
```

```
) -> Result<String, bedrockagentruntime::Error> {
    let mut full_agent_text_response = String::new();

    while let Some(event_result) = event_receiver.recv().await? {
        match event_result {
            ResponseStream::Chunk(chunk) => {
                if let Some(bytes) = chunk.bytes {
                    match String::from_utf8(bytes.into_inner()) {
                        Ok(text_chunk) => {
                            full_agent_text_response.push_str(&text_chunk);
                        }
                        Err(e) => {
                            eprintln!("UTF-8 decoding error for chunk: {}", e);
                        }
                    }
                }
            }
            _ => {
                panic!("received an unhandled event type from Bedrock stream",);
            }
        }
    }
    Ok(full_agent_text_response)
}

#[cfg(test)]
mod test {

    use super::*;

    #[tokio::test]
    async fn test_process_agent_response_stream() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Chunk(
                    aws_sdk_bedrockagentruntime::types::PayloadPart::builder()
                        .set_bytes(Some(aws_smithy_types::Blob::new(vec![
                            116, 101, 115, 116, 32, 99, 111, 109, 112, 108, 101,
 116, 105, 111, 110,
                        ])))
                        .build(),
                ),
            ))
```

```
        });

        // end the stream
        mock.expect_recv().times(1).returning(|| Ok(None));

        let response = process_agent_response_stream(mock).await.unwrap();

        assert_eq!("test completion", response);
    }

    #[tokio::test]
    #[should_panic(expected = "received an unhandled event type from Bedrock
 stream")]
    async fn test_process_agent_response_stream_error() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Trace(

 aws_sdk_bedrockagentruntime::types::TracePart::builder().build(),
                ),
            ))
        });

        let _ = process_agent_response_stream(mock).await.unwrap();
    }
}
```

- Einzelheiten zur API finden Sie InvokeAgentin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA(lo_result) = lo_bdz->invokeagent(
     iv_agentid      = iv_agentid
       iv_agentaliasid = iv_agentaliasid
       iv_enabletrace  = abap_true
       iv_sessionid    = CONV #( cl_system_uuid=>create_uuid_c26_static( ) )
       iv_inputtext    = |Let's play "rock, paper, scissors".  I choose
rock.| ).
   DATA(lo_stream) = lo_result->get_completion( ).
   TRY.
       " loop while there are still events in the stream
       WHILE lo_stream->/aws1/if_rt_stream_reader~data_available( ) = abap_true.
         DATA(lo_evt) = lo_stream->read( ).
         " each /AWS1/CL_BDZRESPONSESTREAM_EV event contains exactly one member
         " all others are INITIAL.  For each event, process the non-initial
         " member if desired
         IF lo_evt->get_chunk( ) IS NOT INITIAL.
           " Process a Chunk event
           DATA(lv_xstr) = lo_evt->get_chunk( )->get_bytes( ).
           DATA(lv_answer) = /aws1/cl_rt_util=>xstring_to_string( lv_xstr ).
           " the answer says something like "I chose paper, so you lost"
         ELSEIF lo_evt->get_files( ) IS NOT INITIAL.
           " process a Files event if desired
         ELSEIF lo_evt->get_returncontrol( ) IS NOT INITIAL.
           " process a ReturnControl event if desired
         ELSEIF lo_evt->get_trace( ) IS NOT INITIAL.
           " process a Trace event if desired
         ENDIF.
       ENDWHILE.
       " the stream of events can possibly contain an exception
       " which will be raised to break the loop
       " catch /AWS1/CX_BDZACCESSDENIEDEX.
       " catch /AWS1/CX_BDZINTERNALSERVEREX.
       " catch /AWS1/CX_BDZMODELNOTREADYEX.
       " catch /AWS1/CX_BDZVALIDATIONEX.
       " catch /AWS1/CX_BDZTHROTTLINGEX.
       " catch /AWS1/CX_BDZDEPENDENCYFAILEDEX.
       " catch /AWS1/CX_BDZBADGATEWAYEX.
       " catch /AWS1/CX_BDZRESOURCENOTFOUNDEX.
       " catch /AWS1/CX_BDZSERVICEQUOTAEXCDEX.
       " catch /AWS1/CX_BDZCONFLICTEXCEPTION.
   ENDTRY.
```

- Einzelheiten zur API finden Sie InvokeAgentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`InvokeFlow`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `InvokeFlow` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Aufrufen eines Flows

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentRuntimeClient,
  InvokeFlowCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * Invokes an alias of a flow to run the inputs that you specify and return
 * the output of each node as a stream.
 *
 * @param {{
 *  flowIdentifier: string,
 *  flowAliasIdentifier: string,
 *  prompt?: string,
 *  region?: string
```

```
 * }} options
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").FlowNodeOutput>} An
 object containing information about the output from flow invocation.
 */
export const invokeBedrockFlow = async ({
  flowIdentifier,
  flowAliasIdentifier,
  prompt = "Hi, how are you?",
  region = "us-east-1",
}) => {
  const client = new BedrockAgentRuntimeClient({ region });

  const command = new InvokeFlowCommand({
    flowIdentifier,
    flowAliasIdentifier,
    inputs: [
      {
        content: {
          document: prompt,
        },
        nodeName: "FlowInputNode",
        nodeOutputName: "document",
      },
    ],
  });

  let flowResponse = {};
  const response = await client.send(command);

  for await (const chunkEvent of response.responseStream) {
    const { flowOutputEvent, flowCompletionEvent } = chunkEvent;

    if (flowOutputEvent) {
      flowResponse = { ...flowResponse, ...flowOutputEvent };
      console.log("Flow output event:", flowOutputEvent);
    } else if (flowCompletionEvent) {
      flowResponse = { ...flowResponse, ...flowCompletionEvent };
      console.log("Flow completion event:", flowCompletionEvent);
    }
  }

  return flowResponse;
};
```

```
// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    flowIdentifier: {
      type: "string",
      required: true,
    },
    flowAliasIdentifier: {
      type: "string",
      required: true,
    },
    prompt: {
      type: "string",
    },
    region: {
      type: "string",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    invokeBedrockFlow(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie InvokeFlowin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aufrufen eines Flows.

```
    def invoke_flow(self, flow_id, flow_alias_id, input_data, execution_id):
        """
        Invoke an Amazon Bedrock flow and handle the response stream.

        Args:
            param flow_id: The ID of the flow to invoke.
            param flow_alias_id: The alias ID of the flow.
            param input_data: Input data for the flow.
            param execution_id: Execution ID for continuing a flow. Use the value
None on first run.

        Return: Response from the flow.
        """
        try:

            request_params = None

            if execution_id is None:
                # Don't pass execution ID for first run.
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "inputs": input_data,
                    "enableTrace": True
                }
            else:
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "executionId": execution_id,
                    "inputs": input_data,
```

```
                "enableTrace": True
            }

        response = self.agents_runtime_client.invoke_flow(**request_params)

        if "executionId" not in request_params:
            execution_id = response['executionId']

        result = ""

        # Get the streaming response
        for event in response['responseStream']:
            result = result + str(event) + '\n'
        print(result)

    except ClientError as e:
        logger.error("Couldn't invoke flow %s.", {e})
        raise

    return result
```

- Einzelheiten zur API finden Sie InvokeFlowin AWS SDK for Python (Boto3) API Reference.

Szenarien für die Verwendung von Amazon Bedrock Agents Runtime AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Bedrock Agents Runtime mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Runtime der Agenten für Amazon Bedrock oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK

- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Ein durchgängiges Beispiel, das zeigt, wie ein Amazon Bedrock-Flow mit einem erstellt und aufgerufen wird AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Flow.
- Erstellen Sie den Flow.
- Stellen Sie den vollständig konfigurierten Flow bereit.
- Rufen Sie den Flow mit vom Benutzer bereitgestellten Prompts auf.
- Löschen aller erstellten Ressourcen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellt eine Musik-Playlist basierend auf dem vom Benutzenden angegebenen Genre und der Anzahl der Songs.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias
```

```
logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music
 playlist based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist
```

```
    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting
of the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
```

```
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
```

```
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the
input of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
```

```
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
```

```
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
```

```
                           flow_id,
                           flow_version,
                           flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
```

```
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()
```

```
            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
```

```
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }
```

```
def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """



    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)
```

```
    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """



    # Trust relationship policy - allows Amazon Bedrock service to assume this
 role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
```

```
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
```

```
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """


    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """
```

```
    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

Python

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finden Sie das aktuelle Repository mit den meisten Trends GitHub und fassen Sie es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step Functions

Codebeispiele für AWS Budgets verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Budgets mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS Budgets Entwicklerhandbuch — Weitere Informationen zu AWS Budgets.
- AWS Budgets API-Referenz — Details zu allen verfügbaren AWS Budgets Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Budgets verwenden AWS SDKs
  - Aktionen für AWS Budgets verwenden AWS SDKs
    - Verwendung von CreateBudget mit einer CLI

Grundlegende Beispiele für AWS Budgets verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Budgets mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS Budgets verwenden AWS SDKs
  - Verwendung von CreateBudget mit einer CLI

Aktionen für AWS Budgets verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Budgets Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Budgets -API-Referenz.

Beispiele

- Verwendung von CreateBudget mit einer CLI

Verwendung von **`CreateBudget`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateBudget` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein Nutzungsbudget

Mit dem folgenden `create-budget`-Befehl wird ein Kosten- und Nutzungsbudget erstellt.

```
aws budgets create-budget \
    --account-id 111122223333 \
    --budget file://budget.json \
    --notifications-with-subscribers file://notifications-with-subscribers.json
```

Inhalt von `budget.json`:

```
{
    "BudgetLimit": {
        "Amount": "100",
        "Unit": "USD"
    },
    "BudgetName": "Example Tag Budget",
    "BudgetType": "COST",
```

```
    "CostFilters": {
        "TagKeyValue": [
            "user:Key$value1",
            "user:Key$value2"
        ]
    },
    "CostTypes": {
        "IncludeCredit": true,
        "IncludeDiscount": true,
        "IncludeOtherSubscription": true,
        "IncludeRecurring": true,
        "IncludeRefund": true,
        "IncludeSubscription": true,
        "IncludeSupport": true,
        "IncludeTax": true,
        "IncludeUpfront": true,
        "UseBlended": false
    },
    "TimePeriod": {
        "Start": 1477958399,
        "End": 3706473600
    },
    "TimeUnit": "MONTHLY"
}
```

Inhalt von `notifications-with-subscribers.json`:

```
[
    {
        "Notification": {
            "ComparisonOperator": "GREATER_THAN",
            "NotificationType": "ACTUAL",
            "Threshold": 80,
            "ThresholdType": "PERCENTAGE"
        },
        "Subscribers": [
            {
                "Address": "example@example.com",
                "SubscriptionType": "EMAIL"
            }
        ]
    }
]
```

- Einzelheiten zur API finden Sie CreateBudgetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues Budget mit den angegebenen Budget- und Zeitvorgaben sowie E-Mail-Benachrichtigungen.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Einzelheiten zur API finden Sie unter CreateBudget AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues Budget mit den angegebenen Budget- und Zeitvorgaben sowie E-Mail-Benachrichtigungen.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
```

```
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}

$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- Einzelheiten zur API finden Sie unter CreateBudget AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für AWS Cloud Map verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Cloud Map mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS Cloud Map Entwicklerhandbuch — Weitere Informationen zu AWS Cloud Map.
- AWS Cloud Map API-Referenz — Details zu allen verfügbaren AWS Cloud Map Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für AWS Cloud Map verwenden AWS SDKs
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Fangen Sie mit Service Discovery an
  - Verwenden von Service Discovery mit benutzerdefinierten Attributen

Szenarien für AWS Cloud Map verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS Cloud Map mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS Cloud Map oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Fangen Sie mit Service Discovery an
- Verwenden von Service Discovery mit benutzerdefinierten Attributen

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
```

```
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
```

```
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"
```

```
    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
```

```
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
```

```
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
```

```
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
```

```
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
```

```
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
```

```
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
```

```
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
```

```
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure
```

```
    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
dependency issues."
    fi
```

```
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices

- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Fangen Sie mit Service Discovery an

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen öffentlichen DNS-Namespace mit gehosteter Route-53-Zonenintegration
- Erstellen Sie Services, die über DNS-Abfragen und API-Aufrufe auffindbar sind.
- Registrieren Sie Service-Instances mit unterschiedlichen Erkennungskonfigurationen.
- Entdecken Sie Services, die sowohl DNS-Abfragen als auch Cloud-Map-API verwenden.
- Überprüfen Sie die Serviceerkennung mithilfe von Dig-Befehlen und API-Aufrufen.
- Bereinigen Sie Ressourcen in der richtigen Reihenfolge (Instances, Services, Namespace).

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Cloud Map Private Namespace Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery
# with DNS queries and API calls

# Exit on error
set -e
```

```
# Configuration
REGION="us-east-2"
NAMESPACE_NAME="cloudmap-tutorial.com"
LOG_FILE="cloudmap-tutorial.log"
CREATOR_REQUEST_ID=$(date +%s)

# Function to log messages
log() {
    local message="$1"
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $message" | tee -a "$LOG_FILE"
}

# Function to check operation status
check_operation() {
    local operation_id="$1"
    local status=""

    log "Checking operation status for $operation_id..."

    while [[ "$status" != "SUCCESS" ]]; do
        sleep 5
        status=$(aws servicediscovery get-operation \
            --operation-id "$operation_id" \
            --region "$REGION" \
            --query "Operation.Status" \
            --output text)

        log "Operation status: $status"

        if [[ "$status" == "FAIL" ]]; then
            log "Operation failed. Exiting."
            exit 1
        fi
    done

    log "Operation completed successfully."
}

# Function to clean up resources
cleanup() {
    log "Starting cleanup process..."

    if [[ -n "$FIRST_INSTANCE_ID" ]]; then
```

```
        log "Deregistering first service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$PUBLIC_SERVICE_ID" \
            --instance-id "$FIRST_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister first instance"
    fi

    if [[ -n "$SECOND_INSTANCE_ID" ]]; then
        log "Deregistering second service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$BACKEND_SERVICE_ID" \
            --instance-id "$SECOND_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister second instance"
    fi

    if [[ -n "$PUBLIC_SERVICE_ID" ]]; then
        log "Deleting public service..."
        aws servicediscovery delete-service \
            --id "$PUBLIC_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete public service"
    fi

    if [[ -n "$BACKEND_SERVICE_ID" ]]; then
        log "Deleting backend service..."
        aws servicediscovery delete-service \
            --id "$BACKEND_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete backend service"
    fi

    if [[ -n "$NAMESPACE_ID" ]]; then
        log "Deleting namespace..."
        aws servicediscovery delete-namespace \
            --id "$NAMESPACE_ID" \
            --region "$REGION" || log "Failed to delete namespace"
    fi

    log "Cleanup completed."
}

# Set up trap for cleanup on script exit
trap cleanup EXIT INT TERM

# Initialize log file
> "$LOG_FILE"
```

```
log "Starting AWS Cloud Map tutorial script"

# Step 1: Create an AWS Cloud Map namespace
log "Creating AWS Cloud Map namespace: $NAMESPACE_NAME"
OPERATION_RESULT=$(aws servicediscovery create-public-dns-namespace \
    --name "$NAMESPACE_NAME" \
    --creator-request-id "cloudmap-tutorial-$CREATOR_REQUEST_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")

OPERATION_ID=$(echo "$OPERATION_RESULT" | jq -r '.OperationId')
log "Namespace creation initiated. Operation ID: $OPERATION_ID"

# Check operation status
check_operation "$OPERATION_ID"

# Get the namespace ID
log "Getting namespace ID..."
NAMESPACE_ID=$(aws servicediscovery list-namespaces \
    --region "$REGION" \
    --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" \
    --output text)

log "Namespace ID: $NAMESPACE_ID"

# Get the hosted zone ID
log "Getting Route 53 hosted zone ID..."
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
    --dns-name "$NAMESPACE_NAME" \
    --query "HostedZones[0].Id" \
    --output text | sed 's|/hostedzone/||')

log "Hosted Zone ID: $HOSTED_ZONE_ID"

# Step 2: Create the AWS Cloud Map services
log "Creating public service..."
PUBLIC_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "public-service" \
    --namespace-id "$NAMESPACE_ID" \
    --dns-config "RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=300}]" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")
```

```
PUBLIC_SERVICE_ID=$(echo "$PUBLIC_SERVICE_RESULT" | jq -r '.Service.Id')
log "Public service created. Service ID: $PUBLIC_SERVICE_ID"

log "Creating backend service..."
BACKEND_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "backend-service" \
    --namespace-id "$NAMESPACE_ID" \
    --type "HTTP" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-
discovery \
    --region "$REGION")

BACKEND_SERVICE_ID=$(echo "$BACKEND_SERVICE_RESULT" | jq -r '.Service.Id')
log "Backend service created. Service ID: $BACKEND_SERVICE_ID"

# Step 3: Register the AWS Cloud Map service instances
log "Registering first service instance..."
FIRST_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$PUBLIC_SERVICE_ID" \
    --instance-id "first" \
    --attributes "AWS_INSTANCE_IPV4=192.168.2.1" \
    --region "$REGION")

FIRST_INSTANCE_ID="first"
FIRST_OPERATION_ID=$(echo "$FIRST_INSTANCE_RESULT" | jq -r '.OperationId')
log "First instance registration initiated. Operation ID: $FIRST_OPERATION_ID"

# Check operation status
check_operation "$FIRST_OPERATION_ID"

log "Registering second service instance..."
SECOND_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$BACKEND_SERVICE_ID" \
    --instance-id "second" \
    --attributes "service-name=backend" \
    --region "$REGION")

SECOND_INSTANCE_ID="second"
SECOND_OPERATION_ID=$(echo "$SECOND_INSTANCE_RESULT" | jq -r '.OperationId')
log "Second instance registration initiated. Operation ID: $SECOND_OPERATION_ID"

# Check operation status
check_operation "$SECOND_OPERATION_ID"
```

```
# Step 4: Discover the AWS Cloud Map service instances
log "Getting Route 53 name servers..."
NAME_SERVERS=$(aws route53 get-hosted-zone \
    --id "$HOSTED_ZONE_ID" \
    --query "DelegationSet.NameServers[0]" \
    --output text)

log "Name server: $NAME_SERVERS"

log "Using dig to query DNS records (this will be simulated)..."
log "Command: dig @$NAME_SERVERS public-service.$NAMESPACE_NAME"
log "Expected output would show: public-service.$NAMESPACE_NAME. 300 IN A
 192.168.2.1"

log "Using AWS CLI to discover backend service instances..."
DISCOVER_RESULT=$(aws servicediscovery discover-instances \
    --namespace-name "$NAMESPACE_NAME" \
    --service-name "backend-service" \
    --region "$REGION")

log "Discovery result: $(echo "$DISCOVER_RESULT" | jq -c '.')"

# Display created resources
log "Resources created:"
log "- Namespace: $NAMESPACE_NAME (ID: $NAMESPACE_ID)"
log "- Public Service: public-service (ID: $PUBLIC_SERVICE_ID)"
log "- Backend Service: backend-service (ID: $BACKEND_SERVICE_ID)"
log "- Service Instance: first (Service: public-service)"
log "- Service Instance: second (Service: backend-service)"

# Ask user if they want to clean up resources
read -p "Do you want to clean up all created resources? (y/n): " CLEANUP_RESPONSE

if [[ "$CLEANUP_RESPONSE" == "y" || "$CLEANUP_RESPONSE" == "Y" ]]; then
    log "User confirmed cleanup. Proceeding with resource deletion."
    # Cleanup function will be called automatically on exit
else
    log "User chose not to clean up resources. Exiting without cleanup."
    trap - EXIT
    exit 0
fi
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreatePublicDnsNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - DiscoverInstances
  - GetOperation
  - ListNamespaces
  - RegisterInstance

Verwenden von Service Discovery mit benutzerdefinierten Attributen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einen Cloud Map-Namespace erstellen
- Erstellen einer DynamoDB-Tabelle
- Erstellen Sie einen Cloud Map-Datendienst und registrieren Sie die DynamoDB-Tabelle
- Erstellen Sie eine IAM-Rolle für Lambda-Funktionen
- Erstellen Sie die Lambda-Funktion zum Schreiben von Daten
- Erstellen Sie einen Cloud Map-App-Service und registrieren Sie die Lambda-Schreibfunktion
- Erstellen Sie die Lambda-Funktion zum Lesen von Daten

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Cloud Map Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery with
 custom attributes

set -euo pipefail

# Set up logging
LOG_FILE="cloudmap-tutorial.log"
echo "AWS Cloud Map Tutorial Script" > "$LOG_FILE"
echo "Started at $(date)" >> "$LOG_FILE"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log commands and their output
log_cmd() {
  echo "$ $1" | tee -a "$LOG_FILE"
  eval "$1" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
  local LINE=$1
  echo "An error occurred at line $LINE" | tee -a "$LOG_FILE"
  echo "Resources created so far:" | tee -a "$LOG_FILE"
  for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
  done
  echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"
  cleanup
  exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Helper function to wait for Cloud Map operations to complete
wait_for_operation() {
  local OPERATION_ID=$1
  local TIMEOUT=300  # 5 minutes timeout
  local START_TIME
```

```
  START_TIME=$(date +%s)

  while true; do
    local STATUS
    STATUS=$(aws servicediscovery get-operation --operation-id "$OPERATION_ID" --
query 'Operation.Status' --output text 2>/dev/null || echo "UNKNOWN")

    if [ "$STATUS" == "SUCCESS" ]; then
      echo "Operation completed successfully" | tee -a "$LOG_FILE"
      break
    elif [ "$STATUS" == "FAIL" ]; then
      echo "Operation failed" | tee -a "$LOG_FILE"
      return 1
    fi

    local CURRENT_TIME
    CURRENT_TIME=$(date +%s)
    if [ $((CURRENT_TIME - START_TIME)) -gt $TIMEOUT ]; then
      echo "Operation timed out" | tee -a "$LOG_FILE"
      return 1
    fi

    sleep 5
  done

  return 0
}

# Function to clean up resources
cleanup() {
  echo "Cleaning up resources..." | tee -a "$LOG_FILE"

  # Reverse the order of created resources for proper deletion
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    echo "Deleting $resource..." | tee -a "$LOG_FILE"

    if [[ $resource == "instance:"* ]]; then
      # Extract service ID and instance ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      INSTANCE_ID=$(echo "$resource" | cut -d':' -f3)

      # Check if instance exists before trying to deregister
```

```
      INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$SERVICE_ID" --query "Instances[?Id=='$INSTANCE_ID'].Id" --output text 2>/dev/
null || echo "")
      if [[ -n "$INSTANCE_EXISTS" ]]; then
        OPERATION_ID=$(aws servicediscovery deregister-instance --service-id
 "$SERVICE_ID" --instance-id "$INSTANCE_ID" --query 'OperationId' --output text)

        # Wait for deregistration to complete
        echo "Waiting for instance deregistration to complete..." | tee -a
 "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Instance $INSTANCE_ID already deregistered" | tee -a "$LOG_FILE"
      fi
    elif [[ $resource == "lambda:"* ]]; then
      # Extract function name
      FUNCTION_NAME=$(echo "$resource" | cut -d':' -f2)
      aws lambda delete-function --function-name "$FUNCTION_NAME" 2>/dev/null ||
 echo "Function already deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "role:"* ]]; then
      # Extract role name
      ROLE_NAME=$(echo "$resource" | cut -d':' -f2)

      # Detach all policies first
      for POLICY_ARN in $(aws iam list-attached-role-policies --role-name
 "$ROLE_NAME" --query 'AttachedPolicies[*].PolicyArn' --output text); do
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
      done

      # Delete the role
      aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Role
 already deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "dynamodb:"* ]]; then
      # Extract table name
      TABLE_NAME=$(echo "$resource" | cut -d':' -f2)
      aws dynamodb delete-table --table-name "$TABLE_NAME" 2>/dev/null || echo
 "Table already deleted" | tee -a "$LOG_FILE"

      # Wait for table deletion to complete
      echo "Waiting for DynamoDB table deletion to complete..." | tee -a
 "$LOG_FILE"
      aws dynamodb wait table-not-exists --table-name "$TABLE_NAME" 2>/dev/null
 || true
```

```
    fi
  done

  # Handle services separately to ensure all instances are deregistered first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "service:"* ]]; then
      # Extract service ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"

      # Make sure all instances are deregistered
      INSTANCES=$(aws servicediscovery list-instances --service-id "$SERVICE_ID"
--query 'Instances[*].Id' --output text 2>/dev/null || echo "")
      if [[ -n "$INSTANCES" ]]; then
        echo "Service still has instances. Waiting before deletion..." | tee -a
"$LOG_FILE"
        sleep 10
      fi

      # Try to delete the service
      aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || echo
"Service already deleted" | tee -a "$LOG_FILE"
      sleep 5
    fi
  done

  # Handle namespaces last to ensure all services are deleted first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "namespace:"* ]]; then
      # Extract namespace ID
      NAMESPACE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"

      # Check if namespace still has services
      SERVICES=$(aws servicediscovery list-services --filters
"Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].Id'
--output text 2>/dev/null || echo "")
      if [[ -n "$SERVICES" ]]; then
        echo "Namespace still has services. Deleting them first..." | tee -a
"$LOG_FILE"
        for SERVICE_ID in $SERVICES; do
          echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"
```

```
          aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null ||
 true
        done
        sleep 5
      fi

      # Try to delete the namespace
      OPERATION_ID=$(aws servicediscovery delete-namespace --id "$NAMESPACE_ID"
 --query 'OperationId' --output text 2>/dev/null || echo "")
      if [[ -n "$OPERATION_ID" ]]; then
        echo "Waiting for namespace deletion to complete..." | tee -a "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Failed to delete namespace or namespace already deleted" | tee -a
 "$LOG_FILE"
      fi
    fi
  done

  echo "Cleanup complete" | tee -a "$LOG_FILE"
}

# Step 1: Create an AWS Cloud Map namespace
echo "Step 1: Creating AWS Cloud Map namespace..." | tee -a "$LOG_FILE"

# Check if namespace already exists
NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$NAMESPACE_ID" || "$NAMESPACE_ID" == "None" ]]; then
  log_cmd "aws servicediscovery create-http-namespace --name cloudmap-tutorial --
creator-request-id namespace-request-\$(date +%s)"
  OPERATION_ID=$(aws servicediscovery create-http-namespace --name cloudmap-
tutorial --creator-request-id "namespace-request-$(date +%s)" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes --
query 'OperationId' --output text)

  # Wait for namespace creation to complete
  echo "Waiting for namespace creation to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"

  # Get the namespace ID
  NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text)
```

```
  echo "Namespace created with ID: $NAMESPACE_ID" | tee -a "$LOG_FILE"
else
  echo "Namespace cloudmap-tutorial already exists with ID: $NAMESPACE_ID" | tee
 -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("namespace:$NAMESPACE_ID")

# Step 2: Create a DynamoDB table
echo "Step 2: Creating DynamoDB table..." | tee -a "$LOG_FILE"

# Check if table already exists
TABLE_EXISTS=$(aws dynamodb describe-table --table-name cloudmap 2>&1 || echo
 "NOT_EXISTS")

if [[ $TABLE_EXISTS == *"ResourceNotFoundException"* || $TABLE_EXISTS ==
 "NOT_EXISTS" ]]; then
  log_cmd "aws dynamodb create-table --table-name cloudmap --
attribute-definitions AttributeName=id,AttributeType=S --key-schema
 AttributeName=id,KeyType=HASH --billing-mode PAY_PER_REQUEST --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes"

  # Wait for DynamoDB table to become active
  echo "Waiting for DynamoDB table to become active..." | tee -a "$LOG_FILE"
  aws dynamodb wait table-exists --table-name cloudmap
else
  echo "DynamoDB table cloudmap already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("dynamodb:cloudmap")

# Step 3: Create an AWS Cloud Map data service
echo "Step 3: Creating AWS Cloud Map data service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
echo "Listing all services in namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"
SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].
[Id,Name]' --output text 2>/dev/null || echo "")
echo "Services found: $SERVICES" | tee -a "$LOG_FILE"

# Check if data service already exists
DATA_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
```

```
  echo "Checking service: ID=$id, Name=$name" | tee -a "$LOG_FILE"
  if [[ "$name" == "data-service" ]]; then
    DATA_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$DATA_SERVICE_ID" ]]; then
  echo "Data service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name data-service --namespace-id
 $NAMESPACE_ID --creator-request-id data-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name data-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "data-
service-request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  DATA_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='data-service'].Id" --output text)
  echo "Data service created with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "Data service already exists with ID: $DATA_SERVICE_ID" | tee -a
 "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$DATA_SERVICE_ID")

# Register DynamoDB table as a service instance
echo "Registering DynamoDB table as a service instance..." | tee -a "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$DATA_SERVICE_ID" --query "Instances[?Id=='data-instance'].Id" --output text
 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  AWS_REGION=$(aws configure get region || echo "us-east-1")
  log_cmd "aws servicediscovery register-instance --service-id $DATA_SERVICE_ID
 --instance-id data-instance --attributes tablename=cloudmap,region=$AWS_REGION"
```

```
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$DATA_SERVICE_ID" --instance-id data-instance --attributes
 "tablename=cloudmap,region=$AWS_REGION" --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance data-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$DATA_SERVICE_ID:data-instance")

# Step 4: Create an IAM role for Lambda
echo "Step 4: Creating IAM role for Lambda..." | tee -a "$LOG_FILE"

# Create a trust policy for Lambda
cat > lambda-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Check if role already exists
echo "Checking if IAM role already exists..." | tee -a "$LOG_FILE"
ROLE_EXISTS=$(aws iam get-role --role-name cloudmap-tutorial-role 2>&1 || echo
 "NOT_EXISTS")

if [[ $ROLE_EXISTS == *"NoSuchEntity"* || $ROLE_EXISTS == "NOT_EXISTS" ]]; then
    log_cmd "aws iam create-role --role-name cloudmap-tutorial-role --assume-
role-policy-document file://lambda-trust-policy.json"
    aws iam tag-role --role-name cloudmap-tutorial-role --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes
else
```

```
    echo "Role cloudmap-tutorial-role already exists, using existing role" | tee
 -a "$LOG_FILE"
fi

# Create a custom policy with least privilege
cat > cloudmap-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:GetItem",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/cloudmap"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicediscovery:DiscoverInstances"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Check if policy already exists
POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text 2>/dev/null || echo "")

if [[ -z "$POLICY_ARN" ]]; then
  echo "Creating CloudMapTutorialPolicy..." | tee -a "$LOG_FILE"
```

```
  echo "$ aws iam create-policy --policy-name CloudMapTutorialPolicy --policy-
document file://cloudmap-policy.json" | tee -a "$LOG_FILE"
  CREATE_OUTPUT=$(aws iam create-policy --policy-name CloudMapTutorialPolicy --
policy-document file://cloudmap-policy.json)
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"
  POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text)
else
  echo "Policy CloudMapTutorialPolicy already exists with ARN: $POLICY_ARN" | tee
 -a "$LOG_FILE"
fi

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn $POLICY_ARN" | tee -a "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 "$POLICY_ARN" | tee -a "$LOG_FILE"

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" | tee -a
 "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole | tee -a
 "$LOG_FILE"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

ROLE_ARN=$(aws iam get-role --role-name cloudmap-tutorial-role --query 'Role.Arn'
 --output text)
CREATED_RESOURCES+=("role:cloudmap-tutorial-role")

# Step 5: Create an AWS Cloud Map app service
echo "Step 5: Creating AWS Cloud Map app service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].
[Id,Name]' --output text 2>/dev/null || echo "")

# Check if app service already exists
APP_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  if [[ "$name" == "app-service" ]]; then
```

```
    APP_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$APP_SERVICE_ID" ]]; then
  echo "App service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name app-service --namespace-id
 $NAMESPACE_ID --creator-request-id app-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name app-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "app-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  APP_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='app-service'].Id" --output text)
  echo "App service created with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "App service already exists with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$APP_SERVICE_ID")

# Step 6: Create a Lambda function to write data
echo "Step 6: Creating Lambda function to write data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > writefunction.py << 'EOF'
import boto3
import json
import random
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
```

```
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-
east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Write data to the table
        table.put_item(
            Item={
                'id': str(random.randint(1, 100)),
                'todo': str(event)
            }
        )

        return {
            'statusCode': 200,
            'body': json.dumps('Data written successfully!')
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip writefunction.zip writefunction.py"

# Create the Lambda function
```

```
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='writefunction'].FunctionName" --output text 2>/dev/null || echo
 "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name writefunction --runtime
 python3.12 --role $ROLE_ARN --handler writefunction.lambda_handler --zip-
file fileb://writefunction.zip --architectures x86_64 --timeout 10 --tags
 project=doc-smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name writefunction --
query 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function writefunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:writefunction")

# Step 7: Register the Lambda write function as an AWS Cloud Map service instance
echo "Step 7: Registering Lambda write function as a service instance..." | tee -
a "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$APP_SERVICE_ID" --query "Instances[?Id=='write-instance'].Id" --output text
 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id write-instance --attributes action=write,functionname=writefunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id write-instance --attributes
 action=write,functionname=writefunction --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for write instance registration to complete..." | tee -a
 "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
```

```
  echo "Instance write-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:write-instance")

# Step 8: Create a Lambda function to read data
echo "Step 8: Creating Lambda function to read data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > readfunction.py << 'EOF'
import boto3
import json
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-
east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Read data from the table
        response = table.scan()
```

```
        return {
            'statusCode': 200,
            'body': json.dumps(response.get('Items', []))
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip readfunction.zip readfunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='readfunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name readfunction --runtime
 python3.12 --role $ROLE_ARN --handler readfunction.lambda_handler --zip-file
 fileb://readfunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name readfunction --
query 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function readfunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:readfunction")

# Step 9: Register the Lambda read function as an AWS Cloud Map service instance
echo "Step 9: Registering Lambda read function as a service instance..." | tee -a
 "$LOG_FILE"

# Check if instance already exists
```

```
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$APP_SERVICE_ID" --query "Instances[?Id=='read-instance'].Id" --output text 2>/
dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id read-instance --attributes action=read,functionname=readfunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id read-instance --attributes
 action=read,functionname=readfunction --query 'OperationId' --output text)

  # Wait for read instance registration to complete
  echo "Waiting for read instance registration to complete..." | tee -a
 "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance read-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:read-instance")

# Step 10: Create Python clients to interact with the services
echo "Step 10: Creating Python clients..." | tee -a "$LOG_FILE"

cat > writeclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'write'}
    )

    if not response.get('Instances'):
        print("No write service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]
```

```
    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        Payload=json.dumps('This is a test data'),
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

cat > readclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'read'}
    )

    if not response.get('Instances'):
        print("No read service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]

    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
```

```
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

chmod +x writeclient.py readclient.py

echo "Running write client..." | tee -a "$LOG_FILE"
log_cmd "python3 writeclient.py" || echo "Write client execution completed with
 warnings" | tee -a "$LOG_FILE"

echo "Running read client..." | tee -a "$LOG_FILE"
log_cmd "python3 readclient.py" || echo "Read client execution completed with
 warnings" | tee -a "$LOG_FILE"

# Step 11: Clean up resources
echo "Resources created:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]}"; do
  echo "- $resource" | tee -a "$LOG_FILE"
done

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "Script completed at $(date)" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateHttpNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - GetOperation
  - ListNamespaces
  - ListServices

- RegisterInstance

Codebeispiele für AWS Cloud9 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Cloud9 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS Cloud9 Benutzerhandbuch — Weitere Informationen zu AWS Cloud9.
- AWS Cloud9 API-Referenz — Details zu allen verfügbaren AWS Cloud9 Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Cloud9 verwenden AWS SDKs
  - Aktionen für AWS Cloud9 verwenden AWS SDKs
    - Verwendung von CreateEnvironmentEc2 mit einer CLI
    - Verwendung von CreateEnvironmentMembership mit einer CLI
    - Verwendung von DeleteEnvironment mit einer CLI
    - Verwendung von DeleteEnvironmentMembership mit einer CLI
    - Verwendung von DescribeEnvironmentMemberships mit einer CLI
    - Verwendung von DescribeEnvironmentStatus mit einer CLI
    - Verwendung von DescribeEnvironments mit einer CLI
    - Verwendung von ListEnvironments mit einer CLI
    - Verwendung von UpdateEnvironment mit einer CLI
    - Verwendung von UpdateEnvironmentMembership mit einer CLI

Grundlegende Beispiele für AWS Cloud9 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Cloud9 mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS Cloud9 verwenden AWS SDKs
  - Verwendung von CreateEnvironmentEc2 mit einer CLI
  - Verwendung von CreateEnvironmentMembership mit einer CLI
  - Verwendung von DeleteEnvironment mit einer CLI
  - Verwendung von DeleteEnvironmentMembership mit einer CLI
  - Verwendung von DescribeEnvironmentMemberships mit einer CLI
  - Verwendung von DescribeEnvironmentStatus mit einer CLI
  - Verwendung von DescribeEnvironments mit einer CLI
  - Verwendung von ListEnvironments mit einer CLI
  - Verwendung von UpdateEnvironment mit einer CLI
  - Verwendung von UpdateEnvironmentMembership mit einer CLI

Aktionen für AWS Cloud9 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Cloud9 Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Cloud9 -API-Referenz.

Beispiele

- Verwendung von CreateEnvironmentEc2 mit einer CLI
- Verwendung von CreateEnvironmentMembership mit einer CLI
- Verwendung von DeleteEnvironment mit einer CLI
- Verwendung von DeleteEnvironmentMembership mit einer CLI
- Verwendung von DescribeEnvironmentMemberships mit einer CLI
- Verwendung von DescribeEnvironmentStatus mit einer CLI

- Verwendung von DescribeEnvironments mit einer CLI
- Verwendung von ListEnvironments mit einer CLI
- Verwendung von UpdateEnvironment mit einer CLI
- Verwendung von UpdateEnvironmentMembership mit einer CLI

Verwendung von **`CreateEnvironmentEc2`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateEnvironmentEc2` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine AWS Cloud9 EC2-Entwicklungsumgebung

Das folgende `create-environment-ec2` Beispiel erstellt eine AWS Cloud9-Entwicklungsumgebung mit den angegebenen Einstellungen, startet eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance und stellt dann eine Verbindung von der Instance zur Umgebung her.

```
aws cloud9 create-environment-ec2 \
    --name my-demo-env \
    --description "My demonstration development environment." \
    --instance-type t2.micro --image-id amazonlinux-2023-x86_64 \
    --subnet-id subnet-1fab8aEX \
    --automatic-stop-time-minutes 60 \
    --owner-arn arn:aws:iam::123456789012:user/MyDemoUser
```

Ausgabe:

```
{
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX"
}
```

Weitere Informationen finden Sie unter Erstellen einer EC2-Umgebung im Benutzerhandbuch für AWS Cloud9.

- Einzelheiten zur API finden Sie CreateEnvironmentEc2in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine AWS Cloud9-Entwicklungsumgebung mit den angegebenen Einstellungen erstellt, eine Amazon Elastic Compute Cloud (Amazon EC2) - Instance gestartet und dann eine Verbindung von der Instance zur Umgebung hergestellt.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Ausgabe:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentEc2 AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine AWS Cloud9-Entwicklungsumgebung mit den angegebenen Einstellungen erstellt, eine Amazon Elastic Compute Cloud (Amazon EC2) - Instance gestartet und dann eine Verbindung von der Instance zur Umgebung hergestellt.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Ausgabe:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentEc2 AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateEnvironmentMembership`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateEnvironmentMembership` verwendet wird.

CLI

AWS CLI

Um ein Umgebungsmitglied zu einer AWS Cloud9-Entwicklungsumgebung hinzuzufügen

In diesem Beispiel wird das angegebene Umgebungsmitglied zur angegebenen AWS Cloud9-Entwicklungsumgebung hinzugefügt.

Befehl:

```
aws cloud9 create-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser --permissions read-write
```

Ausgabe:

```
{
  "membership": {
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
    "userId": "AIDAJ3LOROMOUXTBSU6EX",
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
    "permissions": "read-write"
  }
}
```

- Einzelheiten zur API finden Sie CreateEnvironmentMembershipin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied zur angegebenen AWS Cloud9-Entwicklungsumgebung hinzugefügt.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
write
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied zur angegebenen AWS Cloud9-Entwicklungsumgebung hinzugefügt.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
write
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter CreateEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteEnvironment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteEnvironment` verwendet wird.

CLI

AWS CLI

Um eine AWS Cloud9-Entwicklungsumgebung zu löschen

In diesem Beispiel wird die angegebene AWS Cloud9-Entwicklungsumgebung gelöscht. Wenn die Umgebung mit einer Amazon-EC2-Instance verbunden ist, wird auch die Instance beendet.

Befehl:

```
aws cloud9 delete-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DeleteEnvironmentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene AWS Cloud9-Entwicklungsumgebung gelöscht. Wenn die Umgebung mit einer Amazon-EC2-Instance verbunden ist, wird auch die Instance beendet.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene AWS Cloud9-Entwicklungsumgebung gelöscht. Wenn die Umgebung mit einer Amazon-EC2-Instance verbunden ist, wird auch die Instance beendet.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteEnvironmentMembership`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteEnvironmentMembership` verwendet wird.

CLI

AWS CLI

Um ein Umgebungsmitglied aus einer AWS Cloud9-Entwicklungsumgebung zu löschen

In diesem Beispiel wird das angegebene Umgebungsmitglied aus der angegebenen AWS Cloud9-Entwicklungsumgebung gelöscht.

Befehl:

```
aws cloud9 delete-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DeleteEnvironmentMembershipin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied aus der angegebenen AWS Cloud9-Entwicklungsumgebung gelöscht.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied aus der angegebenen AWS Cloud9-Entwicklungsumgebung gelöscht.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- Einzelheiten zur API finden Sie unter DeleteEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeEnvironmentMemberships`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEnvironmentMemberships` verwendet wird.

CLI

AWS CLI

Um Informationen über Umgebungsmitglieder für eine AWS Cloud9-Entwicklungsumgebung abzurufen

In diesem Beispiel werden Informationen über Umgebungsmitglieder für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

Befehl:

```
aws cloud9 describe-environment-memberships --environment-
id 8a34f51ce1e04a08882f1e811bd706EX
```

Ausgabe:

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJ3LOROMOUXTBSU6EX",
      "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
      "permissions": "read-write"
    },
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
```

```
  ]
}
```

Um Informationen über den Besitzer einer AWS Cloud9-Entwicklungsumgebung zu erhalten

In diesem Beispiel werden Informationen über den Besitzer der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

Befehl:

```
aws cloud9 describe-environment-memberships --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --permissions owner
```

Ausgabe:

```
{
  "memberships": [
    {
      "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

Um Informationen über ein Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen zu erhalten

In diesem Beispiel werden Informationen über das angegebene Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen abgerufen.

Befehl:

```
aws cloud9 describe-environment-memberships --user-
arn arn:aws:iam::123456789012:user/MyDemoUser
```

Ausgabe:

```
{
  "memberships": [
    {
```

```
      "environmentId": "10a75714bd494714929e7f5ec4125aEX",
      "lastAccess": 1516213427.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    },
    {
      "environmentId": "1980b80e5f584920801c09086667f0EX",
      "lastAccess": 1516144884.0,
      "userId": "AIDAJNUEDQAQWFELJDLEX",
      "userArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "permissions": "owner"
    }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeEnvironmentMemberships in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über Umgebungsmitglieder für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 2: In diesem Beispiel werden Informationen über den Besitzer der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 3: In diesem Beispiel werden Informationen über das angegebene Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/
MyDemoUser
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentMemberships AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über Umgebungsmitglieder für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 2: In diesem Beispiel werden Informationen über den Besitzer der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 3: In diesem Beispiel werden Informationen über das angegebene Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/
MyDemoUser
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
```

```
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentMemberships AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeEnvironmentStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEnvironmentStatus` verwendet wird.

CLI

AWS CLI

Um Statusinformationen für eine AWS Cloud9-Entwicklungsumgebung abzurufen

In diesem Beispiel werden Statusinformationen für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

Befehl:

```
aws cloud9 describe-environment-status --environment-
id 685f892f431b45c2b28cb69eadcdb0EX
```

Ausgabe:

```
{
  "status": "ready",
  "message": "Environment is ready to use"
}
```

- Einzelheiten zur API finden Sie DescribeEnvironmentStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Statusinformationen für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Ausgabe:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Statusinformationen für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Ausgabe:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- Einzelheiten zur API finden Sie unter DescribeEnvironmentStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeEnvironments`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEnvironments` verwendet wird.

CLI

AWS CLI

Um Informationen über AWS Cloud9-Entwicklungsumgebungen zu erhalten

In diesem Beispiel werden Informationen zu den angegebenen AWS Cloud9-Entwicklungsumgebungen abgerufen.

Befehl:

```
aws cloud9 describe-environments --environment-
ids 685f892f431b45c2b28cb69eadcdb0EX 349c86d4579e4e7298d500ff57a6b2EX
```

Ausgabe:

```
{
  "environments": [
    {
      "id": "685f892f431b45c2b28cb69eadcdb0EX",
      "name": "my-demo-ec2-env",
      "description": "Created from CodeStar.",
      "type": "ec2",
      "arn": "arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    },
    {
      "id": "349c86d4579e4e7298d500ff57a6b2EX",
      "name": my-demo-ssh-env",
      "description": "",
      "type": "ssh",
      "arn": "arn:aws:cloud9:us-
east-1:123456789012:environment:349c86d4579e4e7298d500ff57a6b2EX",
      "ownerArn": "arn:aws:iam::123456789012:user/MyDemoUser",
      "lifecycle": {
        "status": "CREATED"
      }
    }
  ]
```

```
}
```

- Einzelheiten zur API finden Sie DescribeEnvironmentsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Ausgabe:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Beispiel 2: In diesem Beispiel werden Informationen über den Lebenszyklusstatus der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Ausgabe:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Einzelheiten zur API finden Sie unter DescribeEnvironments AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Ausgabe:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Beispiel 2: In diesem Beispiel werden Informationen über den Lebenszyklusstatus der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Ausgabe:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- Einzelheiten zur API finden Sie unter DescribeEnvironments AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListEnvironments`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListEnvironments` verwendet wird.

CLI

AWS CLI

Um eine Liste der verfügbaren AWS Cloud9-Entwicklungsumgebungskennungen zu erhalten

In diesem Beispiel wird eine Liste verfügbarer AWS Cloud9-Entwicklungsumgebungskennungen abgerufen.

Befehl:

```
aws cloud9 list-environments
```

Ausgabe:

```
{
  "environmentIds": [
    "685f892f431b45c2b28cb69eadcdb0EX",
    "1980b80e5f584920801c09086667f0EX"
  ]
}
```

- Einzelheiten zur API finden Sie ListEnvironmentsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Identifikatoren für die AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentList
```

Ausgabe:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Einzelheiten zur API finden Sie unter ListEnvironments AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Identifikatoren für die AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentList
```

Ausgabe:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- Einzelheiten zur API finden Sie unter ListEnvironments AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateEnvironment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateEnvironment` verwendet wird.

CLI

AWS CLI

Um die Einstellungen einer vorhandenen AWS Cloud9-Entwicklungsumgebung zu ändern

In diesem Beispiel werden die angegebenen Einstellungen der angegebenen vorhandenen AWS Cloud9-Entwicklungsumgebung geändert.

Befehl:

```
aws cloud9 update-environment --environment-id 8a34f51ce1e04a08882f1e811bd706EX
 --name my-changed-demo-env --description "My changed demonstration development
 environment."
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie UpdateEnvironmentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die angegebenen Einstellungen der angegebenen vorhandenen AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Einzelheiten zur API finden Sie unter UpdateEnvironment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die angegebenen Einstellungen der angegebenen vorhandenen AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- Einzelheiten zur API finden Sie unter UpdateEnvironment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateEnvironmentMembership`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateEnvironmentMembership` verwendet wird.

CLI

AWS CLI

Um die Einstellungen eines vorhandenen Umgebungsmitglieds für eine AWS Cloud9-Entwicklungsumgebung zu ändern

In diesem Beispiel werden die Einstellungen des angegebenen vorhandenen Umgebungsmitglieds für die angegebene AWS Cloud9-Entwicklungsumgebung geändert.

Befehl:

```
aws cloud9 update-environment-membership --environment-
id 8a34f51ce1e04a08882f1e811bd706EX --user-arn arn:aws:iam::123456789012:user/
AnotherDemoUser --permissions read-only
```

Ausgabe:

```
{
  "membership": {
    "environmentId": "8a34f51ce1e04a08882f1e811bd706EX",
    "userId": "AIDAJ3LOROMOUXTBSU6EX",
    "userArn": "arn:aws:iam::123456789012:user/AnotherDemoUser",
    "permissions": "read-only"
  }
}
```

- Einzelheiten zur API finden Sie unter UpdateEnvironmentMembership AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Einstellungen des angegebenen vorhandenen Umgebungsmitglieds für die angegebene AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter UpdateEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Einstellungen des angegebenen vorhandenen Umgebungsmitglieds für die angegebene AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- Einzelheiten zur API finden Sie unter UpdateEnvironmentMembership AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für CloudFormation verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CloudFormation mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- CloudFormation Benutzerhandbuch — Weitere Informationen zu CloudFormation.
- CloudFormation API-Referenz — Details zu allen verfügbaren CloudFormation Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für CloudFormation verwenden AWS SDKs
  - Hallo CloudFormation
  - Aktionen für CloudFormation verwenden AWS SDKs
    - Verwendung von CancelUpdateStack mit einer CLI
    - Verwendung von ContinueUpdateRollback mit einer CLI
    - Verwendung von CreateStack mit einer CLI
    - Verwendung von DeleteStack mit einer CLI
    - Verwendung von DescribeStackEvents mit einer CLI
    - Verwendung von DescribeStackResource mit einer CLI
    - Verwendung von DescribeStackResources mit einer CLI
    - Verwenden Sie mit einem DescribeStacks AWS SDK oder CLI
    - Verwendung von EstimateTemplateCost mit einer CLI
    - Verwendung von GetTemplate mit einer CLI
    - Verwendung von ListStackResources mit einer CLI
    - Verwendung von ListStacks mit einer CLI
    - Verwendung von UpdateStack mit einer CLI
    - Verwendung von ValidateTemplate mit einer CLI
- Szenarien für CloudFormation verwenden AWS SDKs
  - Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
  - Erstellen Sie Ihren ersten Infrastruktur-Stack

- Erste Schritte mit verwalteten Kubernetes-Clustern

Grundlegende Beispiele für CloudFormation verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CloudFormation mit AWS SDKs verwendet werden.

Beispiele

- Hallo CloudFormation
- Aktionen für CloudFormation verwenden AWS SDKs
  - Verwendung von CancelUpdateStack mit einer CLI
  - Verwendung von ContinueUpdateRollback mit einer CLI
  - Verwendung von CreateStack mit einer CLI
  - Verwendung von DeleteStack mit einer CLI
  - Verwendung von DescribeStackEvents mit einer CLI
  - Verwendung von DescribeStackResource mit einer CLI
  - Verwendung von DescribeStackResources mit einer CLI
  - Verwenden Sie mit einem DescribeStacks AWS SDK oder CLI
  - Verwendung von EstimateTemplateCost mit einer CLI
  - Verwendung von GetTemplate mit einer CLI
  - Verwendung von ListStackResources mit einer CLI
  - Verwendung von ListStacks mit einer CLI
  - Verwendung von UpdateStack mit einer CLI
  - Verwendung von ValidateTemplate mit einer CLI

Hallo CloudFormation

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudFormation beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Runtime;

namespace CloudFormationActions;

public static class HelloCloudFormation
{
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    static async Task Main(string[] args)
    {
        // Create the CloudFormation client
        _amazonCloudFormation = new AmazonCloudFormationClient();
        Console.WriteLine($"\nIn Region:
 {_amazonCloudFormation.Config.RegionEndpoint}");

        // List the resources for each stack
        await ListResources();
    }

    /// <summary>
    /// Method to list stack resources and other information.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> ListResources()
    {
        try
        {
            Console.WriteLine("Getting CloudFormation stack information...");

            // Get all stacks using the stack paginator.
```

```
            var paginatorForDescribeStacks =
                _amazonCloudFormation.Paginators.DescribeStacks(
                    new DescribeStacksRequest());
            if (paginatorForDescribeStacks.Stacks != null)
            {
                await foreach (Stack stack in paginatorForDescribeStacks.Stacks)
                {
                    // Basic information for each stack
                    Console.WriteLine(
                        "\n-------------------------------------------------");
                    Console.WriteLine($"\nStack: {stack.StackName}");
                    Console.WriteLine($"  Status: {stack.StackStatus.Value}");
                    Console.WriteLine($"  Created: {stack.CreationTime}");

                    // The tags of each stack (etc.)
                    if (stack.Tags != null && stack.Tags.Count > 0)
                    {
                        Console.WriteLine("  Tags:");
                        foreach (Tag tag in stack.Tags)
                            Console.WriteLine($"    {tag.Key}, {tag.Value}");
                    }

                    // The resources of each stack
                    DescribeStackResourcesResponse responseDescribeResources =
                        await _amazonCloudFormation.DescribeStackResourcesAsync(
                            new DescribeStackResourcesRequest
                            {
                                StackName = stack.StackName
                            });
                    if (responseDescribeResources.StackResources != null &&
responseDescribeResources.StackResources.Count > 0)
                    {
                        Console.WriteLine("  Resources:");
                        foreach (StackResource resource in
responseDescribeResources
                                     .StackResources)
                            Console.WriteLine(
                                $"    {resource.LogicalResourceId}:
{resource.ResourceStatus}");
                    }
                }
            }
```

```
Console.WriteLine("\n-----------------------------------------------");
            return true;
        }
        catch (AmazonCloudFormationException ex)
        {
            Console.WriteLine("Unable to get stack information:\n" + ex.Message);
            return false;
        }
        catch (AmazonServiceException ex)
        {
            if (ex.Message.Contains("Unable to get IAM security credentials"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are usnig SSO, be sure to install" +
                                  " the AWSSDK.SSO and AWSSDK.SSOOIDC
packages.");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
        catch (ArgumentNullException ex)
        {
            if (ex.Message.Contains("Options property cannot be empty:
ClientName"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are using SSO, have you logged in?");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeStackResourcesin der AWS SDK für .NET API-Referenz.

Aktionen für CloudFormation verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne CloudFormation Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die CloudFormation API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für CloudFormation verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CloudFormation -API-Referenz.

Beispiele

- Verwendung von CancelUpdateStack mit einer CLI
- Verwendung von ContinueUpdateRollback mit einer CLI
- Verwendung von CreateStack mit einer CLI
- Verwendung von DeleteStack mit einer CLI
- Verwendung von DescribeStackEvents mit einer CLI
- Verwendung von DescribeStackResource mit einer CLI
- Verwendung von DescribeStackResources mit einer CLI
- Verwenden Sie mit einem DescribeStacks AWS SDK oder CLI
- Verwendung von EstimateTemplateCost mit einer CLI
- Verwendung von GetTemplate mit einer CLI
- Verwendung von ListStackResources mit einer CLI
- Verwendung von ListStacks mit einer CLI
- Verwendung von UpdateStack mit einer CLI
- Verwendung von ValidateTemplate mit einer CLI

Verwendung von **`CancelUpdateStack`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelUpdateStack` verwendet wird.

CLI

AWS CLI

So brechen Sie eine Stack-Aktualisierung ab, die gerade ausgeführt wird

Der folgende `cancel-update-stack`-Befehl bricht eine Stackaktualisierung auf dem `myteststack`-Stack ab:

```
aws cloudformation cancel-update-stack --stack-name myteststack
```

- Einzelheiten zur API finden Sie CancelUpdateStackin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Bricht eine Aktualisierung auf dem angegebenen Stapel ab.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter CancelUpdateStack AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Bricht eine Aktualisierung auf dem angegebenen Stapel ab.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter CancelUpdateStack AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ContinueUpdateRollback`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ContinueUpdateRollback` verwendet wird.

CLI

AWS CLI

So wiederholen Sie einen Update-Rollback

Das folgende `continue-update-rollback`-Beispiel setzt einen Rollback-Vorgang nach einer zuvor fehlgeschlagenen Stack-Aktualisierung fort.

```
aws cloudformation continue-update-rollback \
    --stack-name my-stack
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie ContinueUpdateRollbackin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Setzt Rollback des benannten Stapels fort, der sich im Zustand 'UPDATE_ROLLBACK_FAILED' befinden sollte. Wenn das fortgesetzte Rollback erfolgreich ist, geht der Stack in den Zustand 'UPDATE_ROLLBACK_COMPLETE' über.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter ContinueUpdateRollback AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Setzt Rollback des benannten Stapels fort, der sich im Zustand 'UPDATE_ROLLBACK_FAILED' befinden sollte. Wenn das fortgesetzte Rollback erfolgreich ist, geht der Stack in den Zustand 'UPDATE_ROLLBACK_COMPLETE' über.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter ContinueUpdateRollback AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateStack`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateStack` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie Ihren ersten Infrastruktur-Stack
- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Um einen AWS CloudFormation Stapel zu erstellen

Der folgende Befehl `create-stacks` erstellt einen Stapel mit dem Namen `myteststack` unter Verwendung der Vorlage `sampletemplate.json` :

```
aws cloudformation create-stack --stack-name myteststack --template-body file://
sampletemplate.json --parameters ParameterKey=KeyPairName,ParameterValue=TestKey
 ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

Ausgabe:

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896"
}
```

Weitere Informationen finden Sie unter Stacks im AWS CloudFormation Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateStackin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser

Parameter. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von " und ParameterKey 'ParameterValue' angegeben werden. Wenn die Erstellung des Stapels fehlschlägt, wird er nicht zurückgesetzt.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Beispiel 2: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser Parameter. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von " und ParameterKey 'ParameterValue' angegeben werden. Wenn die Erstellung des Stapels fehlschlägt, wird er zurückgerollt.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Beispiel 3: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von " und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false). Die angegebenen Benachrichtigungs-AENs empfangen veröffentlichte stapelbezogene Ereignisse.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- Einzelheiten zur API finden Sie unter CreateStack AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser Parameter. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stapels fehlschlägt, wird er nicht zurückgesetzt.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Beispiel 2: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser Parameter. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von ''

und ParameterKey 'ParameterValue' angegeben werden. Wenn die Erstellung des Stapels fehlschlägt, wird er zurückgerollt.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Beispiel 3: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3 S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Inhalt der Vorlage deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er zurückgesetzt (genauso wie bei der Angabe von - DisableRollback $false). Die angegebenen Benachrichtigungs-AENs empfangen veröffentlichte stapelbezogene Ereignisse.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
                -NotificationARN @( "arn1", "arn2" )
```

- Einzelheiten zur API finden Sie unter CreateStack AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteStack`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteStack` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie Ihren ersten Infrastruktur-Stack
- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

So löschen Sie einen Stack

Das folgende Beispiel `delete-stack` löscht den angegebenen Stapel.

```
aws cloudformation delete-stack \
    --stack-name my-stack
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteStackin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht den angegebenen Stapel.

```
Remove-CFNStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DeleteStack AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht den angegebenen Stapel.

```
Remove-CFNStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DeleteStack AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeStackEvents`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStackEvents` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Stack-Ereignisse

Das folgende Beispiel `describe-stack-events` zeigt die 2 jüngsten Ereignisse für den angegebenen Stapel an.

```
aws cloudformation describe-stack-events \
    --stack-name my-stack \
    --max-items 2

{
    "StackEvents": [
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "EventId": "4e1516d0-e4d6-xmpl-b94f-0a51958a168c",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.556Z",
            "ResourceStatus": "UPDATE_COMPLETE"
        },
        {
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
```

```
            "EventId": "4dd3c810-e4d6-xmpl-bade-0aaf8b31ab7a",
            "StackName": "my-stack",
            "LogicalResourceId": "my-stack",
            "PhysicalResourceId": "arn:aws:cloudformation:us-
west-2:123456789012:stack/my-stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "ResourceType": "AWS::CloudFormation::Stack",
            "Timestamp": "2019-10-02T05:34:29.127Z",
            "ResourceStatus": "UPDATE_COMPLETE_CLEANUP_IN_PROGRESS"
        }
    ],

 "NextToken": "eyJOZXh0VG9XMPLiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==="
}
```

- Einzelheiten zur API finden Sie DescribeStackEventsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle stapelbezogenen Ereignisse für den angegebenen Stapel zurück.

```
Get-CFNStackEvent -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DescribeStackEvents AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle stapelbezogenen Ereignisse für den angegebenen Stapel zurück.

```
Get-CFNStackEvent -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DescribeStackEvents AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeStackResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStackResource` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zu einer Stack-Ressource

Das folgende Beispiel `describe-stack-resource` zeigt Details für die Ressource mit dem Namen `MyFunction` im angegebenen Stapel an.

```
aws cloudformation describe-stack-resource \
    --stack-name MyStack \
    --logical-resource-id MyFunction
```

Ausgabe:

```
{
    "StackResourceDetail": {
        "StackName": "MyStack",
        "StackId": "arn:aws:cloudformation:us-east-2:123456789012:stack/MyStack/
d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
        "LogicalResourceId": "MyFunction",
        "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
        "ResourceType": "AWS::Lambda::Function",
        "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
        "ResourceStatus": "UPDATE_COMPLETE",
        "Metadata": "{}",
        "DriftInformation": {
            "StackResourceDriftStatus": "IN_SYNC"
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeStackResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Beschreibung einer Ressource zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "MyDBInstance" identifiziert wurde.

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Beschreibung einer Ressource zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "MyDBInstance" identifiziert wurde.

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeStackResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStackResources` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zu einer Stack-Ressource

Das folgende Beispiel `describe-stack-resources` zeigt Details zu den Ressourcen im angegebenen Stapel an.

```
aws cloudformation describe-stack-resources \
    --stack-name my-stack
```

Ausgabe:

```
{
    "StackResources": [
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
```

```
            "Timestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "Timestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "StackName": "my-stack",
            "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-
stack/d0a825a0-e4cd-xmpl-b9fb-061c69e99204",
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "Timestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeStackResourcesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die AWS Ressourcenbeschreibungen für bis zu 100 Ressourcen zurück, die dem angegebenen Stack zugeordnet sind. Um Details zu allen Ressourcen zu erhalten, die

einem Stack zugeordnet sind Get-CFNStackResourceSummary, verwenden Sie den, der auch das manuelle Durchsuchen der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -StackName "myStack"
```

Beispiel 2: Gibt die Beschreibung der Amazon EC2-Instance zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "Ec2Instance" identifiziert wurde.

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Beispiel 3: Gibt die Beschreibung von bis zu 100 Ressourcen zurück, die mit dem Stack verbunden sind, der eine Amazon EC2-Instance enthält, die durch die Instance-ID "i-123456" identifiziert wird. Um Details zu allen Ressourcen zu erhalten, die einem Stapel zugeordnet sind, verwenden Sie den Get-CFNStackResourceSummary, der auch manuelles Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Beispiel 4: Gibt die Beschreibung der Amazon EC2-Instance zurück, die durch die logische ID "Ec2Instance" in der Vorlage für einen Stack identifiziert wird. Der Stack wird anhand der physischen Ressourcen-ID einer darin enthaltenen Ressource identifiziert, in diesem Fall also einer Amazon EC2-Instance mit der Instance-ID "i-123456". Je nach Inhalt der Vorlage könnte auch eine andere physische Ressource zur Identifizierung des Stacks verwendet werden, zum Beispiel ein Amazon S3-Bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die AWS Ressourcenbeschreibungen für bis zu 100 Ressourcen zurück, die dem angegebenen Stack zugeordnet sind. Um Details zu allen Ressourcen zu erhalten, die einem Stack zugeordnet sind Get-CFNStackResourceSummary, verwenden Sie den, der auch das manuelle Durchsuchen der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -StackName "myStack"
```

Beispiel 2: Gibt die Beschreibung der Amazon EC2-Instance zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "Ec2Instance" identifiziert wurde.

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Beispiel 3: Gibt die Beschreibung von bis zu 100 Ressourcen zurück, die mit dem Stack verbunden sind, der eine Amazon EC2-Instance enthält, die durch die Instance-ID "i-123456" identifiziert wird. Um Details zu allen Ressourcen zu erhalten, die einem Stapel zugeordnet sind, verwenden Sie den Get-CFNStackResourceSummary, der auch manuelles Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Beispiel 4: Gibt die Beschreibung der Amazon EC2-Instance zurück, die durch die logische ID "Ec2Instance" in der Vorlage für einen Stack identifiziert wird. Der Stack wird anhand der physischen Ressourcen-ID einer darin enthaltenen Ressource identifiziert, in diesem Fall also einer Amazon EC2-Instance mit der Instance-ID "i-123456". Je nach Inhalt der Vorlage könnte auch eine andere physische Ressource zur Identifizierung des Stacks verwendet werden, zum Beispiel ein Amazon S3-Bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- Einzelheiten zur API finden Sie unter DescribeStackResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeStacks`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStacks` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie Ihren ersten Infrastruktur-Stack

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Um AWS CloudFormation Stapel zu beschreiben

Der folgende `describe-stacks`-Befehl zeigt eine Zusammenfassung der Informationen für den `myteststack`-Stack:

```
aws cloudformation describe-stacks --stack-name myteststack
```

Ausgabe:

```
{
    "Stacks":  [
        {
            "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
mytestsstack/466df9e0-0dff-08e3-8e2f-5088487c4896",
            "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
            "Tags": [],
            "Outputs": [
                {
                    "Description": "Name of S3 bucket to hold website content",
                    "OutputKey": "BucketName",
                    "OutputValue": "myteststack-s3bucket-jssofi1zie2w"
                }
            ],
            "StackStatusReason": null,
            "CreationTime": "2013-08-23T01:02:15.422Z",
            "Capabilities": [],
            "StackName": "myteststack",
            "StackStatus": "CREATE_COMPLETE",
            "DisableRollback": false
        }
    ]
}
```

Weitere Informationen finden Sie unter Stacks im AWS CloudFormation Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeStacksin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
```

```
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- Einzelheiten zur API finden Sie DescribeStacksin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt eine Sammlung von Stack-Instances zurück, die alle Stacks des Benutzers beschreiben.

```
Get-CFNStack
```

Beispiel 2: Gibt eine Stack-Instance zurück, die den angegebenen Stack beschreibt

```
Get-CFNStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DescribeStacks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt eine Sammlung von Stack-Instances zurück, die alle Stacks des Benutzers beschreiben.

```
Get-CFNStack
```

Beispiel 2: Gibt eine Stack-Instance zurück, die den angegebenen Stack beschreibt

```
Get-CFNStack -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter DescribeStacks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EstimateTemplateCost`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EstimateTemplateCost` verwendet wird.

CLI

AWS CLI

So schätzen Sie die Kosten für eine Vorlage ein

Im folgenden Beispiel für `estimate-template-cost` wird eine Kostenschätzung für eine Vorlage mit dem Namen `template.yaml` im aktuellen Ordner erstellt.

```
aws cloudformation estimate-template-cost \
    --template-body file://template.yaml
```

Ausgabe:

```
{
    "Url": "http://calculator.s3.amazonaws.com/calc5.html?
key=cloudformation/7870825a-xmpl-4def-92e7-c4f8dd360cca"
}
```

- Einzelheiten zur API finden Sie EstimateTemplateCostin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird von der angegebenen Amazon-S3-URL abgerufen und der einzelne Anpassungsparameter wird angewendet. Der Parameter kann auch mit 'Key' und 'Value' anstelle von '' und ParameterKey '' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-
bucket/templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Beispiel 2: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird anhand des bereitgestellten Inhalts analysiert und die angewendeten Anpassungsparameter angewendet (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType'). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Beispiel 3: Wird verwendet New-Object , um den Satz von Vorlagenparametern zu erstellen, und gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die für die Ausführung der Vorlage erforderlichen Ressourcen beschreibt. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType').

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter
 @( $p1, $p2 )
```

- Einzelheiten zur API finden Sie unter EstimateTemplateCost AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird von der angegebenen Amazon-S3-URL abgerufen und der einzelne

Anpassungsparameter wird angewendet. Der Parameter kann auch mit 'Key' und 'Value' anstelle von '' und ParameterKey '' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-
bucket/templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Beispiel 2: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die zum Ausführen der Vorlage erforderlich sind. Die Vorlage wird anhand des bereitgestellten Inhalts analysiert und die angewendeten Anpassungsparameter angewendet (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType'). Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Beispiel 3: Wird verwendet New-Object , um den Satz von Vorlagenparametern zu erstellen, und gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die für die Ausführung der Vorlage erforderlichen Ressourcen beschreibt. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType').

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter
 @( $p1, $p2 )
```

- Einzelheiten zur API finden Sie unter EstimateTemplateCost AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetTemplate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetTemplate` verwendet wird.

CLI

AWS CLI

Um den Vorlagentext für einen AWS CloudFormation Stack anzuzeigen

Der folgende Befehl `get-template` zeigt die Vorlage für den Stapel `mytestsstack` :

```
aws cloudformation get-template --stack-name mytestsstack
```

Ausgabe:

```
{
    "TemplateBody": {
        "AWSTemplateFormatVersion": "2010-09-09",
        "Outputs": {
            "BucketName": {
                "Description": "Name of S3 bucket to hold website content",
                "Value": {
                    "Ref": "S3Bucket"
                }
            }
        },
        "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample
 template showing how to create a publicly accessible S3 bucket. **WARNING** This
 template creates an S3 bucket. You will be billed for the AWS resources used if
 you create a stack from this template.",
        "Resources": {
            "S3Bucket": {
                "Type": "AWS::S3::Bucket",
                "Properties": {
                    "AccessControl": "PublicRead"
                }
            }
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie GetTemplatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Vorlage zurück, die mit dem angegebenen Stapel verbunden ist.

```
Get-CFNTemplate -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter GetTemplate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Vorlage zurück, die mit dem angegebenen Stapel verbunden ist.

```
Get-CFNTemplate -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter GetTemplate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListStackResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListStackResources` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie Ihren ersten Infrastruktur-Stack

CLI

AWS CLI

So listen Sie Ressourcen in einem Stack auf

Der folgende Befehl zeigt die Liste der Ressourcen im angegebenen Stapel an.

```
aws cloudformation list-stack-resources \
    --stack-name my-stack
```

Ausgabe:

```
{
    "StackResourceSummaries": [
        {
            "LogicalResourceId": "bucket",
            "PhysicalResourceId": "my-stack-bucket-1vc62xmplgguf",
            "ResourceType": "AWS::S3::Bucket",
            "LastUpdatedTimestamp": "2019-10-02T04:34:11.345Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "function",
            "PhysicalResourceId": "my-function-SEZV4XMPL4S5",
            "ResourceType": "AWS::Lambda::Function",
            "LastUpdatedTimestamp": "2019-10-02T05:34:27.989Z",
            "ResourceStatus": "UPDATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        },
        {
            "LogicalResourceId": "functionRole",
            "PhysicalResourceId": "my-functionRole-HIZXMPLEOM9E",
            "ResourceType": "AWS::IAM::Role",
            "LastUpdatedTimestamp": "2019-10-02T04:34:06.350Z",
            "ResourceStatus": "CREATE_COMPLETE",
            "DriftInformation": {
                "StackResourceDriftStatus": "IN_SYNC"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListStackResourcesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Beschreibungen aller Ressourcen zurück, die mit dem angegebenen Stapel verbunden sind.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter ListStackResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Beschreibungen aller Ressourcen zurück, die mit dem angegebenen Stapel verbunden sind.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- Einzelheiten zur API finden Sie unter ListStackResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListStacks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListStacks` verwendet wird.

CLI

AWS CLI

Um AWS CloudFormation Stapel aufzulisten

Der folgende `list-stacks`-Befehl zeigt eine Zusammenfassung aller Stack an, die den Status `CREATE_COMPLETE`haben:

```
aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE
```

Ausgabe:

```
[
```

```
    {
        "StackId": "arn:aws:cloudformation:us-east-1:123456789012:stack/
myteststack/466df9e0-0dff-08e3-8e2f-5088487c4896",
        "TemplateDescription": "AWS CloudFormation Sample Template S3_Bucket:
 Sample template showing how to create a publicly accessible S3 bucket.
 **WARNING** This template creates an S3 bucket. You will be billed for the AWS
 resources used if you create a stack from this template.",
        "StackStatusReason": null,
        "CreationTime": "2013-08-26T03:27:10.190Z",
        "StackName": "myteststack",
        "StackStatus": "CREATE_COMPLETE"
    }
]
```

- Einzelheiten zur API finden Sie ListStacksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt zusammenfassende Informationen für alle Stapel zurück.

```
Get-CFNStackSummary
```

Beispiel 2: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt werden.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Beispiel 3: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt oder aktualisiert werden.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- Einzelheiten zur API finden Sie unter ListStacks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt zusammenfassende Informationen für alle Stapel zurück.

```
Get-CFNStackSummary
```

Beispiel 2: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt werden.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Beispiel 3: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt oder aktualisiert werden.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS",
 "UPDATE_IN_PROGRESS")
```

- Einzelheiten zur API finden Sie unter ListStacks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateStack`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateStack` verwendet wird.

CLI

AWS CLI

Um AWS CloudFormation Stacks zu aktualisieren

Der folgende Befehl `update-stack` aktualisiert die Vorlage und die Eingabeparameter für den Stapel `mystack` :

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template --
parameters ParameterKey=KeyPairName,ParameterValue=SampleKeyPair
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,SampleSubnetID2
```

Der folgende Befehl `update-stack` aktualisiert nur den Wert des Parameters `SubnetIDs` für den Stapel `mystack` . Wenn Sie keinen Parameterwert angeben, wird der Standardwert verwendet, der in der Vorlage angegeben ist:

```
aws cloudformation update-stack --stack-name mystack --
template-url https://s3.amazonaws.com/sample/updated.template
```

```
 --parameters ParameterKey=KeyPairName,UsePreviousValue=true
 ParameterKey=SubnetIDs,ParameterValue=SampleSubnetID1\\,UpdatedSampleSubnetID2
```

Der folgende `update-stack`-Befehl fügt dem `mystack`-Stack zwei Themen für Stackbenachrichtigungen hinzu:

```
aws cloudformation update-stack --stack-name mystack --use-previous-template --
notification-arns "arn:aws:sns:use-east-1:123456789012:mytopic1" "arn:aws:sns:us-
east-1:123456789012:mytopic2"
```

Weitere Informationen finden Sie unter AWS CloudFormation Stack-Updates im AWS CloudFormation Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateStackin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV1' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 2: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 3: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV2' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Aktualisiert den Stack 'myStack' mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 5: Aktualisiert den Stack 'myStack', von dem in diesem Beispiel angenommen wird, dass er IAM-Ressourcen enthält, mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden. Bei Stacks, die IAM-Ressourcen enthalten, müssen Sie den Capability-Parameter „CAPABILITY_IAM“ angeben. Andernfalls schlägt das Update mit einem Fehler '' fehl. InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Einzelheiten zur API finden Sie unter UpdateStackCmdlet-Referenz (V4).AWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV1' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 2: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 3: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV2' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Aktualisiert den Stack 'myStack' mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
```

```
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 5: Aktualisiert den Stack 'myStack', von dem in diesem Beispiel angenommen wird, dass er IAM-Ressourcen enthält, mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden. Bei Stacks, die IAM-Ressourcen enthalten, müssen Sie den Capability-Parameter „CAPABILITY_IAM“ angeben. Andernfalls schlägt das Update mit einem Fehler '' fehl. InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- Einzelheiten zur API finden Sie unter UpdateStackCmdlet-Referenz (V5).AWS -Tools für PowerShell

Verwendung von **`ValidateTemplate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ValidateTemplate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie Ihren ersten Infrastruktur-Stack

CLI

AWS CLI

Um eine AWS CloudFormation Vorlage zu validieren

Der folgende Befehl `validate-template` validiert die Vorlage `sampletemplate.json` :

```
aws cloudformation validate-template --template-body file://sampletemplate.json
```

Ausgabe:

```
{
    "Description": "AWS CloudFormation Sample Template S3_Bucket: Sample template
 showing how to create a publicly accessible S3 bucket. **WARNING** This template
 creates an S3 bucket. You will be billed for the AWS resources used if you
 create a stack from this template.",
    "Parameters": [],
    "Capabilities": []
}
```

Weitere Informationen finden Sie im AWS CloudFormation Benutzerhandbuch unter Arbeiten mit AWS CloudFormation Vorlagen.

- Einzelheiten zur API finden Sie ValidateTemplatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Validiert den angegebenen Vorlageninhalt. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Beispiel 2: Validiert die angegebene Vorlage, auf die über eine Amazon S3 URL zugegriffen wird. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Einzelheiten zur API finden Sie unter ValidateTemplate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Validiert den angegebenen Vorlageninhalt. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Beispiel 2: Validiert die angegebene Vorlage, auf die über eine Amazon S3 URL zugegriffen wird. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- Einzelheiten zur API finden Sie unter ValidateTemplate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für CloudFormation verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien CloudFormation mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb CloudFormation oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Erstellen Sie Ihren ersten Infrastruktur-Stack
- Erste Schritte mit verwalteten Kubernetes-Clustern

Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die anhand fiktiver Daten ein System zur Nachverfolgung täglicher Fälle COVID-19 in den Vereinigte Staaten simuliert.

Python

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon AWS Lambda DynamoDB verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle von Infektionen COVID-19 in Vereinigte Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Erstellen Sie Ihren ersten Infrastruktur-Stack

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine CloudFormation Vorlage
- Testen Sie den Webserver
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudFormation Getting Started Script
# This script creates a CloudFormation stack with a web server and security
 group,
# monitors the stack creation, and provides cleanup options.

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="cloudformation-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS CloudFormation Getting Started Tutorial"
echo "====================================================="
echo "This script will create a CloudFormation stack with:"
echo "- An EC2 instance running a simple web server"
echo "- A security group allowing HTTP access from your IP"
echo ""
echo "Starting at: $(date)"
echo ""

# Function to validate IP address format
validate_ip() {
    local ip=$1
    if ! [[ "$ip" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        return 1
    fi
```

```
    local IFS=.
    local -a octets=($ip)
    for octet in "${octets[@]}"; do
        if ((octet > 255)); then
            return 1
        fi
    done
    return 0
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME" --region
 "${AWS_REGION:-us-east-1}" || true

        echo "Waiting for stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
 --region "${AWS_REGION:-us-east-1}" 2>/dev/null || true

        echo "Stack deletion complete."
    fi

    if [ -f "${TEMPLATE_FILE:-}" ]; then
        echo "Removing local template file: $TEMPLATE_FILE"
        shred -vfz -n 3 "$TEMPLATE_FILE" 2>/dev/null || rm -f "$TEMPLATE_FILE"
    fi

    echo "Cleanup completed at: $(date)"
}

# Function to handle errors
handle_error() {
    echo ""
    echo "==================================================="
    echo "ERROR: $1"
    echo "==================================================="
```

```
    echo "Resources created before error:"
    if [ -n "${STACK_NAME:-}" ]; then
        echo "- CloudFormation stack: $STACK_NAME"
    fi
    echo ""

    echo "Cleaning up resources automatically..."
    cleanup

    exit 1
}

# Set up trap for script interruption
trap 'handle_error "Script interrupted"' INT TERM
trap 'cleanup' EXIT

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS_REGION format: $AWS_REGION"
fi

# Generate a unique stack name with timestamp
TIMESTAMP=$(date +%s)
STACK_NAME="MyTestStack-${TIMESTAMP}"
TEMPLATE_FILE="webserver-template-${TIMESTAMP}.yaml"

# Verify AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Verify curl is installed
if ! command -v curl &> /dev/null; then
    handle_error "curl is not installed or not in PATH"
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Step 1: Create the CloudFormation template file
echo "Creating CloudFormation template file: $TEMPLATE_FILE"
```

```
cat > "$TEMPLATE_FILE" << 'EOF'
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Metadata:
  AWS::CloudFormation::Init:
    config:
      packages:
        yum:
          httpd: []
      services:
        sysvinit:
          httpd:
            enabled: true
            ensureRunning: true

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerRole:
    Type: AWS::IAM::Role
    Properties:
```

```
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'

  WebServerInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref WebServerRole

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via specified IP address
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
          Description: Allow HTTP outbound
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
          Description: Allow HTTPS outbound for package updates
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP
          Description: HTTP access from specified IP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref WebServerInstanceProfile
```

```
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      Monitoring: true
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
        HttpPutResponseHopLimit: 1
      UserData: !Base64 |
        #!/bin/bash
        set -euo pipefail
        exec > >(tee /var/log/user-data.log)
        exec 2>&1
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
        chmod 644 /var/www/html/index.html

Outputs:
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
  InstanceId:
    Value: !Ref WebServer
    Description: EC2 Instance ID
  SecurityGroupId:
    Value: !Ref WebServerSecurityGroup
    Description: Security Group ID
EOF

chmod 600 "$TEMPLATE_FILE"

if [ ! -f "$TEMPLATE_FILE" ]; then
    handle_error "Failed to create template file"
fi

# Step 2: Validate the template
echo ""
echo "Validating CloudFormation template..."
```

```
if ! VALIDATION_RESULT=$(aws cloudformation validate-template \
    --template-body "file://$TEMPLATE_FILE" \
    --region "$AWS_REGION" 2>&1); then
    handle_error "Template validation failed: $VALIDATION_RESULT"
fi
echo "Template validation successful."

# Step 3: Get the user's public IP address
echo ""
echo "Retrieving your public IP address..."

MY_IP=""
for endpoint in "https://checkip.amazonaws.com" "https://api.ipify.org" "https://
icanhazip.com"; do
    if MY_IP=$(curl -s --max-time 5 "$endpoint" 2>/dev/null); then
        MY_IP="${MY_IP//[[:space:]]/}"
        if validate_ip "$MY_IP"; then
            break
        fi
        MY_IP=""
    fi
done

if [ -z "$MY_IP" ]; then
    handle_error "Failed to retrieve public IP address from multiple sources"
fi

MY_IP_CIDR="${MY_IP}/32"
echo "Your public IP address: $MY_IP_CIDR"

# Step 4: Create the CloudFormation stack
echo ""
echo "Creating CloudFormation stack: $STACK_NAME"
echo "This will create an EC2 instance and security group."
if ! CREATE_RESULT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-body "file://$TEMPLATE_FILE" \
  --parameters \
    ParameterKey=InstanceType,ParameterValue=t2.micro \
    ParameterKey=MyIP,ParameterValue="$MY_IP_CIDR" \
  --capabilities CAPABILITY_IAM \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudformation-gs \
  --region "$AWS_REGION" \
  --output text 2>&1); then
```

```
    handle_error "Stack creation failed: $CREATE_RESULT"
fi

STACK_ID="${CREATE_RESULT//[[:space:]]/}"
echo "Stack creation initiated. Stack ID: $STACK_ID"

# Step 5: Monitor stack creation
echo ""
echo "Monitoring stack creation..."
echo "This may take a few minutes."

if ! aws cloudformation wait stack-create-complete \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" 2>/dev/null; then
    STACK_STATUS=$(aws cloudformation describe-stacks \
        --stack-name "$STACK_NAME" \
        --query "Stacks[0].StackStatus" \
        --region "$AWS_REGION" \
        --output text 2>/dev/null || echo "UNKNOWN")
    if [[ "$STACK_STATUS" =~ ROLLBACK|FAILED ]]; then
        handle_error "Stack creation failed. Status: $STACK_STATUS"
    fi
fi

echo "Stack creation completed successfully."

# Step 6: List stack resources
echo ""
echo "Resources created by the stack:"
aws cloudformation list-stack-resources \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "StackResourceSummaries[*].{LogicalID:LogicalResourceId,
 Type:ResourceType, Status:ResourceStatus}" \
    --output table

# Step 7: Get stack outputs
echo ""
echo "Stack outputs:"
if ! OUTPUTS=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs" \
    --output json 2>&1); then
```

```
    handle_error "Failed to retrieve stack outputs"
fi

echo "$OUTPUTS"

# Extract the WebsiteURL
WEBSITE_URL=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs[?OutputKey=='WebsiteURL'].OutputValue" \
    --output text 2>/dev/null || echo "")

if [ -z "$WEBSITE_URL" ]; then
    handle_error "Failed to extract WebsiteURL from stack outputs"
fi

echo ""
echo "WebsiteURL: $WEBSITE_URL"
echo ""
echo "You can access the web server by opening the above URL in your browser."
echo "You should see a simple 'Hello World!' message."

# Step 8: Test the connection via CLI with timeout
echo ""
echo "Testing connection to the web server..."
MAX_RETRIES=5
RETRY_COUNT=0

while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
    if HTTP_RESPONSE=$(curl -s -o /dev/null -w "%{http_code}" --max-time 10
 "$WEBSITE_URL" 2>/dev/null); then
        if [ "$HTTP_RESPONSE" == "200" ]; then
            echo "Connection successful! HTTP status code: $HTTP_RESPONSE"
            break
        elif [ "$HTTP_RESPONSE" == "000" ]; then
            RETRY_COUNT=$((RETRY_COUNT + 1))
            if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
                echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES -
 retrying in 10 seconds..."
                sleep 10
            else
                echo "Warning: Connection test failed after $MAX_RETRIES
 attempts"
            fi
```

```
        else
            echo "Warning: Connection test returned HTTP status code:
 $HTTP_RESPONSE"
            break
        fi
    else
        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying in
 10 seconds..."
            sleep 10
        else
            echo "Warning: Connection test failed after $MAX_RETRIES attempts"
        fi
    fi
done

# Step 9: Auto-confirm cleanup
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Resources created:"
echo "- CloudFormation stack: $STACK_NAME"
echo "  - EC2 instance"
echo "  - Security group"
echo "  - IAM role and instance profile"
echo ""
echo "Proceeding with cleanup of all created resources..."

cleanup

echo ""
echo "==================================================="
echo "Tutorial completed at: $(date)"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateStack
  - DeleteStack

- DescribeStacks
- ListStackResources
- ValidateTemplate
- Wait

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
```

```
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
```

```
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
```

```
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."
```

```
# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
```

```
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
```

```
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
```

```
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi
```

```
# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks

- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Codebeispiele für die CloudFront Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon CloudFront mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- CloudFront Entwicklerhandbuch — Weitere Informationen zu CloudFront.
- CloudFront API-Referenz — Details zu allen verfügbaren CloudFront Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CloudFront AWS SDKs
  - Aktionen zur Verwendung CloudFront AWS SDKs
    - Verwenden Sie CreateDistributionmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem CreateFunction AWS SDK
    - Verwendung von CreateInvalidation mit einer CLI
    - Verwenden Sie mit einem CreateKeyGroup AWS SDK

    - Verwenden Sie CreatePublicKeymit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDistributionmit einem AWS SDK oder CLI
    - Verwendung von GetCloudFrontOriginAccessIdentity mit einer CLI
    - Verwendung von GetCloudFrontOriginAccessIdentityConfig mit einer CLI
    - Verwendung von GetDistribution mit einer CLI
    - Verwenden Sie mit einem GetDistributionConfig AWS SDK oder CLI
    - Verwendung von ListCloudFrontOriginAccessIdentities mit einer CLI
    - Verwenden Sie mit einem ListDistributions AWS SDK oder CLI
    - Verwenden Sie UpdateDistributionmit einem AWS SDK oder CLI
- Szenarien für die Verwendung CloudFront AWS SDKs
  - SaaS-Manager-Ressourcen erstellen AWS SDK
  - Löschen Sie Signaturressourcen CloudFront mit AWS SDK
  - Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten
  - Erste Schritte mit Firewalls für Webanwendungen
  - Erstellen Sie signierte URLs und Cookies mit einem AWS SDK
- CloudFront Funktionen, Beispiele für CloudFront
  - Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis HTTP-Sicherheitsheader hinzu
  - Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis einen CORS-Header hinzu
  - Fügen Sie einem Antwortereignis des CloudFront Functions-Viewers einen Cache-Control-Header hinzu
  - Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen echten Client-IP-Header hinzu
  - Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen Origin-Header hinzu
  - Fügen Sie index.html hinzu, um URLs ohne Dateinamen in einem CloudFront Functions-Viewer-Anforderungsereignis anzufordern
  - Normalisieren Sie die Parameter der Abfragezeichenfolge in einer CloudFront Functions Viewer-Anfrage
  - Leiten Sie in einem CloudFront Functions-Viewer-Anforderungsereignis zu einer neuen URL weiter
  - Schreiben Sie eine Anforderungs-URI auf der Grundlage der KeyValueStore Konfiguration für ein CloudFront Functions-Viewer-Anforderungsereignis neu

- Leiten Sie Anfragen in einem CloudFront Functions-Viewer-Anforderungsereignis an einen Ursprung weiter, der näher am Betrachter liegt
- Verwenden Sie Schlüssel-Wert-Paare in einer CloudFront Functions-Viewer-Anfrage
- Validieren Sie ein einfaches Token in einer CloudFront Functions-Viewer-Anfrage

Grundlegende Beispiele für die Verwendung CloudFront AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon CloudFront mit AWS SDKs verwenden können.

Beispiele

- Aktionen zur Verwendung CloudFront AWS SDKs
  - Verwenden Sie CreateDistributionmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem CreateFunction AWS SDK
  - Verwendung von CreateInvalidation mit einer CLI
  - Verwenden Sie mit einem CreateKeyGroup AWS SDK
  - Verwenden Sie CreatePublicKeymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDistributionmit einem AWS SDK oder CLI
  - Verwendung von GetCloudFrontOriginAccessIdentity mit einer CLI
  - Verwendung von GetCloudFrontOriginAccessIdentityConfig mit einer CLI
  - Verwendung von GetDistribution mit einer CLI
  - Verwenden Sie mit einem GetDistributionConfig AWS SDK oder CLI
  - Verwendung von ListCloudFrontOriginAccessIdentities mit einer CLI
  - Verwenden Sie mit einem ListDistributions AWS SDK oder CLI
  - Verwenden Sie UpdateDistributionmit einem AWS SDK oder CLI

Aktionen zur Verwendung CloudFront AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne CloudFront Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die CloudFront API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung CloudFront AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon CloudFront API-Referenz.

Beispiele

- Verwenden Sie CreateDistributionmit einem AWS SDK oder CLI
- Verwenden Sie mit einem CreateFunction AWS SDK
- Verwendung von CreateInvalidation mit einer CLI
- Verwenden Sie mit einem CreateKeyGroup AWS SDK
- Verwenden Sie CreatePublicKeymit einem AWS SDK oder CLI
- Verwenden Sie DeleteDistributionmit einem AWS SDK oder CLI
- Verwendung von GetCloudFrontOriginAccessIdentity mit einer CLI
- Verwendung von GetCloudFrontOriginAccessIdentityConfig mit einer CLI
- Verwendung von GetDistribution mit einer CLI
- Verwenden Sie mit einem GetDistributionConfig AWS SDK oder CLI
- Verwendung von ListCloudFrontOriginAccessIdentities mit einer CLI
- Verwenden Sie mit einem ListDistributions AWS SDK oder CLI
- Verwenden Sie UpdateDistributionmit einem AWS SDK oder CLI

Verwenden Sie **`CreateDistribution`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDistribution` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen einer Multi-Tenant-Distribution und eines Distributionsmandanten
- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

CLI

AWS CLI

Beispiel 1: Um eine CloudFront Distribution zu erstellen

Im folgenden `create-distribution`-Beispiel wird eine Distribution für einen S3-Bucket mit dem Namen `amzn-s3-demo-bucket` erstellt und `index.html` als Standardstammobjekt mithilfe von Befehlszeilenargumenten festgelegt.

```
aws cloudfront create-distribution \
    --origin-domain-name amzn-s3-demo-bucket.s3.amazonaws.com \
    --default-root-object index.html
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
```

```
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
```

```
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
```

```
}
```

Beispiel 2: Um eine CloudFront Distribution mit einer JSON-Datei zu erstellen

Im folgenden `create-distribution`-Beispiel wird eine Distribution für einen S3-Bucket mit dem Namen `amzn-s3-demo-bucket` erstellt und `index.html` als Standardstammobjekt mithilfe einer JSON-Datei festgelegt.

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

Inhalt von `dist-config.json`:

```
{
    "CallerReference": "cli-example",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
```

```
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
```

```
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Beispiel 3: Um eine CloudFront Multi-Tenant-Distribution mit einem Zertifikat zu erstellen

Im folgenden `create-distribution` Beispiel wird eine CloudFront Distribution mit Mehrmandantenunterstützung erstellt und ein TLS-Zertifikat angegeben.

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

Inhalt von `dist-config.json`:

```
{
    "CallerReference": "cli-example-with-cert",
    "Comment": "CLI example distribution",
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
```

```
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "DomainName": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "OriginPath": "/{{tenantName}}",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
        "ViewerProtocolPolicy": "allow-all",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["HEAD", "GET"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["HEAD", "GET"]
            }
        }
    },
    "Enabled": true,
    "ViewerCertificate": {
        "ACMCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/191306a1-db01-49ca-90ef-fc414ee5dabc",
        "SSLSupportMethod": "sni-only"
    },
    "HttpVersion": "http2",
    "ConnectionMode": "tenant-only",
    "TenantConfig": {
        "ParameterDefinitions": [
            {
                "Name": "tenantName",
                "Definition": {
                    "StringSchema": {
                        "Comment": "tenantName parameter",
                        "DefaultValue": "root",
                        "Required": false
```

```
                    }
                }
            }
        ]
    }
}
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2020-05-31/distribution/
E1HVIAU7UABC",
    "ETag": "E20LT7R1BABC",
    "Distribution": {
        "Id": "E1HVIAU7U12ABC",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/E1HVIAU7U12ABC",
        "Status": "InProgress",
        "LastModifiedTime": "2025-07-10T20:33:31.117000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "example.com",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example-with-cert",
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                        "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                        "OriginPath": "/{{tenantName}}",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
```

```
                            "OriginAccessIdentity": ""
                        },
                        "ConnectionAttempts": 3,
                        "ConnectionTimeout": 10,
                        "OriginShield": {
                            "Enabled": false
                        },
                        "OriginAccessControlId": ""
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                "TrustedKeyGroups": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": ["HEAD", "GET"],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": ["HEAD", "GET"]
                    }
                },
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": "",
                "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
                "GrpcConfig": {
                    "Enabled": false
                }
            },
            "CacheBehaviors": {
```

```
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "CLI example distribution",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": false,
                "ACMCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/1954f095-11b6-4daf-9952-0c308a00abc",
                "SSLSupportMethod": "sni-only",
                "MinimumProtocolVersion": "TLSv1.2_2021",
                "Certificate": "arn:aws:acm:us-
east-1:123456789012:certificate/1954f095-11b6-4daf-9952-0c308a00abc",
                "CertificateSource": "acm"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "TenantConfig": {
                "ParameterDefinitions": [
                    {
                        "Name": "tenantName",
                        "Definition": {
                            "StringSchema": {
                                "Comment": "tenantName parameter",
                                "DefaultValue": "root",
                                "Required": false
                            }
                        }
                    }
                ]
```

```
            },
            "ConnectionMode": "tenant-only"
        }
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Distributionen im Amazon CloudFront Developer Guide.

Beispiel 4: So erstellen Sie eine CloudFront Multi-Tenant-Distribution ohne Zertifikat

Im folgenden `create-distribution` Beispiel wird eine CloudFront Distribution mit Mehrmandantenunterstützung, aber ohne TLS-Zertifikat erstellt.

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

Inhalt von `dist-config.json`:

```
{
    "CallerReference": "cli-example",
    "Comment": "CLI example distribution",
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "DomainName": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                "OriginPath": "/{{tenantName}}",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
        "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
```

```
        "ViewerProtocolPolicy": "allow-all",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        }
    },
    "Enabled": true,
    "HttpVersion": "http2",
    "ConnectionMode": "tenant-only",
    "TenantConfig": {
        "ParameterDefinitions": [
            {
                "Name": "tenantName",
                "Definition": {
                    "StringSchema": {
                        "Comment": "tenantName parameter",
                        "DefaultValue": "root",
                        "Required": false
                    }
                }
            }
        ]
    }
}
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2020-05-31/distribution/
E2GJ5J9QN12ABC",
    "ETag": "E37YLVVQIABC",
    "Distribution": {
        "Id": "E2GJ5J9QNABC",
```

```
        "ARN": "arn:aws:cloudfront::123456789012:distribution/E2GJ5J9QN12ABC",
        "Status": "InProgress",
        "LastModifiedTime": "2025-07-10T20:35:20.565000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "example.com",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example-no-cert",
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.us-east-1.amazonaws.com",
                        "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                        "OriginPath": "/{{tenantName}}",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        },
                        "ConnectionAttempts": 3,
                        "ConnectionTimeout": 10,
                        "OriginShield": {
                            "Enabled": false
                        },
                        "OriginAccessControlId": ""
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
```

```
                "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                "TrustedKeyGroups": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": "",
                "CachePolicyId": "658327ea-f89d-4fab-a63d-7e88639e5ABC",
                "GrpcConfig": {
                    "Enabled": false
                }
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "CLI example distribution",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
```

```
                "Bucket": "",
                "Prefix": ""
            },
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "SSLSupportMethod": "sni-only",
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "TenantConfig": {
                "ParameterDefinitions": [
                    {
                        "Name": "tenantName",
                        "Definition": {
                            "StringSchema": {
                                "Comment": "tenantName parameter",
                                "DefaultValue": "root",
                                "Required": false
                            }
                        }
                    }
                ]
            },
            "ConnectionMode": "tenant-only"
        }
    }
}
```

Weitere Informationen finden Sie unter Distributionen konfigurieren im Amazon CloudFront Developer Guide.

- Einzelheiten zur API finden Sie CreateDistributionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im folgenden Beispiel wird ein Bucket von Amazon Simple Storage Service(Amazon S3) als Inhaltsquelle verwendet.

Nach dem Erstellen der Distribution erstellt der Code eine Option, CloudFrontWaiterum zu warten, bis die Distribution bereitgestellt ist, bevor die Distribution zurückgegeben wird.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {
```

```
                final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

.domainName(originDomain)

.id(originId)

.s3OriginConfig(builder4 -> builder4

              .originAccessIdentity(

                              ""))

.originAccessControlId(

              originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)
                                                                .minTTL(200L)

.forwardedValues(b5 -> b5

.cookies(cp -> cp

              .forward(ItemSelection.NONE))
```

```
.queryString(true))

.trustedKeyGroups(b3 -> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

.allowedMethods(b4 -> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

              .quantity(2)

              .items(Method.HEAD,

                             Method.GET))))
                                                .cacheBehaviors(b -> b
                                                                .quantity(1)
                                                                .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

              ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

              .quantity(1)

              .items(keyGroupId)

              .enabled(true))
```

```
.minTTL(200L)

.forwardedValues(b4 -> b4

              .cookies(cp -> cp

                              .forward(ItemSelection.NONE))

              .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

              .items(Method.HEAD,

                              Method.GET)

              .cachedMethods(b6 -> b6

                              .quantity(2)

                              .items(Method.HEAD,

                                             Method.GET)))))
                                              .enabled(true)
                                              .comment("Distribution built with
java")

.callerReference(Instant.now().toString())));

               final Distribution distribution =
createDistResponse.distribution();
               logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
distribution.domainName(),
                               distribution.id());
               logger.info("Waiting for distribution to be deployed ...");
               try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                      ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                                      .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                                      .matched();
                      responseOrException.response()
```

```
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- Einzelheiten zur API finden Sie CreateDistributionunter AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt eine CloudFront Basisdistribution, die mit Protokollierung und Caching konfiguriert ist.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
```

```
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- Einzelheiten zur API finden Sie unter CreateDistribution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt eine CloudFront Basisdistribution, die mit Protokollierung und Caching konfiguriert ist.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
      -Logging_Prefix "help/" `
      -DistributionConfig_CallerReference Client1 `
      -DistributionConfig_DefaultRootObject index.html `
      -DefaultCacheBehavior_TargetOriginId $origin.Id `
      -ForwardedValues_QueryString $true `
      -Cookies_Forward all `
      -WhitelistedNames_Quantity 0 `
      -TrustedSigners_Enabled $false `
      -TrustedSigners_Quantity 0 `
      -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
      -DefaultCacheBehavior_MinTTL 1000 `
      -DistributionConfig_PriceClass "PriceClass_All" `
      -CacheBehaviors_Quantity 0 `
      -Aliases_Quantity 0
```

- Einzelheiten zur API finden Sie unter CreateDistribution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateFunction`** AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateFunction`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <functionName> <filepath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName,
filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();
```

```
            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK for Java 2.x API-Referenz.

Verwendung von **`CreateInvalidation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateInvalidation` verwendet wird.

CLI

AWS CLI

Um eine Ungültigerklärung für eine CloudFront Distribution zu erstellen

Das folgende `create-invalidation` Beispiel erstellt eine Invalidierung für die angegebenen Dateien in der angegebenen CloudFront Distribution:

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",
    "Invalidation": {
        "Id": "I1JLWSDAP8FU89",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:24:51.407Z",
        "InvalidationBatch": {
```

```
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file2.png",
                    "/example-path/example-file.jpg"
                ]
            },
            "CallerReference": "cli-1575570291-670203"
        }
    }
}
```

Im vorherigen Beispiel generierte die AWS CLI automatisch ein zufälliges Ergebnis`CallerReference`. Wenn Sie Ihre eigene `CallerReference` angeben oder die Invalidierungsparameter nicht als Befehlszeilenargumente übergeben möchten, können Sie eine JSON-Datei verwenden. Im folgende Beispiel wird eine Invalidierung für zwei Dateien erstellt, indem die Invalidierungsparameter in einer JSON-Datei mit dem Namen `inv-batch.json` angegeben werden.

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --invalidation-batch file://inv-batch.json
```

Inhalt von `inv-batch.json`:

```
{
    "Paths": {
        "Quantity": 2,
        "Items": [
            "/example-path/example-file.jpg",
            "/example-path/example-file2.png"
        ]
    },
    "CallerReference": "cli-example"
}
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUYOIK",
```

```
    "Invalidation": {
        "Id": "I2J0I21PCUYOIK",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:40:49.413Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file.jpg",
                    "/example-path/example-file2.png"
                ]
            },
            "CallerReference": "cli-example"
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateInvalidationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Invalidierung für eine Distribution mit der ID EXAMPLENSTXAXE erstellt. Das CallerReference ist eine vom Benutzer gewählte eindeutige ID. In diesem Fall wird ein Zeitstempel verwendet, der den 15. Mai 2019 um 9:00 Uhr darstellt. Die Variable $Paths speichert drei Pfade zu Bild- und Mediendateien, die der Benutzer nicht als Teil des Caches der Distribution haben möchte. Der Wert des Paths_Quantity Parameters - ist die Gesamtzahl der im Paths_Item Parameter - angegebenen Pfade.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

Ausgabe:

```
Invalidation                        Location

------------                        --------
```

```
Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Einzelheiten zur API finden Sie unter CreateInvalidation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Invalidierung für eine Distribution mit der ID EXAMPLENSTXAXE erstellt. Das CallerReference ist eine vom Benutzer gewählte eindeutige ID. In diesem Fall wird ein Zeitstempel verwendet, der den 15. Mai 2019 um 9:00 Uhr darstellt. Die Variable $Paths speichert drei Pfade zu Bild- und Mediendateien, die der Benutzer nicht als Teil des Caches der Distribution haben möchte. Der Wert des Paths_Quantity Parameters - ist die Gesamtzahl der im Paths_Item Parameter - angegebenen Pfade.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

Ausgabe:

```
Invalidation                        Location

------------                        --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- Einzelheiten zur API finden Sie unter CreateInvalidation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateKeyGroup`** AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateKeyGroup`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Schlüsselgruppe erfordert mindestens einen öffentlichen Schlüssel, der zur Überprüfung signierter URLs oder Cookies verwendet wird.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
 b.keyGroupConfig(c -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- Einzelheiten zur API finden Sie CreateKeyGroupin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreatePublicKey`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePublicKey` verwendet wird.

CLI

AWS CLI

Um einen CloudFront öffentlichen Schlüssel zu erstellen

Im folgenden Beispiel wird ein CloudFront öffentlicher Schlüssel erstellt, indem die Parameter in einer JSON-Datei mit dem Namen bereitgestellt `pub-key-config.json` werden. Bevor Sie diesen Befehl verwenden können, benötigen Sie einen PEM-encoded öffentlichen Schlüssel. Weitere Informationen finden Sie unter Create an RSA Key Pair im Amazon CloudFront Developer Guide.

```
aws cloudfront create-public-key \
    --public-key-config file://pub-key-config.json
```

Die Datei `pub-key-config.json` ist ein JSON-Dokument im aktuellen Ordner, das Folgendes enthält. Beachten Sie, dass der öffentliche Schlüssel im PEM-Format verschlüsselt ist.

```
{
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
    "Comment": "example public key"
}
```

Ausgabe:

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
        "Id": "KDFB19YGCR002",
```

```
        "CreatedTime": "2019-12-05T18:51:43.781Z",
        "PublicKeyConfig": {
            "CallerReference": "cli-example",
            "Name": "ExampleKey",
            "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
            "Comment": "example public key"
        }
    }
}
```

- Einzelheiten zur API finden Sie unter CreatePublicKey AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Codebeispiel liest einen öffentlichen Schlüssel ein und lädt ihn auf Amazon CloudFront hoch.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;
```

```
public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
 String publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreatePublicKeyin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteDistribution`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDistribution` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

CLI

AWS CLI

Um eine CloudFront Distribution zu löschen

Im folgenden Beispiel wird die CloudFront Distribution mit der ID `EDFDVBD6EXAMPLE` gelöscht. Bevor Sie eine Distribution löschen können, müssen Sie sie deaktivieren. Verwenden Sie zum Deaktivieren einer Distribution den Befehl „update-distribution“. Weitere Informationen finden Sie in den Beispielen zu „update-distribution“.

Wenn eine Distribution deaktiviert ist, können Sie sie löschen. Um eine Distribution zu löschen, müssen Sie die `--if-match`-Option verwenden, um den `ETag` der Distribution anzugeben. Um den `ETag` zu erhalten, verwenden Sie bitte den Befehl „get-Distribution“ oder „get-Distribution-Config“.

```
aws cloudfront delete-distribution \
    --id EDFDVBD6EXAMPLE \
    --if-match E2QWRUHEXAMPLE
```

Bei erfolgreicher Ausführung gibt dieser Befehl keine Ausgabe aus.

- Einzelheiten zur API finden Sie unter DeleteDistribution AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Codebeispiel aktualisiert eine Distribution auf deaktiviert, verwendet einen Waiter, der auf die Bereitstellung der Änderung wartet, und löscht anschließend die Distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
```

```
public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
 response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

 .cacheBehaviors(distConfig.cacheBehaviors())

 .defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

 .callerReference(distConfig.callerReference())

 .defaultCacheBehavior(distConfig.defaultCacheBehavior())

 .priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

 .defaultRootObject(distConfig.defaultRootObject())

 .customErrorResponses(distConfig.customErrorResponses())

 .httpVersion(distConfig.httpVersion())

 .isIPV6Enabled(distConfig.isIPV6Enabled())

 .restrictions(distConfig.restrictions())
```

```
 .viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

 .originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for
 deployment before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                }

                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
 {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- Einzelheiten zur API finden Sie DeleteDistributionin der AWS SDK for Java 2.x API-Referenz.

Verwendung von **`GetCloudFrontOriginAccessIdentity`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetCloudFrontOriginAccessIdentity` verwendet wird.

CLI

AWS CLI

Um eine CloudFront ursprüngliche Zugriffsidentität zu erhalten

Im folgenden Beispiel wird die ursprüngliche CloudFront Zugriffsidentität (OAI) mit der ID abgerufen`E74FTE3AEXAMPLE`, einschließlich ihrer `ETag` und der zugehörigen kanonischen S3-ID. Die OAI-ID wird in der Ausgabe der Befehle create-cloud-front-origin-access-identity und list-cloud-front-origin-access-identities zurückgegeben.

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

Ausgabe:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentity": {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "CloudFrontOriginAccessIdentityConfig": {
            "CallerReference": "cli-example",
            "Comment": "Example OAI"
        }
    }
}
```

- Einzelheiten zur API finden Sie GetCloudFrontOriginAccessIdentityin AWS CLI der Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine bestimmte CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Obwohl der -Id-Parameter nicht erforderlich ist, werden keine Ergebnisse zurückgegeben, wenn Sie ihn nicht angeben.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
     CloudFrontOriginAccessIdentityConfig    Id
    S3CanonicalUserId
     -----------------------------------    --
    -----------------
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentity AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine bestimmte CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Obwohl der -Id-Parameter nicht erforderlich ist, werden keine Ergebnisse zurückgegeben, wenn Sie ihn nicht angeben.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
     CloudFrontOriginAccessIdentityConfig    Id
    S3CanonicalUserId
     -----------------------------------    --
    -----------------
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentity AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetCloudFrontOriginAccessIdentityConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetCloudFrontOriginAccessIdentityConfig` verwendet wird.

CLI

AWS CLI

Um eine Konfiguration der CloudFront Origin-Zugriffsidentität zu erhalten

Im folgenden Beispiel werden Metadaten zur ursprünglichen CloudFront Zugriffsidentität (OAI) mit der ID abgerufen`E74FTE3AEXAMPLE`, einschließlich ihrer`ETag`. Die OAI-ID wird in der Ausgabe der Befehle create-cloud-front-origin-access-identity und list-cloud-front-origin-access-identities zurückgegeben.

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

Ausgabe:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
        "CallerReference": "cli-example",
        "Comment": "Example OAI"
    }
}
```

- Einzelheiten zur API finden Sie GetCloudFrontOriginAccessIdentityConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Konfigurationsinformationen zu einer einzelnen CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Fehler treten auf, wenn kein -Id-Parameter angegeben ist.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
      CallerReference                                          Comment
```

```
     ---------------                                                  -------
     mycallerreference: 2/1/2011 1:16:32 PM                           Caller
reference: 2/1/2011 1:16:32 PM
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentityConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Konfigurationsinformationen zu einer einzelnen CloudFront Amazon-Ursprungszugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Fehler treten auf, wenn kein -Id-Parameter angegeben ist.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
     CallerReference                                                  Comment
     ---------------                                                  -------
     mycallerreference: 2/1/2011 1:16:32 PM                           Caller
reference: 2/1/2011 1:16:32 PM
```

- Einzelheiten zur API finden Sie unter GetCloudFrontOriginAccessIdentityConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDistribution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDistribution` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

CLI

AWS CLI

Um eine CloudFront Distribution zu bekommen

Im folgenden `get-distribution` Beispiel wird die CloudFront Distribution mit der ID abgerufen`EDFDVBD6EXAMPLE`, einschließlich ihrer`ETag`. Die Distributions-ID wird in den Befehlen „create-distribution“ und „list-distributions“ zurückgegeben.

```
aws cloudfront get-distribution \
    --id EDFDVBD6EXAMPLE
```

Ausgabe:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "Deployed",
        "LastModifiedTime": "2019-12-04T23:35:41.433Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
```

```
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
```

```
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- Einzelheiten zur API finden Sie GetDistributionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft die Informationen für eine bestimmte Distribution ab.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistribution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft die Informationen für eine bestimmte Distribution ab.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistribution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetDistributionConfig`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDistributionConfig` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

CLI

AWS CLI

Um eine CloudFront Distributionskonfiguration zu erhalten

Im folgenden Beispiel werden Metadaten über die CloudFront Distribution mit der ID abgerufen`EDFDVBD6EXAMPLE`, einschließlich ihrer`ETag`. Die Distributions-ID wird in den Befehlen „create-distribution“ und „list-distributions“ zurückgegeben.

```
aws cloudfront get-distribution-config \
    --id EDFDVBD6EXAMPLE
```

Ausgabe:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
                    "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "amzn-s3-demo-bucket.s3.amazonaws.com-cli-example",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
```

```
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
```

```
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- Einzelheiten zur API finden Sie GetDistributionConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft die Konfiguration für eine bestimmte Distribution ab.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistributionConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft die Konfiguration für eine bestimmte Distribution ab.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- Einzelheiten zur API finden Sie unter GetDistributionConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
```

```
        )
        print("Done!")
```

- Einzelheiten zur API finden Sie GetDistributionConfigin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`ListCloudFrontOriginAccessIdentities`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListCloudFrontOriginAccessIdentities` verwendet wird.

CLI

AWS CLI

Um die ursprünglichen CloudFront Zugriffsidentitäten aufzulisten

Im folgenden Beispiel wird eine Liste der CloudFront Origin-Zugriffsidentitäten (OAIs) in Ihrem Konto abgerufen: AWS

```
aws cloudfront list-cloud-front-origin-access-identities
```

Ausgabe:

```
{
    "CloudFrontOriginAccessIdentityList": {
        "Items": [
            {
                "Id": "E74FTE3AEXAMPLE",
                "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
                "Comment": "Example OAI"
            },
            {
                "Id": "EH1HDMBEXAMPLE",
                "S3CanonicalUserId":
 "1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
                "Comment": "Test OAI"
```

```
            },
            {
                "Id": "E2X2C9TEXAMPLE",
                "S3CanonicalUserId":
 "cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
                "Comment": "Example OAI #2"
            }
        ]
    }
}
```

- Einzelheiten zur API finden Sie ListCloudFrontOriginAccessIdentitiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der CloudFront Amazon-Origin-Zugriffsidentitäten zurückgegeben. Da der MaxItem Parameter - den Wert 2 angibt, enthalten die Ergebnisse zwei Identitäten.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Ausgabe:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Einzelheiten zur API finden Sie unter ListCloudFrontOriginAccessIdentities AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der CloudFront Amazon-Origin-Zugriffsidentitäten zurückgegeben. Da der MaxItem Parameter - den Wert 2 angibt, enthalten die Ergebnisse zwei Identitäten.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Ausgabe:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- Einzelheiten zur API finden Sie unter ListCloudFrontOriginAccessIdentities AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListDistributions`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDistributions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Webanwendungs-Firewalls

CLI

AWS CLI

Um CloudFront Distributionen aufzulisten

Im folgenden Beispiel wird eine Liste der CloudFront Verteilungen in Ihrem AWS Konto abgerufen.

```
aws cloudfront list-distributions
```

Ausgabe:

```
{
    "DistributionList": {
        "Items": [
            {
```

```
                "Id": "E23YS8OEXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
E23YS8OEXAMPLE",
                "Status": "Deployed",
                "LastModifiedTime": "2024-08-05T18:23:40.375000+00:00",
                "DomainName": "abcdefgh12ijk.cloudfront.net",
                "Aliases": {
                    "Quantity": 0
                },
                "Origins": {
                    "Quantity": 1,
                    "Items": [
                        {
                            "Id": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                            "DomainName": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                            "OriginPath": "",
                            "CustomHeaders": {
                                "Quantity": 0
                            },
                            "S3OriginConfig": {
                                "OriginAccessIdentity": ""
                            },
                            "ConnectionAttempts": 3,
                            "ConnectionTimeout": 10,
                            "OriginShield": {
                                "Enabled": false
                            },
                            "OriginAccessControlId": "EIAP8PEXAMPLE"
                        }
                    ]
                },
                "OriginGroups": {
                    "Quantity": 0
                },
                "DefaultCacheBehavior": {
                    "TargetOriginId": "amzn-s3-demo-bucket.s3.us-
east-1.amazonaws.com",
                    "TrustedSigners": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "TrustedKeyGroups": {
```

```
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "ViewerProtocolPolicy": "allow-all",
                    "AllowedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ],
                        "CachedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "Compress": true,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": "",
                    "CachePolicyId": "658327ea-f89d-4fab-a63d-7e886EXAMPLE"
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "SSLSupportMethod": "vip",
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
```

```
                "Restrictions": {
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "HTTP2",
                "IsIPV6Enabled": true,
                "Staging": false
            }
        ]
    }
}
```

- Einzelheiten zur API finden Sie ListDistributionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Distributionen zurück.

```
Get-CFDistributionList
```

- Einzelheiten zur API finden Sie unter ListDistributions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Distributionen zurück.

```
Get-CFDistributionList
```

- Einzelheiten zur API finden Sie unter ListDistributions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
 "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- Einzelheiten zur API finden Sie ListDistributionsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_fnt->listdistributions( ). " oo_result is returned for
testing purposes. "
       MESSAGE 'Retrieved list of CloudFront distributions.' TYPE 'I'.
     CATCH /aws1/cx_fntinvalidargument.
       MESSAGE 'Invalid argument provided.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ListDistributionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateDistribution`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateDistribution` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

CLI

AWS CLI

Beispiel 1: Um das Standard-Root-Objekt einer CloudFront Distribution zu aktualisieren

Im folgenden Beispiel wird das Standard-Stammobjekt `index.html` für die CloudFront Distribution mit der ID aktualisiert`EDFDVBD6EXAMPLE`.

```
aws cloudfront update-distribution \
    --id EDFDVBD6EXAMPLE \
    --default-root-object index.html
```

Ausgabe:

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:55:39.870Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "example-website",
                        "DomainName": "www.example.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 443,
                            "OriginProtocolPolicy": "match-viewer",
                            "OriginSslProtocols": {
```

```
                                "Quantity": 2,
                                "Items": [
                                    "SSLv3",
                                    "TLSv1"
                                ]
                            },
                            "OriginReadTimeout": 30,
                            "OriginKeepaliveTimeout": 5
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "example-website",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 1,
                        "Items": [
                            "*"
                        ]
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
```

```
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
```

```
            "HttpVersion": "http1.1",
            "IsIPV6Enabled": true
        }
    }
}
```

Beispiel 2: Um eine CloudFront Distribution zu aktualisieren

Im folgenden Beispiel wird die CloudFront Distribution mit der ID deaktiviert, `EMLARXS9EXAMPLE` indem die Verteilungskonfiguration in einer JSON-Datei mit dem Namen `dist-config-disable.json` bereitgestellt wird. Um eine Distribution zu aktualisieren, müssen Sie die Option `--if-match` verwenden, um das `ETag` der Distribution anzugeben. Um den `ETag` zu erhalten, verwenden Sie bitte den Befehl „get-Distribution“ oder „get-Distribution-Config“. Beachten Sie, dass das Feld `Enabled` in der JSON-Datei auf `false` gesetzt ist.

Nachdem Sie das folgende Beispiel verwendet haben, um die Distribution zu deaktivieren, können Sie den Befehl „delete-distribution“ verwenden, um sie zu löschen.

```
aws cloudfront update-distribution \
    --id EMLARXS9EXAMPLE \
    --if-match E2QWRUHEXAMPLE \
    --distribution-config file://dist-config-disable.json
```

Inhalt von `dist-config-disable.json`:

```
{
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "amzn-s3-demo-bucket.s3.amazonaws.com-1574382155-273939",
                "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
```

```
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
```

```
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

Ausgabe:

```
{
```

```
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:32:35.553Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1574382155-496510",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
                        "DomainName": "amzn-s3-demo-bucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "amzn-s3-demo-
bucket.s3.amazonaws.com-1574382155-273939",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
```

```
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
```

```
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": false,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- Einzelheiten zur API finden Sie unter UpdateDistribution AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
```

```
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
 software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
 idVal) {
        try {
```

```
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();

            GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to
comment and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

        } catch (CloudFrontException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie UpdateDistributionin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]
```

```
        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- Einzelheiten zur API finden Sie UpdateDistributionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Get the current distribution configuration and ETag "
        DATA(lo_distribution_config_result) = lo_fnt-
>getdistributionconfig( iv_id = iv_distribution_id ).
        DATA(lo_old_config) = lo_distribution_config_result-
>get_distributionconfig( ).
        DATA(lv_etag) = lo_distribution_config_result->get_etag( ).

        " Create a new distribution config with the updated comment "
        " Since the config object is immutable, we need to create a new one with
 all existing values "
        DATA(lo_new_config) = NEW /aws1/cl_fntdistributionconfig(
          iv_callerreference = lo_old_config->get_callerreference( )
```

```
          io_aliases = lo_old_config->get_aliases( )
          iv_defaultrootobject = lo_old_config->get_defaultrootobject( )
          io_origins = lo_old_config->get_origins( )
          io_origingroups = lo_old_config->get_origingroups( )
          io_defaultcachebehavior = lo_old_config->get_defaultcachebehavior( )
          io_cachebehaviors = lo_old_config->get_cachebehaviors( )
          io_customerrorresponses = lo_old_config->get_customerrorresponses( )
          iv_comment = iv_comment
          io_logging = lo_old_config->get_logging( )
          iv_priceclass = lo_old_config->get_priceclass( )
          iv_enabled = lo_old_config->get_enabled( )
          io_viewercertificate = lo_old_config->get_viewercertificate( )
          io_restrictions = lo_old_config->get_restrictions( )
          iv_webaclid = lo_old_config->get_webaclid( )
          iv_httpversion = lo_old_config->get_httpversion( )
          iv_isipv6enabled = lo_old_config->get_isipv6enabled( ) ).

        " Update the distribution with the modified configuration "
        lo_fnt->updatedistribution(
          io_distributionconfig = lo_new_config
          iv_id = iv_distribution_id
          iv_ifmatch = lv_etag ).
        MESSAGE 'CloudFront distribution updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_fntnosuchdistribution.
        MESSAGE 'Distribution does not exist.' TYPE 'E'.
      CATCH /aws1/cx_fntpreconditionfailed.
        MESSAGE 'Precondition failed - ETag mismatch.' TYPE 'E'.
      CATCH /aws1/cx_fntinvalidifmatchvrs.
        MESSAGE 'Invalid If-Match version.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateDistributionin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung CloudFront AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien CloudFront mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb CloudFront oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- SaaS-Manager-Ressourcen erstellen AWS SDK
- Löschen Sie Signaturressourcen CloudFront mit AWS SDK
- Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten
- Erste Schritte mit Firewalls für Webanwendungen
- Erstellen Sie signierte URLs und Cookies mit einem AWS SDK

SaaS-Manager-Ressourcen erstellen AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine mandantenfähige Distribution und einen Distributionsmandanten mit verschiedenen Konfigurationen erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Beispiel zeigt, wie Sie eine Multi-Tenant-Distribution mit Parametern und einem Platzhalterzertifikat erstellen.

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
```

```
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
                                                                        S3Client
 s3Client,
                                                                        final String
 bucketName,
                                                                        final String
 certificateArn) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with
 java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .viewerCertificate(certBuilder -> certBuilder
                                .acmCertificateArn(certificateArn)
                                .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
```

```
                                                                    ""))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName
 value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }

    public static Distribution
 CreateMultiTenantDistributionNoCert(CloudFrontClient cloudFrontClient,
                                                                S3Client s3Client,
                                                                final String
 bucketName) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
```

```
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with
 java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName
 value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));
```

```
        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }
}
```

Das folgende Beispiel zeigt, wie Sie einen dieser Vorlage zugeordneten Distributionsmandanten erstellen und dabei den oben definierten Parameter verwenden. Beachten Sie, dass Sie hier keine Zertifikatsinformationen hinzufügen müssen, da Ihre Domain bereits durch die übergeordnete Vorlage abgedeckt ist.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantNoCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
```

```
                                                                        String
distributionId,
                                                                        String
domain,
                                                                        String
hostedZoneId) {
        CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("no-cert-tenant")
        );

        final DistributionTenant distributionTenant =
createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection
group that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

.value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
```

```
    }

}
```

Wenn das Viewer-Zertifikat in der übergeordneten Vorlage weggelassen wurde, müssen Sie stattdessen Zertifikatsinformationen für die damit verknüpften Mandanten hinzufügen. Das folgende Beispiel zeigt, wie Sie dies mithilfe eines ACM-Zertifikats-ARN tun, der die erforderliche Domain für den Mandanten abdeckt.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,

 Route53Client route53Client,
                                                                        String
 distributionId,
                                                                        String
 domain,
                                                                        String
 hostedZoneId,
                                                                        String
 certificateArn) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
```

```
                        .domain(domain))
                .enabled(false)
                .name("tenant-with-cert")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .customizations(b3 -> b3
                        .certificate(b4 -> b4
                                .arn(certificateArn))) // NOTE: Cert must be in
 Us-East-1 and cover the domain provided in this request

        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection
 group that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

Das folgende Beispiel zeigt, wie Sie dies mit einer CloudFront-hosted verwalteten Zertifikatsanforderung tun können. Diese Methode eignet sich ideal, wenn noch kein Datenverkehr zu Ihrer Domain besteht. In diesem Fall erstellen wir eine, ConnectionGroup um eine zu generieren RoutingEndpoint. Dann verwenden wir das, RoutingEndpoint um DNS-Einträge zu erstellen, die den Domainbesitz verifizieren und auf die wir verweisen CloudFront. CloudFront stellt dann automatisch ein Token bereit, um den Domainbesitz zu validieren und ein verwaltetes Zertifikat zu erstellen.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,

 Route53Client route53Client,
                                                                        String
 distributionId,
                                                                        String
 domain,
                                                                        String
 hostedZoneId) throws InterruptedException {
        CreateConnectionGroupResponse createConnectionGroupResponse =
 cloudFrontClient.createConnectionGroup(builder -> builder
                .ipv6Enabled(true)
                .name("cf-hosted-connection-group")
                .enabled(true));

        route53Client.changeResourceRecordSets(builder -> builder
```

```
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("cf-hosted domain validation record")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type(RRType.CNAME)
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

.value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));

        // Give the R53 record time to propagate, if it isn't being returned by
servers yet, the following call will fail
        Thread.sleep(60000);

        CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))

.connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
                .enabled(false)
                .name("cf-hosted-tenant")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

Das folgende Beispiel zeigt, wie Sie dies mit einer selbst gehosteten verwalteten Zertifikatsanforderung durchführen können. Dies ist ideal, wenn bereits Datenverkehr zu

Ihrer Domain besteht und Sie während einer Migration keine Ausfallzeiten in Kauf nehmen können. Am Ende dieses Beispiels wird der Mandant in einem Zustand erstellt, in dem die Domainvalidierung und die DNS-Einrichtung noch ausstehen. Folgen Sie den Schritten [hier] (https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certificates.html#complete -domain-ownership), um die Einrichtung abzuschließen, wenn Sie bereit sind, Traffic zu migrieren.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import
 software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                        String
 distributionId,
                                                                        String
 domain) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
```

```
                )
        );

        return createResponse.distributionTenant();
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDistribution
  - CreateDistributionTenant

Löschen Sie Signaturressourcen CloudFront mit AWS SDK

Das folgende Codebeispiel zeigt, wie Ressourcen gelöscht werden, die für den Zugriff auf eingeschränkte Inhalte in einem Bucket von Amazon Simple Storage Service(Amazon S3) verwendet werden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
```

```
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
 final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));
```

```
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Beginnen Sie mit einem grundlegenden Netzwerk zur Verteilung von Inhalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Laden Sie Inhalte in den Bucket hoch
- Erstellen Sie eine CloudFront Distribution mit OAC
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudFront Getting Started Tutorial Script
```

```
# This script creates an S3 bucket, uploads sample content, creates a CloudFront
 distribution with OAC,
# and demonstrates how to access content through CloudFront.

set -euo pipefail

# Set up logging
LOG_FILE="cloudfront-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting CloudFront Getting Started Tutorial at $(date)"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created before error:"
    if [ -n "${BUCKET_NAME:-}" ]; then
        echo "- S3 Bucket: $BUCKET_NAME"
    fi
    if [ -n "${OAC_ID:-}" ]; then
        echo "- CloudFront Origin Access Control: $OAC_ID"
    fi
    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "- CloudFront Distribution: $DISTRIBUTION_ID"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "Cleaning up resources..."

    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "Disabling CloudFront distribution $DISTRIBUTION_ID..."

        # Get the current configuration and ETag in one call
        DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
            echo "Failed to get distribution config. Continuing with cleanup..."
            DIST_CONFIG=""
        }
```

```
        if [ -n "$DIST_CONFIG" ]; then
            ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')

            # Modify and update distribution in one pipeline
            if echo "$DIST_CONFIG" | jq '.DistributionConfig.Enabled = false' | \
                aws cloudfront update-distribution \
                    --id "$DISTRIBUTION_ID" \
                    --distribution-config "$(cat)" \
                    --if-match "$ETAG" 2>/dev/null; then

                echo "Waiting for distribution to be disabled (this may take
 several minutes)..."
                aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID"
 2>/dev/null || {
                    echo "Distribution deployment wait timed out. Proceeding with
 deletion..."
                }

                # Get fresh ETag for deletion
                DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
                    echo "Failed to get updated config. Skipping distribution
 deletion..."
                    DIST_CONFIG=""
                }

                if [ -n "$DIST_CONFIG" ]; then
                    ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')
                    aws cloudfront delete-distribution --id "$DISTRIBUTION_ID" --
if-match "$ETAG" 2>/dev/null && \
                        echo "CloudFront distribution deleted." || \
                        echo "Failed to delete distribution. You may need to
 delete it manually."
                fi
            else
                echo "Failed to disable distribution. Continuing with cleanup..."
            fi
        fi
    fi

    if [ -n "${OAC_ID:-}" ]; then
        echo "Deleting Origin Access Control $OAC_ID..."
```

```
        OAC_DATA=$(aws cloudfront get-origin-access-control --id "$OAC_ID" 2>/
dev/null) || {
            echo "Failed to get Origin Access Control. You may need to delete it
 manually."
            OAC_DATA=""
        }

        if [ -n "$OAC_DATA" ]; then
            OAC_ETAG=$(echo "$OAC_DATA" | jq -r '.ETag')
            aws cloudfront delete-origin-access-control --id "$OAC_ID" --if-match
 "$OAC_ETAG" 2>/dev/null && \
                echo "Origin Access Control deleted." || \
                echo "Failed to delete Origin Access Control. You may need to
 delete it manually."
        fi
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "$BUCKET_IS_SHARED" != "true" ]; then
        echo "Deleting S3 bucket $BUCKET_NAME and its contents..."
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || {
            echo "Failed to remove bucket contents. Continuing with bucket
 deletion..."
        }

        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null && \
            echo "S3 bucket deleted." || \
            echo "Failed to delete bucket. You may need to delete it manually."
    fi

    # Clean up temporary files
    rm -f temp_disabled_config.json distribution-config.json bucket-policy.json
 2>/dev/null || true
    rm -rf temp_content 2>/dev/null || true
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Validate jq is available
```

```
if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

# Validate AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Initialize variables
BUCKET_NAME=""
OAC_ID=""
DISTRIBUTION_ID=""
BUCKET_IS_SHARED=false

# Generate a random identifier for the bucket name using secure random
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
    handle_error "Failed to generate random identifier"
fi

# Check for shared prereq bucket and get account ID in parallel calls
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null) || PREREQ_BUCKET=""

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="cloudfront-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Get AWS account ID early
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
if [ $? -ne 0 ]; then
    handle_error "Failed to get AWS account ID"
fi

# Validate account ID format
```

```
if ! [[ "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    handle_error "Invalid AWS account ID format: $ACCOUNT_ID"
fi

# Create a temporary directory for content with restrictive permissions
TEMP_DIR="temp_content"
mkdir -p "$TEMP_DIR/css"
chmod 700 "$TEMP_DIR"
if [ $? -ne 0 ]; then
    handle_error "Failed to create temporary directory"
fi

# Step 1: Create an S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" != "true" ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    aws s3 mb "s3://$BUCKET_NAME" --region us-east-1
    if [ $? -ne 0 ]; then
        handle_error "Failed to create S3 bucket"
    fi

    aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=cloudfront-
gettingstarted}]'

    # Batch bucket configuration calls for efficiency
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled &
    aws s3api put-public-access-block \
        --bucket "$BUCKET_NAME" \
        --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 &
    aws s3api put-bucket-encryption \
        --bucket "$BUCKET_NAME" \
        --server-side-encryption-configuration '{
            "Rules": [
                {
                    "ApplyServerSideEncryptionByDefault": {
                        "SSEAlgorithm": "AES256"
                    }
                }
            ]
        }' &
    wait
```

```
fi

# Step 2: Create sample content
echo "Creating sample content..."
cat > "$TEMP_DIR/index.html" << 'EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
EOF

cat > "$TEMP_DIR/css/styles.css" << 'EOF'
body {
    font-family: Arial, sans-serif;
    margin: 40px;
    background-color: #f5f5f5;
}
h1 {
    color: #333;
    text-align: center;
}
EOF

# Set restrictive permissions on content files
chmod 600 "$TEMP_DIR/index.html" "$TEMP_DIR/css/styles.css"

# Step 3: Upload content to the S3 bucket with encryption and metadata
echo "Uploading content to S3 bucket..."
aws s3 cp "$TEMP_DIR/" "s3://$BUCKET_NAME/" --recursive \
    --sse AES256 \
    --metadata "Source=CloudFrontTutorial"
if [ $? -ne 0 ]; then
    handle_error "Failed to upload content to S3 bucket"
fi

# Step 4: Create Origin Access Control
echo "Creating Origin Access Control..."
OAC_RESPONSE=$(aws cloudfront create-origin-access-control \
```

```
    --origin-access-control-config Name="oac-for-
$BUCKET_NAME",SigningProtocol=sigv4,SigningBehavior=always,OriginAccessControlOriginType=

if [ $? -ne 0 ]; then
    handle_error "Failed to create Origin Access Control"
fi

OAC_ID=$(echo "$OAC_RESPONSE" | jq -r '.OriginAccessControl.Id')
if [ -z "$OAC_ID" ] || [ "$OAC_ID" = "null" ]; then
    handle_error "Failed to extract OAC ID from response"
fi

# Validate OAC ID format (alphanumeric and hyphens)
if ! [[ "$OAC_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid OAC ID format: $OAC_ID"
fi

echo "Created Origin Access Control with ID: $OAC_ID"

# Step 5: Create CloudFront distribution
echo "Creating CloudFront distribution..."

# Validate bucket name format
if ! [[ "$BUCKET_NAME" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]]; then
    handle_error "Invalid S3 bucket name format: $BUCKET_NAME"
fi

# Create distribution configuration with improved security settings
cat > distribution-config.json << EOF
{
    "CallerReference": "cli-tutorial-$(date +%s)",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "S3-$BUCKET_NAME",
                "DomainName": "$BUCKET_NAME.s3.amazonaws.com",
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                },
                "OriginAccessControlId": "$OAC_ID"
            }
        ]
    },
```

```
    "DefaultCacheBehavior": {
        "TargetOriginId": "S3-$BUCKET_NAME",
        "ViewerProtocolPolicy": "redirect-to-https",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["GET", "HEAD"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["GET", "HEAD"]
            }
        },
        "DefaultTTL": 86400,
        "MinTTL": 0,
        "MaxTTL": 31536000,
        "Compress": true,
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            }
        }
    },
    "Comment": "CloudFront distribution for tutorial",
    "Enabled": true,
    "WebACLId": "",
    "HttpVersion": "http2and3"
}
EOF

# Set restrictive permissions on config file before passing credentials
chmod 600 distribution-config.json

DIST_RESPONSE=$(aws cloudfront create-distribution --distribution-config file://
distribution-config.json)
if [ $? -ne 0 ]; then
    handle_error "Failed to create CloudFront distribution"
fi

DISTRIBUTION_ID=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.Id')
DOMAIN_NAME=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.DomainName')

if [ -z "$DISTRIBUTION_ID" ] || [ "$DISTRIBUTION_ID" = "null" ] || [ -z
 "$DOMAIN_NAME" ] || [ "$DOMAIN_NAME" = "null" ]; then
    handle_error "Failed to extract distribution ID or domain name from response"
```

```
fi

# Validate distribution ID format
if ! [[ "$DISTRIBUTION_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid distribution ID format: $DISTRIBUTION_ID"
fi

echo "Created CloudFront distribution with ID: $DISTRIBUTION_ID"
echo "CloudFront domain name: $DOMAIN_NAME"

# Tag the CloudFront distribution
aws cloudfront tag-resource --resource "arn:aws:cloudfront::
$ACCOUNT_ID:distribution/$DISTRIBUTION_ID" --tags 'Items=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=cloudfront-gettingstarted}]'

# Step 6: Update S3 bucket policy
echo "Updating S3 bucket policy..."

cat > bucket-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::$BUCKET_NAME/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::
$ACCOUNT_ID:distribution/$DISTRIBUTION_ID"
                }
            }
        }
    ]
}
EOF

# Set restrictive permissions on policy file
chmod 600 bucket-policy.json
```

```
aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://bucket-
policy.json
if [ $? -ne 0 ]; then
    handle_error "Failed to update S3 bucket policy"
fi

# Step 7: Wait for distribution to deploy
echo "Waiting for CloudFront distribution to deploy (this may take 5-10
 minutes)..."
aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID" 2>/dev/null ||
 {
    echo "Warning: Distribution deployment wait timed out. The distribution may
 still be deploying."
}

echo "CloudFront distribution is now deployed."

# Step 8: Display access information
echo ""
echo "===== CloudFront Distribution Setup Complete ====="
echo "You can access your content at: https://$DOMAIN_NAME/index.html"
echo ""
echo "Resources created:"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- CloudFront Origin Access Control: $OAC_ID"
echo "- CloudFront Distribution: $DISTRIBUTION_ID"
echo ""
echo "To clean up resources, run: cleanup"
echo ""

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateDistribution
  - CreateOriginAccessControl
  - DeleteDistribution
  - DeleteOriginAccessControl
  - GetDistribution
  - GetDistributionConfig

- GetOriginAccessControl
- UpdateDistribution
- WaitDistributionDeployed

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleich von Zeichenketten hinzu
- Fügen Sie verwaltete Regeln hinzu
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
```

```
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}
```

```
# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
```

```
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
```

```
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "==================================================="
echo "STEP 2: Adding String Match Rule"
echo "==================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
```

```
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
```

```
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
```

```
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "==================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "==================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true
```

```
    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
```

```
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done
```

```
# Step 4: List CloudFront distributions
echo ""
echo "==================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "==================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "==================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "==================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
```

```
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
```

```
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Erstellen Sie signierte URLs und Cookies mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie signierte URLs und Cookies erstellt werden, die den Zugriff auf eingeschränkte Ressourcen ermöglichen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie den CannedSignerRequestKurs, um URLs oder Cookies mit einer vorgefertigten Richtlinie zu signieren.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
```

```
public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

Verwenden Sie die CustomSignerRequestKlasse, um URLs oder Cookies mit einer benutzerdefinierten Richtlinie zu signieren. Die Methoden `activeDate` und `ipRange` sind optional.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
```

```
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

Das folgende Beispiel zeigt, wie die CloudFrontUtilitiesKlasse verwendet wird, um signierte Cookies und URLs zu erzeugen. Sehen Sie sich dieses Codebeispiel unter an GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
```

```
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- Einzelheiten zur API finden Sie CloudFrontUtilitiesin der AWS SDK for Java 2.x API-Referenz.

CloudFront Funktionen, Beispiele für CloudFront

Die folgenden Codebeispiele zeigen, wie die Verwendung CloudFront mit AWS SDKs funktioniert.

Beispiele

- Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis HTTP-Sicherheitsheader hinzu
- Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis einen CORS-Header hinzu
- Fügen Sie einem Antwortereignis des CloudFront Functions-Viewers einen Cache-Control-Header hinzu
- Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen echten Client-IP-Header hinzu
- Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen Origin-Header hinzu
- Fügen Sie index.html hinzu, um URLs ohne Dateinamen in einem CloudFront Functions-Viewer-Anforderungsereignis anzufordern
- Normalisieren Sie die Parameter der Abfragezeichenfolge in einer CloudFront Functions Viewer-Anfrage
- Leiten Sie in einem CloudFront Functions-Viewer-Anforderungsereignis zu einer neuen URL weiter
- Schreiben Sie eine Anforderungs-URI auf der Grundlage der KeyValueStore Konfiguration für ein CloudFront Functions-Viewer-Anforderungsereignis neu
- Leiten Sie Anfragen in einem CloudFront Functions-Viewer-Anforderungsereignis an einen Ursprung weiter, der näher am Betrachter liegt
- Verwenden Sie Schlüssel-Wert-Paare in einer CloudFront Functions-Viewer-Anfrage
- Validieren Sie ein einfaches Token in einer CloudFront Functions-Viewer-Anfrage

Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis HTTP-Sicherheitsheader hinzu

Das folgende Codebeispiel zeigt, wie einem Antwortereignis des CloudFront Functions-Viewers HTTP-Sicherheitsheader hinzugefügt werden.

JavaScript

JavaScript Runtime 2.0 für Funktionen CloudFront

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};
    headers['referrer-policy'] = {value: 'same-origin'};

    // Return the response to viewers
    return response;
}
```

Fügen Sie einem CloudFront Functions-Viewer-Antwortereignis einen CORS-Header hinzu

Das folgende Codebeispiel zeigt, wie ein CORS-Header zu einem Antwortereignis des CloudFront Functions-Viewers hinzugefügt wird.

JavaScript

JavaScript Runtime 2.0 für Funktionen CloudFront

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event)  {
    var request = event.request;
    var response  = event.response;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!response.headers['access-control-allow-origin'] &&
 request.headers['origin']) {
        response.headers['access-control-allow-origin'] = {value:
 request.headers['origin'].value};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

Fügen Sie einem Antwortereignis des CloudFront Functions-Viewers einen Cache-Control-Header hinzu

Das folgende Codebeispiel zeigt, wie ein Cache-Control-Header zu einem Antwortereignis des CloudFront Functions-Viewers hinzugefügt wird.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var response = event.response;
    var headers = response.headers;

    if (response.statusCode >= 200 && response.statusCode < 400) {
        // Set the cache-control header
        headers['cache-control'] = {value: 'public, max-age=63072000'};
    }

    // Return response to viewers
    return response;
}
```

Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen echten Client-IP-Header hinzu

Das folgende Codebeispiel zeigt, wie einem CloudFront Functions-Viewer-Anforderungsereignis ein echter Client-IP-Header hinzugefügt wird.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

Fügen Sie einem CloudFront Functions-Viewer-Anforderungsereignis einen Origin-Header hinzu

Das folgende Codebeispiel zeigt, wie ein Origin-Header zu einem CloudFront Functions-Viewer-Anforderungsereignis hinzugefügt wird.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

Fügen Sie index.html hinzu, um URLs ohne Dateinamen in einem CloudFront Functions-Viewer-Anforderungsereignis anzufordern

Das folgende Codebeispiel zeigt, wie index.html hinzugefügt wird, um URLs ohne Dateinamen in einem CloudFront Functions-Viewer-Anforderungsereignis anzufordern.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var request = event.request;
    var uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

Normalisieren Sie die Parameter der Abfragezeichenfolge in einer CloudFront Functions Viewer-Anfrage

Das folgende Codebeispiel zeigt, wie Abfragezeichenfolgenparameter in einer CloudFront Functions Viewer-Anfrage normalisiert werden.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
function handler(event) {
     var qs=[];
     for (var key in event.request.querystring) {
         if (event.request.querystring[key].multiValue) {
             event.request.querystring[key].multiValue.forEach((mv) =>
 {qs.push(key + "=" + mv.value)});
         } else {
             qs.push(key + "=" + event.request.querystring[key].value);
         }
     };

     event.request.querystring = qs.sort().join('&');


     return event.request;
}
```

Leiten Sie in einem CloudFront Functions-Viewer-Anforderungsereignis zu einer neuen URL weiter

Das folgende Codebeispiel zeigt, wie in einem CloudFront Functions-Viewer-Anforderungsereignis zu einer neuen URL umgeleitet wird.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
async function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;
    var country = 'DE' // Choose a country code
    var newurl = `https://${host}/de/index.html`; // Change the redirect URL to
 your choice

    if (headers['cloudfront-viewer-country']) {
        var countryCode = headers['cloudfront-viewer-country'].value;
        if (countryCode === country) {
            var response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

Schreiben Sie eine Anforderungs-URI auf der Grundlage der KeyValueStore Konfiguration für ein CloudFront Functions-Viewer-Anforderungsereignis neu

Das folgende Codebeispiel zeigt, wie eine Anforderungs-URI auf der Grundlage der KeyValueStore Konfiguration für ein CloudFront Functions-Viewer-Anforderungsereignis neu geschrieben wird.

JavaScript

JavaScript Runtime 2.0 für Funktionen CloudFront

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
import cf from 'cloudfront';

// (Optional) Replace KVS_ID with actual KVS ID
const kvsId = "KVS_ID";
// enable stickiness by setting a cookie from origin or using another edge
 function
const stickinessCookieName = "appversion";
// set to true to enable console logging
const loggingEnabled = false;

// function rewrites the request uri based on configuration in KVS
// example config in KVS in key:value format
// "latest": {"a_weightage": .8, "a_url": "v1", "b_url": "v2"}
// given above key and value in KVS the request uri will be rewritten
// for example http(s)://domain/latest/something/else will be rewritten as
 http(s)://domain/v1/something/else or http(s)://domain/v2/something/else
 depending on weightage
// if no configuration is found, then the request is returned as is
async function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function
 with a distribution.
    const request = event.request;
    const pathSegments = request.uri.split('/');
    const key = pathSegments[1];

    // if empty path segment or if there is valid stickiness cookie
    // then skip call to KVS and let the request continue.
    if (!key || hasValidSticknessCookie(request.cookies[stickinessCookieName],
 key)) {
        return event.request;
    }
```

```
    try {
        // get the prefix replacement from KVS
        const replacement = await getPathPrefixByWeightage(key);
        if (!replacement) {
            return event.request;
        }
        //Replace the first path with the replacement
        pathSegments[1] = replacement;
        log(`using prefix ${pathSegments[1]}`)
        const newUri = pathSegments.join('/');
        log(`${request.uri} -> ${newUri}`);
        request.uri = newUri;

        return request;
    } catch (err) {
        // No change to the path if the key is not found or any other error
        log(`request uri: ${request.uri}, error: ${err}`);
    }
    // no change to path - return request
    return event.request;
}

// function to get the prefix from KVS
async function getPathPrefixByWeightage(key) {
    const kvsHandle = cf.kvs(kvsId);
    // get the weightage config from KVS
    const kvsResponse = await kvsHandle.get(key);
    const weightageConfig = JSON.parse(kvsResponse);
    // no configuration - return null
    if (!weightageConfig || !isFinite(weightageConfig.a_weightage)) {
        return null;
    }
    // return the url based on weightage
    // return null if no url is configured
    if (Math.random() <= weightageConfig.a_weightage) {
        return weightageConfig.a_url ? weightageConfig.a_url: null;
    } else {
        return weightageConfig.b_url ? weightageConfig.b_url : null;
    }
}

// function to check if the stickiness cookie is valid
function hasValidStickinessCookie(stickinessCookie, pathSegment) {
```

```
    // if the value exists and it matches pathSegment
    return (stickinessCookie && stickinessCookie.value === pathSegment)
}

function log(message) {
    if (loggingEnabled) {
        console.log(message);
    }
}
```

Leiten Sie Anfragen in einem CloudFront Functions-Viewer-Anforderungsereignis an einen Ursprung weiter, der näher am Betrachter liegt

Das folgende Codebeispiel zeigt, wie Anfragen in einem CloudFront Functions-Viewer-Anforderungsereignis an einen Ursprung weitergeleitet werden, der näher am Betrachter liegt.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
import cf from 'cloudfront';

function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const country = headers['cloudfront-viewer-country'] &&
        headers['cloudfront-viewer-country'].value;

    //List of Regions with S3 buckets containing content
    const countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
```

```
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    };

    const DEFAULT_REGION = 'us-east-1';

    const selectedRegion = (country && countryToRegion[country]) ||
 DEFAULT_REGION;

    const domainName =
        `cloudfront-functions-demo-bucket-in-${selectedRegion}.s3.
${selectedRegion}.amazonaws.com`;

    cf.updateRequestOrigin({
        "domainName": domainName,
        "originAccessControlConfig": {
            "enabled": true,
            "region": selectedRegion,
            "signingBehavior": "always",
            "signingProtocol": "sigv4",
            "originType": "s3"
        },
    });

    return request;
}
```

Verwenden Sie Schlüssel-Wert-Paare in einer CloudFront Functions-Viewer-Anfrage

Das folgende Codebeispiel zeigt, wie Schlüssel-Wert-Paare in einer CloudFront Functions-Viewer-Anfrage verwendet werden.

JavaScript

JavaScript Runtime 2.0 für Funktionen CloudFront

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
import cf from 'cloudfront';

// This fails if there is no key value store associated with the function
const kvsHandle = cf.kvs();

// Remember to associate the KVS with your function before referencing KVS in
 your code.
// https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/kvs-with-
functions-associate.html
async function handler(event) {
    const request = event.request;
    // Use the first segment of the pathname as key
    // For example http(s)://domain/<key>/something/else
    const pathSegments = request.uri.split('/')
    const key = pathSegments[1]
    try {
        // Replace the first path of the pathname with the value of the key
        // For example http(s)://domain/<value>/something/else
        pathSegments[1] = await kvsHandle.get(key);
        const newUri = pathSegments.join('/');
        console.log(`${request.uri} -> ${newUri}`)
        request.uri = newUri;
    } catch (err) {
        // No change to the pathname if the key is not found
        console.log(`${request.uri} | ${err}`);
    }
    return request;
}
```

Validieren Sie ein einfaches Token in einer CloudFront Functions-Viewer-Anfrage

Das folgende Codebeispiel zeigt, wie ein einfaches Token in einer CloudFront Functions-Viewer-Anfrage validiert wird.

JavaScript

JavaScript Runtime 2.0 für CloudFront Funktionen

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel und Informationen zur Einrichtung und Ausführung finden Sie im CloudFront Functions-Beispiel-Repository.

```
import crypto from 'crypto';
import cf from 'cloudfront';

//Response when JWT is not valid.
const response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

// Remember to associate the KVS with your function before calling the const
 kvsKey = 'jwt.secret'.
// https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/kvs-with-
functions-associate.html
const kvsKey = 'jwt.secret';
// set to true to enable console logging
const loggingEnabled = false;


function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    const segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    const headerSeg = segments[0];
    const payloadSeg = segments[1];
```

```
    const signatureSeg = segments[2];

    // base64 decode and parse JSON
    const payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        const signingMethod = 'sha256';
        const signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        const signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType,
 signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    let xor = 0;
    for (let i = 0; i < a.length; i++) {
    xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
```

```
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}

async function handler(event) {
    let request = event.request;

    //Secret key used to verify JWT token.
    //Update with your own key.
    const secret_key = await getSecret()

    if(!secret_key) {
        return response401;
    }

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        log("Error: No JWT in the querystring");
        return response401;
    }

    const jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, secret_key);
    }
    catch(e) {
        log(e);
        return response401;
```

```
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    log("Valid JWT token");
    return request;
}

// get secret from key value store
async function getSecret() {
    // initialize cloudfront kv store and get the key value
    try {
        const kvsHandle = cf.kvs();
        return await kvsHandle.get(kvsKey);
    } catch (err) {
        log(`Error reading value for key: ${kvsKey}, error: ${err}`);
        return null;
    }

}

function log(message) {
    if (loggingEnabled) {
        console.log(message);
    }
}
```

Codebeispiele für die CloudTrail Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CloudTrail mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CloudTrail Entwicklerhandbuch — Weitere Informationen zu CloudTrail.
- CloudTrail API-Referenz — Details zu allen verfügbaren CloudTrail Aktionen.

- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CloudTrail AWS SDKs
  - Aktionen zur Verwendung CloudTrail AWS SDKs
    - Verwenden Sie CreateTrailmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteTrailmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeTrailmit einem AWS SDK
    - Verwendung von DescribeTrails mit einer CLI
    - Verwendung von GetTrailStatus mit einer CLI
    - Verwenden Sie mit einem ListTrails AWS SDK
    - Verwenden Sie LookupEventsmit einem AWS SDK oder CLI
    - Verwendung von StartLogging mit einer CLI
    - Verwendung von StopLogging mit einer CLI
    - Verwendung von UpdateTrail mit einer CLI

Grundlegende Beispiele für die Verwendung CloudTrail AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CloudTrail mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung CloudTrail AWS SDKs
  - Verwenden Sie CreateTrailmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTrailmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeTrailmit einem AWS SDK
  - Verwendung von DescribeTrails mit einer CLI
  - Verwendung von GetTrailStatus mit einer CLI
  - Verwenden Sie mit einem ListTrails AWS SDK
  - Verwenden Sie LookupEventsmit einem AWS SDK oder CLI

- Verwendung von StartLogging mit einer CLI
- Verwendung von StopLogging mit einer CLI
- Verwendung von UpdateTrail mit einer CLI

Aktionen zur Verwendung CloudTrail AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CloudTrail Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CloudTrail -API-Referenz.

Beispiele

- Verwenden Sie CreateTrailmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTrailmit einem AWS SDK oder CLI
- Verwenden Sie DescribeTrailmit einem AWS SDK
- Verwendung von DescribeTrails mit einer CLI
- Verwendung von GetTrailStatus mit einer CLI
- Verwenden Sie mit einem ListTrails AWS SDK
- Verwenden Sie LookupEventsmit einem AWS SDK oder CLI
- Verwendung von StartLogging mit einer CLI
- Verwendung von StopLogging mit einer CLI
- Verwendung von UpdateTrail mit einer CLI

Verwenden Sie **`CreateTrail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTrail` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which creates an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param bucketName: The Amazon S3 bucket designate for publishing logs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTrailin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Trail

Im folgenden Beispiel für `create-trail` wird ein Multi-Region-Trail mit dem Namen `Trail1` erstellt und ein S3-Bucket angegeben.

```
aws cloudtrail create-trail \
    --name Trail1 \
    --s3-bucket-name amzn-s3-demo-bucket \
    --is-multi-region-trail
```

Ausgabe:

```
{
    "IncludeGlobalServiceEvents": true,
    "Name": "Trail1",
    "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",
    "LogFileValidationEnabled": false,
    "IsMultiRegionTrail": true,
    "S3BucketName": "amzn-s3-demo-bucket"
}
```

- Einzelheiten zur API finden Sie CreateTrailin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Beispiel 2: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet. Die S3-Objekte, die die Protokolle darstellen, verfügen über das gemeinsame Schlüsselpräfix „mylogs“. Wenn neue Protokolle an den Bucket geliefert werden, wird eine Benachrichtigung an das SNS-Thema „mlog-deliverytopic” gesendet.

In diesem Beispiel wird Splatting verwendet, um die Parameterwerte an das Cmdlet zu übergeben.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Einzelheiten zur API finden Sie unter CreateTrail AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Beispiel 2: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet. Die S3-Objekte, die die Protokolle darstellen, verfügen über das gemeinsame Schlüsselpräfix „mylogs“. Wenn neue Protokolle an den Bucket geliefert werden, wird eine Benachrichtigung an das SNS-Thema „mlog-deliverytopic” gesendet.
In diesem Beispiel wird Splatting verwendet, um die Parameterwerte an das Cmdlet zu übergeben.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
}
New-CTTrail @params
```

- Einzelheiten zur API finden Sie unter CreateTrail AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => "arn:aws:s3:::#{bucket_name}"
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:PutObject',
          'Resource' => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
```

```
          'Condition' => {
            'StringEquals' => {
              's3:x-amz-acl' => 'bucket-owner-full-control'
            }
          }
        }
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket_name,
      policy: policy
    )
    puts "Successfully added policy to bucket #{bucket_name}"
  end

  begin
    cloudtrail_client.create_trail({
                                     name: trail_name, # required
                                     s3_bucket_name: bucket_name # required
                                   })

    puts "Successfully created trail: #{trail_name}."
  rescue StandardError => e
    puts "Got error trying to create trail #{trail_name}:\n #{e}"
    puts e
    exit 1
  end
```

- Einzelheiten zur API finden Sie CreateTrailin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`DeleteTrail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTrail` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which deletes an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTrailin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie einen Trail

Der folgende `delete-trail`-Befehl löscht einen Trail mit dem Namen `Trail1`:

```
aws cloudtrail delete-trail --name Trail1
```

- Einzelheiten zur API finden Sie DeleteTrailin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht den angegebenen Trail. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigung zu unterdrücken, fügen Sie den Schalterparameter -Force hinzu.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Einzelheiten zur API finden Sie unter DeleteTrail AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht den angegebenen Trail. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigung zu unterdrücken, fügen Sie den Schalterparameter -Force hinzu.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- Einzelheiten zur API finden Sie unter DeleteTrail AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: #{trail_name}"
rescue StandardError => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
end
```

- Einzelheiten zur API finden Sie DeleteTrailin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`DescribeTrail`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeTrail`.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which describes the AWS CloudTrail trails in an account.
/*!
```

```
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/

bool AwsDoc::CloudTrail::describeTrails(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;

    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
 outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeTrailin der AWS SDK für C++ API-Referenz.

Verwendung von **`DescribeTrails`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrails` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen Trail

Im folgenden Beispiel für `describe-trails` wird der Wert für `Trail1` und `Trail2` zurückgegeben.

```
aws cloudtrail describe-trails \
```

```
    --trail-name-list Trail1 Trail2
```

Ausgabe:

```
{
    "trailList": [
        {
            "IncludeGlobalServiceEvents": true,
            "Name": "Trail1",
            "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail1",
            "LogFileValidationEnabled": false,
            "IsMultiRegionTrail": false,
            "S3BucketName": "amzn-s3-demo-bucket",
            "CloudWatchLogsRoleArn": "arn:aws:iam::123456789012:role/
CloudTrail_CloudWatchLogs_Role",
            "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-
east-1:123456789012:log-group:CloudTrail:*",
            "SnsTopicName": "my-topic",
            "HomeRegion": "us-east-1"
        },
        {
            "IncludeGlobalServiceEvents": true,
            "Name": "Trail2",
            "S3KeyPrefix": "my-prefix",
            "TrailARN": "arn:aws:cloudtrail:us-east-1:123456789012:trail/Trail2",
            "LogFileValidationEnabled": false,
            "IsMultiRegionTrail": false,
            "S3BucketName": "amzn-s3-demo-bucket2",
            "KmsKeyId": "arn:aws:kms:us-
east-1:123456789012:key/4c5ae5ac-3c13-421e-8335-c7868ef6a769",
            "HomeRegion": "us-east-1"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeTrailsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Einstellungen aller Trails zurück, die der aktuellen Region Ihres Kontos zugeordnet sind.

```
Get-CTTrail
```

Beispiel 2: Gibt die Einstellungen für die angegebenen Trails zurück.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Beispiel 3: Gibt die Einstellungen für die angegebenen Trails zurück, die in einer anderen Region als der aktuellen Shell-Standardregion (in diesem Fall der Region Frankfurt (eu-central-1)) erstellt wurden).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter DescribeTrails AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Einstellungen aller Trails zurück, die der aktuellen Region Ihres Kontos zugeordnet sind.

```
Get-CTTrail
```

Beispiel 2: Gibt die Einstellungen für die angegebenen Trails zurück.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Beispiel 3: Gibt die Einstellungen für die angegebenen Trails zurück, die in einer anderen Region als der aktuellen Shell-Standardregion (in diesem Fall der Region Frankfurt (eu-central-1)) erstellt wurden).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter DescribeTrails AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetTrailStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetTrailStatus` verwendet wird.

CLI

AWS CLI

So rufen Sie den Status eines Trails ab

Der folgende `get-trail-status`-Befehl gibt die Liefer- und Protokollierungsdetails für `Trail1` zurück:

```
aws cloudtrail get-trail-status --name Trail1
```

Ausgabe:

```
{
  "LatestNotificationTime": 1454022144.869,
  "LatestNotificationAttemptSucceeded": "2016-01-28T23:02:24Z",
  "LatestDeliveryAttemptTime": "2016-01-28T23:02:24Z",
  "LatestDeliveryTime": 1454022144.869,
  "TimeLoggingStarted": "2015-11-06T18:36:38Z",
  "LatestDeliveryAttemptSucceeded": "2016-01-28T23:02:24Z",
  "IsLogging": true,
  "LatestCloudWatchLogsDeliveryTime": 1454022144.918,
  "StartLoggingTime": 1446834998.695,
  "StopLoggingTime": 1446834996.933,
  "LatestNotificationAttemptTime": "2016-01-28T23:02:24Z",
  "TimeLoggingStopped": "2015-11-06T18:36:36Z"
}
```

- Einzelheiten zur API finden Sie GetTrailStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Statusinformationen für den Trail mit dem Namen „myExampleTrail“ zurück. Die zurückgegebenen Daten umfassen Informationen zu Zustellungsfehlern, Amazon-SNS- und Amazon-S3-Fehlern sowie Start- und Endzeiten für die Protokollierung des Trails. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Get-CTTrailStatus -Name myExampleTrail
```

Beispiel 2: Gibt Statusinformationen für einen Trail zurück, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter GetTrailStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Statusinformationen für den Trail mit dem Namen „my“ ExampleTrail zurück. Die zurückgegebenen Daten umfassen Informationen zu Zustellungsfehlern, Amazon-SNS- und Amazon-S3-Fehlern sowie Start- und Endzeiten für die Protokollierung des Trails. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Get-CTTrailStatus -Name myExampleTrail
```

Beispiel 2: Gibt Statusinformationen für einen Trail zurück, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter GetTrailStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListTrails`** AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListTrails`.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           #{trail.name}"
    puts "S3 bucket name: #{trail.s3_bucket_name}"
    puts
  end
```

- Einzelheiten zur API finden Sie ListTrailsin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`LookupEvents`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `LookupEvents` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which looks up events captured by AWS CloudTrail.
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::lookupEvents(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;

    size_t count = 0;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
 cloudtrail.LookupEvents(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
 outcome.GetResult().GetEvents();
            count += events.size();
            allEvents.insert(allEvents.end(), events.begin(), events.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
```

```
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!nextToken.empty() && count <= 50); // Limit to 50 events.

    std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

    for (auto &event: allEvents) {
        std::cout << "Event name: " << event.GetEventName() << std::endl;
        std::cout << "Event source: " << event.GetEventSource() << std::endl;
        std::cout << "Event id: " << event.GetEventId() << std::endl;
        std::cout << "Resources: " << std::endl;
        for (auto &resource: event.GetResources()) {
            std::cout << "  " << resource.GetResourceName() << std::endl;
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie LookupEventsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Ereignisse für einen Trail ab

Der folgende `lookup-events`-Befehl ruft API-Aktivitätsereignisse anhand des Attributs `EventName` ab:

```
aws cloudtrail lookup-events --lookup-
attributes AttributeKey=EventName,AttributeValue=ConsoleLogin
```

Ausgabe:

```
{
  "Events": [
     {
         "EventId": "654ccbc0-ba0d-486a-9076-dbf7274677a7",
```

```
            "Username": "my-session-name",
            "EventTime": "2021-11-18T09:41:02-08:00",
            "CloudTrailEvent": "{\"eventVersion\":\"1.02\",\"userIdentity
\":{\"type\":\"AssumedRole\",\"principalId\":\"AROAJIKPFTA72SWU4L7T4:my-
session-name\",\"arn\":\"arn:aws:sts::123456789012:assumed-role/my-
role/my-session-name\",\"accountId\":\"123456789012\",\"sessionContext
\":{\"attributes\":{\"mfaAuthenticated\":\"false\",\"creationDate\":
\"2016-01-26T21:42:12Z\"},\"sessionIssuer\":{\"type\":\"Role\",\"principalId
\":\"AROAJIKPFTA72SWU4L7T4\",\"arn\":\"arn:aws:iam::123456789012:role/my-
role\",\"accountId\":\"123456789012\",\"userName\":\"my-role\"}}},\"eventTime
\":\"2016-01-26T21:42:12Z\",\"eventSource\":\"signin.amazonaws.com\",
\"eventName\":\"ConsoleLogin\",\"awsRegion\":\"us-east-1\",\"sourceIPAddress\":
\"72.21.198.70\",\"userAgent\":\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36\",
\"requestParameters\":null,\"responseElements\":{\"ConsoleLogin\":\"Success\"},
\"additionalEventData\":{\"MobileVersion\":\"No\",\"MFAUsed\":\"No\"},\"eventID
\":\"654ccbc0-ba0d-486a-9076-dbf7274677a7\",\"eventType\":\"AwsConsoleSignIn\",
\"recipientAccountId\":\"123456789012\"}",
            "EventName": "ConsoleLogin",
            "Resources": []
        }
    ]
}
```

- Einzelheiten zur API finden Sie LookupEventsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind. Das Cmdlet führt standardmäßig automatisch mehrere Aufrufe aus, um alle Ereignisse bereitzustellen, und beendet die Ausführung, wenn der Service angibt, dass keine weiteren Daten verfügbar sind.

```
Find-CTEvent
```

Beispiel 2: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind, und gibt dabei eine Region an, die nicht dem aktuellen Shell-Standard entspricht.

```
Find-CTEvent -Region eu-central-1
```

Beispiel 3: Gibt alle Ereignisse zurück, die mit dem RunInstances API-Aufruf verknüpft sind.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Beispiel 4: Gibt die ersten 5 verfügbaren Ereignisse zurück.

```
Find-CTEvent -MaxResult 5
```

- Einzelheiten zur API finden Sie unter LookupEvents AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind. Das Cmdlet führt standardmäßig automatisch mehrere Aufrufe aus, um alle Ereignisse bereitzustellen, und beendet die Ausführung, wenn der Service angibt, dass keine weiteren Daten verfügbar sind.

```
Find-CTEvent
```

Beispiel 2: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind, und gibt dabei eine Region an, die nicht dem aktuellen Shell-Standard entspricht.

```
Find-CTEvent -Region eu-central-1
```

Beispiel 3: Gibt alle Ereignisse zurück, die mit dem RunInstances API-Aufruf verknüpft sind.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Beispiel 4: Gibt die ersten 5 verfügbaren Ereignisse zurück.

```
Find-CTEvent -MaxResult 5
```

- Einzelheiten zur API finden Sie unter LookupEvents AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts 'Resources:'

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ''
    end
  end
end
```

- Einzelheiten zur API finden Sie LookupEventsin der AWS SDK für Ruby API-Referenz.

Verwendung von **`StartLogging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StartLogging` verwendet wird.

CLI

AWS CLI

So starten Sie das Protokollieren für einen Trail

Der folgende `start-logging`-Befehl aktiviert das Protokollieren für `Trail1`:

```
aws cloudtrail start-logging --name Trail1
```

- Einzelheiten zur API finden Sie StartLoggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „myExampleTrail“. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Start-CTLogging -Name myExampleTrail
```

Beispiel 2: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als dem aktuellen Shell-Standard erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter StartLogging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „my“ExampleTrail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Start-CTLogging -Name myExampleTrail
```

Beispiel 2: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als dem aktuellen Shell-Standard erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter StartLogging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StopLogging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StopLogging` verwendet wird.

CLI

AWS CLI

So beenden Sie die Protokollierung eines Trails

Der folgende `stop-logging`-Befehl deaktiviert die Protokollierung für `Trail1`:

```
aws cloudtrail stop-logging --name Trail1
```

- Einzelheiten zur API finden Sie StopLoggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „my“ExampleTrail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Stop-CTLogging -Name myExampleTrail
```

Beispiel 2: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (StopLoggingV4).

Tools für V5 PowerShell

Beispiel 1: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „my“ExampleTrail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Stop-CTLogging -Name myExampleTrail
```

Beispiel 2: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (StopLoggingV5).

Verwendung von **`UpdateTrail`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateTrail` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie einen Trail

Im folgenden Beispiel für `update-trail` wird ein Trail aktualisiert, um einen vorhandenen Bucket für die Protokollzustellung zu verwenden.

```
aws cloudtrail update-trail \
    --name Trail1 \
    --s3-bucket-name amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "IncludeGlobalServiceEvents": true,
    "Name": "Trail1",
    "TrailARN": "arn:aws:cloudtrail:us-west-2:123456789012:trail/Trail1",
    "LogFileValidationEnabled": false,
    "IsMultiRegionTrail": true,
    "S3BucketName": "amzn-s3-demo-bucket"
}
```

- Einzelheiten zur API finden Sie UpdateTrailin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert den angegebenen Trail, sodass globale Serviceereignisse (z. B. von IAM) aufgezeichnet werden, und ändert das gemeinsame Schlüsselpräfix der Protokolldateien künftig in „globallogs“.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Beispiel 2: Aktualisiert den angegebenen Trail, sodass Benachrichtigungen über neue Protokollbereitstellungen an das angegebene SNS-Thema gesendet werden.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Beispiel 3: Aktualisiert den angegebenen Trail, sodass Protokolle in einen anderen Bucket zugestellt werden.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Einzelheiten zur API finden Sie unter UpdateTrail AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert den angegebenen Trail, sodass globale Serviceereignisse (z. B. von IAM) aufgezeichnet werden, und ändert das gemeinsame Schlüsselpräfix der Protokolldateien künftig in „globallogs“.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Beispiel 2: Aktualisiert den angegebenen Trail, sodass Benachrichtigungen über neue Protokollbereitstellungen an das angegebene SNS-Thema gesendet werden.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Beispiel 3: Aktualisiert den angegebenen Trail, sodass Protokolle in einen anderen Bucket zugestellt werden.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- Einzelheiten zur API finden Sie unter UpdateTrail AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Verwendung CloudWatch AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon CloudWatch mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- CloudWatch Benutzerhandbuch — Weitere Informationen zu CloudWatch.
- CloudWatch API-Referenz — Details zu allen verfügbaren CloudWatch Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CloudWatch AWS SDKs
  - Hallo CloudWatch
  - Erfahren Sie mehr über die Kernoperationen für die CloudWatch Verwendung eines AWS SDK
  - Aktionen zur CloudWatch Verwendung AWS SDKs
    - Verwenden Sie DeleteAlarmsmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteAnomalyDetectormit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDashboardsmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem DescribeAlarmHistory AWS SDK oder CLI
    - Verwenden Sie DescribeAlarmsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeAlarmsForMetricmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeAnomalyDetectorsmit einem AWS SDK oder CLI
    - Verwenden Sie DisableAlarmActionsmit einem AWS SDK oder CLI
    - Verwenden Sie EnableAlarmActionsmit einem AWS SDK oder CLI
    - Verwenden Sie GetDashboardmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem GetMetricData AWS SDK oder CLI
    - Verwenden Sie GetMetricStatisticsmit einem AWS SDK oder CLI
    - Verwenden Sie GetMetricWidgetImagemit einem AWS SDK oder CLI
    - Verwenden Sie ListDashboardsmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem ListMetrics AWS SDK oder CLI
    - Verwenden Sie PutAnomalyDetectormit einem AWS SDK oder CLI
    - Verwenden Sie PutDashboardmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem PutMetricAlarm AWS SDK oder CLI
    - Verwenden Sie PutMetricDatamit einem AWS SDK oder CLI
- Szenarien für die Verwendung CloudWatch AWS SDKs
  - Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
  - Beginnen Sie mit CloudWatch Alarmen mit einem AWS SDK
  - Verwalten Sie CloudWatch Metriken und Alarme mit einem AWS SDK
  - Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK

  - Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Grundlegende Beispiele für die Verwendung CloudWatch AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon CloudWatch mit AWS SDKs verwenden können.

Beispiele

- Hallo CloudWatch
- Erfahren Sie mehr über die Kernoperationen für die CloudWatch Verwendung eines AWS SDK
- Aktionen zur CloudWatch Verwendung AWS SDKs
  - Verwenden Sie DeleteAlarmsmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteAnomalyDetectormit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDashboardsmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem DescribeAlarmHistory AWS SDK oder CLI
  - Verwenden Sie DescribeAlarmsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeAlarmsForMetricmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeAnomalyDetectorsmit einem AWS SDK oder CLI
  - Verwenden Sie DisableAlarmActionsmit einem AWS SDK oder CLI
  - Verwenden Sie EnableAlarmActionsmit einem AWS SDK oder CLI
  - Verwenden Sie GetDashboardmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem GetMetricData AWS SDK oder CLI
  - Verwenden Sie GetMetricStatisticsmit einem AWS SDK oder CLI
  - Verwenden Sie GetMetricWidgetImagemit einem AWS SDK oder CLI
  - Verwenden Sie ListDashboardsmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem ListMetrics AWS SDK oder CLI
  - Verwenden Sie PutAnomalyDetectormit einem AWS SDK oder CLI
  - Verwenden Sie PutDashboardmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem PutMetricAlarm AWS SDK oder CLI
  - Verwenden Sie PutMetricDatamit einem AWS SDK oder CLI

Hallo CloudWatch

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von CloudWatch beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;

public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
```

```
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        if (response.Metrics != null)
        {
            foreach (var metric in response.Metrics.Take(5))
            {
                Console.WriteLine($"\tMetric: {metric.MetricName}");
                Console.WriteLine($"\tNamespace: {metric.Namespace}");
                Console.WriteLine(
                    $"\tDimensions: {string.Join(", ", metric.Dimensions.Select(m
 => $"{m.Name}:{m.Value}"))}");
                Console.WriteLine();
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
```

```
                    .forEach(metrics -> System.out.println(" Retrieved metric is:
 " + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
```

```
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListMetricsin der API-Referenz zum AWS SDK für Kotlin.

Erfahren Sie mehr über die Kernoperationen für die CloudWatch Verwendung eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Listet CloudWatch Namespaces und Metriken auf.
- Rufen Sie Statistiken für eine Metrik und die geschätzte Fakturierung ab.
- Erstellen und aktualisieren Sie ein Dashboard.
- Erstellen Sie eine Metrik und fügen Sie ihr Daten hinzu.
- Erstellen und lösen Sie einen Alarm aus und zeigen Sie dann den Alarmverlauf an.
- Fügen Sie einen Anomaliedetektor hinzu.
- Ermitteln Sie ein Metrik-Image, dann bereinigen Sie die Ressourcen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
    */
```

```
   private static ILogger logger = null!;
   private static CloudWatchWrapper _cloudWatchWrapper = null!;
   private static IConfiguration _configuration = null!;
   private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
   private static SingleMetricAnomalyDetector? anomalyDetector = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonCloudWatch>()
           .AddTransient<CloudWatchWrapper>()
       )
       .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
           .CreateLogger<CloudWatchScenario>();

       _cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();

       Console.WriteLine(new string('-', 80));
       Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
       Console.WriteLine(new string('-', 80));

       try
       {
           var selectedNamespace = await SelectNamespace();
```

```
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace,
selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }
```

```
        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d
=> d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
```

```
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string
metricNamespace, Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last
day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };
```

```
        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for
{metric} in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
```

```
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
```

```
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
```

```
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm
{alarmName}? (y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");
```

```
        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for
{alarm.MetricName} {alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
```

```
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            if (metricData[i].Values != null)
            {
                for (int j = 0; j < metricData[i].Values.Count; j++)
                {
                    Console.WriteLine(
                        $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
                }
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger
an alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
```

```
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
```

```
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10
retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
```

```
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
```

```
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state
{detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream,
"MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;

        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
```

```
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
 anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
```

```
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Wrapper-Methoden, die vom Szenario für CloudWatch Aktionen verwendet werden.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
 listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
```

```
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
```

```
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
```

```
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
 metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
 JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
```

```
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
```

```
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
```

```
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
 null)
```

```
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }

    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
```

```
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
```

```
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
```

```
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Beispielwerte für settings.json für das Szenario.

```
{
  "dashboardName": "example-new-dashboard",
  "exampleAlarmName": "example-metric-alarm",
  "accountId": "1234567890",
  "region": "us-east-1",
  "emailTopic": "Default_CloudWatch_Alarms_Topic",
  "customMetricNamespace": "example-namespace",
  "customMetricName": "example-custom-metric",
  "dashboardExampleBody": {
    "widgets": [
      {
        "height": 6,
        "width": 6,
        "y": 0,
        "x": 0,
        "type": "text",
        "properties": {
          "markdown": "# Code Example Dashboard \nThis dashboard was created by
 example code.\n"
        }
      },
      {
        "height": 8,
```

```
        "width": 8,
        "y": 0,
        "x": 6,
        "type": "metric",
        "properties": {
          "metrics": [
            [
              "AWS/Billing",
              "EstimatedCharges",
              "Currency",
              "USD",
              { "region": "us-east-1" }
            ]
          ],
          "view": "timeSeries",
          "region": "us-east-1",
          "stat": "Maximum",
          "period": 86400,
          "yAxis": {
            "left": {
              "min": 0,
              "max": 100
            }
          },
          "stacked": false,
          "title": "Estimated Billing",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      },
      {
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 14,
        "type": "metric",
        "properties": {
          "metrics": [
            [ "AWS/Usage", "CallCount", "Type", "API", "Resource", "ListMetrics",
 "Service", "CloudWatch", "Class", "None" ],
            [ "...", "GetMetricStatistics", ".", ".", ".", "." ],
            [ "...", "GetMetricData", ".", ".", ".", "." ],
```

```
            [ "...", "PutDashboard", ".", ".", ".", "." ],
            [ "...", "PutMetricData", ".", ".", ".", "." ]
          ],
          "view": "timeSeries",
          "yAxis": {
            "left": {
              "min": 0,
              "max": 200
            }
          },
          "stacked": false,
          "region": "us-east-1",
          "stat": "Sum",
          "period": 300,
          "title": "CloudWatch Usage",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      }
    ]
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage

- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem CloudWatch Funktionen demonstriert werden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    static CloudWatchActions cwActions = new CloudWatchActions();

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchScenario.class);
    static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) throws Throwable {
```

```
        final String usage = """

            Usage:
              <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

            Where:
              myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s
              costDateWeek - The start date to use to get AWS/Billing statistics.
(For example, 2023-01-11T18:35:24.00Z.)\s
              dashboardName - The name of the dashboard to create.\s
              dashboardJson - The location of a JSON file to use to create a
dashboard. (See jsonWidgets.json in javav2/example_code/cloudwatch.)\s
              dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
              settings - The location of a JSON file from which various values
are read. (See settings.json in javav2/example_code/cloudwatch.)\s
              metricImage - The location of a BMP file that is used to create a
graph.\s
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }
        String myDate = args[0];
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
        String settings = args[5];
        String metricImage = args[6];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
        logger.info("""
            Amazon CloudWatch is a comprehensive monitoring and observability
service
            provided by Amazon Web Services (AWS). It is designed to help you
monitor your
            AWS resources, applications, and services, as well as on-premises
resources,
```

```
            in real-time.

            CloudWatch collects and tracks various types of data, including
metrics,
            logs, and events, from your AWS and on-premises resources. It allows
you to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they
impact your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch
tasks by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings,
String metricImage ) throws Throwable {
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
        1. List at least five available unique namespaces from Amazon
CloudWatch.
        Select one from the list.
        """);
        String selectedNamespace;
        String selectedMetrics;
```

```
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building
blocks of CloudWatch
            and provide data points that represent a specific aspect of your
system or application over time.

            Select a metric from the list.
            """);
```

```
        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension =
cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and
displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last
day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified
time period.
            """);
        waitForInputToContinue(scanner);
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and
displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is
the ability to
```

```
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
```

```
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future =
cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been
reached: Error message: {}, Error code {}", cwEx.getMessage(),
cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
```

```
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the
alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
```

```
            of your metrics and establish a baseline.

            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential
issues or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
```

```
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        try {
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }
```

```
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für CloudWatch SDK-Methoden.

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
```

```
     *
     * @return the asynchronous CloudWatch client instance
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of
the Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root
node
```

```
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly
Detector", exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
```

```
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse>
deleteDashboardAsync(String dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: "
+ dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String
fileName) {
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
           """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
           .metricWidget(myJSON)
           .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
           .thenCompose(response -> {
               SdkBytes sdkBytes = response.metricWidgetImage();
               byte[] bytes = sdkBytes.asByteArray();
               return CompletableFuture.runAsync(() -> {
                   try {
                       File outputFile = new File(fileName);
                       try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
```

```
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to
file", e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly
detectors have been described
     * @throws RuntimeException if there is a failure during the operation, such
as when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName)
{
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });
```

```
        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response
-> {
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly
detectors", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
```

```
    * @return a {@link CompletableFuture} that completes when the anomaly
detector has been added
    */
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root
node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .stat("Maximum")
                   .build();

               PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                   .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                   .build();

               return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response
-> {
                   logger.info("Added anomaly detector for metric {}",
customMetricName);
               });
           } catch (Exception e) {
```

```
                throw new RuntimeException("Failed to create anomaly detector",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
```

```
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response ->
response.alarmHistoryItems().forEach(item -> {
                        logger.info("History summary: {}",
item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
```

```
    * Checks for a metric alarm in AWS CloudWatch.
    *
    * @param fileName the name of the file containing the JSON configuration for
the custom metric
    * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
    */
   public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return checkForAlarmAsync(metricRequest, customMetricName, 10);

           } catch (IOException e) {
               throw new RuntimeException("Failed to parse JSON content", e);
           }
       }).whenComplete((result, exception) -> {
           if (exception != null) {
```

```
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified
metric.
     *
     * @param metricRequest    the request to describe the alarms for the
specified metric
     * @param customMetricName the name of the custom metric to check for an
alarm
     * @param retries          the number of retries to perform if no alarm is
found
     * @return a {@link CompletableFuture} that completes when an alarm is found
or the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }

        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response ->
{
            if (response.hasMetricAlarms()) {
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
```

```
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }


    /**
     * Adds metric data for an alarm asynchronously.
     *
     * @param fileName the name of the JSON file containing the metric data
     * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
     */
    public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for
further processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
```

```
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed
to parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " +
exception.getMessage(), exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric
information
    * @return a {@link CompletableFuture} that completes when the metric data
has been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               // Parse the JSON string to extract relevant values.
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               // Set the current time and date range for metric query.
               Instant nowDate = Instant.now();
               long hours = 1;
               long minutes = 30;
               Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

               Metric met = Metric.builder()
                   .metricName(customMetricName)
```

```
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}",
item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ",
alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}",
ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm
configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
```

```
    * @throws RuntimeException if an exception occurs while reading the JSON
file or creating the alarm
    */
   public CompletableFuture<String> createAlarmAsync(String fileName) {
       com.fasterxml.jackson.databind.JsonNode rootNode;
       try {
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           rootNode = new ObjectMapper().readTree(parser);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the alarm configuration
file", e);
       }

       // Extract values from the JSON node.
       String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
       String customMetricName =
rootNode.findValue("customMetricName").asText();
       String alarmName = rootNode.findValue("exampleAlarmName").asText();
       String emailTopic = rootNode.findValue("emailTopic").asText();
       String accountId = rootNode.findValue("accountId").asText();
       String region = rootNode.findValue("region").asText();

       // Create a List for alarm actions.
       List<String> alarmActions = new ArrayList<>();
       alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

       PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
           .alarmActions(alarmActions)
           .alarmDescription("Example metric alarm")
           .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
           .threshold(100.00)
           .metricName(customMetricName)
           .namespace(customMetricNamespace)
           .evaluationPeriods(1)
           .period(10)
           .statistic("Maximum")
           .datapointsToAlarm(1)
           .treatMissingData("ignore")
           .build();
```

```
        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
     * @return a {@link CompletableFuture} representing the asynchronous
operation, which will complete with a
     * {@link PutDashboardResponse} when the dashboard is successfully updated
     */
    public CompletableFuture<PutDashboardResponse>
addMetricToDashboardAsync(String fileName, String dashboardName) {
        String dashboardBody;
        try {
            dashboardBody = readFileAsString(fileName);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the dashboard file", e);
        }

        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to update dashboard: {}",
ex.getMessage());
                    throw new RuntimeException("Error updating dashboard", ex);
                } else {
```

```
                    logger.info("{} was successfully updated.", dashboardName);
                    return response;
                }
            });
    }

    /**
     * Creates a new custom metric.
     *
     * @param dataPoint the data point to be added to the custom metric
     * @return a {@link CompletableFuture} representing the asynchronous
operation of adding the custom metric
     */
    public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
```

```
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing
the dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
     * @param dashboardName the name of the dashboard to be created
     * @param fileName      the name of the file containing the dashboard body
     * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the dashboard
     * @throws IOException if there is an error reading the dashboard body from
the file
     */
```

```
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}",
ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new
Dashboard.");
                   } else {
                       for (DashboardValidationMessage message : messages) {
                           logger.info("Message: {}", message.message());
                       }
                   }
                   return response; // Return the response for further use
               }
           });
   }


   /**
    * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
    *
    * @param costDateWeek the start date for the metric statistics, in the
format of an ISO-8601 date string (e.g., "2023-04-05")
    * @return a {@link CompletableFuture} that, when completed, contains the
{@link GetMetricStatisticsResponse} with the retrieved metric statistics
    * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
```

```
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();

        // Define dimension
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400) // One day period
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics:
" + exception.getMessage(), exception);
                }
            });
    }
```

```
   /**
    * Retrieves and displays metric statistics for the specified parameters.
    *
    * @param nameSpace    the namespace for the metric
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics,
in the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric
statistics have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

           String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
```

```
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric
statistics: " + exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
```

```
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or
dimensions are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
```

```
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception)
 -> {
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
 exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
 dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
 dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData

- GetMetricStatistics
- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem CloudWatch Funktionen demonstriert werden.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
```

```
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
```

```
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
 CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select
 one from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
```

```
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension)
    println(DASHES)
```

```
    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("13. Check the alarm state using the action
 DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
```

```
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
```

```
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
```

```
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10
 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
```

```
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
```

```
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
```

```
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
```

```
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
```

```
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
```

```
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
```

```
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
```

```
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Aktionen zur CloudWatch Verwendung AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CloudWatch Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die CloudWatch API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung CloudWatch AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon CloudWatch API-Referenz.

Beispiele

- Verwenden Sie DeleteAlarmsmit einem AWS SDK oder CLI
- Verwenden Sie DeleteAnomalyDetectormit einem AWS SDK oder CLI
- Verwenden Sie DeleteDashboardsmit einem AWS SDK oder CLI
- Verwenden Sie mit einem DescribeAlarmHistory AWS SDK oder CLI
- Verwenden Sie DescribeAlarmsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeAlarmsForMetricmit einem AWS SDK oder CLI
- Verwenden Sie DescribeAnomalyDetectorsmit einem AWS SDK oder CLI
- Verwenden Sie DisableAlarmActionsmit einem AWS SDK oder CLI
- Verwenden Sie EnableAlarmActionsmit einem AWS SDK oder CLI
- Verwenden Sie GetDashboardmit einem AWS SDK oder CLI
- Verwenden Sie mit einem GetMetricData AWS SDK oder CLI
- Verwenden Sie GetMetricStatisticsmit einem AWS SDK oder CLI
- Verwenden Sie GetMetricWidgetImagemit einem AWS SDK oder CLI
- Verwenden Sie ListDashboardsmit einem AWS SDK oder CLI
- Verwenden Sie mit einem ListMetrics AWS SDK oder CLI
- Verwenden Sie PutAnomalyDetectormit einem AWS SDK oder CLI
- Verwenden Sie PutDashboardmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutMetricAlarm AWS SDK oder CLI

- Verwenden Sie PutMetricDatamit einem AWS SDK oder CLI

Verwenden Sie **`DeleteAlarms`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAlarms` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Alarmen
- Metriken und Alarme verwalten
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Löschen Sie den Alarm.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie einen Alarm

Im folgenden Beispiel wird der `delete-alarms` Befehl verwendet, um den CloudWatch Amazon-Alarm mit dem Namen „myalarm“ zu löschen:

```
aws cloudwatch delete-alarms --alarm-names myalarm
```

Ausgabe:

```
This command returns to the prompt if successful.
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
```

```
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
 alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
 alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
```

```
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};
```

```
cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
 metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
 metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteAlarmsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_cwt->deletealarms(
      it_alarmnames = it_alarm_names ).
    MESSAGE 'Alarms deleted.' TYPE 'I'.
  CATCH /aws1/cx_cwtresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteAnomalyDetector`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAnomalyDetector` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteAnomalyDetectorin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie ein bestimmtes Anomalieerkennungsmodell

Im folgenden Beispiel für `delete-anomaly-detector` wird ein Anomalieerkennungsmodell im angegebenen Konto gelöscht.

```
aws cloudwatch delete-anomaly-detector \
    --namespace AWS/Logs \
    --metric-name IncomingBytes \
    --stat SampleCount
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines Modells zur Erkennung von Anomalien im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAnomalyDetectorin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an Anomaly Detector.
    *
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of
the Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root
node
           } catch (IOException e) {
```

```
                throw new RuntimeException("Failed to read or parse the file",
e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly
Detector", exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- Einzelheiten zur API finden Sie DeleteAnomalyDetectorin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAnomalyDetectorin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteDashboards`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDashboards` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteDashboardsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie bestimmte Dashboards

Im folgenden Beispiel für `delete-dashboards` werden zwei Dashboards mit den Namen `Dashboard-A` und `Dashboard-B` im angegebenen Konto gelöscht.

```
aws cloudwatch delete-dashboards \
    --dashboard-names Dashboard-A Dashboard-B
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter CloudWatch Amazon-Dashboards im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteDashboardsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse>
deleteDashboardAsync(String dashboardName) {
```

```
        DeleteDashboardsRequest dashboardsRequest =
 DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: "
 + dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

- Einzelheiten zur API finden Sie DeleteDashboardsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteDashboardsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Löscht das angegebene Dashboard und bittet um Bestätigung, bevor der Vorgang fortgesetzt wird. Um die Bestätigung zu umgehen, fügen Sie dem Befehl den Schalter -Force hinzu.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter DeleteDashboards AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht das angegebene Dashboard und bittet um Bestätigung, bevor der Vorgang fortgesetzt wird. Um die Bestätigung zu umgehen, fügen Sie dem Befehl den Schalter -Force hinzu.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter DeleteDashboards AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeAlarmHistory`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAlarmHistory` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmHistoryin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den Verlauf eines Alarms ab

Im folgenden Beispiel wird der `describe-alarm-history` Befehl verwendet, um den Verlauf für den CloudWatch Amazon-Alarm mit dem Namen „myalarm“ abzurufen:

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-
type StateUpdate
```

Ausgabe:

```
{
    "AlarmHistoryItems": [
        {
            "Timestamp": "2014-04-09T18:59:06.442Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"ALARM\",\"stateReason\":\"testing purposes\"},\"newState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were not
 greater than the threshold (70.0). The most recent datapoints: [38.958,
 40.292].\",\"stateReasonData\":{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}}}",
            "HistorySummary": "Alarm updated from ALARM to OK"
        },
        {
            "Timestamp": "2014-04-09T18:59:05.805Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
 not greater than the threshold (70.0). The most recent datapoints:
 [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
\"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
\"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
\"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState
\":{\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
            "HistorySummary": "Alarm updated from OK to ALARM"
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAlarmHistoryin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the alarm history for a given alarm name and date range.
    *
    * @param fileName the path to the JSON file containing the alarm name
    * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
    * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
    */
   public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });
```

```
        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response ->
response.alarmHistoryItems().forEach(item -> {
                        logger.info("History summary: {}",
item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
```

```
        });
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmHistoryin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAlarmHistoryin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeAlarms`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAlarms` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Alarmen
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Informationen über einen Alarm auf

Im folgenden Beispiel wird der `describe-alarms`-Befehl verwendet, um Informationen über den Alarm mit dem Namen „myalarm“ bereitzustellen:

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

Ausgabe:

```
{
    "MetricAlarms": [
```

```
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:myalarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myalarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ",
alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}",
ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
```

```
                }
            });
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt alle Alarme zurück, einschließlich zusammengesetzter und metrischer Alarme von CloudWatch.

```
Get-CWAlarm -MaxRecords 1
```

Ausgabe:

```
CompositeAlarms MetricAlarms         NextToken
--------------- ------------         ---------
                {MetricAlarms-01}    NextToken-01
                {MetricAlarms-02}    NextToken-02
                {MetricAlarms-03}    NextToken-03
```

Beispiel 2: Gibt nur die zusammengesetzten Alarmdaten von CloudWatch nach dem Einstellen des AlarmType Parameters auf zurück CompositeAlarms.

```
Get-CWAlarm -AlarmType 'CompositeAlarms'
```

Ausgabe:

```
CompositeAlarms        MetricAlarms NextToken
---------------        ------------ ---------
{CompositeAlarms-01}
{CompositeAlarms-02}
{CompositeAlarms-03}
```

- Einzelheiten zur API finden Sie unter DescribeAlarms AWS -Tools für PowerShellCmdlet-Referenz (V4).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwt->describealarms(                 " oo_result is
 returned for testing purposes. "
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAlarmsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeAlarmsForMetric`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAlarmsForMetric` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Beschreiben Sie die Alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
```

```
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So zeigen Sie Informationen über Alarme an, die einer Metrik zugeordnet sind

Im folgenden Beispiel wird der `describe-alarms-for-metric`-Befehl verwendet, um Informationen über alle Alarme anzuzeigen, die der Amazon-EC2-Metrik CPUUtilization und der Instance mit der ID i-0c986c72 zugeordnet sind:

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --
namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

Ausgabe:

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 10,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
            "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
            "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:NotifyMe"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2013-10-30T03:03:51.479+0000\",\"startDate\":\"2013-10-30T02:08:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":
[40.698,39.612,42.432,39.796,38.816,42.28,42.854,40.088,40.760000000000005,41.316],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 10 datapoints were not
 greater than or equal to the threshold (70.0). The most recent datapoints:
 [40.760000000000005, 41.316].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        },
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
```

```
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": false,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Checks for a metric alarm in AWS CloudWatch.
    *
    * @param fileName the name of the file containing the JSON configuration for
the custom metric
    * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
    */
   public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified
metric.
     *
     * @param metricRequest    the request to describe the alarms for the
specified metric
     * @param customMetricName the name of the custom metric to check for an
alarm
     * @param retries          the number of retries to perform if no alarm is
found
     * @return a {@link CompletableFuture} that completes when an alarm is found
or the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }
```

```
        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response ->
{
            if (response.hasMetricAlarms()) {
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });
```

```
cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
```

```
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10
 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
```

```
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
```

```
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts "Name:           #{alarm.alarm_name}"
      puts "State value:    #{alarm.state_value}"
      puts "State reason:   #{alarm.state_reason}"
      puts "Metric:         #{alarm.metric_name}"
      puts "Namespace:      #{alarm.namespace}"
      puts "Statistic:      #{alarm.statistic}"
      puts "Period:         #{alarm.period}"
      puts "Unit:           #{alarm.unit}"
      puts "Eval. periods:  #{alarm.evaluation_periods}"
      puts "Threshold:      #{alarm.threshold}"
      puts "Comp. operator: #{alarm.comparison_operator}"

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts 'Alarm actions:'
        alarm.alarm_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
         alarm.insufficient_data_actions.count.positive?
        puts 'Insufficient data actions:'
        alarm.insufficient_data_actions.each do |a|
          puts "  #{a}"
        end
      end

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: #{d.name}, Value: #{d.value}"
        end
      else
        puts '  None for this alarm.'
```

```
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ''

  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwt->describealarmsformetric(          " oo_result is
 returned for testing purposes. "
          iv_namespace   = iv_namespace
          iv_metricname  = iv_metric_name
          it_dimensions  = it_dimensions
          iv_statistic   = iv_statistic
          iv_period      = iv_period
          iv_unit        = iv_unit ).
        MESSAGE 'Alarms for metric retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeAnomalyDetectors`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAnomalyDetectors` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- Einzelheiten zur API finden Sie DescribeAnomalyDetectorsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste von Modellen zur Erkennung von Anomalien ab

Im folgenden Beispiel für `describe-anomaly-detectors` werden Informationen zu Anomalieerkennungsmodellen angezeigt, die dem `AWS/Logs`-Namespace im angegebenen Konto zugeordnet sind.

```
aws cloudwatch describe-anomaly-detectors \
    --namespace AWS/Logs
```

Ausgabe:

```
{
    "AnomalyDetectors": [
        {
            "Namespace": "AWS/Logs",
            "MetricName": "IncomingBytes",
            "Dimensions": [],
            "Stat": "SampleCount",
            "Configuration": {
                "ExcludedTimeRanges": []
            },
            "StateValue": "TRAINED",
            "SingleMetricAnomalyDetector": {
                "AccountId": "123456789012",
                "Namespace": "AWS/Logs",
                "MetricName": "IncomingBytes",
                "Dimensions": [],
                "Stat": "SampleCount"
            }
        },
        {
            "Namespace": "AWS/Logs",
            "MetricName": "IncomingBytes",
            "Dimensions": [
                {
                    "Name": "LogGroupName",
                    "Value": "demo"
                }
            ],
```

```
            "Stat": "Average",
            "Configuration": {
                "ExcludedTimeRanges": []
            },
            "StateValue": "PENDING_TRAINING",
            "SingleMetricAnomalyDetector": {
                "AccountId": "123456789012",
                "Namespace": "AWS/Logs",
                "MetricName": "IncomingBytes",
                "Dimensions": [
                    {
                        "Name": "LogGroupName",
                        "Value": "demo"
                    }
                ],
                "Stat": "Average"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden der CloudWatch Anomalieerkennung im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAnomalyDetectorsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
```

```
    * @return a {@link CompletableFuture} that completes when the anomaly
detectors have been described
    * @throws RuntimeException if there is a failure during the operation, such
as when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName)
{
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response
-> {
                   List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                   for (AnomalyDetector detector : anomalyDetectorList) {
                       logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                       logger.info("State: {} ", detector.stateValue());
                   }
```

```
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly
 detectors", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
 exception);
            }
        });
    }
```

- Einzelheiten zur API finden Sie DescribeAnomalyDetectorsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAnomalyDetectorsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DisableAlarmActions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableAlarmActions` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Alarmen
- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
```

```
    {
        var disableAlarmActionsResult = await
 _amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Deaktivieren Sie die Alarmaktionen.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);
```

```
        auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So deaktivieren Sie Aktionen für einen Alarm

Das folgende Beispiel verwendet den `disable-alarm-actions`-Befehl, um alle Aktionen für den Alarm mit dem Namen „myalarm“ zu deaktivieren:

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

Wenn dieser Befehl erfolgreich war, kehrt er zum Prompt zurück.

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
```

```
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });
```

```
  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
```

```
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
 they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
```

```
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
```

```
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableAlarmActions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableAlarmActions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
```

```
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Aktivieren Sie die Alarmaktionen.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
```

```
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);

    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktivieren Sie alle Aktionen für einen Alarm

Das folgende Beispiel verwendet den `enable-alarm-actions`-Befehl, um alle Aktionen für den Alarm mit dem Namen „myalarm“ zu aktivieren:

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

Wenn dieser Befehl erfolgreich war, kehrt er zur Eingabeaufforderung zurück.

- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request =
 EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
 they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
```

```
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetDashboard`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDashboard` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- Einzelheiten zur API finden Sie GetDashboardin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen zu einem Dashboard ab

Im folgenden Beispiel für `get-dashboard` werden Informationen zum Dashboard `Dashboard-A` im angegebenen Konto angezeigt.

```
aws cloudwatch get-dashboard \
    --dashboard-name Dashboard-A
```

Ausgabe:

```
{
    "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/Dashboard-A",
    "DashboardBody": "{\"widgets\":[{\"type\":\"metric\",\"x\":0,\"y\":0,\"width
\":6,\"height\":6,\"properties\":{\"view\":\"timeSeries\",\"stacked\":false,
\"metrics\":[[\"AWS/EC2\",\"NetworkIn\",\"InstanceId\",\"i-0131f062232ade043\"],
[\".\",\"NetworkOut\",\".\",\".\"]],\"region\":\"us-east-1\"}}]}",
    "DashboardName": "Dashboard-A"
}
```

Weitere Informationen finden Sie unter CloudWatch Amazon-Dashboards im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetDashboardin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt den ARN und den Inhalt des angegebenen Dashboards zurück.

```
Get-CWDashboard -DashboardName Dashboard1
```

Ausgabe:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Einzelheiten zur API finden Sie unter GetDashboard AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt den ARN und den Inhalt des angegebenen Dashboards zurück.

```
Get-CWDashboard -DashboardName Dashboard1
```

Ausgabe:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- Einzelheiten zur API finden Sie unter GetDashboard AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetMetricData`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetMetricData` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
```

```
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }
```

- Einzelheiten zur API finden Sie GetMetricDatain der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So rufen Sie die durchschnittlichen Gesamt-IOPS für die angegebene EC2-Instance mithilfe eines mathematischen Ausdrucks ab

Im folgenden `get-metric-data` Beispiel werden CloudWatch Metrikwerte für die EC2-Instance mit InstanceID abgerufen, wobei ein metrischer mathematischer Ausdruck `i-abcdef` verwendet wird, der Metriken kombiniert. `EBSReadOps` `EBSWriteOps`

```
aws cloudwatch get-metric-data \
    --metric-data-queries file://file.json \
    --start-time 2024-09-29T22:10:00Z \
    --end-time 2024-09-29T22:15:00Z
```

Inhalt von `file.json`:

```
[
    {
```

```
        "Id": "m3",
        "Expression": "(m1+m2)/300",
        "Label": "Avg Total IOPS"
    },
    {
        "Id": "m1",
        "MetricStat": {
            "Metric": {
                "Namespace": "AWS/EC2",
                "MetricName": "EBSReadOps",
                "Dimensions": [
                    {
                        "Name": "InstanceId",
                        "Value": "i-abcdef"
                    }
                ]
            },
            "Period": 300,
            "Stat": "Sum",
            "Unit": "Count"
        },
        "ReturnData": false
    },
    {
        "Id": "m2",
        "MetricStat": {
            "Metric": {
                "Namespace": "AWS/EC2",
                "MetricName": "EBSWriteOps",
                "Dimensions": [
                    {
                        "Name": "InstanceId",
                        "Value": "i-abcdef"
                    }
                ]
            },
            "Period": 300,
            "Stat": "Sum",
            "Unit": "Count"
        },
        "ReturnData": false
    }
]
```

Ausgabe:

```
{
    "MetricDataResults": [
        {
            "Id": "m3",
            "Label": "Avg Total IOPS",
            "Timestamps": [
                "2024-09-29T22:10:00+00:00"
            ],
            "Values": [
                96.85
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Beispiel 2: Um die geschätzten Gebühren anhand von Abrechnungskennzahlen zu überwachen AWS CloudWatch

Im folgenden `get-metric-data` Beispiel wird die `EstimatedCharges` CloudWatch Metrik aus dem Namespace AWS/Billing abgerufen.

```
aws cloudwatch get-metric-data \
    --metric-data-queries '[{"Id":"m1","MetricStat":{"Metric":
{"Namespace":"AWS/Billing","MetricName":"EstimatedCharges","Dimensions":
[{"Name":"Currency","Value":"USD"}]},"Period":21600,"Stat":"Maximum"}}]' \
    --start-time 2024-09-26T12:00:00Z \
    --end-time 2024-09-26T18:00:00Z \
    --region us-east-1
```

Ausgabe:

```
{
    "MetricDataResults": [
        {
            "Id": "m1",
            "Label": "EstimatedCharges",
            "Timestamps": [
                "2024-09-26T12:00:00+00:00"
```

```
            ],
            "Values": [
                542.38
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Weitere Informationen finden Sie unter Verwenden von mathematischen Ausdrücken mit CloudWatch Metriken im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetMetricDatain der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric
information
    * @return a {@link CompletableFuture} that completes when the metric data
has been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
```

```
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for
further processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
```

```
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}",
item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- Einzelheiten zur API finden Sie GetMetricDatain der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
```

```
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetMetricDatain der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetMetricStatistics`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetMetricStatistics` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
```

```
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }
```

- Einzelheiten zur API finden Sie GetMetricStatisticsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die CPU-Auslastung pro EC2-Instance ab

Im folgenden Beispiel wird der `get-metric-statistics`-Befehl verwendet, um die CPU-Auslastung für eine EC2-Instance mit der ID i-abcdef abzurufen.

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-
time 2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --
namespace AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-
abcdef
```

Ausgabe:

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-04-09T11:18:00Z",
            "Maximum": 44.79,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T20:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T19:18:00Z",
```

```
            "Maximum": 50.85,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T09:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T03:18:00Z",
            "Maximum": 76.84,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T21:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T14:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T08:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T16:18:00Z",
            "Maximum": 45.55,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T06:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T13:18:00Z",
            "Maximum": 45.08,
            "Unit": "Percent"
        },
        {
```

```
            "Timestamp": "2014-04-09T05:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T18:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T17:18:00Z",
            "Maximum": 52.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T07:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T02:18:00Z",
            "Maximum": 51.23,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T12:18:00Z",
            "Maximum": 47.67,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-08T23:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T10:18:00Z",
            "Maximum": 51.91,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T04:18:00Z",
            "Maximum": 47.13,
            "Unit": "Percent"
        },
```

```
        {
            "Timestamp": "2014-04-09T15:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T00:18:00Z",
            "Maximum": 48.16,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T01:18:00Z",
            "Maximum": 49.18,
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}
```

Angeben mehrerer Dimensionen

Das folgende Beispiel zeigt, wie mehrere Dimensionen angegeben werden können. Jede Dimension wird als Name/Value Paar angegeben, wobei zwischen dem Namen und dem Wert ein Komma steht. Mehrere Dimensionen werden durch ein Leerzeichen getrennt. Wenn eine einzelne Metrik mehrere Dimensionen enthält, müssen Sie für jede definierte Dimension einen Wert angeben.

Weitere Beispiele für die Verwendung des `get-metric-statistics` Befehls finden Sie unter Get Statistics for a Metric im Amazon CloudWatch Developer Guide.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --
namespace MyNameSpace --dimensions Name=InstanceID,Value=i-
abcdef Name=InstanceType,Value=m1.small --start-time 2016-10-15T04:00:00Z --end-
time 2016-10-19T07:00:00Z --statistics Average --period 60
```

- Einzelheiten zur API finden Sie GetMetricStatisticsunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves and displays metric statistics for the specified parameters.
    *
    * @param nameSpace    the namespace for the metric
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics,
in the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric
statistics have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

           String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();
```

```
        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric
statistics: " + exception.getMessage(), exception);
            });
    }
```

- Einzelheiten zur API finden Sie GetMetricStatisticsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
```

```
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie GetMetricStatisticsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
 stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
 grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
 period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
 must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
 value
                           or maximum value.
        :return: The retrieved statistics for the metric.
```

```
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats
```

- Einzelheiten zur API finden Sie GetMetricStatisticsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_cwt->getmetricstatistics(              " oo_result is
returned for testing purposes. "
         iv_namespace    = iv_namespace
         iv_metricname   = iv_metric_name
         iv_starttime    = iv_start_time
         iv_endtime      = iv_end_time
```

```
          iv_period      = iv_period
          it_statistics  = it_statistics ).
        MESSAGE 'Metric statistics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetMetricStatisticsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetMetricWidgetImage`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetMetricWidgetImage` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
```

```
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- Einzelheiten zur API finden Sie GetMetricWidgetImagein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie ein Snapshot-Diagramm der CPU-Auslastung ab

Im folgenden Beispiel für `get-metric-widget-image` wird das Snapshot-Diagramm für die Metrik `CPUUtilization` der EC2-Instance mit der ID `i-abcde` abgerufen und das abgerufene Bild als Datei mit dem Namen „image.png“ auf Ihrem lokalen Computer gespeichert.

```
aws cloudwatch get-metric-widget-image \
    --metric-widget '{"metrics":[["AWS/EC2","CPUUtilization","InstanceId","i-abcde"]]}' \
    --output-format png \
    --output text | base64 --decode > image.png
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie GetMetricWidgetImagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves and saves a custom metric image to a file.
     *
     * @param fileName the name of the file to save the metric image to
     * @return a {@link CompletableFuture} that completes when the image has been saved to the file
     */
    public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName) {
```

```
        logger.info("Getting Image data for custom metric.");
        String myJSON = """
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
                  "stacked ": false,
                  "period": 10,
                  "width": 1400,
                  "height": 600,
                  "metrics": [
                      [
                      "AWS/Billing",
                      "EstimatedCharges",
                      "Currency",
                      "USD"
                     ]
                  ]
              }
            """;

        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to
file", e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Error getting and saving metric
 image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }
```

- Einzelheiten zur API finden Sie GetMetricWidgetImagein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""
```

```
    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- Einzelheiten zur API finden Sie GetMetricWidgetImagein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListDashboards`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDashboards` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
```

```
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }
```

- Einzelheiten zur API finden Sie ListDashboardsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste von Dashboards ab

Im folgenden Beispiel für `list-dashboards` werden alle Dashboards im angegebenen Konto aufgelistet.

```
aws cloudwatch list-dashboards
```

Ausgabe:

```
{
    "DashboardEntries": [
        {
            "DashboardName": "Dashboard-A",
            "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/
Dashboard-A",
            "LastModified": "2024-10-11T18:40:11+00:00",
            "Size": 271
        },
        {
            "DashboardName": "Dashboard-B",
```

```
            "DashboardArn": "arn:aws:cloudwatch::123456789012:dashboard/
Dashboard-B",
            "LastModified": "2024-10-11T18:44:41+00:00",
            "Size": 522
        }
    ]
}
```

Weitere Informationen finden Sie unter CloudWatch Amazon-Dashboards im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListDashboardsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists the available dashboards.
    *
    * @return a {@link CompletableFuture} that completes when the operation is
finished.
    * The future will complete exceptionally if an error occurs while listing
the dashboards.
    */
   public CompletableFuture<Void> listDashboardsAsync() {
       ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
       ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
       return paginator.subscribe(response -> {
           response.dashboardEntries().forEach(entry -> {
               logger.info("Dashboard name is: {} ", entry.dashboardName());
               logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
           });
       }).exceptionally(ex -> {
```

```
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
 ex);
        });
    }
```

- Einzelheiten zur API finden Sie ListDashboardsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListDashboardsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt die Sammlung von Dashboards für Ihr Konto zurück.

```
Get-CWDashboardList
```

Ausgabe:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Beispiel 2: Gibt die Sammlung von Dashboards für Ihr Konto zurück, deren Namen mit dem Präfix „dev“ beginnen.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Einzelheiten zur API finden Sie unter ListDashboards AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Sammlung von Dashboards für Ihr Konto zurück.

```
Get-CWDashboardList
```

Ausgabe:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Beispiel 2: Gibt die Sammlung von Dashboards für Ihr Konto zurück, deren Namen mit dem Präfix „dev“ beginnen.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- Einzelheiten zur API finden Sie unter ListDashboards AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListMetrics`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListMetrics` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }
```

```
        return results;
    }
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Listen Sie die Metriken auf.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }
```

```
        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
```

```
        }
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie die Metriken für Amazon SNS auf

Im folgenden Beispiel für `list-metrics` werden die Metriken für Amazon SNS angezeigt.

```
aws cloudwatch list-metrics \
    --namespace "AWS/SNS"
```

Ausgabe:

```
{
    "Metrics": [
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
```

```
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
```

```
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an
{@link ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();
```

```
        ListMetricsPublisher metricsPaginator =
 getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
 exception.getMessage(), exception);
            });
    }
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  CloudWatchServiceException,
  ListMetricsCommand,
} from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = async () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
```

```
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });

  try {
    const response = await client.send(command);
    console.log(`Metrics count: ${response.Metrics?.length}`);
    return response;
  } catch (caught) {
    if (caught instanceof CloudWatchServiceException) {
      console.error(`Error from CloudWatch. ${caught.name}: ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
```

```
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- Einzelheiten zur API finden Sie ListMetricsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
```

```
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- Einzelheiten zur API finden Sie ListMetricsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
```

```
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end

# Example usage:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
    'example.com',
    5_885.0,
    'Count'
  )
```

```
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "The following list-metrics example displays the metrics for Amazon
 CloudWatch."
```

```
    TRY.
        oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
          iv_namespace = iv_namespace ).
        DATA(lt_metrics) = oo_result->get_metrics( ).
        MESSAGE 'Metrics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListMetricsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutAnomalyDetector`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutAnomalyDetector` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
```

```
    {
        var putAlarmDetectorResult = await
 _amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie PutAnomalyDetectorin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Anomalieerkennungsmodell

Im folgenden `put-anomaly-detector` Beispiel wird ein Modell zur Erkennung von Anomalien für eine CloudWatch Metrik erstellt.

```
aws cloudwatch put-anomaly-detector \
    --namespace AWS/Logs \
    --metric-name IncomingBytes \
    --stat SampleCount
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden der CloudWatch Anomalieerkennung im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutAnomalyDetectorin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Adds an anomaly detector for the given file.
    *
    * @param fileName the name of the file containing the anomaly detector
configuration
    * @return a {@link CompletableFuture} that completes when the anomaly
detector has been added
    */
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root
node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file",
e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
```

```
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response
-> {
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- Einzelheiten zur API finden Sie PutAnomalyDetectorin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}
```

- Einzelheiten zur API finden Sie PutAnomalyDetectorin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutDashboard`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutDashboard` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
```

```
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }
```

- Einzelheiten zur API finden Sie PutDashboardin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Dashboard

Im folgenden Beispiel für `put-dashboard` wird ein Dashboard mit dem Namen `Dashboard-A` im angegebenen Konto erstellt.

```
aws cloudwatch put-dashboard \
    --dashboard-name Dashboard-A \
    --dashboard-body '{"widgets":
[{"height":6,"width":6,"y":0,"x":0,"type":"metric","properties":
{"view":"timeSeries","stacked":false,"metrics":
[["Namespace","CPUUtilization","Environment","Prod","Type","App"]],"region":"us-
east-1"}}]}'
```

Ausgabe:

```
{
    "DashboardValidationMessages": []
}
```

Weitere Informationen finden Sie unter Erstellen eines CloudWatch Dashboards im CloudWatch Amazon-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutDashboardin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from
the file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}",
ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
```

```
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new
 Dashboard.");
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }
```

- Einzelheiten zur API finden Sie PutDashboardin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
```

```
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie PutDashboardin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Erstellt oder aktualisiert das Dashboard mit dem Namen „Dashboard1“, sodass es zwei Metrik-Widgets nebeneinander enthält.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
```

```
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
                      "BucketName",
                      "amzn-s3-demo-bucket"
                   ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Beispiel 2: Erstellt oder aktualisiert das Dashboard und leitet den Inhalt, der das Dashboard beschreibt, an das Cmdlet weiter.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter PutDashboard AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt oder aktualisiert das Dashboard mit dem Namen „Dashboard1“, sodass es zwei Metrik-Widgets nebeneinander enthält.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
                      "BucketName",
                      "amzn-s3-demo-bucket"
                   ]
                ],
```

```
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Beispiel 2: Erstellt oder aktualisiert das Dashboard und leitet den Inhalt, der das Dashboard beschreibt, an das Cmdlet weiter.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- Einzelheiten zur API finden Sie unter PutDashboard AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutMetricAlarm`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutMetricAlarm` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Alarmen
- Metriken und Alarme verwalten
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Add a metric alarm to send an email when the metric passes a threshold.
   /// </summary>
   /// <param name="alarmDescription">A description of the alarm.</param>
   /// <param name="alarmName">The name for the alarm.</param>
   /// <param name="comparison">The type of comparison to use.</param>
   /// <param name="metricName">The name of the metric for the alarm.</param>
   /// <param name="metricNamespace">The namespace of the metric.</param>
   /// <param name="threshold">The threshold value for the alarm.</param>
   /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
       string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
   {
       try
       {
           var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
               new PutMetricAlarmRequest()
               {
                   AlarmActions = alarmActions,
                   AlarmDescription = alarmDescription,
                   AlarmName = alarmName,
                   ComparisonOperator = comparison,
                   Threshold = threshold,
                   Namespace = metricNamespace,
                   MetricName = metricName,
                   EvaluationPeriods = 1,
                   Period = 10,
```

```
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Erstellen Sie den Alarm, um die Metrik zu beobachten.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
```

```
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So senden Sie eine E-Mail-Nachricht von Amazon Simple Notification Service, wenn die CPU-Auslastung 70 % übersteigt

Im folgenden Beispiel wird der `put-metric-alarm`-Befehl verwendet, um eine E-Mail-Nachricht von Amazon Simple Notification Service zu senden, wenn die CPU-Auslastung 70 % übersteigt:

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
EC2 --statistic Average --period 300 --threshold 70 --comparison-
operator GreaterThanThreshold  --dimensions "Name=InstanceId,Value=i-12345678" --
evaluation-periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic
 --unit Percent
```

Wenn dieser Befehl erfolgreich war, kehrt er zur Eingabeaufforderung zurück. Wenn ein Alarm mit demselben Namen bereits vorhanden ist, wird er durch den neuen Alarm überschrieben.

So geben Sie mehrere Dimensionen an

Das folgende Beispiel zeigt, wie mehrere Dimensionen angegeben werden können. Jede Dimension wird als Name/Value Paar angegeben, wobei zwischen dem Namen und dem Wert ein Komma steht. Mehrere Dimensionen werden durch ein Leerzeichen getrennt:

```
aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS"
 --metric-name Default_Test --statistic Average --period 60 --evaluation-
periods 3 --threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --
dimensions Name=key1,Value=value1 Name=key2,Value=value2
```

- Einzelheiten zur API finden Sie unter PutMetricAlarm AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an alarm based on the configuration provided in a JSON file.
    *
    * @param fileName the name of the JSON file containing the alarm
configuration
    * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
    * @throws RuntimeException if an exception occurs while reading the JSON
file or creating the alarm
    */
   public CompletableFuture<String> createAlarmAsync(String fileName) {
       com.fasterxml.jackson.databind.JsonNode rootNode;
       try {
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           rootNode = new ObjectMapper().readTree(parser);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the alarm configuration
file", e);
       }

       // Extract values from the JSON node.
```

```
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
```

```
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
```

```
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
```

```
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server
 CPU utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
 the
                             alarm threshold before the alarm is set into an
 alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
 statistic.
        :param comparison_op: The comparison operation used to compare the
 threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
```

```
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- Einzelheiten zur API finden Sie PutMetricAlarmin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
```

```
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                 = iv_alarm_name
      iv_comparisonoperator        = iv_comparison_operator
      iv_evaluationperiods         = iv_evaluation_periods
      iv_metricname                = iv_metric_name
      iv_namespace                 = iv_namespace
      iv_statistic                 = iv_statistic
      iv_threshold                 = iv_threshold
      iv_actionsenabled            = iv_actions_enabled
      iv_alarmdescription          = iv_alarm_description
      iv_unit                      = iv_unit
      iv_period                    = iv_period
      it_dimensions                = it_dimensions ).
    MESSAGE 'Alarm created.' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutMetricData`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutMetricData` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen

- Metriken und Alarme verwalten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }
```

```
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
 customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
```

```
#include <iostream>
```

Fügen Sie Daten in eine Metrik ein.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine benutzerdefinierte Metrik auf Amazon zu veröffentlichen CloudWatch

Im folgenden Beispiel wird der `put-metric-data` Befehl verwendet, um eine benutzerdefinierte Metrik auf Amazon zu veröffentlichen CloudWatch:

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://
metric.json
```

Die Werte für die Metrik selbst werden in der JSON-Datei `metric.json` gespeichert.

Hier ist der Inhalt dieser Datei:

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

Weitere Informationen finden Sie unter Veröffentlichen benutzerdefinierter Metriken im Amazon CloudWatch Developer Guide.

So geben Sie mehrere Dimensionen an

Das folgende Beispiel zeigt, wie mehrere Dimensionen angegeben werden können. Jede Dimension wird als Name/Wert-Paar angegeben. Mehrere Dimensionen sind durch ein Komma getrennt:

```
aws cloudwatch put-metric-data --metric-name Buffers --
namespace MyNameSpace --unit Bytes --value 231434333 --
dimensions InstanceID=1-23456789,InstanceType=m1.small
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for
further processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
 CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed
 to parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " +
 exception.getMessage(), exception);
            } else {
                logger.info("Added metric values for metric.");
            }
```

```
        });
    }
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
```

```
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
```

```
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- Einzelheiten zur API finden Sie PutMetricDatain der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Erstellt ein neues MetricDatum Objekt und schreibt es in Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Einzelheiten zur API finden Sie unter PutMetricData AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues MetricDatum Objekt und schreibt es in Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- Einzelheiten zur API finden Sie unter PutMetricData AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
 given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
 name)
            raise
```

Setze einen Datensatz in eine CloudWatch Metrik um.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
```

```
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise
```

- Einzelheiten zur API finden Sie PutMetricDatain AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
```

```
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

"Create metric data object.
DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
  iv_metricname = iv_metric_name
  iv_value      = iv_value
  iv_unit       = iv_unit ).

INSERT lo_metricdatum INTO TABLE lt_metricdata.

TRY.
    lo_cwt->putmetricdata(
      iv_namespace   = iv_namespace
      it_metricdata  = lt_metricdata ).
    MESSAGE 'Metric data added.' TYPE 'I'.
  CATCH /aws1/cx_cwtinvparamvalueex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
ENDTRY.
```

Setze einen Datensatz in eine CloudWatch Metrik um.

```
DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

"Create metric data object with values and counts.
DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
  iv_metricname = iv_metric_name
  iv_timestamp  = iv_timestamp
  iv_unit       = iv_unit
  it_values     = it_values
```

```
      it_counts     = it_counts ).

    INSERT lo_metricdatum INTO TABLE lt_metricdata.

    TRY.
        lo_cwt->putmetricdata(
          iv_namespace   = iv_namespace
          it_metricdata  = lt_metricdata ).
        MESSAGE 'Metric data set added.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutMetricDatain der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung CloudWatch AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien CloudWatch mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb CloudWatch oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Beginnen Sie mit CloudWatch Alarmen mit einem AWS SDK
- Verwalten Sie CloudWatch Metriken und Alarme mit einem AWS SDK
- Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
```

```
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
```

```
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Beginnen Sie mit CloudWatch Alarmen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Alarm.
- Deaktivieren Sie Alarmaktionen.
- Beschreiben Sie einen Alarm.
- Löschen Sie einen Alarm.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
    DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

    "Create an alarm"
    TRY.
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions ).
        MESSAGE 'Alarm created' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.

    "Create an ABAP internal table for the created alarm."
    lo_alarmname = NEW #( iv_value = iv_alarm_name ).
    INSERT lo_alarmname INTO TABLE lt_alarmnames.

    "Disable alarm actions."
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames               = lt_alarmnames ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception-
>av_err_code }" - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
```

```
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound.
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

Verwalten Sie CloudWatch Metriken und Alarme mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Alarm, um eine CloudWatch Metrik zu beobachten.
- Geben Sie Daten in eine Metrik ein und lösen Sie den Alarm aus.
- Rufen Sie Daten aus dem Alarm ab.
- Löschen Sie den Alarm.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die CloudWatch Operationen umschließt.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
```

```
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
```

```
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
```

```
        """
        Sends a single data value to CloudWatch for a metric. This metric is
given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
value
                           or maximum value.
```

```
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
```

```
            metric = self.cloudwatch_resource.Metric(metric_namespace,
 metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

Verwenden Sie die Wrapper-Klasse, um Daten in eine Metrik einzugeben, einen Alarm auszulösen, der die Metrik überwacht, und Daten aus dem Alarm abzurufen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the
 "
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
```

```
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the
threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
```

```
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteAlarms

- DescribeAlarmsForMetric
- DisableAlarmActions
- EnableAlarmActions
- GetMetricStatistics
- ListMetrics
- PutMetricAlarm
- PutMetricData

Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie die Verwendung von DynamoDB durch eine Anwendung zur Leistungsüberwachung konfiguriert wird.

Java

SDK für Java 2.x

Im folgenden Codebeispiel wird gezeigt, wie eine Java-Anwendung zur Leistungsüberwachung von DynamoDB konfiguriert wird. Die Anwendung sendet Metrikdaten, CloudWatch an die Sie die Leistung überwachen können.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch
- DynamoDB

Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Experimentvorlage

- Führen Sie das Experiment durch
- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
```

```
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
```

```
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")
```

```
echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
```

```
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")
```

```
# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
```

```
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"
```

```
# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
```

```
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
```

```
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
```

```
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"
```

```
# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
```

```
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
```

```
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances
  - Wait

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Lambda-Funktionen für die Überwachung erstellen
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
```

```
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi
```

```
    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
```

```
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
```

```
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
```

```
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
```

```
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
```

```
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Codebeispiele für CloudWatch Ereignisse mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon CloudWatch Events mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CloudWatch Benutzerleitfaden für Veranstaltungen — Weitere Informationen zu CloudWatch Veranstaltungen.
- CloudWatch Events-API-Referenz — Details zu allen verfügbaren CloudWatch Event-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für CloudWatch Ereignisse mit AWS SDKs
  - Aktionen für CloudWatch Ereignisse mit AWS SDKs
    - Verwenden Sie PutEventsmit einem AWS SDK
    - Verwenden Sie PutRulemit einem AWS SDK
    - Verwenden Sie PutTargetsmit einem AWS SDK

Grundlegende Beispiele für CloudWatch Ereignisse mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon CloudWatch Events mit AWS SDKs verwenden können.

Beispiele

- Aktionen für CloudWatch Ereignisse mit AWS SDKs
  - Verwenden Sie PutEventsmit einem AWS SDK
  - Verwenden Sie PutRulemit einem AWS SDK
  - Verwenden Sie PutTargetsmit einem AWS SDK

Aktionen für CloudWatch Ereignisse mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne CloudWatch Events-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon CloudWatch Events API-Referenz.

Beispiele

- Verwenden Sie PutEventsmit einem AWS SDK
- Verwenden Sie PutRulemit einem AWS SDK
- Verwenden Sie PutTargetsmit einem AWS SDK

Verwenden Sie **`PutEvents`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutEvents` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import
 software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <resourceArn>

                Where:
                   resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String
 resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();
```

```
            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutEventsCommand({
    // The list of events to send to Amazon CloudWatch Events.
    Entries: [
      {
        // The name of the application or service that is sending the event.
        Source: "my.app",

        // The name of the event that is being sent.
        DetailType: "My Custom Event",

        // The data that is sent with the event.
        Detail: JSON.stringify({ timeOfEvent: new Date().toISOString() }),
```

```
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
```

```
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

cwevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`PutRule`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutRule` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> roleArn>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  roleArn - A role ARN value (for example,
 arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName,
 String roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                    .name(ruleName)
                    .roleArn(roleArn)
                    .scheduleExpression("rate(5 minutes)")
                    .state(RuleState.ENABLED)
```

```
                    .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  // Request parameters for PutRule.
  // https://docs.aws.amazon.com/eventbridge/latest/APIReference/
API_PutRule.html#API_PutRule_RequestParameters
  const command = new PutRuleCommand({
    Name: process.env.CLOUDWATCH_EVENTS_RULE,

    // The event pattern for the rule.
    //  Example: {"source": ["my.app"]}
    EventPattern: process.env.CLOUDWATCH_EVENTS_RULE_PATTERN,
```

```
    // The state of the rule. Valid values: ENABLED, DISABLED
    State: "ENABLED",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutRulein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
```

```
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

cwevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutRulein der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`PutTargets`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutTargets` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
```

```
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> <functionArn> <targetId>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                  targetId - A target id value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }

    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
 String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                    .arn(functionArn)
                    .id(targetId)
                    .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                    .targets(target)
```

```
                    .rule(ruleName)
                    .build();

            cwe.putTargets(request);
            System.out.printf(
                    "Successfully created CloudWatch events target for rule %s",
                    ruleName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutTargetsCommand({
    // The name of the Amazon CloudWatch Events rule.
    Rule: process.env.CLOUDWATCH_EVENTS_RULE,

    // The targets to add to the rule.
    Targets: [
      {
        Arn: process.env.CLOUDWATCH_EVENTS_TARGET_ARN,
```

```
        // The ID of the target. Choose a unique ID for each target.
        Id: process.env.CLOUDWATCH_EVENTS_TARGET_ID,
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });
```

```
var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

cwevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für JavaScript API-Referenz.

Codebeispiele für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon CloudWatch Logs mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- CloudWatch Logs-Benutzerhandbuch — Weitere Informationen zu CloudWatch Logs.
- CloudWatch Logs-API-Referenz — Details zu allen verfügbaren CloudWatch Logs-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für CloudWatch Logs mit AWS SDKs
  - Aktionen für CloudWatch Logs mit AWS SDKs
    - Verwenden Sie AssociateKmsKeymit einem AWS SDK
    - Verwenden Sie CancelExportTaskmit einem AWS SDK
    - Verwenden Sie CreateExportTaskmit einem AWS SDK
    - Verwenden Sie CreateLogGroupmit einem AWS SDK oder CLI
    - Verwenden Sie CreateLogStreammit einem AWS SDK oder CLI
    - Verwenden Sie DeleteLogGroupmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteSubscriptionFiltermit einem AWS SDK
    - Verwenden Sie DescribeExportTasksmit einem AWS SDK
    - Verwenden Sie DescribeLogGroupsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeLogStreamsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeSubscriptionFiltersmit einem AWS SDK
    - Verwenden Sie GetLogEventsmit einem AWS SDK oder CLI
    - Verwenden Sie GetQueryResultsmit einem AWS SDK
    - Verwenden Sie PutSubscriptionFiltermit einem AWS SDK
    - Verwenden Sie StartLiveTailmit einem AWS SDK
    - Verwenden Sie mit StartQueryeinem AWS SDK
- Szenarien für CloudWatch Protokolle mit AWS SDKs
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Erstellen Sie Ihre erste serverlose Funktion
  - Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen
  - Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Grundlegende Beispiele für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon CloudWatch Logs mit AWS SDKs verwenden können.

Beispiele

- Aktionen für CloudWatch Logs mit AWS SDKs
  - Verwenden Sie AssociateKmsKeymit einem AWS SDK
  - Verwenden Sie CancelExportTaskmit einem AWS SDK
  - Verwenden Sie CreateExportTaskmit einem AWS SDK
  - Verwenden Sie CreateLogGroupmit einem AWS SDK oder CLI
  - Verwenden Sie CreateLogStreammit einem AWS SDK oder CLI
  - Verwenden Sie DeleteLogGroupmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteSubscriptionFiltermit einem AWS SDK
  - Verwenden Sie DescribeExportTasksmit einem AWS SDK
  - Verwenden Sie DescribeLogGroupsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeLogStreamsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeSubscriptionFiltersmit einem AWS SDK
  - Verwenden Sie GetLogEventsmit einem AWS SDK oder CLI
  - Verwenden Sie GetQueryResultsmit einem AWS SDK
  - Verwenden Sie PutSubscriptionFiltermit einem AWS SDK
  - Verwenden Sie StartLiveTailmit einem AWS SDK
  - Verwenden Sie mit StartQueryeinem AWS SDK

Aktionen für CloudWatch Logs mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne CloudWatch Logs-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die CloudWatch Logs-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für CloudWatch Protokolle mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon CloudWatch Logs API-Referenz.

Beispiele

- Verwenden Sie AssociateKmsKeymit einem AWS SDK
- Verwenden Sie CancelExportTaskmit einem AWS SDK
- Verwenden Sie CreateExportTaskmit einem AWS SDK
- Verwenden Sie CreateLogGroupmit einem AWS SDK oder CLI
- Verwenden Sie CreateLogStreammit einem AWS SDK oder CLI
- Verwenden Sie DeleteLogGroupmit einem AWS SDK oder CLI
- Verwenden Sie DeleteSubscriptionFiltermit einem AWS SDK
- Verwenden Sie DescribeExportTasksmit einem AWS SDK
- Verwenden Sie DescribeLogGroupsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeLogStreamsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeSubscriptionFiltersmit einem AWS SDK
- Verwenden Sie GetLogEventsmit einem AWS SDK oder CLI
- Verwenden Sie GetQueryResultsmit einem AWS SDK
- Verwenden Sie PutSubscriptionFiltermit einem AWS SDK
- Verwenden Sie StartLiveTailmit einem AWS SDK
- Verwenden Sie mit StartQueryeinem AWS SDK

Verwenden Sie **`AssociateKmsKey`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`AssociateKmsKey`.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to associate an AWS Key Management Service (AWS KMS) key with
    /// an Amazon CloudWatch Logs log group.
    /// </summary>
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };

            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully associated KMS key ID:
 {kmsKeyId} with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
 {kmsKeyId} and {groupName}.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie AssociateKmsKeyin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`CancelExportTask`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CancelExportTask`.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task.
/// </summary>
public class CancelExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskId = "exampleTaskId";

        var request = new CancelExportTaskRequest
        {
            TaskId = taskId,
        };
```

```
            var response = await client.CancelExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{taskId} successfully canceled.");
            }
            else
            {
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CancelExportTaskin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`CreateExportTask`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateExportTask`.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon
S3)
    /// bucket.
```

```
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "amzn-s3-demo-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateExportTaskin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`CreateLogGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLogGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new CreateLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.CreateLogGroupAsync(request);
```

```
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully create log group with ID:
 {logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateLogGroupin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt eine Protokollgruppe mit dem Namen `my-logs`.

```
aws logs create-log-group --log-group-name my-logs
```

- Einzelheiten zur API finden Sie CreateLogGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie CreateLogGroupin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateLogStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLogStream` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
    /// log group.
```

```
    /// </summary>
    public class CreateLogStream
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string logStreamName = "cloudwatchlogs-example-logstream";

            var request = new CreateLogStreamRequest
            {
                LogGroupName = logGroupName,
                LogStreamName = logStreamName,
            };

            var response = await client.CreateLogStreamAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create stream.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateLogStreamin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Im folgenden Beispiel wird ein Protokollstream mit dem Namen `20150601` in der Protokollgruppe `my-logs` erstellt.

```
aws logs create-log-stream --log-group-name my-logs --log-stream-name 20150601
```

- Einzelheiten zur API finden Sie CreateLogStreamin der AWS CLI Befehlsreferenz.

Verwenden Sie **`DeleteLogGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLogGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie Ihre erste serverlose Funktion

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
```

```
            var request = new DeleteLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.DeleteLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted CloudWatch log group,
 {logGroupName}.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DeleteLogGroupin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird eine Protokollgruppe mit dem Namen `my-logs` gelöscht:

```
aws logs delete-log-group --log-group-name my-logs
```

- Einzelheiten zur API finden Sie DeleteLogGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";
```

```
const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie DeleteLogGroupin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DeleteSubscriptionFilter`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteSubscriptionFilter` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Löschen Sie den Abonnementfilter.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);

        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
                  logGroup - The name of the log group. (for example, testgroup).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
```

```
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
```

```
};

export default run();
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeExportTasks`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeExportTasks`.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
```

```
                    Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has
status: {t.Status}");
                });
            }
            while (response.NextToken is not null);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeExportTasksin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`DescribeLogGroups`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLogGroups` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
```

```
            // user. If you need to work with resources in another
            // AWS Region than the one defined for the default user,
            // pass the AWS Region as a parameter to the client constructor.
            var client = new AmazonCloudWatchLogsClient();

            bool done = false;
            string newToken = null;

            var request = new DescribeLogGroupsRequest
            {
                Limit = 5,
            };

            DescribeLogGroupsResponse response;

            do
            {
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on:
{lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
                    done = true;
                }
                else
                {
                    newToken = response.NextToken;
                }
            }
            while (!done);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie DescribeLogGroupsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl beschreibt eine Protokollgruppe mit dem Namen `my-logs`:

```
aws logs describe-log-groups --log-group-name-prefix my-logs
```

Ausgabe:

```
{
    "logGroups": [
        {
            "storedBytes": 0,
            "metricFilterCount": 0,
            "creationTime": 1433189500783,
            "logGroupName": "my-logs",
            "retentionInDays": 5,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:*"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLogGroupsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- Einzelheiten zur API finden Sie DescribeLogGroupsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DescribeLogStreams`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLogStreams` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie Ihre erste serverlose Funktion

CLI

AWS CLI

Der folgende Befehl zeigt alle Protokollstreams in der Protokollgruppe `my-logs` an, die mit dem Präfix `2015` beginnen:

```
aws logs describe-log-streams --log-group-name my-logs --log-stream-name-
prefix 2015
```

Ausgabe:

```
{
    "logStreams": [
        {
            "creationTime": 1433189871774,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150531",
            "logStreamName": "20150531",
            "storedBytes": 0
        },
        {
            "creationTime": 1433189873898,
            "arn": "arn:aws:logs:us-west-2:0123456789012:log-group:my-logs:log-
stream:20150601",
            "logStreamName": "20150601",
            "storedBytes": 0
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLogStreamsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sucht innerhalb einer angegebenen Protokollgruppe nach Protokollströmen, die einem bestimmten Präfix entsprechen.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
 pattern);
    }

    /**
```

```
    * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
    *
    * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
    * @param logGroupName    the name of the log group to search within
    * @param logStreamPrefix the prefix of the log streams to search for
    * @param pattern         the pattern to filter log events by
    */
   public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
       DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
               .logGroupName(logGroupName)
               .logStreamNamePrefix(logStreamPrefix)
               .build();

       DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
       List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

       for (LogStream logStream : logStreams) {
           String logStreamName = logStream.logStreamName();
           System.out.println("Searching in log stream: " + logStreamName);

           FilterLogEventsRequest filterLogEventsRequest =
FilterLogEventsRequest.builder()
                   .logGroupName(logGroupName)
                   .logStreamNames(logStreamName)
                   .filterPattern(pattern)
                   .build();

           FilterLogEventsResponse filterLogEventsResponse =
cwlClient.filterLogEvents(filterLogEventsRequest);

           for (FilteredLogEvent event : filterLogEventsResponse.events()) {
               System.out.println(event.message());
           }


System.out.println("--------------------------------------------------"); //
Separator for better readability
       }
   }
```

```
}
```

Druckt Metadaten über den neuesten Protokollstream in einer angegebenen Protokollgruppe.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
```

```
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient
 logsClient, String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " +
 stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeLogStreamsunter AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeSubscriptionFilters`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeSubscriptionFilters` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

Listen Sie die Abonnementfilter auf.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
```

```
                    request);
            if (!outcome.IsSuccess()) {
                std::cout << "Failed to describe CloudWatch subscription filters
"
                    << "for log group " << log_group << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }
```

```
   public static void describeFilters(CloudWatchLogsClient logs, String
logGroup) {
       try {
           boolean done = false;
           String newToken = null;

           while (!done) {
               DescribeSubscriptionFiltersResponse response;
               if (newToken == null) {
                   DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                           .logGroupName(logGroup)
                           .limit(1).build();

                   response = logs.describeSubscriptionFilters(request);
               } else {
                   DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                           .nextToken(newToken)
                           .logGroupName(logGroup)
                           .limit(1).build();
                   response = logs.describeSubscriptionFilters(request);
               }

               for (SubscriptionFilter filter : response.subscriptionFilters())
{
                   System.out.printf("Retrieved filter with name %s, " +
"pattern %s " + "and destination arn %s",
                           filter.filterName(),
                           filter.filterPattern(),
                           filter.destinationArn());
               }

               if (response.nextToken() == null) {
                   done = true;
               } else {
                   newToken = response.nextToken();
               }
           }

       } catch (CloudWatchException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
        System.out.printf("Done");
    }
}
```

- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient
 ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetLogEvents`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetLogEvents` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie Ihre erste serverlose Funktion

CLI

AWS CLI

Der folgende Befehl ruft Protokollereignisse aus einem Protokollstream mit dem Namen `20150601` in der Protokollgruppe `my-logs` ab:

```
aws logs get-log-events --log-group-name my-logs --log-stream-name 20150601
```

Ausgabe:

```
{
    "nextForwardToken":
 "f/31961209122447488583055879464742346735121166569214640130",
    "events": [
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190516679,
            "timestamp": 1433190184356,
            "message": "Example Event 1"
        },
        {
            "ingestionTime": 1433190494190,
            "timestamp": 1433190184358,
            "message": "Example Event 2"
        }
    ],
    "nextBackwardToken":
 "b/31961209122358285602261756944988674324553373268216709120"
}
```

- Einzelheiten zur API finden Sie GetLogEventsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
```

```
                  logGroupName - The name of the log group (for example,
 myloggroup).
                  logStreamName - The name of the log stream (for example,
 mystream).

               """;

      // if (args.length != 2) {
      //     System.out.print(usage);
      //     System.exit(1);
//        }

       String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
       String logStreamName = "weathertop-java-stream" ; //args[1];

       Region region = Region.US_EAST_1 ;
       CloudWatchLogsClient cloudWatchLogsClient =
 CloudWatchLogsClient.builder()
               .region(region)
               .build();

       getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
       cloudWatchLogsClient.close();
   }

   public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                     String logGroupName,
                                     String logStreamPrefix) {
       try {
           // First, find the exact log stream name
           DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                   .logGroupName(logGroupName)
                   .logStreamNamePrefix(logStreamPrefix)
                   .limit(1) // get the first matching stream
                   .build();

           DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

           if (describeResponse.logStreams().isEmpty()) {
               System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
               return;
```

```
            }

            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7,
 ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest =
 GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not
 prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetLogEventsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetQueryResults`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetQueryResults` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Ausführen einer umfangreichen Abfrage

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string
queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while getting query results:
{ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie GetQueryResultsin der AWS SDK für .NET API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- Einzelheiten zur API finden Sie GetQueryResultsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Einzelheiten zur API finden Sie GetQueryResultsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwl->getqueryresults(
          iv_queryid = iv_query_id ).

        " Display query status and result count
```

```
        DATA(lv_status) = oo_result->get_status( ).
        DATA(lt_results) = oo_result->get_results( ).
        DATA(lv_result_count) = lines( lt_results ).

        MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetQueryResultsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutSubscriptionFilter`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutSubscriptionFilter` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie den Abonnementfilter.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;
```

```
/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
                  logGroup - A log group name (testgroup).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
 PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();

            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter
 %s",
                    filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter
 pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
```

```
    console.error(err);
  }
};

export default run();
```

- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`StartLiveTail`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartLiveTail` verwendet wird.

.NET

SDK für .NET

Binden Sie die erforderlichen Dateien ein.

```
using Amazon;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

Starten Sie die Live-Tail-Sitzung.

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Sie können die Ereignisse aus der Live Tail-Sitzung auf zwei Arten handhaben:

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail
session at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
sessionResult.Message);
                        }
                    }
                    if (item is LiveTailSessionStart)
                    {
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
```

```
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
 session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
                {
                    foreach (LiveTailSessionLogEvent logEvent in
 e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
                    Console.WriteLine($"ERROR:
 {e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- Einzelheiten zur API finden Sie StartLiveTailin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

Binden Sie die erforderlichen Dateien ein.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
    return
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Starten Sie die Live-Tail-Sitzung.

```
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error, " + err.Error())
 }
 client := cloudwatchlogs.NewFromConfig(cfg)

 request := &cloudwatchlogs.StartLiveTailInput{
  LogGroupIdentifiers:   logGroupIdentifiers,
```

```
 LogStreamNames:        logStreamNames,
 LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- Einzelheiten zur API finden Sie StartLiveTailin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Binden Sie die erforderlichen Dateien ein.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
```

```
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
    private static StartLiveTailResponseHandler
 getStartLiveTailResponseStreamHandler(
            AtomicReference<Subscription> subscriptionAtomicReference) {
        return StartLiveTailResponseHandler.builder()
            .onResponse(r -> System.out.println("Received initial response"))
            .onError(throwable -> {
                CloudWatchLogsException e = (CloudWatchLogsException)
 throwable.getCause();
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            })
            .subscriber(() -> new FlowableSubscriber<>() {
                @Override
                public void onSubscribe(@NonNull Subscription s) {
                    subscriptionAtomicReference.set(s);
                    s.request(Long.MAX_VALUE);
                }

                @Override
                public void onNext(StartLiveTailResponseStream event) {
                    if (event instanceof LiveTailSessionStart) {
                        LiveTailSessionStart sessionStart =
 (LiveTailSessionStart) event;
                        System.out.println(sessionStart);
                    } else if (event instanceof LiveTailSessionUpdate) {
                        LiveTailSessionUpdate sessionUpdate =
 (LiveTailSessionUpdate) event;
                        List<LiveTailSessionLogEvent> logEvents =
 sessionUpdate.sessionResults();
```

```
                        logEvents.forEach(e -> {
                            long timestamp = e.timestamp();
                            Date date = new Date(timestamp);
                            System.out.println("[" + date + "] " + e.message());
                        });
                    } else {
                        throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                    }
                }

                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Starten Sie die Live-Tail-Sitzung.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
AtomicReference<>(null);
```

```
        cloudWatchLogsAsyncClient.startLiveTail(request,
 getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- Einzelheiten zur API finden Sie StartLiveTailin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Binden Sie die erforderlichen Dateien ein.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
async function handleResponseAsync(response) {
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
```

```
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Starten Sie die Live-Tail-Sitzung.

```
    const client = new CloudWatchLogsClient();

    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
    try{
        const response = await client.send(command);
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
    /* Set a timeout to close the client. This will stop the Live Tail session.
 */
    setTimeout(function() {
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- Einzelheiten zur API finden Sie StartLiveTailin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Binden Sie die erforderlichen Dateien ein.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Starten Sie die Live-Tail-Sitzung.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
 10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is
 StartLiveTailResponseStream.SessionUpdate) {
```

```
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- Einzelheiten zur API finden Sie StartLiveTailin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

Binden Sie die erforderlichen Dateien ein.

```
import boto3
import time
from datetime import datetime
```

Starten Sie die Live-Tail-Sitzung.

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
```

```
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
                    print('[{date}]
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['me
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- Einzelheiten zur API finden Sie StartLiveTailin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`StartQuery`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartQuery` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Ausführen einer umfangreichen Abfrage

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</
param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };

            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
```

```
        }
        catch (InvalidParameterException ex)
        {
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie StartQueryin der AWS SDK für .NET API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
```

```
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
```

- Einzelheiten zur API finden Sie StartQueryin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
```

```
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
```

```
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- Einzelheiten zur API finden Sie StartQueryin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_log_group_name = '/aws/lambda/my-function'
```

```
        " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
        " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms
since Jan 1, 1970 00:00:00 UTC)
        oo_result = lo_cwl->startquery(
          iv_loggroupname = iv_log_group_name
          iv_starttime    = iv_start_time
          iv_endtime      = iv_end_time
          iv_querystring  = iv_query_string
          iv_limit        = iv_limit ).

        " Display the query ID for tracking
        DATA(lv_query_id) = oo_result->get_queryid( ).
        MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwllimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cwlmalformedqueryex.
        MESSAGE 'Malformed query.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartQueryin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für CloudWatch Protokolle mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in CloudWatch Logs with AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in CloudWatch Logs aufrufen oder mit anderen AWS-Services kombinieren. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie Ihre erste serverlose Funktion
- Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits
```

```
set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}
```

```
# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
```

```
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
```

```
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }
```

```
    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
```

```
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
```

```
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
```

```
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
```

```
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
```

```
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
```

```
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup

- DeleteNamespace
- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Eine IAM-Rolle für Lambda erstellen
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES
```

```
##############################################################################
# Helper functions
##############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
```

```
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
```

```
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
```

```
##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
```

```
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

###############################################################################
# Step 1: Create IAM execution role
###############################################################################

echo ""
echo "==========================================="
echo "Step 1: Create IAM execution role"
echo "==========================================="
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10
```

```
##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi
```

```
echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
```

```
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "============================================"
echo "Step 4: View CloudWatch Logs"
echo "============================================"
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
```

```
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy

- CreateFunction
- CreateRole
- DeleteFunction
- DeleteLogGroup
- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

Verwenden Sie CloudWatch Logs, um eine umfangreiche Abfrage auszuführen

Die folgenden Codebeispiele zeigen, wie CloudWatch Logs verwendet werden kann, um mehr als 10.000 Datensätze abzufragen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Hauptworkflow, der das Szenario mit großen Abfragen demonstriert.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.

    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.

    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-
query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";
```

```
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
```

```
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query
Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to
retrieve more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
```

```
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack
resource .yaml file (or press Enter for default '{_stackResourcePath}'): ");
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
```

```
                    }
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);

                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
                        return true;
                    }
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
```

```
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
```

```
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
```

```
        {
            // Calculate timestamps
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");

            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }

                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query
 API)
```

```
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

            Console.WriteLine($"Query start date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-
ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Query end date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log
 entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample
 logs.");
            return false;
        }
    }

    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000)
", 10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
 _queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();

        Console.WriteLine();
        Console.WriteLine($"Queries finished in
 {stopwatch.Elapsed.TotalSeconds:F3} seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");
```

```
        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count -
1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();

        var viewSample = !_interactive || GetYesNoResponse("Would you like to see
a sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
```

```
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
 startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
        }

        var startDate =
 DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-
MM-ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than
 limit of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and
 recurse.");

        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
```

```
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");

        // Parse the timestamp and add 1 millisecond to avoid querying the same
 log again
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-
MM-ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid
 overlapping with logs in the same second
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s
 ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-MM-
ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing:
 offsetLastLogTime={offsetLastLogTime}s vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }
```

```
        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);

        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
 {range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
 {range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
 range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");

        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
 range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);

        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
 ({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
```

```
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }

        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
 range2End) SplitDateRange(long startTime, long endTime)
    {
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
 queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;
```

```
        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources?
(y/n) ");

        if (cleanup)
        {
            try
            {
```

```
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }

        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is
complete.");
        return true;
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

        if (stackDeleted)
        {
```

```
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }

                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
```

```
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                      ");
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

```
    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
```

```
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'):
");
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }

    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
FindDuplicateLogs(List<List<ResultField>> logs)
    {
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
```

```
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count:
 kvp.Value);
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - GetQueryResults
  - StartQuery

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}

const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs
 found: ${cloudWatchQuery.results.length}`,
);
```

Dies ist eine Klasse, die Abfragen bei Bedarf in mehrere Schritte aufteilt.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient}
 client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
 { limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
```

```
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[]
[]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
```

```
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();

    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }

  /**
   * Starts a query and waits for it to complete.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   */
  async _query(dateRange, maxLogs) {
    try {
      const { queryId } = await this._startQuery(dateRange, maxLogs);
      const { results } = await this._waitUntilQueryDone(queryId);
      return results ?? [];
    } catch (err) {
      /**
       * This error is thrown when StartQuery returns an error indicating
```

```
     * that the query's start or end date occur before the log group was
     * created.
     */
    if (err instanceof DateOutOfBoundsError) {
      return [];
    }
    throw err;
  }
}

/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
      // before the log group was created.
      throw new DateOutOfBoundsError(message);
    }

    throw err;
  }
}

/**
 * Call GetQueryResultsCommand until the query is done.
 * @param {string} queryId
```

```
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - GetQueryResults
  - StartQuery

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Diese Datei ruft ein Beispielmodul für die Verwaltung von CloudWatch Abfragen mit mehr als 10.000 Ergebnissen auf.

```
import logging
import os
import sys

import boto3
from botocore.config import Config

from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date
 utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()
```

```
    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts":
10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and
end dates.
        Fetches the environment variable for log group, returning the default
value if it
        does not exist.

        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables
are required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
```

```
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using
 default value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 =
 self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided
 date range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
```

```
        :type log_group: str
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds.
 Total logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the
 query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601,
 log_group=log_group)


if __name__ == "__main__":
    main()
```

Dieses Modul verarbeitet CloudWatch Abfragen mit mehr als 10.000 Ergebnissen.

```
import logging
import time
```

```
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """

    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
 str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and
 calculates the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
```

```
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
        :return: None if the recursive fetching is continued or stops when the
final batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
```

```
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log
```

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []
```

```
    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for
the query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
```

```
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - GetQueryResults
  - StartQuery

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Codebeispiele für die CodeBuild Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CodeBuild mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CodeBuild Benutzerhandbuch — Weitere Informationen zu CodeBuild.
- CodeBuild API-Referenz — Details zu allen verfügbaren CodeBuild Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CodeBuild AWS SDKs
  - Aktionen zur Verwendung CodeBuild AWS SDKs
    - Verwenden Sie CreateProjectmit einem AWS SDK oder CLI
    - Verwenden Sie ListBuildsmit einem AWS SDK oder CLI
    - Verwenden Sie ListProjectsmit einem AWS SDK oder CLI
    - Verwenden Sie StartBuildmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung CodeBuild AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CodeBuild mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung CodeBuild AWS SDKs
  - Verwenden Sie CreateProjectmit einem AWS SDK oder CLI
  - Verwenden Sie ListBuildsmit einem AWS SDK oder CLI
  - Verwenden Sie ListProjectsmit einem AWS SDK oder CLI
  - Verwenden Sie StartBuildmit einem AWS SDK oder CLI

Aktionen zur Verwendung CodeBuild AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CodeBuild Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CodeBuild -API-Referenz.

Beispiele

- Verwenden Sie CreateProjectmit einem AWS SDK oder CLI
- Verwenden Sie ListBuildsmit einem AWS SDK oder CLI
- Verwenden Sie ListProjectsmit einem AWS SDK oder CLI
- Verwenden Sie StartBuildmit einem AWS SDK oder CLI

Verwenden Sie **`CreateProject`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateProject` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um ein AWS CodeBuild Build-Projekt zu erstellen

Im folgenden `create-project` Beispiel wird ein CodeBuild Build-Projekt mit Quelldateien aus einem S3-Bucket erstellt

```
aws codebuild create-project \
    --name "my-demo-project" \
    --source "{\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-input-bucket/my-source.zip\"}" \
    --artifacts {"\"type\": \"S3\",\"location\": \"codebuild-us-west-2-123456789012-output-bucket\""} \
    --environment "{\"type\": \"LINUX_CONTAINER\",\"image\": \"aws/codebuild/standard:1.0\",\"computeType\": \"BUILD_GENERAL1_SMALL\"}" \
    --service-role "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role"
```

Ausgabe:

```
{
    "project": {
        "arn": "arn:aws:codebuild:us-west-2:123456789012:project/my-demo-project",
        "name": "my-cli-demo-project",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-service-role",
        "lastModified": 1556839783.274,
        "badge": {
            "badgeEnabled": false
        },
        "queuedTimeoutInMinutes": 480,
        "environment": {
            "image": "aws/codebuild/standard:1.0",
            "computeType": "BUILD_GENERAL1_SMALL",
            "type": "LINUX_CONTAINER",
```

```
            "imagePullCredentialsType": "CODEBUILD",
            "privilegedMode": false,
            "environmentVariables": []
        },
        "artifacts": {
            "location": "codebuild-us-west-2-123456789012-output-bucket",
            "name": "my-cli-demo-project",
            "namespaceType": "NONE",
            "type": "S3",
            "packaging": "NONE",
            "encryptionDisabled": false
        },
        "source": {
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip",
            "insecureSsl": false
        },
        "timeoutInMinutes": 60,
        "cache": {
            "type": "NO_CACHE"
        },
        "created": 1556839783.274
    }
}
```

Beispiel 2: Um ein AWS CodeBuild Build-Projekt mit einer JSON-Eingabedatei für die Parameter zu erstellen

Im folgenden `create-project` Beispiel wird ein CodeBuild Build-Projekt erstellt, indem alle erforderlichen Parameter in einer JSON-Eingabedatei übergeben werden. Erstellen Sie die Vorlage für die Eingabedatei, indem Sie den Befehl nur mit `--generate-cli-skeleton parameter` ausführen.

```
aws codebuild create-project --cli-input-json file://create-project.json
```

Die JSON-Eingabedatei `create-project.json` enthält Folgendes:

```
{
    "name": "codebuild-demo-project",
    "source": {
        "type": "S3",
```

```
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "artifacts": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-output-bucket"
    },
    "environment": {
        "type": "LINUX_CONTAINER",
        "image": "aws/codebuild/standard:1.0",
        "computeType": "BUILD_GENERAL1_SMALL"
    },
    "serviceRole": "serviceIAMRole"
}
```

Ausgabe:

```
{
    "project": {
        "name": "codebuild-demo-project",
        "serviceRole": "serviceIAMRole",
        "tags": [],
        "artifacts": {
            "packaging": "NONE",
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-output-bucket",
            "name": "message-util.zip"
        },
        "lastModified": 1472661575.244,
        "timeoutInMinutes": 60,
        "created": 1472661575.244,
        "environment": {
            "computeType": "BUILD_GENERAL1_SMALL",
            "image": "aws/codebuild/standard:1.0",
            "type": "LINUX_CONTAINER",
            "environmentVariables": []
        },
        "source": {
            "type": "S3",
            "location": "codebuild-region-ID-account-ID-input-bucket/
MessageUtil.zip"
        },
        "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
```

```
        "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Build-Projekts (AWS CLI) im AWS CodeBuild Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateProjectunter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ein Projekt zu erstellen.

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
```

```
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of
 "computeType" and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-
ref-available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
        // The location of the repository that contains the source code to be
 built.
        location: githubUrl,
      },
    }),
  );
  console.log(response);
  //   {
  //     '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //     },
  //     project: {
```

```
  //       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
  //       artifacts: {
  //         encryptionDisabled: false,
  //         location: 'xxxxxx-xxxxxxx-xxxxxx',
  //         name: 'MyCodeBuilder',
  //         namespaceType: 'NONE',
  //         packaging: 'NONE',
  //         type: 'S3'
  //       },
  //       badge: { badgeEnabled: false },
  //       cache: { type: 'NO_CACHE' },
  //       created: 2023-08-18T14:46:48.979Z,
  //       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
  //       environment: {
  //         computeType: 'BUILD_GENERAL1_SMALL',
  //         environmentVariables: [],
  //         image: 'aws/codebuild/standard:7.0',
  //         imagePullCredentialsType: 'CODEBUILD',
  //         privilegedMode: false,
  //         type: 'LINUX_CONTAINER'
  //       },
  //       lastModified: 2023-08-18T14:46:48.979Z,
  //       name: 'MyCodeBuilder',
  //       projectVisibility: 'PRIVATE',
  //       queuedTimeoutInMinutes: 480,
  //       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
  //       source: {
  //         insecureSsl: false,
  //         location: 'https://...',
  //         reportBuildStatus: false,
  //         type: 'GITHUB'
  //       },
  //       timeoutInMinutes: 60
  //     }
  //   }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateProjectin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`ListBuilds`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListBuilds` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
 codeBuildClient.ListBuilds(
                listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
 listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {
```

```
                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
 codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);

                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
 getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." <<
 std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;
                    }
                } else {
                    std::cerr << "Error getting builds"
                              << getBuildsOutcome.GetError().GetMessage() <<
 std::endl;
                    return false;
                }
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.

            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!nextToken.

            empty()

            );

    return true;
}
```

- Einzelheiten zur API finden Sie ListBuildsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine Liste der AWS CodeBuild Build-IDs zu erhalten.

Im folgenden `list-builds` Beispiel wird eine Liste von CodeBuild IDs abgerufen, die in aufsteigender Reihenfolge sortiert sind.

```
aws codebuild list-builds --sort-order ASCENDING
```

Die Ausgabe enthält einen `nextToken`-Wert, der angibt, dass mehr Ausgaben verfügbar sind.

```
{
    "nextToken": "4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==",
    "ids": [
        "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
        "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
    ]
}
```

Führen Sie diesen Befehl erneut aus und geben Sie den `nextToken`-Wert aus der vorherigen Antwort als Parameter an, um den nächsten Teil der Ausgabe abzurufen. Wiederholen Sie den Vorgang, bis Sie in der Antwort keinen `nextToken`-Wert mehr erhalten.

```
aws codebuild list-builds --sort-order ASCENDING --next-
token 4AEA6u7J...The full token has been omitted for brevity...MzY2OA==
```

Nächster Teil der Ausgabe:

```
{
    "ids": [
```

```
        "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
        "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
            ... The full list of build IDs has been omitted for brevity ...
        "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
    ]
}
```

Weitere Informationen finden Sie unter Anzeigen einer Liste von Build-IDs (AWS CLI) im AWS CodeBuild Benutzerhandbuch

- Einzelheiten zur API finden Sie ListBuildsunter AWS CLI Befehlsreferenz.

Verwenden Sie **`ListProjects`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListProjects` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType
 sortType,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
```

```
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(),
 projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" <<
 outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListProjectsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine Liste der AWS CodeBuild Build-Projektnamen zu erhalten.

Im folgenden `list-projects` Beispiel wird eine Liste von CodeBuild Build-Projekten abgerufen, die in aufsteigender Reihenfolge nach Namen sortiert sind.

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

Die Ausgabe enthält einen `nextToken`-Wert, der angibt, dass mehr Ausgaben verfügbar sind.

```
{
    "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U
+AkMx8=",
    "projects": [
        "codebuild-demo-project",
        "codebuild-demo-project2",
            ... The full list of build project names has been omitted for
 brevity ...
        "codebuild-demo-project99"
    ]
}
```

Führen Sie diesen Befehl erneut aus und geben Sie den `nextToken`-Wert aus der vorherigen Antwort als Parameter an, um den nächsten Teil der Ausgabe abzurufen. Wiederholen Sie den Vorgang, bis Sie in der Antwort keinen `nextToken`-Wert mehr erhalten.

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-
token Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=

{
    "projects": [
        "codebuild-demo-project100",
        "codebuild-demo-project101",

  ... The full list of build project names has been omitted for brevity ...
        "codebuild-demo-project122"

  ]
}
```

Weitere Informationen finden Sie im AWS CodeBuild Benutzerhandbuch unter Anzeigen einer Liste von Build-Projektnamen (AWS CLI).

- Einzelheiten zur API finden Sie ListProjectsin der AWS CLI Befehlsreferenz.

Verwenden Sie **`StartBuild`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartBuild` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Start an AWS CodeBuild project build.
/*!
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome =
 codeBuildClient.StartBuild(
            startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie StartBuildin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um mit der Ausführung eines AWS CodeBuild Build-Projekts zu beginnen.

Im folgenden `start-build` Beispiel wird ein Build für das angegebene CodeBuild Projekt gestartet. Der Build überschreibt sowohl die Einstellung des Projekts für die Anzahl der Minuten, die der Build in die Warteschlange gestellt werden darf, bis eine Zeitüberschreitung auftritt, als auch die Artefakteinstellungen des Projekts.

```
aws codebuild start-build \
    --project-name "my-demo-project" \
    --queued-timeout-in-minutes-override 5 \
    --artifacts-override {"\"type\": \"S3\",\"location\":
 \"arn:aws:s3:::artifacts-override\",\"overrideArtifactName\":true"}
```

Ausgabe:

```
{
    "build": {
        "serviceRole": "arn:aws:iam::123456789012:role/service-role/my-codebuild-
service-role",
        "buildStatus": "IN_PROGRESS",
        "buildComplete": false,
        "projectName": "my-demo-project",
        "timeoutInMinutes": 60,
        "source": {
            "insecureSsl": false,
            "type": "S3",
            "location": "codebuild-us-west-2-123456789012-input-bucket/my-
source.zip"
        },
        "queuedTimeoutInMinutes": 5,
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3",
        "currentPhase": "QUEUED",
```

```
        "startTime": 1556905683.568,
        "environment": {
            "computeType": "BUILD_GENERAL1_MEDIUM",
            "environmentVariables": [],
            "type": "LINUX_CONTAINER",
            "privilegedMode": false,
            "image": "aws/codebuild/standard:1.0",
            "imagePullCredentialsType": "CODEBUILD"
        },
        "phases": [
            {
                "phaseStatus": "SUCCEEDED",
                "startTime": 1556905683.568,
                "phaseType": "SUBMITTED",
                "durationInSeconds": 0,
                "endTime": 1556905684.524
            },
            {
                "startTime": 1556905684.524,
                "phaseType": "QUEUED"
            }
        ],
        "logs": {
            "deepLink": "https://console.aws.amazon.com/cloudwatch/home?
region=us-west-2#logEvent:group=null;stream=null"
        },
        "artifacts": {
            "encryptionDisabled": false,
            "location": "arn:aws:s3:::artifacts-override/my-demo-project",
            "overrideArtifactName": true
        },
        "cache": {
            "type": "NO_CACHE"
        },
        "id": "my-demo-project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE",
        "initiator": "my-aws-account-name",
        "arn": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-
project::12345678-a1b2-c3d4-e5f6-11111EXAMPLE"
    }
}
```

Weitere Informationen finden Sie unter Run a Build (AWS CLI) im AWS CodeBuild Benutzerhandbuch.

- Einzelheiten zur API finden Sie StartBuildin der AWS CLI Befehlsreferenz.

Codebeispiele für die CodeCommit Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CodeCommit mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CodeCommit Benutzerhandbuch — Weitere Informationen zu CodeCommit.
- CodeCommit API-Referenz — Details zu allen verfügbaren CodeCommit Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CodeCommit AWS SDKs
  - Aktionen zur Verwendung CodeCommit AWS SDKs
    - Verwendung von BatchGetRepositories mit einer CLI
    - Verwendung von CreateBranch mit einer CLI
    - Verwendung von CreateRepository mit einer CLI
    - Verwendung von DeleteRepository mit einer CLI
    - Verwendung von GetBranch mit einer CLI
    - Verwendung von GetRepository mit einer CLI
    - Verwendung von ListBranches mit einer CLI
    - Verwendung von ListRepositories mit einer CLI
    - Verwendung von UpdateDefaultBranch mit einer CLI
    - Verwendung von UpdateRepositoryDescription mit einer CLI
    - Verwendung von UpdateRepositoryName mit einer CLI

Grundlegende Beispiele für die Verwendung CodeCommit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CodeCommit mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung CodeCommit AWS SDKs
  - Verwendung von BatchGetRepositories mit einer CLI
  - Verwendung von CreateBranch mit einer CLI
  - Verwendung von CreateRepository mit einer CLI
  - Verwendung von DeleteRepository mit einer CLI
  - Verwendung von GetBranch mit einer CLI
  - Verwendung von GetRepository mit einer CLI
  - Verwendung von ListBranches mit einer CLI
  - Verwendung von ListRepositories mit einer CLI
  - Verwendung von UpdateDefaultBranch mit einer CLI
  - Verwendung von UpdateRepositoryDescription mit einer CLI
  - Verwendung von UpdateRepositoryName mit einer CLI

Aktionen zur Verwendung CodeCommit AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CodeCommit Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CodeCommit -API-Referenz.

Beispiele

- Verwendung von BatchGetRepositories mit einer CLI
- Verwendung von CreateBranch mit einer CLI
- Verwendung von CreateRepository mit einer CLI
- Verwendung von DeleteRepository mit einer CLI
- Verwendung von GetBranch mit einer CLI

- Verwendung von GetRepository mit einer CLI
- Verwendung von ListBranches mit einer CLI
- Verwendung von ListRepositories mit einer CLI
- Verwendung von UpdateDefaultBranch mit einer CLI
- Verwendung von UpdateRepositoryDescription mit einer CLI
- Verwendung von UpdateRepositoryName mit einer CLI

Verwendung von **`BatchGetRepositories`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetRepositories` verwendet wird.

CLI

AWS CLI

So zeigen Sie Details zu mehreren Repositorys an

Dieses Beispiel zeigt Details zu mehreren AWS CodeCommit Repositorys.

```
aws codecommit batch-get-repositories \
    --repository-names MyDemoRepo MyOtherDemoRepo
```

Ausgabe:

```
{
    "repositoriesNotFound": [],
    "repositories": [
         {
            "creationDate": 1429203623.625,
            "defaultBranch": "main",
            "repositoryName": "MyDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyDemoRepo",
            "lastModifiedDate": 1430783812.0869999,
            "repositoryDescription": "My demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyDemoRepo",
            "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyDemoRepo"
            "accountId": "111111111111"
        },
```

```
        {
            "creationDate": 1429203623.627,
            "defaultBranch": "main",
            "repositoryName": "MyOtherDemoRepo",
            "cloneUrlSsh": "ssh://git-codecommit.us-east-2.amazonaws.com/v1/
repos/MyOtherDemoRepo",
            "lastModifiedDate": 1430783812.0889999,
            "repositoryDescription": "My other demonstration repository",
            "cloneUrlHttp": "https://codecommit.us-east-2.amazonaws.com/v1/repos/
MyOtherDemoRepo",
            "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE",
            "Arn": "arn:aws:codecommit:us-east-2:111111111111:MyOtherDemoRepo"
            "accountId": "111111111111"
        }
    ],
    "repositoriesNotFound": []
}
```

- Einzelheiten zur API finden Sie BatchGetRepositoriesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel bestätigt, welche der angegebenen Repositorys gefunden wurden und welche nicht.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Ausgabe:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Einzelheiten zur API finden Sie unter BatchGetRepositories AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel bestätigt, welche der angegebenen Repositorys gefunden wurden und welche nicht.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Ausgabe:

```
Repositories                            RepositoriesNotFound
------------                            --------------------
{MyDemoRepo, MyNewRepo}                {AMissingRepo}
```

- Einzelheiten zur API finden Sie unter BatchGetRepositories AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateBranch`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateBranch` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Branch

In diesem Beispiel wird ein Branch in einem AWS CodeCommit Repository erstellt. Dieser Befehl liefert nur eine Ausgabe, wenn Fehler aufgetreten sind.

Befehl:

```
aws codecommit create-branch --repository-name MyDemoRepo --branch-
name MyNewBranch --commit-id 317f8570EXAMPLE
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie CreateBranchin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer Branch mit dem angegebenen Namen für das angegebene Repository und der angegebenen Commit-ID erstellt.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter CreateBranch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Branch mit dem angegebenen Namen für das angegebene Repository und der angegebenen Commit-ID erstellt.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter CreateBranch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateRepository`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRepository` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein Repository

In diesem Beispiel wird ein Repository erstellt und es dem AWS Konto des Benutzers zugeordnet.

Befehl:

```
aws codecommit create-repository --repository-name MyDemoRepo --repository-
description "My demonstration repository"
```

Ausgabe:

```
{
    "repositoryMetadata": {
        "repositoryName": "MyDemoRepo",
                "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/
repos/MyDemoRepo",
                "lastModifiedDate": 1444766838.027,
        "repositoryDescription": "My demonstration repository",
                "cloneUrlHttp": "https://git-codecommit.us-east-1.amazonaws.com/
v1/repos/MyDemoRepo",
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
                "Arn": "arn:aws:codecommit:us-
east-1:111111111111EXAMPLE:MyDemoRepo",
        "accountId": "111111111111"
    }
}
```

- Einzelheiten zur API finden Sie CreateRepositoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues Repository mit dem angegebenen Namen und der angegebenen Beschreibung erstellt.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
```

```
RepositoryId           : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName         : MyDemoRepo
```

- Einzelheiten zur API finden Sie unter CreateRepository AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Repository mit dem angegebenen Namen und der angegebenen Beschreibung erstellt.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- Einzelheiten zur API finden Sie unter CreateRepository AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteRepository`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRepository` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Repository

Dieses Beispiel zeigt, wie ein AWS CodeCommit Repository gelöscht wird.

Befehl:

```
aws codecommit delete-repository --repository-name MyDemoRepo
```

Ausgabe:

```
{
  "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE"
}
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Repository zwangsweise gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Repository ohne Prompt zu löschen.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Einzelheiten zur API finden Sie unter DeleteRepository AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Repository zwangsweise gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Repository ohne Prompt zu löschen.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- Einzelheiten zur API finden Sie unter DeleteRepository AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBranch`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBranch` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einem Branch ab

In diesem Beispiel werden Informationen über einen Branch in einem AWS CodeCommit Repository abgerufen.

Befehl:

```
aws codecommit get-branch --repository-name MyDemoRepo --branch-name MyNewBranch
```

Ausgabe:

```
{
  "BranchInfo": {
        "commitID": "317f8570EXAMPLE",
                "branchName": "MyNewBranch"
  }
}
```

- Einzelheiten zur API finden Sie GetBranchin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Branch für das spezifizierte Repository abgerufen.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Ausgabe:

```
BranchName                                CommitId
----------                                --------
MyNewBranch                               7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter GetBranch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Branch für das spezifizierte Repository abgerufen.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Ausgabe:

```
BranchName                                CommitId
----------                                --------
MyNewBranch                               7763222d...561fc9c9
```

- Einzelheiten zur API finden Sie unter GetBranch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetRepository`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetRepository` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einem Repository ab

Dieses Beispiel zeigt Details zu einem AWS CodeCommit Repository.

```
aws codecommit get-repository \
```

```
    --repository-name MyDemoRepo
```

Ausgabe:

```
{
    "repositoryMetadata": {
        "creationDate": 1429203623.625,
        "defaultBranch": "main",
        "repositoryName": "MyDemoRepo",
        "cloneUrlSsh": "ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/v1/
repos/MyDemoRepo",
        "lastModifiedDate": 1430783812.0869999,
        "repositoryDescription": "My demonstration repository",
        "cloneUrlHttp": "https://codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo",
        "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
        "Arn": "arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
        "accountId": "111111111111"
    }
}
```

- Einzelheiten zur API finden Sie GetRepositoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen für das angegebene Repository abgerufen.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
```

```
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Einzelheiten zur API finden Sie unter GetRepository AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen für das angegebene Repository abgerufen.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- Einzelheiten zur API finden Sie unter GetRepository AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListBranches`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListBranches` verwendet wird.

CLI

AWS CLI

Um eine Liste von Branchennamen anzuzeigen

In diesem Beispiel werden alle Zweignamen in einem AWS CodeCommit Repository aufgeführt.

```
aws codecommit list-branches \
    --repository-name MyDemoRepo
```

Ausgabe:

```
{
    "branches": [
        "MyNewBranch",
        "main"
    ]
}
```

- Einzelheiten zur API finden Sie ListBranchesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von Branch-Namen für das angegebene Repository abgerufen.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Ausgabe:

```
master
MyNewBranch
```

- Einzelheiten zur API finden Sie unter ListBranches AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Branch-Namen für das angegebene Repository abgerufen.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Ausgabe:

```
master
MyNewBranch
```

- Einzelheiten zur API finden Sie unter ListBranches AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListRepositories`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListRepositories` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine Liste der Repositorys an

In diesem Beispiel werden alle AWS CodeCommit Repositorys aufgeführt, die dem AWS Konto des Benutzers zugeordnet sind.

Befehl:

```
aws codecommit list-repositories
```

Ausgabe:

```
{
  "repositories": [
      {
         "repositoryName": "MyDemoRepo"
         "repositoryId": "f7579e13-b83e-4027-aaef-650c0EXAMPLE",
      },
      {
         "repositoryName": "MyOtherDemoRepo"
         "repositoryId": "cfc29ac4-b0cb-44dc-9990-f6f51EXAMPLE"
      }
  ]
}
```

- Einzelheiten zur API finden Sie ListRepositoriesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Repositorys in aufsteigender Reihenfolge nach Repository-Namen aufgelistet.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Ausgabe:

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Einzelheiten zur API finden Sie unter ListRepositories AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Repositorys in aufsteigender Reihenfolge nach Repository-Namen aufgelistet.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Ausgabe:

```
RepositoryId                            RepositoryName
------------                            --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- Einzelheiten zur API finden Sie unter ListRepositories AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateDefaultBranch`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateDefaultBranch` verwendet wird.

CLI

AWS CLI

So ändern Sie den Standard-Branch für ein Repository

In diesem Beispiel wird der Standardzweig für ein AWS CodeCommit Repository geändert. Dieser Befehl liefert nur eine Ausgabe, wenn Fehler aufgetreten sind.

Befehl:

```
aws codecommit update-default-branch --repository-name MyDemoRepo --default-branch-name MyNewBranch
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie UpdateDefaultBranchin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Standard-Branch für das angegebene Repository in den spezifischen Branch geändert.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Einzelheiten zur API finden Sie unter UpdateDefaultBranch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Standard-Branch für das angegebene Repository in den spezifischen Branch geändert.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- Einzelheiten zur API finden Sie unter UpdateDefaultBranch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateRepositoryDescription`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateRepositoryDescription` verwendet wird.

CLI

AWS CLI

So ändern Sie die Beschreibung eines Repositorys

In diesem Beispiel wird die Beschreibung für ein AWS CodeCommit Repository geändert. Dieser Befehl liefert nur eine Ausgabe, wenn Fehler aufgetreten sind.

Befehl:

```
aws codecommit update-repository-description --repository-name MyDemoRepo --
repository-description "This description was changed"
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie UpdateRepositoryDescriptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Repository geändert.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryDescription AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Repository geändert.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryDescription AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateRepositoryName`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateRepositoryName` verwendet wird.

CLI

AWS CLI

So ändern Sie den Namen eines Repositorys

In diesem Beispiel wird der Name eines AWS CodeCommit Repositorys geändert. Dieser Befehl liefert nur eine Ausgabe, wenn Fehler aufgetreten sind. Wenn Sie den Namen des AWS CodeCommit Repositorys ändern, ändern sich auch die SSH- und HTTPS-URLs, die Benutzer benötigen, um sich mit dem Repository zu verbinden. Benutzer können erst eine Verbindung mit diesem Repository herstellen, wenn sie ihre Verbindungseinstellungen aktualisiert haben. Da sich auch der ARN des Repositorys ändert, wird durch die Änderung des Repository-Namens jede IAM-Benutzerrichtlinie ungültig, die den ARN dieses Repositorys verwendet.

Befehl:

```
aws codecommit update-repository-name --old-name MyDemoRepo --new-
name MyRenamedDemoRepo
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie UpdateRepositoryNamein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Name des angegebenen Repositorys geändert.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryName AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Name des angegebenen Repositorys geändert.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- Einzelheiten zur API finden Sie unter UpdateRepositoryName AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Verwendung CodeDeploy AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CodeDeploy mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CodeDeploy Entwicklerhandbuch — Weitere Informationen zu CodeDeploy.
- CodeDeploy API-Referenz — Details zu allen verfügbaren CodeDeploy Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CodeDeploy AWS SDKs
  - Aktionen zur Verwendung CodeDeploy AWS SDKs
    - Verwendung von AddTagsToOnPremisesInstances mit einer CLI
    - Verwendung von BatchGetApplications mit einer CLI
    - Verwendung von BatchGetDeployments mit einer CLI

- Verwendung von BatchGetOnPremisesInstances mit einer CLI
- Verwendung von CreateApplication mit einer CLI
- Verwendung von CreateDeployment mit einer CLI
- Verwendung von CreateDeploymentConfig mit einer CLI
- Verwendung von CreateDeploymentGroup mit einer CLI
- Verwendung von DeleteApplication mit einer CLI
- Verwendung von DeleteDeploymentConfig mit einer CLI
- Verwendung von DeleteDeploymentGroup mit einer CLI
- Verwendung von DeregisterOnPremisesInstance mit einer CLI
- Verwendung von GetApplication mit einer CLI
- Verwendung von GetApplicationRevision mit einer CLI
- Verwendung von GetDeployment mit einer CLI
- Verwendung von GetDeploymentConfig mit einer CLI
- Verwendung von GetDeploymentGroup mit einer CLI
- Verwendung von GetDeploymentInstance mit einer CLI
- Verwendung von GetOnPremisesInstance mit einer CLI
- Verwendung von ListApplicationRevisions mit einer CLI
- Verwendung von ListApplications mit einer CLI
- Verwendung von ListDeploymentConfigs mit einer CLI
- Verwendung von ListDeploymentGroups mit einer CLI
- Verwendung von ListDeploymentInstances mit einer CLI
- Verwendung von ListDeployments mit einer CLI
- Verwendung von ListOnPremisesInstances mit einer CLI
- Verwendung von RegisterApplicationRevision mit einer CLI
- Verwendung von RegisterOnPremisesInstance mit einer CLI
- Verwendung von RemoveTagsFromOnPremisesInstances mit einer CLI
- Verwendung von StopDeployment mit einer CLI
- Verwendung von UpdateApplication mit einer CLI
- Verwendung von UpdateDeploymentGroup mit einer CLI

Grundlegende Beispiele für die Verwendung CodeDeploy AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CodeDeploy mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung CodeDeploy AWS SDKs
  - Verwendung von AddTagsToOnPremisesInstances mit einer CLI
  - Verwendung von BatchGetApplications mit einer CLI
  - Verwendung von BatchGetDeployments mit einer CLI
  - Verwendung von BatchGetOnPremisesInstances mit einer CLI
  - Verwendung von CreateApplication mit einer CLI
  - Verwendung von CreateDeployment mit einer CLI
  - Verwendung von CreateDeploymentConfig mit einer CLI
  - Verwendung von CreateDeploymentGroup mit einer CLI
  - Verwendung von DeleteApplication mit einer CLI
  - Verwendung von DeleteDeploymentConfig mit einer CLI
  - Verwendung von DeleteDeploymentGroup mit einer CLI
  - Verwendung von DeregisterOnPremisesInstance mit einer CLI
  - Verwendung von GetApplication mit einer CLI
  - Verwendung von GetApplicationRevision mit einer CLI
  - Verwendung von GetDeployment mit einer CLI
  - Verwendung von GetDeploymentConfig mit einer CLI
  - Verwendung von GetDeploymentGroup mit einer CLI
  - Verwendung von GetDeploymentInstance mit einer CLI
  - Verwendung von GetOnPremisesInstance mit einer CLI
  - Verwendung von ListApplicationRevisions mit einer CLI
  - Verwendung von ListApplications mit einer CLI
  - Verwendung von ListDeploymentConfigs mit einer CLI
  - Verwendung von ListDeploymentGroups mit einer CLI
  - Verwendung von ListDeploymentInstances mit einer CLI
  - Verwendung von ListDeployments mit einer CLI

- Verwendung von ListOnPremisesInstances mit einer CLI
- Verwendung von RegisterApplicationRevision mit einer CLI
- Verwendung von RegisterOnPremisesInstance mit einer CLI
- Verwendung von RemoveTagsFromOnPremisesInstances mit einer CLI
- Verwendung von StopDeployment mit einer CLI
- Verwendung von UpdateApplication mit einer CLI
- Verwendung von UpdateDeploymentGroup mit einer CLI

Aktionen zur Verwendung CodeDeploy AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CodeDeploy Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CodeDeploy -API-Referenz.

Beispiele

- Verwendung von AddTagsToOnPremisesInstances mit einer CLI
- Verwendung von BatchGetApplications mit einer CLI
- Verwendung von BatchGetDeployments mit einer CLI
- Verwendung von BatchGetOnPremisesInstances mit einer CLI
- Verwendung von CreateApplication mit einer CLI
- Verwendung von CreateDeployment mit einer CLI
- Verwendung von CreateDeploymentConfig mit einer CLI
- Verwendung von CreateDeploymentGroup mit einer CLI
- Verwendung von DeleteApplication mit einer CLI
- Verwendung von DeleteDeploymentConfig mit einer CLI
- Verwendung von DeleteDeploymentGroup mit einer CLI
- Verwendung von DeregisterOnPremisesInstance mit einer CLI
- Verwendung von GetApplication mit einer CLI
- Verwendung von GetApplicationRevision mit einer CLI
- Verwendung von GetDeployment mit einer CLI

- Verwendung von GetDeploymentConfig mit einer CLI
- Verwendung von GetDeploymentGroup mit einer CLI
- Verwendung von GetDeploymentInstance mit einer CLI
- Verwendung von GetOnPremisesInstance mit einer CLI
- Verwendung von ListApplicationRevisions mit einer CLI
- Verwendung von ListApplications mit einer CLI
- Verwendung von ListDeploymentConfigs mit einer CLI
- Verwendung von ListDeploymentGroups mit einer CLI
- Verwendung von ListDeploymentInstances mit einer CLI
- Verwendung von ListDeployments mit einer CLI
- Verwendung von ListOnPremisesInstances mit einer CLI
- Verwendung von RegisterApplicationRevision mit einer CLI
- Verwendung von RegisterOnPremisesInstance mit einer CLI
- Verwendung von RemoveTagsFromOnPremisesInstances mit einer CLI
- Verwendung von StopDeployment mit einer CLI
- Verwendung von UpdateApplication mit einer CLI
- Verwendung von UpdateDeploymentGroup mit einer CLI

Verwendung von **`AddTagsToOnPremisesInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddTagsToOnPremisesInstances` verwendet wird.

CLI

AWS CLI

So fügen Sie Tags zu On-Premises-Instances hinzu

Im folgenden `add-tags-to-on-premises-instances` Beispiel wird AWS CodeDeploy dasselbe lokale Instanz-Tag zwei lokalen Instanzen zugeordnet. Die lokalen Instanzen werden nicht bei registriert. AWS CodeDeploy

```
aws deploy add-tags-to-on-premises-instances \
    --instance-names AssetTag12010298EX AssetTag23121309EX \
    --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie AddTagsToOnPremisesInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Tag einer On-Premises-Instance mit dem angegebenen Schlüssel und Wert für die angegebene On-Premises-Instance hinzugefügt.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter AddTagsToOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Tag einer On-Premises-Instance mit dem angegebenen Schlüssel und Wert für die angegebene On-Premises-Instance hinzugefügt.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter AddTagsToOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`BatchGetApplications`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetApplications` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu mehreren Anwendungen ab

Im folgenden `batch-get-applications` Beispiel werden Informationen zu mehreren Anwendungen angezeigt, die dem AWS Konto des Benutzers zugeordnet sind.

```
aws deploy batch-get-applications --application-names WordPress_App MyOther_App
```

Ausgabe:

```
{
    "applicationsInfo": [
        {
            "applicationName": "WordPress_App",
            "applicationId": "d9dd6993-f171-44fa-a811-211e4EXAMPLE",
            "createTime": 1407878168.078,
            "linkedToGitHub": false
        },
        {
            "applicationName": "MyOther_App",
            "applicationId": "8ca57519-31da-42b2-9194-8bb16EXAMPLE",
            "createTime": 1407453571.63,
            "linkedToGitHub": false
        }
    ]
}
```

- Einzelheiten zur API finden Sie BatchGetApplicationsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Anwendungen abgerufen.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName              CreateTime
            LinkedToGitHub
-------------                           ---------------              ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

```
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Einzelheiten zur API finden Sie unter BatchGetApplications AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Anwendungen abgerufen.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName              CreateTime
            LinkedToGitHub
-------------                           ---------------              ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- Einzelheiten zur API finden Sie unter BatchGetApplications AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`BatchGetDeployments`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetDeployments` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu mehreren Bereitstellungen ab

Im folgenden `batch-get-deployments` Beispiel werden Informationen zu mehreren Bereitstellungen angezeigt, die dem AWS Konto des Benutzers zugeordnet sind.

```
aws deploy batch-get-deployments --deployment-ids d-A1B2C3111 d-A1B2C3222
```

Ausgabe:

```
{
    "deploymentsInfo": [
        {
            "applicationName": "WordPress_App",
            "status": "Failed",
            "deploymentOverview": {
                "Failed": 0,
                "InProgress": 0,
                "Skipped": 0,
                "Succeeded": 1,
                "Pending": 0
            },
            "deploymentConfigName": "CodeDeployDefault.OneAtATime",
            "creator": "user",
            "deploymentGroupName": "WordPress_DG",
            "revision": {
                "revisionType": "S3",
                "s3Location": {
                "bundleType": "zip",
                "version": "uTecLusEXAMPLEFXtfUcyfV8bEXAMPLE",
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPressApp.zip"
                }
            },
            "deploymentId": "d-A1B2C3111",
            "createTime": 1408480721.9,
            "completeTime": 1408480741.822
        },
        {
            "applicationName": "MyOther_App",
            "status": "Failed",
            "deploymentOverview": {
                "Failed": 1,
                "InProgress": 0,
                "Skipped": 0,
                "Succeeded": 0,
                "Pending": 0
            },
            "deploymentConfigName": "CodeDeployDefault.OneAtATime",
```

```
            "creator": "user",
            "errorInformation": {
                "message": "Deployment failed: Constraint default violated: No
 hosts succeeded.",
                "code": "HEALTH_CONSTRAINTS"
            },
            "deploymentGroupName": "MyOther_DG",
            "revision": {
                "revisionType": "S3",
                "s3Location": {
                "bundleType": "zip",
                "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",
                "bucket": "amzn-s3-demo-bucket",
                "key": "MyOtherApp.zip"
                }
            },
            "deploymentId": "d-A1B2C3222",
            "createTime": 1409764576.589,
            "completeTime": 1409764596.101
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter BatchGetDeployments AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Bereitstellungen abgerufen.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Ausgabe:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
```

```
DeploymentId                   : d-QZMRGSTEX
DeploymentOverview             : Amazon.CodeDeploy.Model.DeploymentOverview
Description                    :
ErrorInformation               :
IgnoreApplicationStopFailures  : False
Revision                       : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                      : 1/1/0001 12:00:00 AM
Status                         : Succeeded

ApplicationName                : CodePipelineDemoApplication
CompleteTime                   : 7/23/2015 6:07:30 PM
CreateTime                     : 7/23/2015 6:06:29 PM
Creator                        : user
DeploymentConfigName           : CodeDeployDefault.OneAtATime
DeploymentGroupName            : CodePipelineDemoFleet
DeploymentId                   : d-RR0T5KTEX
DeploymentOverview             : Amazon.CodeDeploy.Model.DeploymentOverview
Description                    :
ErrorInformation               :
IgnoreApplicationStopFailures  : False
Revision                       : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                      : 1/1/0001 12:00:00 AM
Status                         : Succeeded
```

- Einzelheiten zur API finden Sie unter BatchGetDeployments AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Bereitstellungen abgerufen.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Ausgabe:

```
ApplicationName                : CodeDeployDemoApplication
CompleteTime                   : 7/23/2015 11:26:04 PM
CreateTime                     : 7/23/2015 11:24:43 PM
Creator                        : user
DeploymentConfigName           : CodeDeployDefault.OneAtATime
DeploymentGroupName            : CodeDeployDemoFleet
DeploymentId                   : d-QZMRGSTEX
```

```
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded

ApplicationName               : CodePipelineDemoApplication
CompleteTime                  : 7/23/2015 6:07:30 PM
CreateTime                    : 7/23/2015 6:06:29 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodePipelineDemoFleet
DeploymentId                  : d-RR0T5KTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

- Einzelheiten zur API finden Sie unter BatchGetDeployments AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`BatchGetOnPremisesInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetOnPremisesInstances` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einzelnen oder mehreren On-Premises-Instances ab

Im folgenden Beispiel für `batch-get-on-premises-instances` werden Informationen zu zwei On-Premises-Instances abgerufen.

```
aws deploy batch-get-on-premises-instances --instance-
names AssetTag12010298EX AssetTag23121309EX
```

Ausgabe:

```
{
    "instanceInfos": [
        {
            "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag12010298EX",
            "tags": [
                {
                    "Value": "CodeDeployDemo-OnPrem",
                    "Key": "Name"
                }
            ],
            "instanceName": "AssetTag12010298EX",
            "registerTime": 1425579465.228,
            "instanceArn": "arn:aws:codedeploy:us-west-2:123456789012:instance/
AssetTag12010298EX_4IwLNI2Alh"
        },
        {
            "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag23121309EX",
            "tags": [
                {
                    "Value": "CodeDeployDemo-OnPrem",
                    "Key": "Name"
                }
            ],
            "instanceName": "AssetTag23121309EX",
            "registerTime": 1425595585.988,
            "instanceArn": "arn:aws:codedeploy:us-west-2:80398EXAMPLE:instance/
AssetTag23121309EX_PomUy64Was"
        }
    ]
}
```

- Einzelheiten zur API finden Sie BatchGetOnPremisesInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX,
 AssetTag12010298EX-2
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Einzelheiten zur API finden Sie unter BatchGetOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX,
 AssetTag12010298EX-2
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
```

```
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Einzelheiten zur API finden Sie unter BatchGetOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateApplication`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateApplication` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Anwendung

Im folgenden `create-application` Beispiel wird eine Anwendung erstellt und sie dem AWS Konto des Benutzers zugeordnet.

```
aws deploy create-application --application-name MyOther_App
```

Ausgabe:

```
{
    "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
}
```

- Einzelheiten zur API finden Sie CreateApplicationunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Anwendung mit dem angegebenen Namen erstellt.

```
New-CDApplication -ApplicationName MyNewApplication
```

Ausgabe:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateApplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Anwendung mit dem angegebenen Namen erstellt.

```
New-CDApplication -ApplicationName MyNewApplication
```

Ausgabe:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateApplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateDeployment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDeployment` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine CodeDeploy Bereitstellung mithilfe der EC2/On-premises Rechenplattform

Das folgende `create-deployment` Beispiel erstellt eine Bereitstellung und ordnet sie dem AWS Konto des Benutzers zu.

```
aws deploy create-deployment \
    --application-name WordPress_App \
    --deployment-config-name CodeDeployDefault.OneAtATime \
    --deployment-group-name WordPress_DG \
    --description "My demo deployment" \
    --s3-location bucket=amzn-s3-demo-
bucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPressApp.zip
```

Ausgabe:

```
{
    "deploymentId": "d-A1B2C3111"
}
```

Beispiel 2: So erstellen Sie eine CodeDeploy Bereitstellung mit der Amazon ECS-Rechenplattform

Im folgenden Beispiel für `create-deployment` werden die folgenden zwei Dateien verwendet, um einen Amazon-ECS-Service bereitzustellen.

Inhalt der `create-deployment.json`-Datei:

```
{
    "applicationName": "ecs-deployment",
    "deploymentGroupName": "ecs-deployment-dg",
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bucket": "ecs-deployment-bucket",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

Diese Datei wiederum ruft die folgende `appspec.yaml`-Datei aus einem S3-Bucket mit dem Namen `ecs-deployment-bucket` ab.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:region:123456789012:task-definition/ecs-
task-def:2"
        LoadBalancerInfo:
          ContainerName: "sample-app"
          ContainerPort: 80
        PlatformVersion: "LATEST"
```

Befehl:

```
aws deploy create-deployment \
    --cli-input-json file://create-deployment.json \
    --region us-east-1
```

Ausgabe:

```
{
    "deploymentId": "d-1234ABCDE"
}
```

Weitere Informationen finden Sie unter CreateDeployment in der AWS CodeDeploy -API-Referenz.

- Einzelheiten zur API finden Sie CreateDeploymentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellung für die angegebene Anwendung und Bereitstellungsgruppe mit der angegebenen Bereitstellungskonfiguration und Anwendungsversion erstellt.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup -
```

```
IgnoreApplicationStopFailures $True -S3Location_Key aws-codedeploy_linux-
master.zip -RevisionType S3
```

Ausgabe:

```
d-ZHROG7UEX
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup
 -IgnoreApplicationStopFailures $True -S3Location_Key aws-
codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Ausgabe:

```
d-ZHROG7UEX
```

- Einzelheiten zur API finden Sie unter CreateDeployment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellung für die angegebene Anwendung und Bereitstellungsgruppe mit der angegebenen Bereitstellungskonfiguration und Anwendungsversion erstellt.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup -
IgnoreApplicationStopFailures $True -S3Location_Key aws-codedeploy_linux-
master.zip -RevisionType S3
```

Ausgabe:

```
d-ZHROG7UEX
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -DeploymentConfigName
 CodeDeployDefault.OneAtATime -DeploymentGroupName MyNewDeploymentGroup
 -IgnoreApplicationStopFailures $True -S3Location_Key aws-
codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Ausgabe:

```
d-ZHROG7UEX
```

- Einzelheiten zur API finden Sie unter CreateDeployment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateDeploymentConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDeploymentConfig` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine benutzerdefinierte Bereitstellungskonfiguration

Im folgenden `create-deployment-config` Beispiel wird eine benutzerdefinierte Bereitstellungskonfiguration erstellt und sie dem AWS Benutzerkonto zugeordnet.

```
aws deploy create-deployment-config \
    --deployment-config-name ThreeQuartersHealthy \
    --minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

Ausgabe:

```
{
    "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
```

```
}
```

- Einzelheiten zur API finden Sie CreateDeploymentConfigunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellungskonfiguration mit dem angegebenen Namen und Verhalten erstellt.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Ausgabe:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellungskonfiguration mit dem angegebenen Namen und Verhalten erstellt.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Ausgabe:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateDeploymentGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDeploymentGroup` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Bereitstellungsgruppe

Im folgenden `create-deployment-group` Beispiel wird eine Bereitstellungsgruppe erstellt und sie der angegebenen Anwendung und dem AWS Benutzerkonto zugeordnet.

```
aws deploy create-deployment-group \
    --application-name WordPress_App \
    --auto-scaling-groups CodeDeployDemo-ASG \
    --deployment-config-name CodeDeployDefault.OneAtATime \
    --deployment-group-name WordPress_DG \
    --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
    --service-role-arn arn:aws:iam::123456789012:role/CodeDeployDemoRole
```

Ausgabe:

```
{
    "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
}
```

- Einzelheiten zur API finden Sie CreateDeploymentGroupunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Bereitstellungsgruppe mit dem angegebenen Namen, der Auto-Scaling-Gruppe, der Bereitstellungskonfiguration, dem Tag und der Servicerolle für die angegebene Anwendung erstellt.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Bereitstellungsgruppe mit dem angegebenen Namen, der Auto-Scaling-Gruppe, der Bereitstellungskonfiguration, dem Tag und der Servicerolle für die angegebene Anwendung erstellt.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- Einzelheiten zur API finden Sie unter CreateDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteApplication`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteApplication` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Anwendung

Im folgenden `delete-application` Beispiel wird die angegebene Anwendung gelöscht, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy delete-application --application-name WordPress_App
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter DeleteApplication AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Anwendung mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Anwendung ohne Prompt zu löschen.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Einzelheiten zur API finden Sie unter DeleteApplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Anwendung mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Anwendung ohne Prompt zu löschen.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- Einzelheiten zur API finden Sie unter DeleteApplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteDeploymentConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDeploymentConfig` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Bereitstellungskonfiguration

Im folgenden `delete-deployment-config` Beispiel wird eine benutzerdefinierte Bereitstellungskonfiguration gelöscht, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter DeleteDeploymentConfig AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Bereitstellungskonfiguration mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt

wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungskonfiguration ohne Prompt zu löschen.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Bereitstellungskonfiguration mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungskonfiguration ohne Prompt zu löschen.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteDeploymentGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDeploymentGroup` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Bereitstellungsgruppe

Im folgenden Beispiel für `delete-deployment-group` wird eine Bereitstellungsgruppe gelöscht, die der angegebenen Anwendung zugeordnet ist.

```
aws deploy delete-deployment-group \
    --application-name WordPress_App \
    --deployment-group-name WordPress_DG
```

Ausgabe:

```
{
    "hooksNotCleanedUp": []
```

```
}
```

- Einzelheiten zur API finden Sie DeleteDeploymentGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Bereitstellungsgruppe mit dem angegebenen Namen für die spezifizierte Anwendung gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungsgruppe ohne Prompt zu löschen.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Bereitstellungsgruppe mit dem angegebenen Namen für die spezifizierte Anwendung gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungsgruppe ohne Prompt zu löschen.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- Einzelheiten zur API finden Sie unter DeleteDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterOnPremisesInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterOnPremisesInstance` verwendet wird.

CLI

AWS CLI

So heben Sie die Registrierung einer On-Premises-Instance auf

Im folgenden `deregister-on-premises-instance` Beispiel wird die Registrierung einer lokalen Instance aufgehoben AWS CodeDeploy, der mit der Instance verknüpfte IAM-Benutzer wird jedoch nicht gelöscht, und es wird auch nicht die Zuordnung in den lokalen Instance-Tags zu AWS CodeDeploy der Instance aufgehoben. Außerdem wird weder der AWS CodeDeploy Agent von der Instanz deinstalliert noch die lokale Konfigurationsdatei aus der Instanz entfernt.

```
aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter DeregisterOnPremisesInstance AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Registrierung der On-Premises-Instance mit dem angegebenen Namen aufgehoben.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter DeregisterOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung der On-Premises-Instance mit dem angegebenen Namen aufgehoben.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter DeregisterOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetApplication`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetApplication` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Anwendung ab

Im folgenden `get-application` Beispiel werden Informationen zu einer Anwendung angezeigt, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy get-application --application-name WordPress_App
```

Ausgabe:

```
{
    "application": {
        "applicationName": "WordPress_App",
        "applicationId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "createTime": 1407878168.078,
        "linkedToGitHub": false
    }
}
```

- Einzelheiten zur API finden Sie GetApplicationunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendung abgerufen.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName             CreateTime
            LinkedToGitHub
-------------                           ---------------             ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication   7/20/2015
 9:49:48 PM    False
```

- Einzelheiten zur API finden Sie unter GetApplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendung abgerufen.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName              CreateTime
            LinkedToGitHub
-------------                           ---------------              ----------
            --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- Einzelheiten zur API finden Sie unter GetApplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetApplicationRevision`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetApplicationRevision` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Anwendungsrevision ab

Im folgenden Beispiel für `get-application-revision` werden Informationen zu einer Anwendungsrevision angezeigt, die der angegebenen Anwendung zugeordnet ist.

```
aws deploy get-application-revision \
    --application-name WordPress_App \
    --s3-location bucket=amzn-s3-demo-
bucket,bundleType=zip,eTag=dd56cfdEXAMPLE8e768f9d77fEXAMPLE,key=WordPressApp.zip
```

Ausgabe:

```
{
    "applicationName": "WordPress_App",
    "revisionInfo": {
        "description": "Application revision registered by Deployment ID: d-
A1B2C3111",
        "registerTime": 1411076520.009,
        "deploymentGroups": "WordPress_DG",
        "lastUsedTime": 1411076520.009,
        "firstUsedTime": 1411076520.009
    },
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bundleType": "zip",
            "eTag": "dd56cfdEXAMPLE8e768f9d77fEXAMPLE",
            "bucket": "amzn-s3-demo-bucket",
            "key": "WordPressApp.zip"
        }
    }
}
```

- Einzelheiten zur API finden Sie GetApplicationRevisionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendungsrevision abgerufen.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Ausgabe:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Einzelheiten zur API finden Sie unter GetApplicationRevision AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendungsrevision abgerufen.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Ausgabe:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- Einzelheiten zur API finden Sie unter GetApplicationRevision AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDeployment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDeployment` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Bereitstellung ab

Im folgenden `get-deployment` Beispiel werden Informationen zu einer Bereitstellung angezeigt, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy get-deployment --deployment-id d-A1B2C3123
```

Ausgabe:

```
{
```

```
    "deploymentInfo": {
        "applicationName": "WordPress_App",
        "status": "Succeeded",
        "deploymentOverview": {
            "Failed": 0,
            "InProgress": 0,
            "Skipped": 0,
            "Succeeded": 1,
            "Pending": 0
        },
        "deploymentConfigName": "CodeDeployDefault.OneAtATime",
        "creator": "user",
        "description": "My WordPress app deployment",
        "revision": {
            "revisionType": "S3",
            "s3Location":  {
            "bundleType": "zip",
            "eTag": "\"dd56cfdEXAMPLE8e768f9d77fEXAMPLE\"",
            "bucket": "amzn-s3-demo-bucket",
            "key": "WordPressApp.zip"
            }
        },
        "deploymentId": "d-A1B2C3123",
        "deploymentGroupName": "WordPress_DG",
        "createTime": 1409764576.589,
        "completeTime": 1409764596.101,
        "ignoreApplicationStopFailures": false
    }
}
```

- Einzelheiten zur API finden Sie GetDeploymentunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellung abgerufen.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Beispiel 2: In diesem Beispiel werden Informationen zum Status der Instances abgerufen, die an der angegebenen Bereitstellung beteiligt sind.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Ausgabe:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Beispiel 3: Dieses Beispiel ruft Informationen zur Anwendungsrevision für die angegebene Bereitstellung ab.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Ausgabe:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Einzelheiten zur API finden Sie unter GetDeployment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellung abgerufen.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Beispiel 2: In diesem Beispiel werden Informationen zum Status der Instances abgerufen, die an der angegebenen Bereitstellung beteiligt sind.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Ausgabe:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Beispiel 3: Dieses Beispiel ruft Informationen zur Anwendungsrevision für die angegebene Bereitstellung ab.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Ausgabe:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- Einzelheiten zur API finden Sie unter GetDeployment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDeploymentConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDeploymentConfig` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Bereitstellungskonfiguration ab

Im folgenden `get-deployment-config` Beispiel werden Informationen zu einer Bereitstellungskonfiguration angezeigt, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy get-deployment-config --deployment-config-name ThreeQuartersHealthy
```

Ausgabe:

```
{
    "deploymentConfigInfo": {
        "deploymentConfigId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "minimumHealthyHosts": {
            "type": "FLEET_PERCENT",
            "value": 75
        },
        "createTime": 1411081164.379,
        "deploymentConfigName": "ThreeQuartersHealthy"
    }
}
```

- Einzelheiten zur API finden Sie GetDeploymentConfigunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellungskonfiguration abgerufen.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Ausgabe:

```
CreateTime              DeploymentConfigId
 DeploymentConfigName    MinimumHealthyHosts
----------              ------------------
 --------------------    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE
 ThreeQuartersHealthy    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Beispiel 2: In diesem Beispiel werden Informationen zur Definition der angegebenen Bereitstellungskonfiguration abgerufen.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Ausgabe:

```
Type            Value
----            -----
FLEET_PERCENT   75
```

- Einzelheiten zur API finden Sie unter GetDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellungskonfiguration abgerufen.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Ausgabe:

```
CreateTime              DeployментConfigId
 DeploymentConfigName    MinimumHealthyHosts
----------              -----------------
 --------------------    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE
 ThreeQuartersHealthy    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Beispiel 2: In diesem Beispiel werden Informationen zur Definition der angegebenen Bereitstellungskonfiguration abgerufen.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Ausgabe:

```
Type            Value
----            -----
FLEET_PERCENT   75
```

- Einzelheiten zur API finden Sie unter GetDeploymentConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDeploymentGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDeploymentGroup` verwendet wird.

CLI

AWS CLI

So zeigen Sie Informationen zu einer Bereitstellungsgruppe an

Im folgenden Beispiel für `get-deployment-group` werden Informationen zu einer Bereitstellungsgruppe angezeigt, die der angegebenen Anwendung zugeordnet ist.

```
aws deploy get-deployment-group \
    --application-name WordPress_App \
    --deployment-group-name WordPress_DG
```

Ausgabe:

```
{
    "deploymentGroupInfo": {
        "applicationName": "WordPress_App",
        "autoScalingGroups": [
            "CodeDeployDemo-ASG"
        ],
        "deploymentConfigName": "CodeDeployDefault.OneAtATime",
        "ec2TagFilters": [
            {
                "Type": "KEY_AND_VALUE",
                "Value": "CodeDeployDemo",
                "Key": "Name"
            }
        ],
        "deploymentGroupId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
        "serviceRoleArn": "arn:aws:iam::123456789012:role/CodeDeployDemoRole",
        "deploymentGroupName": "WordPress_DG"
    }
}
```

- Einzelheiten zur API finden Sie GetDeploymentGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
```

```
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Einzelheiten zur API finden Sie unter GetDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- Einzelheiten zur API finden Sie unter GetDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDeploymentInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDeploymentInstance` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Bereitstellungs-Instance ab

Im folgenden Beispiel für `get-deployment-instance` werden Informationen zu einer Bereitstellungs-Instance angezeigt, die der angegebenen Anwendung zugeordnet ist.

```
aws deploy get-deployment-instance --deployment-id d-QA4G4F9EX --instance-
id i-902e9fEX
```

Ausgabe:

```
{
    "instanceSummary": {
        "instanceId": "arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-902e9fEX",
        "lifecycleEvents": [
            {
                "status": "Succeeded",
                "endTime": 1408480726.569,
                "startTime": 1408480726.437,
                "lifecycleEventName": "ApplicationStop"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480728.016,
                "startTime": 1408480727.665,
                "lifecycleEventName": "DownloadBundle"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480729.744,
                "startTime": 1408480729.125,
                "lifecycleEventName": "BeforeInstall"
            },
            {
                "status": "Succeeded",
                "endTime": 1408480730.979,
                "startTime": 1408480730.844,
                "lifecycleEventName": "Install"
            },
            {
                "status": "Failed",
                "endTime": 1408480732.603,
                "startTime": 1408480732.1,
                "lifecycleEventName": "AfterInstall"
            },
            {
```

```
                "status": "Skipped",
                "endTime": 1408480732.606,
                "lifecycleEventName": "ApplicationStart"
            },
            {
                "status": "Skipped",
                "endTime": 1408480732.606,
                "lifecycleEventName": "ValidateService"
            }
        ],
        "deploymentId": "d-QA4G4F9EX",
        "lastUpdatedAt": 1408480733.152,
        "status": "Failed"
    }
}
```

- Einzelheiten zur API finden Sie GetDeploymentInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Instance für die spezifizierte Bereitstellung abgerufen.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Ausgabe:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Einzelheiten zur API finden Sie unter GetDeploymentInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Instance für die spezifizierte Bereitstellung abgerufen.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Ausgabe:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- Einzelheiten zur API finden Sie unter GetDeploymentInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetOnPremisesInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetOnPremisesInstance` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer On-Premises-Instance ab

Im folgenden Beispiel für `get-on-premises-instance` werden Informationen zur angegebenen On-Premises-Instance abgerufen.

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

Ausgabe:

```
{
    "instanceInfo": {
    "iamUserArn": "arn:aws:iam::123456789012:user/AWS/CodeDeploy/
AssetTag12010298EX",
        "tags": [
        {
            "Value": "CodeDeployDemo-OnPrem",
            "Key": "Name"
        }
        ],
        "instanceName": "AssetTag12010298EX",
```

```
        "registerTime": 1425579465.228,
        "instanceArn": "arn:aws:codedeploy:us-east-1:123456789012:instance/
AssetTag12010298EX_4IwLNI2Alh"
    }
}
```

- Einzelheiten zur API finden Sie GetOnPremisesInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- Einzelheiten zur API finden Sie unter GetOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
```

```
InstanceArn     : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName    : AssetTag12010298EX
RegisterTime    : 4/3/2015 6:36:24 PM
Tags            : {Name}
```

- Einzelheiten zur API finden Sie unter GetOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListApplicationRevisions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListApplicationRevisions` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Anwendungsrevisionen ab

Im folgenden Beispiel für `list-application-revisions` werden Informationen zu allen Anwendungsrevisionen angezeigt, die der angegebenen Anwendung zugeordnet sind.

```
aws deploy list-application-revisions \
    --application-name WordPress_App \
    --s-3-bucket amzn-s3-demo-bucket \
    --deployed exclude \
    --s-3-key-prefix WordPress_ \
    --sort-by lastUsedTime \
    --sort-order descending
```

Ausgabe:

```
{
    "revisions": [
        {
            "revisionType": "S3",
            "s3Location": {
                "version": "uTecLusvCB_JqHFXtfUcyfV8bEXAMPLE",
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPress_App.zip",
                "bundleType": "zip"
            }
```

```
        },
        {
            "revisionType": "S3",
            "s3Location": {
                "version": "tMk.UxgDpMEVb7V187ZM6wVAWEXAMPLE",
                "bucket": "amzn-s3-demo-bucket",
                "key": "WordPress_App_2-0.zip",
                "bundleType": "zip"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListApplicationRevisionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu den verfügbaren Revisionen für die angegebene Anwendung abgerufen.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Ausgabe:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
```

```
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Einzelheiten zur API finden Sie unter ListApplicationRevisions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu den verfügbaren Revisionen für die angegebene Anwendung abgerufen.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Ausgabe:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- Einzelheiten zur API finden Sie unter ListApplicationRevisions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListApplications`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListApplications` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Anwendungen ab

Im folgenden `list-applications` Beispiel werden Informationen zu allen Anwendungen angezeigt, die dem AWS Konto des Benutzers zugeordnet sind.

```
aws deploy list-applications
```

Ausgabe:

```
{
    "applications": [
        "WordPress_App",
        "MyOther_App"
    ]
}
```

- Einzelheiten zur API finden Sie ListApplicationsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Anwendungen abgerufen.

```
Get-CDApplicationList
```

Ausgabe:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Einzelheiten zur API finden Sie unter ListApplications AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Anwendungen abgerufen.

```
Get-CDApplicationList
```

Ausgabe:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- Einzelheiten zur API finden Sie unter ListApplications AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDeploymentConfigs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDeploymentConfigs` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Bereitstellungskonfigurationen ab

Im folgenden `list-deployment-configs` Beispiel werden Informationen zu allen Bereitstellungskonfigurationen angezeigt, die dem AWS Konto des Benutzers zugeordnet sind.

```
aws deploy list-deployment-configs
```

Ausgabe:

```
{
    "deploymentConfigsList": [
        "ThreeQuartersHealthy",
        "CodeDeployDefault.AllAtOnce",
        "CodeDeployDefault.HalfAtATime",
        "CodeDeployDefault.OneAtATime"
    ]
}
```

- Einzelheiten zur API finden Sie ListDeploymentConfigsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Bereitstellungskonfigurationen abgerufen.

```
Get-CDDeploymentConfigList
```

Ausgabe:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Einzelheiten zur API finden Sie unter ListDeploymentConfigs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Bereitstellungskonfigurationen abgerufen.

```
Get-CDDeploymentConfigList
```

Ausgabe:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- Einzelheiten zur API finden Sie unter ListDeploymentConfigs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDeploymentGroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDeploymentGroups` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Bereitstellungsgruppen ab

Im folgenden Beispiel für `list-deployment-groups` werden Informationen zu allen Bereitstellungsgruppen angezeigt, die der angegebenen Anwendung zugeordnet sind.

```
aws deploy list-deployment-groups --application-name WordPress_App
```

Ausgabe:

```
{
    "applicationName": "WordPress_App",
    "deploymentGroups": [
        "WordPress_DG",
        "WordPress_Beta_DG"
    ]
}
```

- Einzelheiten zur API finden Sie ListDeploymentGroupsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungsgruppen für die angegebene Anwendung abgerufen.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Einzelheiten zur API finden Sie unter ListDeploymentGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungsgruppen für die angegebene Anwendung abgerufen.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- Einzelheiten zur API finden Sie unter ListDeploymentGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDeploymentInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDeploymentInstances` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Bereitstellungs-Instances ab

Im folgenden Beispiel für `list-deployment-instances` werden Informationen zu allen Bereitstellungs-Instances angezeigt, die der angegebenen Anwendung zugeordnet sind.

```
aws deploy list-deployment-instances \
    --deployment-id d-A1B2C3111 \
    --instance-status-filter Succeeded
```

Ausgabe:

```
{
```

```
    "instancesList": [
        "i-EXAMPLE11",
        "i-EXAMPLE22"
    ]
}
```

- Einzelheiten zur API finden Sie ListDeploymentInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von Instance-IDs für die angegebene Bereitstellung abgerufen.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Einzelheiten zur API finden Sie unter ListDeploymentInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Instance-IDs für die angegebene Bereitstellung abgerufen.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- Einzelheiten zur API finden Sie unter ListDeploymentInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDeployments`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDeployments` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu Bereitstellungen ab

Im folgenden Beispiel für `list-deployments` werden Informationen zu allen Bereitstellungen angezeigt, die der angegebenen Anwendung und Bereitstellungsgruppe zugeordnet sind.

```
aws deploy list-deployments \
    --application-name WordPress_App \
    --create-time-range start=2014-08-19T00:00:00,end=2014-08-20T00:00:00 \
    --deployment-group-name WordPress_DG \
    --include-only-statuses Failed
```

Ausgabe:

```
{
    "deployments": [
        "d-EXAMPLE11",
        "d-EXAMPLE22",
        "d-EXAMPLE33"
    ]
}
```

- Einzelheiten zur API finden Sie ListDeploymentsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungs-IDs für die angegebene Anwendung und Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Einzelheiten zur API finden Sie unter ListDeployments AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungs-IDs für die angegebene Anwendung und Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- Einzelheiten zur API finden Sie unter ListDeployments AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListOnPremisesInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListOnPremisesInstances` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einzelnen oder mehreren On-Premises-Instances ab

Im folgenden `list-on-premises-instances` Beispiel wird eine Liste verfügbarer lokaler Instanznamen für Instanzen abgerufen, die registriert sind AWS CodeDeploy und denen auch das angegebene lokale Instanz-Tag zugeordnet ist. AWS CodeDeploy

```
aws deploy list-on-premises-instances \
    --registration-status Registered \
    --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
```

Ausgabe:

```
{
    "instanceNames": [
        "AssetTag12010298EX"
    ]
}
```

- Einzelheiten zur API finden Sie unter Befehlsreferenz ListOnPremisesInstances.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren On-Premises-Instance-Namen abgerufen.

```
Get-CDOnPremiseInstanceList
```

Ausgabe:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Einzelheiten zur API finden Sie unter ListOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren On-Premises-Instance-Namen abgerufen.

```
Get-CDOnPremiseInstanceList
```

Ausgabe:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- Einzelheiten zur API finden Sie unter ListOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterApplicationRevision`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterApplicationRevision` verwendet wird.

CLI

AWS CLI

So registrieren Sie Informationen zu einer bereits hochgeladenen Anwendungsrevision

Das folgende `register-application-revision` Beispiel registriert Informationen über eine bereits hochgeladene Anwendungsrevision, die in Amazon S3 gespeichert ist, mit. AWS CodeDeploy

```
aws deploy register-application-revision \
    --application-name WordPress_App \
    --description "Revised WordPress application" \
    --s3-location bucket=amzn-s3-demo-
bucket,key=RevisedWordPressApp.zip,bundleType=zip,eTag=cecc9b8a08eac650a6e71fdb88EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter RegisterApplicationRevision AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Anwendungsrevision mit dem angegebenen Amazon-S3-Standort für die angegebene Anwendung registriert.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -
S3Location_Bucket amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key
 aws-codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Einzelheiten zur API finden Sie unter RegisterApplicationRevision AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Anwendungsrevision mit dem angegebenen Amazon-S3-Standort für die angegebene Anwendung registriert.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -
S3Location_Bucket amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key
 aws-codedeploy_linux-master.zip -Revision_RevisionType S3
```

- Einzelheiten zur API finden Sie unter RegisterApplicationRevision AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterOnPremisesInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterOnPremisesInstance` verwendet wird.

CLI

AWS CLI

So registrieren Sie eine On-Premises-Instance

Im folgenden `register-on-premises-instance` Beispiel wird eine lokale Instanz bei AWS CodeDeploy registriert. Es erstellt weder den angegebenen IAM-Benutzer noch ordnet es der AWS CodeDeploy registrierten Instanz in lokalen Instanzen Tags zu.

```
aws deploy register-on-premises-instance \
    --instance-name AssetTag12010298EX \
    --iam-user-arn arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser-OnPrem
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie RegisterOnPremisesInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine On-Premises-Instance mit dem angegebenen Namen und IAM-Benutzer registriert.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter RegisterOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine On-Premises-Instance mit dem angegebenen Namen und IAM-Benutzer registriert.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- Einzelheiten zur API finden Sie unter RegisterOnPremisesInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveTagsFromOnPremisesInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTagsFromOnPremisesInstances` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einzelnen oder mehreren On-Premises-Instances

Im folgenden `remove-tags-from-on-premises-instances` Beispiel wird die Zuordnung der angegebenen On-Premises-Tags zu lokalen AWS CodeDeploy Instanzen aufgehoben. Es wird weder die Registrierung der lokalen Instanzen in der Instanz aufgehoben noch der AWS CodeDeploy Agent von der Instanz deinstalliert AWS CodeDeploy, noch wird die lokale Konfigurationsdatei aus den Instanzen entfernt, noch werden die IAM-Benutzer gelöscht, die den Instanzen zugeordnet sind.

```
aws deploy remove-tags-from-on-premises-instances \
    --instance-names AssetTag12010298EX AssetTag23121309EX \
    --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. RemoveTagsFromOnPremisesInstancesAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Tag für die On-Premises-Instance mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Tag ohne Prompt zu löschen.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag für die On-Premises-Instance mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Tag ohne Prompt zu löschen.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromOnPremisesInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StopDeployment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StopDeployment` verwendet wird.

CLI

AWS CLI

So versuchen Sie, eine Bereitstellung zu beenden

Im folgenden `stop-deployment` Beispiel wird versucht, eine laufende Bereitstellung zu beenden, die dem AWS Konto des Benutzers zugeordnet ist.

aws deploy stop-deployment --deployment-id d-A1B2C3111

Ausgabe:

```
{
    "status": "Succeeded",
    "statusMessage": "No more commands will be scheduled for execution in the
 deployment instances"
}
```

- Einzelheiten zur API finden Sie unter StopDeployment AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird versucht, die Bereitstellung mit der angegebenen Bereitstellungs-ID zu beenden.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Ausgabe:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Einzelheiten zur API finden Sie unter StopDeployment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird versucht, die Bereitstellung mit der angegebenen Bereitstellungs-ID zu beenden.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Ausgabe:

```
Status     StatusMessage
------     -------------
```

```
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- Einzelheiten zur API finden Sie unter StopDeployment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateApplication`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateApplication` verwendet wird.

CLI

AWS CLI

So ändern Sie die Details einer Anwendung

Im folgenden `update-application` Beispiel wird der Name einer Anwendung geändert, die dem AWS Konto des Benutzers zugeordnet ist.

```
aws deploy update-application \
    --application-name WordPress_App \
    --new-application-name My_WordPress_App
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie UpdateApplicationunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Anwendung geändert.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Einzelheiten zur API finden Sie unter UpdateApplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Anwendung geändert.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- Einzelheiten zur API finden Sie unter UpdateApplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **UpdateDeploymentGroup** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateDeploymentGroup` verwendet wird.

CLI

AWS CLI

So ändern Sie Informationen zu einer Bereitstellungsgruppe

Im folgenden Beispiel für `update-deployment-group` werden die Einstellungen einer Bereitstellungsgruppe geändert, die der angegebenen Anwendung zugeordnet ist.

```
aws deploy update-deployment-group \
    --application-name WordPress_App \
    --auto-scaling-groups My_CodeDeployDemo_ASG \
    --current-deployment-group-name WordPress_DG \
    --deployment-config-name CodeDeployDefault.AllAtOnce \
    --ec2-tag-filters Key=Name,Type=KEY_AND_VALUE,Value=My_CodeDeployDemo \
    --new-deployment-group-name My_WordPress_DepGroup \
    --service-role-arn arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo-2
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie UpdateDeploymentGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Bereitstellungsgruppe für die angegebene Anwendung geändert.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication
 -CurrentDeploymentGroupName MyNewDeploymentGroup -
NewDeploymentGroupName MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

- Einzelheiten zur API finden Sie unter UpdateDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Bereitstellungsgruppe für die angegebene Anwendung geändert.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen werden kann.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication
 -CurrentDeploymentGroupName MyNewDeploymentGroup -
NewDeploymentGroupName MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{K
```

- Einzelheiten zur API finden Sie unter UpdateDeploymentGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Verwendung CodePipeline AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS CodePipeline mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- CodePipeline Benutzerhandbuch — Weitere Informationen zu CodePipeline.
- CodePipeline API-Referenz — Details zu allen verfügbaren CodePipeline Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung CodePipeline AWS SDKs
  - Aktionen zur Verwendung CodePipeline AWS SDKs
    - Verwendung von AcknowledgeJob mit einer CLI
    - Verwendung von CreateCustomActionType mit einer CLI
    - Verwendung von CreatePipeline mit einer CLI
    - Verwendung von DeleteCustomActionType mit einer CLI
    - Verwendung von DeletePipeline mit einer CLI
    - Verwendung von DisableStageTransition mit einer CLI
    - Verwendung von EnableStageTransition mit einer CLI
    - Verwendung von GetJobDetails mit einer CLI
    - Verwendung von GetPipeline mit einer CLI
    - Verwendung von GetPipelineState mit einer CLI
    - Verwendung von ListActionTypes mit einer CLI
    - Verwendung von ListPipelines mit einer CLI
    - Verwendung von PollForJobs mit einer CLI
    - Verwendung von StartPipelineExecution mit einer CLI

- Verwendung von UpdatePipeline mit einer CLI

Grundlegende Beispiele für die Verwendung CodePipeline AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS CodePipeline mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung CodePipeline AWS SDKs
  - Verwendung von AcknowledgeJob mit einer CLI
  - Verwendung von CreateCustomActionType mit einer CLI
  - Verwendung von CreatePipeline mit einer CLI
  - Verwendung von DeleteCustomActionType mit einer CLI
  - Verwendung von DeletePipeline mit einer CLI
  - Verwendung von DisableStageTransition mit einer CLI
  - Verwendung von EnableStageTransition mit einer CLI
  - Verwendung von GetJobDetails mit einer CLI
  - Verwendung von GetPipeline mit einer CLI
  - Verwendung von GetPipelineState mit einer CLI
  - Verwendung von ListActionTypes mit einer CLI
  - Verwendung von ListPipelines mit einer CLI
  - Verwendung von PollForJobs mit einer CLI
  - Verwendung von StartPipelineExecution mit einer CLI
  - Verwendung von UpdatePipeline mit einer CLI

Aktionen zur Verwendung CodePipeline AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne CodePipeline Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS CodePipeline -API-Referenz.

Beispiele

- Verwendung von AcknowledgeJob mit einer CLI
- Verwendung von CreateCustomActionType mit einer CLI
- Verwendung von CreatePipeline mit einer CLI
- Verwendung von DeleteCustomActionType mit einer CLI
- Verwendung von DeletePipeline mit einer CLI
- Verwendung von DisableStageTransition mit einer CLI
- Verwendung von EnableStageTransition mit einer CLI
- Verwendung von GetJobDetails mit einer CLI
- Verwendung von GetPipeline mit einer CLI
- Verwendung von GetPipelineState mit einer CLI
- Verwendung von ListActionTypes mit einer CLI
- Verwendung von ListPipelines mit einer CLI
- Verwendung von PollForJobs mit einer CLI
- Verwendung von StartPipelineExecution mit einer CLI
- Verwendung von UpdatePipeline mit einer CLI

Verwendung von **`AcknowledgeJob`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AcknowledgeJob` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einem angegebenen Auftrag ab

In diesem Beispiel werden Informationen zu einem angegebenen Auftrag zurückgegeben, einschließlich des Status dieses Jobs, falls vorhanden. Dieser Befehl wird nur für Auftragsworker und benutzerdefinierte Aktionen verwendet. Verwenden Sie „aws codepipeline poll-for-jobs“, um den nonce-Wert und die Job-ID zu ermitteln.

Befehl:

```
aws codepipeline acknowledge-job --job-id f4f4ff82-2d11-EXAMPLE --nonce 3
```

Ausgabe:

```
{
  "status": "InProgress"
}
```

- Einzelheiten zur API finden Sie AcknowledgeJobin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status eines bestimmten Auftrags abgerufen.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Ausgabe:

```
Value
-----
InProgress
```

- Einzelheiten zur API finden Sie unter AcknowledgeJob AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status eines bestimmten Auftrags abgerufen.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Ausgabe:

```
Value
-----
InProgress
```

- Einzelheiten zur API finden Sie unter AcknowledgeJob AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateCustomActionType`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCustomActionType` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine benutzerdefinierte Aktion

In diesem Beispiel wird eine benutzerdefinierte Aktion für die AWS CodePipeline Verwendung einer bereits erstellten JSON-Datei (hier benannt MyCustomAction.json) erstellt, die die Struktur der benutzerdefinierten Aktion enthält. Weitere Informationen zu den Anforderungen für die Erstellung einer benutzerdefinierten Aktion, einschließlich der Struktur der Datei, finden Sie im AWS CodePipeline Benutzerhandbuch.

```
aws codepipeline create-custom-action-type --cli-input-json file://
MyCustomAction.json
```

Inhalt der JSON-Datei `MyCustomAction.json`:

```
{
    "category": "Build",
    "provider": "MyJenkinsProviderName",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://192.0.2.4/job/{Config:ProjectName}/
lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "configurationProperties": [
        {
            "name": "MyJenkinsExampleBuildProject",
            "required": true,
            "key": true,
            "secret": false,
            "queryable": false,
            "description": "The name of the build project must be provided when
 this action is added to the pipeline.",
            "type": "String"
        }
    ],
```

```
    "inputArtifactDetails": {
        "maximumCount": 1,
        "minimumCount": 0
    },
    "outputArtifactDetails": {
        "maximumCount": 1,
        "minimumCount": 0
    }
}
```

Dieser Befehl gibt die Struktur der benutzerdefinierten Aktion zurück.

- Einzelheiten zur API finden Sie CreateCustomActionTypein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue benutzerdefinierte Aktion mit den angegebenen Eigenschaften erstellt.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is
 added to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Ausgabe:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Einzelheiten zur API finden Sie unter CreateCustomActionType AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue benutzerdefinierte Aktion mit den angegebenen Eigenschaften erstellt.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is
 added to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Ausgabe:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- Einzelheiten zur API finden Sie unter CreateCustomActionType AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreatePipeline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePipeline` verwendet wird.

CLI

AWS CLI

Um eine Pipeline zu erstellen

In diesem Beispiel wird eine Pipeline AWS CodePipeline unter Verwendung einer bereits erstellten JSON-Datei (hier benannt MySecondPipeline.json) erstellt, die die Struktur der Pipeline enthält. Weitere Informationen zu den Anforderungen für die Erstellung einer Pipeline, einschließlich der Struktur der Datei, finden Sie im AWS CodePipeline Benutzerhandbuch.

Befehl:

```
aws codepipeline create-pipeline --cli-input-json file://MySecondPipeline.json
```

Beispielinhalt für die JSON-Datei:

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "configuration": {
            "S3Bucket": "awscodepipeline-demo-bucket",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Beta",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
```

```
          "actionTypeId": {
            "category": "Deploy",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeDeploy"
          },
          "outputArtifacts": [],
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-1-11EXAMPLE11"
  },
  "name": "MySecondPipeline",
  "version": 1
 }
}
```

Ausgabe:

```
This command returns the structure of the pipeline.
```

- Einzelheiten zur API finden Sie CreatePipelinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Pipeline mit den angegebenen Einstellungen erstellt.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
```

```
$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Einzelheiten zur API finden Sie unter CreatePipeline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Pipeline mit den angegebenen Einstellungen erstellt.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"
```

```
$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- Einzelheiten zur API finden Sie unter CreatePipeline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteCustomActionType`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCustomActionType` verwendet wird.

CLI

AWS CLI

So löschen Sie eine benutzerdefinierte Aktion

In diesem Beispiel AWS CodePipeline wird eine benutzerdefinierte Aktion mithilfe einer bereits erstellten JSON-Datei (hier benannt DeleteMyCustomAction.json) gelöscht, die den Aktionstyp,

den Anbieternamen und die Versionsnummer der zu löschenden Aktion enthält. Verwenden Sie den Befehl „list-action-types“, um die richtigen Werte für Kategorie, Version und Anbieter anzuzeigen.

Befehl:

```
aws codepipeline delete-custom-action-type --cli-input-json file:// DeleteMyCustomAction.json
```

Beispielinhalt für die JSON-Datei:

```
{
  "category": "Build",
  "version": "1",
  "provider": "MyJenkinsProviderName"
}
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DeleteCustomActionTypein AWS CLI der Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene benutzerdefinierte Aktion gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die benutzerdefinierte Aktion ohne Prompt zu löschen.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version
 1
```

- Einzelheiten zur API finden Sie unter DeleteCustomActionType AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene benutzerdefinierte Aktion gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die benutzerdefinierte Aktion ohne Prompt zu löschen.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version
 1
```

- Einzelheiten zur API finden Sie unter DeleteCustomActionType AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeletePipeline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePipeline` verwendet wird.

CLI

AWS CLI

Um eine Pipeline zu löschen

In diesem Beispiel wird eine Pipeline mit dem Namen MySecondPipeline von AWS CodePipeline gelöscht. Verwenden Sie den Befehl list-pipelines, um eine Liste der Pipelines anzuzeigen, die Ihrem Konto zugeordnet sind. AWS

Befehl:

```
aws codepipeline delete-pipeline --name MySecondPipeline
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie unter DeletePipelineBefehlsreferenz.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Pipeline gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Pipeline ohne Prompt zu löschen.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter DeletePipeline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Pipeline gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Pipeline ohne Prompt zu löschen.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter DeletePipeline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DisableStageTransition`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisableStageTransition` verwendet wird.

CLI

AWS CLI

So deaktivieren Sie den Übergang zu einer Phase in einer Pipeline

In diesem Beispiel werden Übergänge in die Betaphase der MyFirstPipeline Pipeline in AWS CodePipeline deaktiviert.

Befehl:

```
aws codepipeline disable-stage-transition \
```

```
--pipeline-name MyFirstPipeline \
--stage-name Beta  \
--transition-type Inbound \
--reason "An example reason"
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DisableStageTransitionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline deaktiviert.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter DisableStageTransition AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline deaktiviert.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter DisableStageTransition AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EnableStageTransition`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableStageTransition` verwendet wird.

CLI

AWS CLI

So aktivieren Sie einen Übergang zu einer Phase in einer Pipeline

Dieses Beispiel ermöglicht Übergänge in die Beta-Phase der MyFirstPipeline Pipeline in AWS CodePipeline.

Befehl:

```
aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
name Beta  --transition-type Inbound
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie EnableStageTransitionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline aktiviert.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter EnableStageTransition AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline aktiviert.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- Einzelheiten zur API finden Sie unter EnableStageTransition AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetJobDetails`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetJobDetails` verwendet wird.

CLI

AWS CLI

So rufen Sie Details zu einem Auftrag ab

In diesem Beispiel werden Details zu einem Auftrag zurückgegeben, dessen ID durch „f4f4ff82-2d11-EXAMPLE“ dargestellt wird. Dieser Befehl wird nur für benutzerdefinierte Aktionen verwendet. Wenn dieser Befehl aufgerufen wird, werden temporäre Anmeldeinformationen für den Amazon S3 S3-Bucket AWS CodePipeline zurückgegeben, der zum Speichern von Artefakten für die Pipeline verwendet wird, falls dies für die benutzerdefinierte Aktion erforderlich ist. Dieser Befehl gibt auch alle geheimen Werte zurück, die für die Aktion definiert wurden, sofern welche definiert wurden.

Befehl:

```
aws codepipeline get-job-details --job-id f4f4ff82-2d11-EXAMPLE
```

Ausgabe:

```
{
 "jobDetails": {
  "accountId": "111111111111",
  "data": {
    "actionConfiguration": {
      "__type": "ActionConfiguration",
      "configuration": {
        "ProjectName": "MyJenkinsExampleTestProject"
      }
    },
    "actionTypeId": {
      "__type": "ActionTypeId",
      "category": "Test",
      "owner": "Custom",
```

```
      "provider": "MyJenkinsProviderName",
      "version": "1"
    },
    "artifactCredentials": {
      "__type": "AWSSessionCredentials",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
      "sessionToken":
 "fICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQH
+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQE
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8
    },
    "inputArtifacts": [
      {
        "__type": "Artifact",
        "location": {
          "s3Location": {
            "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
            "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
          },
          "type": "S3"
        },
        "name": "MyAppBuild"
      }
    ],
    "outputArtifacts": [],
    "pipelineContext": {
      "__type": "PipelineContext",
      "action": {
        "name": "MyJenkinsTest-Action"
      },
      "pipelineName": "MySecondPipeline",
      "stage": {
        "name": "Testing"
      }
    }
  },
  "id": "f4f4ff82-2d11-EXAMPLE"
 }
}
```

- Einzelheiten zur API finden Sie GetJobDetailsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zum angegebenen Auftrag abgerufen.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Ausgabe:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zum angegebenen Auftrag abgerufen.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
```

```
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Ausgabe:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Einzelheiten zur API finden Sie unter GetJobDetails AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zum angegebenen Auftrag abgerufen.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Ausgabe:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zum angegebenen Auftrag abgerufen.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Ausgabe:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- Einzelheiten zur API finden Sie unter GetJobDetails AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPipeline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPipeline` verwendet wird.

CLI

AWS CLI

So zeigen Sie die Struktur einer Pipeline an

In diesem Beispiel wird die Struktur einer Pipeline mit dem Namen zurückgegeben MyFirstPipeline.

Befehl:

```
aws codepipeline get-pipeline --name MyFirstPipeline
```

Ausgabe:

```
{
  "pipeline": {
      "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
      "stages": [
          {
              "name": "Source",
              "actions": [
                  {
                      "inputArtifacts": [],
                      "name": "Source",
                      "actionTypeId": {
                          "category": "Source",
                          "owner": "AWS",
                          "version": "1",
                          "provider": "S3"
                      },
                      "outputArtifacts": [
                          {
                              "name": "MyApp"
                          }
                      ],
                      "configuration": {
                          "S3Bucket": "awscodepipeline-demo-bucket",
                          "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                      },
                      "runOrder": 1
                  }
              ]
          },
          {
              "name": "Beta",
              "actions": [
                  {
                      "inputArtifacts": [
                          {
                              "name": "MyApp"
                          }
                      ],
                      "name": "CodePipelineDemoFleet",
                      "actionTypeId": {
                          "category": "Deploy",
```

```
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "CodePipelineDemoApplication",
                            "DeploymentGroupName": "CodePipelineDemoFleet"
                        },
                        "runOrder": 1
                    }
                ]
            }
        ],
        "artifactStore": {
            "type": "S3",
            "location": "codepipeline-us-east-1-11EXAMPLE11"
        },
        "name": "MyFirstPipeline",
        "version": 1
    }
}
```

- Einzelheiten zur API finden Sie GetPipelineunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zur angegebenen Pipeline abgerufen.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zur angegebenen Pipeline abgerufen.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Ausgabe:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
```

```
    Type = S3
  Stages:
    Name = Source
      Actions:
        Name = Source
          Category = Source
          Owner = ThirdParty
          Provider = GitHub
          Version = 1
          Configuration:
            Branch = master
            OAuthToken = ****
            Owner = my-user-name
            Repo = MyRepoName
          InputArtifacts:
            MyApp
          RunOrder = 1
    Name = Build
      Actions:
        Name = Build
          Category = Build
          Owner = Custom
          Provider = MyCustomProviderName
          Version = 1
          Configuration:
            ProjectName = MyProjectName
          InputArtifacts:
            MyApp
            MyAppBuild
          RunOrder = 1
    Name = Beta
      Actions:
        Name = CodePipelineDemoFleet
          Category = Deploy
          Owner = AWS
          Provider = CodeDeploy
          Version = 1
          Configuration:
            ApplicationName = CodePipelineDemoApplication
            DeploymentGroupName = CodePipelineDemoFleet
          InputArtifacts:
            MyAppBuild
          RunOrder = 1
    Name = TestStage
```

```
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Einzelheiten zur API finden Sie unter GetPipeline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zur angegebenen Pipeline abgerufen.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zur angegebenen Pipeline abgerufen.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
```

```
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Ausgabe:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
  Type = S3
Stages:
  Name = Source
    Actions:
      Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
```

```
          Owner = my-user-name
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- Einzelheiten zur API finden Sie unter GetPipeline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPipelineState`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPipelineState` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zum Status einer Pipeline ab

In diesem Beispiel wird der letzte Status einer Pipeline mit dem Namen zurückgegeben MyFirstPipeline.

Befehl:

```
aws codepipeline get-pipeline-state --name MyFirstPipeline
```

Ausgabe:

```
{
 "created": 1446137312.204,
 "pipelineName": "MyFirstPipeline",
 "pipelineVersion": 1,
 "stageStates": [
  {
    "actionStates": [
      {
        "actionName": "Source",
        "entityUrl": "https://console.aws.amazon.com/s3/home?#",
        "latestExecution": {
          "lastStatusChange": 1446137358.328,
          "status": "Succeeded"
        }
      }
    ],
    "stageName": "Source"
  },
  {
    "actionStates": [
```

```
      {
        "actionName": "CodePipelineDemoFleet",
        "entityUrl": "https://console.aws.amazon.com/codedeploy/
home?#/applications/CodePipelineDemoApplication/deployment-groups/
CodePipelineDemoFleet",
        "latestExecution": {
          "externalExecutionId": "d-EXAMPLE",
          "externalExecutionUrl": "https://console.aws.amazon.com/codedeploy/
home?#/deployments/d-EXAMPLE",
          "lastStatusChange": 1446137493.131,
          "status": "Succeeded",
          "summary": "Deployment Succeeded"
        }
      }
    ],
    "inboundTransitionState": {
      "enabled": true
    },
    "stageName": "Beta"
  }
 ],
 "updated": 1446137312.204
}
```

- Einzelheiten zur API finden Sie GetPipelineStateunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zu den Phasen der angegebenen Pipeline abgerufen.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Ausgabe:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen über den Status der angegebenen Pipeline abgerufen.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Ausgabe:

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time
 limit. If this is a custom action, make sure that the job worker is configured
 correctly.
```

```
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
```

```
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
      RevisionUrl =
```

- Einzelheiten zur API finden Sie unter GetPipelineState AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zu den Phasen der angegebenen Pipeline abgerufen.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Ausgabe:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen über den Status der angegebenen Pipeline abgerufen.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
```

```
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Ausgabe:

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
```

```
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time
 limit. If this is a custom action, make sure that the job worker is configured
 correctly.
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
```

```
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
```

```
        Summary = Finished
      RevisionUrl =
```

- Einzelheiten zur API finden Sie unter GetPipelineState AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListActionTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListActionTypes` verwendet wird.

CLI

AWS CLI

So zeigen Sie die verfügbaren Aktionstypen an

Der Befehl list-action-types wird von sich aus verwendet und gibt die Struktur aller Aktionen zurück, die für Ihr Konto verfügbar sind. AWS In diesem Beispiel wird die Option „--action-owner-filter“ verwendet, um nur benutzerdefinierte Aktionen zurückzugeben.

Befehl:

```
aws codepipeline list-action-types --action-owner-filter Custom
```

Ausgabe:

```
{
  "actionTypes": [
      {
          "inputArtifactDetails": {
              "maximumCount": 5,
              "minimumCount": 0
          },
          "actionConfigurationProperties": [
              {
                  "secret": false,
                  "required": true,
                  "name": "MyJenkinsExampleBuildProject",
                  "key": true,
                  "queryable": true
              }
          ],
```

```
            "outputArtifactDetails": {
                "maximumCount": 5,
                "minimumCount": 0
            },
            "id": {
                "category": "Build",
                "owner": "Custom",
                "version": "1",
                "provider": "MyJenkinsProviderName"
            },
            "settings": {
                "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
                "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
            }
        },
        {
            "inputArtifactDetails": {
                "maximumCount": 5,
                "minimumCount": 0
            },
            "actionConfigurationProperties": [
                {
                    "secret": false,
                    "required": true,
                    "name": "MyJenkinsExampleTestProject",
                    "key": true,
                    "queryable": true
                }
            ],
            "outputArtifactDetails": {
                "maximumCount": 5,
                "minimumCount": 0
            },
            "id": {
                "category": "Test",
                "owner": "Custom",
                "version": "1",
                "provider": "MyJenkinsProviderName"
            },
            "settings": {
                "entityUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}",
                "executionUrlTemplate": "http://192.0.2.4/job/{Config:ProjectName}/
{ExternalExecutionId}"
```

```
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz ListActionTypes.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über alle verfügbaren Aktionen für den angegebenen Besitzer abgerufen.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version =
 " + $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " +
 $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
```

```
}
```

Ausgabe:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
```

```
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/
home?#/applications/{Config:ApplicationName}/deployment-groups/
{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Einzelheiten zur API finden Sie unter ListActionTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle verfügbaren Aktionen für den angegebenen Besitzer abgerufen.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version =
 " + $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
```

```
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " +
 $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
}
```

Ausgabe:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
```

```
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/
home?#/applications/{Config:ApplicationName}/deployment-groups/
{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- Einzelheiten zur API finden Sie unter ListActionTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListPipelines`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListPipelines` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine Liste von Pipelines an

In diesem Beispiel werden alle AWS CodePipeline Pipelines aufgeführt, die dem AWS Konto des Benutzers zugeordnet sind.

Befehl:

```
aws codepipeline list-pipelines
```

Ausgabe:

```
{
  "pipelines": [
```

```
        {
            "updated": 1439504274.641,
            "version": 1,
            "name": "MyFirstPipeline",
            "created": 1439504274.641
        },
        {
            "updated": 1436461837.992,
            "version": 2,
            "name": "MySecondPipeline",
            "created": 1436460801.381
        }
        ]
}
```

- Einzelheiten zur API finden Sie unter ListPipelines AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Pipelines abgerufen.

```
Get-CPPipelineList
```

Ausgabe:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- Einzelheiten zur API finden Sie unter ListPipelines AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Pipelines abgerufen.

```
Get-CPPipelineList
```

Ausgabe:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- Einzelheiten zur API finden Sie unter ListPipelines AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PollForJobs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PollForJobs` verwendet wird.

CLI

AWS CLI

So zeigen Sie alle verfügbaren Jobs an

In diesem Beispiel werden Informationen zu allen Aufträgen zurückgegeben, auf die ein Auftragsworker reagieren kann. In diesem Beispiel wird eine vordefinierte JSON-Datei (MyActionTypeInfo.json) verwendet, um Informationen über den Aktionstyp bereitzustellen, für den der Job Worker Jobs verarbeitet. Dieser Befehl wird nur für benutzerdefinierte Aktionen verwendet. Wenn dieser Befehl aufgerufen wird, werden temporäre Anmeldeinformationen für den Amazon S3 S3-Bucket AWS CodePipeline zurückgegeben, der zum Speichern von Artefakten für die Pipeline verwendet wird. Dieser Befehl gibt auch alle geheimen Werte zurück, die für die Aktion definiert wurden, sofern welche definiert wurden.

Befehl:

```
aws codepipeline poll-for-jobs --cli-input-json file://MyActionTypeInfo.json
```

Beispielinhalt für die JSON-Datei:

```
{
  "actionTypeId": {
    "category": "Test",
    "owner": "Custom",
    "provider": "MyJenkinsProviderName",
    "version": "1"
  },
```

```
  "maxBatchSize": 5,
  "queryParam": {
      "ProjectName": "MyJenkinsTestProject"
  }
}
```

Ausgabe:

```
{
 "jobs": [
  {
    "accountId": "111111111111",
    "data": {
      "actionConfiguration": {
        "__type": "ActionConfiguration",
        "configuration": {
          "ProjectName": "MyJenkinsExampleTestProject"
        }
      },
      "actionTypeId": {
        "__type": "ActionTypeId",
        "category": "Test",
        "owner": "Custom",
        "provider": "MyJenkinsProviderName",
        "version": "1"
      },
      "artifactCredentials": {
        "__type": "AWSSessionCredentials",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "secretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "sessionToken":
 "fICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQH
+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQE
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8
      },
      "inputArtifacts": [
        {
          "__type": "Artifact",
          "location": {
            "s3Location": {
              "bucketName": "codepipeline-us-east-1-11EXAMPLE11",
```

```
                "objectKey": "MySecondPipeline/MyAppBuild/EXAMPLE"
              },
              "type": "S3"
            },
            "name": "MyAppBuild"
          }
        ],
        "outputArtifacts": [],
        "pipelineContext": {
          "__type": "PipelineContext",
          "action": {
            "name": "MyJenkinsTest-Action"
          },
          "pipelineName": "MySecondPipeline",
          "stage": {
            "name": "Testing"
          }
        }
      },
      "id": "ef66c259-64f9-EXAMPLE",
      "nonce": "3"
    }
  ]
}
```

- Einzelheiten zur API finden Sie PollForJobsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über alle Aufträge abgerufen, die für die angegebene Aktionskategorie, den Besitzer, den Anbieter, die Version und die Abfrageparameter ausgeführt werden können.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Ausgabe:

```
AccountId       Data                                Id
            Nonce
---------       ----                                --
            -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Einzelheiten zur API finden Sie unter PollForJobs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle Aufträge abgerufen, die für die angegebene Aktionskategorie, den Besitzer, den Anbieter, die Version und die Abfrageparameter ausgeführt werden können.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Ausgabe:

```
AccountId       Data                                Id
            Nonce
---------       ----                                --
            -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- Einzelheiten zur API finden Sie unter PollForJobs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StartPipelineExecution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StartPipelineExecution` verwendet wird.

CLI

AWS CLI

So führen Sie die neueste Revision über eine Pipeline aus

In diesem Beispiel wird die neueste Version, die sich in der Quellphase einer Pipeline befindet, über die Pipeline mit dem Namen "MyFirstPipeline" ausgeführt.

Befehl:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline
```

Ausgabe:

```
{
  "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE"
}
```

- Einzelheiten zur API finden Sie StartPipelineExecutionunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird mit der Ausführung der angegebenen Pipeline begonnen.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter StartPipelineExecution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird mit der Ausführung der angegebenen Pipeline begonnen.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- Einzelheiten zur API finden Sie unter StartPipelineExecution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdatePipeline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdatePipeline` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie die Struktur einer Pipeline

In diesem Beispiel wird der Befehl „update-pipeline“ mit dem Argument „--cli-input-json“ verwendet. In diesem Beispiel wird eine vordefinierte JSON-Datei (MyFirstPipeline.json) verwendet, um die Struktur einer Pipeline zu aktualisieren. AWS CodePipeline erkennt den in der JSON-Datei enthaltenen Pipelinenamen und wendet dann alle Änderungen an geänderten Feldern in der Pipeline-Struktur an, um die Pipeline zu aktualisieren.

Beachten Sie beim Erstellen der vordefinierten JSON-Datei die folgenden Richtlinien:

Wenn Sie mit einer Pipeline-Struktur arbeiten, die mit dem Befehl „get-pipeline“ abgerufen wurde, müssen Sie den Metadatenabschnitt aus der Pipeline-Struktur in der JSON-Datei entfernen (die Zeilen mit „metadata“: {} und die darin enthaltenen Felder „created“, „pipelineARN“ und „updated“). Der Pipeline-Name kann nicht geändert werden.

Befehl:

```
aws codepipeline update-pipeline --cli-input-json file://MyFirstPipeline.json
```

Inhalt einer JSON-Beispieldatei:

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
```

```
              "name": "MyApp"
            }
          ],
          "configuration": {
            "S3Bucket": "awscodepipeline-demo-bucket2",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Beta",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
          "actionTypeId": {
            "category": "Deploy",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeDeploy"
          },
          "outputArtifacts": [],
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "codepipeline-us-east-1-11EXAMPLE11"
  },
  "name": "MyFirstPipeline",
  "version": 1
}
```

```
}
```

Ausgabe:

```
{
 "pipeline": {
  "artifactStore": {
    "location": "codepipeline-us-east-1-11EXAMPLE11",
    "type": "S3"
  },
  "name": "MyFirstPipeline",
  "roleArn": "arn:aws:iam::111111111111:role/AWS-CodePipeline-Service",
  "stages": [
    {
      "actions": [
        {
          "actionTypeId": {
            "__type": "ActionTypeId",
            "category": "Source",
            "owner": "AWS",
            "provider": "S3",
            "version": "1"
          },
          "configuration": {
            "S3Bucket": "awscodepipeline-demo-bucket2",
            "S3ObjectKey": "aws-codepipeline-s3-aws-codedeploy_linux.zip"
          },
          "inputArtifacts": [],
          "name": "Source",
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "runOrder": 1
        }
      ],
      "name": "Source"
    },
    {
      "actions": [
        {
          "actionTypeId": {
```

```
            "__type": "ActionTypeId",
            "category": "Deploy",
            "owner": "AWS",
            "provider": "CodeDeploy",
            "version": "1"
          },
          "configuration": {
            "ApplicationName": "CodePipelineDemoApplication",
            "DeploymentGroupName": "CodePipelineDemoFleet"
          },
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "CodePipelineDemoFleet",
          "outputArtifacts": [],
          "runOrder": 1
        }
      ],
      "name": "Beta"
    }
  ],
  "version": 3
 }
}
```

- Einzelheiten zur API finden Sie UpdatePipelinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene vorhandene Pipeline mit den angegebenen Einstellungen aktualisiert.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
```

```
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
```

```
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Einzelheiten zur API finden Sie unter UpdatePipeline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene vorhandene Pipeline mit den angegebenen Einstellungen aktualisiert.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object
 Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName",
 "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
```

```
$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- Einzelheiten zur API finden Sie unter UpdatePipeline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Amazon Cognito Identity mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Cognito Identity mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch zu Amazon Cognito Identity – Weitere Informationen zu Amazon Cognito Identity.
- API-Referenz zu Amazon Cognito Identity – Details zu allen verfügbaren Aktionen von Amazon Cognito Identity.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Cognito Identity AWS SDKs
  - Aktionen für Amazon Cognito Identity mit AWS SDKs
    - Verwenden Sie CreateIdentityPoolmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteIdentityPoolmit einem AWS SDK oder CLI
    - Verwendung von DescribeIdentityPool mit einer CLI
    - Verwenden Sie mit einem GetCredentialsForIdentity AWS SDK
    - Verwendung von GetIdentityPoolRoles mit einer CLI
    - Verwenden Sie mit einem ListIdentityPools AWS SDK oder CLI
    - Verwendung von SetIdentityPoolRoles mit einer CLI
    - Verwendung von UpdateIdentityPool mit einer CLI
- Szenarien für die Verwendung von Amazon Cognito Identity AWS SDKs
  - Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Grundlegende Beispiele für die Verwendung von Amazon Cognito Identity AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Cognito Identity mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Cognito Identity mit AWS SDKs
  - Verwenden Sie CreateIdentityPoolmit einem AWS SDK oder CLI

- Verwenden Sie DeleteIdentityPoolmit einem AWS SDK oder CLI
- Verwendung von DescribeIdentityPool mit einer CLI
- Verwenden Sie mit einem GetCredentialsForIdentity AWS SDK
- Verwendung von GetIdentityPoolRoles mit einer CLI
- Verwenden Sie mit einem ListIdentityPools AWS SDK oder CLI
- Verwendung von SetIdentityPoolRoles mit einer CLI
- Verwendung von UpdateIdentityPool mit einer CLI

Aktionen für Amazon Cognito Identity mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Cognito Identity-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Cognito–Identity-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Cognito Identity AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Cognito Identity-API-Referenz.

Beispiele

- Verwenden Sie CreateIdentityPoolmit einem AWS SDK oder CLI
- Verwenden Sie DeleteIdentityPoolmit einem AWS SDK oder CLI
- Verwendung von DescribeIdentityPool mit einer CLI
- Verwenden Sie mit einem GetCredentialsForIdentity AWS SDK
- Verwendung von GetIdentityPoolRoles mit einer CLI
- Verwenden Sie mit einem ListIdentityPools AWS SDK oder CLI
- Verwendung von SetIdentityPoolRoles mit einer CLI
- Verwendung von UpdateIdentityPool mit einer CLI

Verwenden Sie **`CreateIdentityPool`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateIdentityPool` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Identitätspool mit dem Cognito-Identitätspool-Anbieter

In diesem Beispiel wird ein Identitätspool mit dem Namen erstellt MyIdentityPool. Der Pool hat einen Cognito-Identitätspool-Anbieter. Nicht authentifizierte Identitäten sind nicht zulässig.

Befehl:

```
aws cognito-identity create-identity-pool --identity-pool-
name MyIdentityPool --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

Ausgabe:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Einzelheiten zur API finden Sie CreateIdentityPoolin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateIdentityPoolin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt einen neuen Identitätspool, der nicht authentifizierte Identitäten zulässt.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- Einzelheiten zur API finden Sie unter CreateIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt einen neuen Identitätspool, der nicht authentifizierte Identitäten zulässt.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- Einzelheiten zur API finden Sie unter CreateIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V5).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        do {
            let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
 "com.exampleco.CognitoIdentityDemo",
                                                           identityPoolName:
 name)

            let result = try await
 cognitoIdentityClient.createIdentityPool(input: cognitoInputCall)
            guard let poolId = result.identityPoolId else {
                return nil
            }

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- Einzelheiten zur API finden Sie CreateIdentityPoolin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteIdentityPool`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteIdentityPool` verwendet wird.

CLI

AWS CLI

Löschen eines Identitätspools

Im folgenden `delete-identity-pool`-Beispiel wird der angegebene Identitätspool gelöscht.

Befehl:

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteIdentityPoolin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {
```

```
            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteIdentityPoolin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht einen bestimmten Identitätspool.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Einzelheiten zur API finden Sie unter DeleteIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht einen bestimmten Identitätspool.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- Einzelheiten zur API finden Sie unter DeleteIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V5).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        do {
            let input = DeleteIdentityPoolInput(
                identityPoolId: id
            )

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- Einzelheiten zur API finden Sie DeleteIdentityPoolin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DescribeIdentityPool`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeIdentityPool` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen Identitätspool

In diesem Beispiel wird ein Identitätspool beschrieben.

Befehl:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Ausgabe:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeIdentityPoolin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft Informationen über einen bestimmten Identitätspool anhand seiner ID ab.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- Einzelheiten zur API finden Sie unter DescribeIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft Informationen über einen bestimmten Identitätspool anhand seiner ID ab.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- Einzelheiten zur API finden Sie unter DescribeIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetCredentialsForIdentity`** AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetCredentialsForIdentity`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetCredentialsForIdentityin der AWS SDK for Java 2.x API-Referenz.

Verwendung von **`GetIdentityPoolRoles`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetIdentityPoolRoles` verwendet wird.

CLI

AWS CLI

So rufen Sie Rollen für einen Identitätspool ab

In diesem Beispiel werden Rollen für einen Identitätspool abgerufen.

Befehl:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

Ausgabe:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- Einzelheiten zur API finden Sie GetIdentityPoolRolesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft die Informationen über Rollen für einen bestimmten Identitätspool ab.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/CommonTests1Role]}
```

```
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter GetIdentityPoolRoles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft die Informationen über Rollen für einen bestimmten Identitätspool ab.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter GetIdentityPoolRoles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListIdentityPools`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListIdentityPools` verwendet wird.

CLI

AWS CLI

So listen Sie Identitätspools auf

In diesem Beispiel werden Identitätspools aufgeführt. Es werden maximal 20 Identitäten aufgeführt.

Befehl:

```
aws cognito-identity list-identity-pools --max-results 20
```

Ausgabe:

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- Einzelheiten zur API finden Sie ListIdentityPoolsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListIdentityPoolsin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft eine Liste vorhandener Identitätspools ab.

```
Get-CGIIdentityPoolList
```

Ausgabe:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- Einzelheiten zur API finden Sie unter ListIdentityPools AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft eine Liste vorhandener Identitätspools ab.

```
Get-CGIIdentityPoolList
```

Ausgabe:

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- Einzelheiten zur API finden Sie unter ListIdentityPools AWS -Tools für PowerShellCmdlet-Referenz (V5).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
 "ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
 listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
 contents.")
                    continue
                }

                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
```

```
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

Rufen Sie ID eines vorhandenen Identitätspools ab oder erstellen Sie sie, wenn sie nicht bereits vorhanden ist.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        do {
            guard let poolId = try await getIdentityPoolID(name: name) else {
                return try await createIdentityPool(name: name)
            }

            return poolId
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- Einzelheiten zur API finden Sie ListIdentityPoolsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`SetIdentityPoolRoles`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetIdentityPoolRoles` verwendet wird.

CLI

AWS CLI

So legen Sie Rollen für einen Identitätspool fest

Im folgenden Beispiel für `set-identity-pool-roles` wird eine Rolle für einen Identitätspool festgelegt.

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- Einzelheiten zur API finden Sie SetIdentityPoolRolesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Konfiguriert den spezifischen Identitätspool so, dass er über eine nicht authentifizierte IAM-Rolle verfügt.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Einzelheiten zur API finden Sie unter SetIdentityPoolRoles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Konfiguriert den spezifischen Identitätspool so, dass er über eine nicht authentifizierte IAM-Rolle verfügt.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- Einzelheiten zur API finden Sie unter SetIdentityPoolRoles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateIdentityPool`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateIdentityPool` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie einen Identitätspool

In diesem Beispiel wird ein Identitätspool aktualisiert. Es setzt den Namen auf MyIdentityPool. Cognito wird als Identitätsanbieter hinzugefügt. Nicht authentifizierte Identitäten werden nicht zugelassen.

Befehl:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-
name "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

Ausgabe:

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
```

```
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- Einzelheiten zur API finden Sie UpdateIdentityPoolin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert einige Eigenschaften des Identitätspools, in diesem Fall den Namen des Identitätspools.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Einzelheiten zur API finden Sie unter UpdateIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert einige Eigenschaften des Identitätspools, in diesem Fall den Namen des Identitätspools.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- Einzelheiten zur API finden Sie unter UpdateIdentityPool AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Amazon Cognito Identity AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Cognito Identity mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Cognito Identity oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.

- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Codebeispiele für Amazon Cognito Identity Provider mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Cognito Identity Provider mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Cognito Identity Provider – Weitere Informationen zu Amazon Cognito Identity Provider.
- API-Referenz für Amazon Cognito Identity Provider – Details zu allen verfügbaren Aktionen von Amazon Cognito Identity Provider.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Cognito Identity Provider AWS SDKs

- Hello Amazon Cognito
- Aktionen für Amazon Cognito Identity Provider mit AWS SDKs
  - Verwenden Sie AdminCreateUsermit einem AWS SDK oder CLI
  - Verwenden Sie AdminGetUsermit einem AWS SDK oder CLI
  - Verwenden Sie AdminInitiateAuthmit einem AWS SDK oder CLI
  - Verwenden Sie AdminRespondToAuthChallengemit einem AWS SDK oder CLI
  - Verwenden Sie AdminSetUserPasswordmit einem AWS SDK oder CLI
  - Verwenden Sie AssociateSoftwareTokenmit einem AWS SDK oder CLI
  - Verwenden Sie ConfirmDevicemit einem AWS SDK oder CLI
  - Verwenden Sie mit ConfirmForgotPasswordeinem AWS SDK oder CLI
  - Verwenden Sie ConfirmSignUpmit einem AWS SDK oder CLI
  - Verwenden Sie CreateUserPoolmit einem AWS SDK oder CLI
  - Verwenden Sie CreateUserPoolClientmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteUsermit einem AWS SDK oder CLI
  - Verwenden Sie ForgotPasswordmit einem AWS SDK oder CLI
  - Verwenden Sie InitiateAuthmit einem AWS SDK oder CLI
  - Verwenden Sie mit ListUserPoolseinem AWS SDK oder CLI
  - Verwenden Sie ListUsersmit einem AWS SDK oder CLI
  - Verwenden Sie ResendConfirmationCodemit einem AWS SDK oder CLI
  - Verwenden Sie RespondToAuthChallengemit einem AWS SDK oder CLI
  - Verwenden Sie mit SignUpeinem AWS SDK oder CLI
  - Verwenden Sie UpdateUserPoolmit einem AWS SDK oder CLI
  - Verwenden Sie VerifySoftwareTokenmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Cognito Identity Provider AWS SDKs
  - Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
  - Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
  - Erste Schritte mit Benutzerpools
  - Registrieren Sie einen Benutzer mit einem Amazon Cognito Cognito-Benutzerpool, für den MFA erforderlich ist, mithilfe eines AWS SDK

- Verwenden Sie Amazon Cognito Cognito-Identitätspools und Authentifizierungsabläufe
- Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Cognito Identity Provider AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Cognito Identity Provider mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Cognito
- Aktionen für Amazon Cognito Identity Provider mit AWS SDKs
  - Verwenden Sie AdminCreateUsermit einem AWS SDK oder CLI
  - Verwenden Sie AdminGetUsermit einem AWS SDK oder CLI
  - Verwenden Sie AdminInitiateAuthmit einem AWS SDK oder CLI
  - Verwenden Sie AdminRespondToAuthChallengemit einem AWS SDK oder CLI
  - Verwenden Sie AdminSetUserPasswordmit einem AWS SDK oder CLI
  - Verwenden Sie AssociateSoftwareTokenmit einem AWS SDK oder CLI
  - Verwenden Sie ConfirmDevicemit einem AWS SDK oder CLI
  - Verwenden Sie mit ConfirmForgotPasswordeinem AWS SDK oder CLI
  - Verwenden Sie ConfirmSignUpmit einem AWS SDK oder CLI
  - Verwenden Sie CreateUserPoolmit einem AWS SDK oder CLI
  - Verwenden Sie CreateUserPoolClientmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteUsermit einem AWS SDK oder CLI
  - Verwenden Sie ForgotPasswordmit einem AWS SDK oder CLI
  - Verwenden Sie InitiateAuthmit einem AWS SDK oder CLI
  - Verwenden Sie mit ListUserPoolseinem AWS SDK oder CLI
  - Verwenden Sie ListUsersmit einem AWS SDK oder CLI
  - Verwenden Sie ResendConfirmationCodemit einem AWS SDK oder CLI
  - Verwenden Sie RespondToAuthChallengemit einem AWS SDK oder CLI
  - Verwenden Sie mit SignUpeinem AWS SDK oder CLI

- Verwenden Sie UpdateUserPoolmit einem AWS SDK oder CLI
- Verwenden Sie VerifySoftwareTokenmit einem AWS SDK oder CLI

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()
```

```
# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_cognito.cpp.

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
```

```
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
 listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
 listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
```

```
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListUserPoolsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});
```

```
  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
```

```
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- Einzelheiten zur API finden Sie ListUserPoolsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
```

```
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für Ruby API-Referenz.

Aktionen für Amazon Cognito Identity Provider mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Cognito Identity Provider-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Cognito–Identity-Provider-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Cognito Identity Provider AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Die vollständige Liste finden Sie in der Amazon Cognito Identity Provider API-Referenz.

Beispiele

- Verwenden Sie AdminCreateUsermit einem AWS SDK oder CLI

- Verwenden Sie AdminGetUsermit einem AWS SDK oder CLI
- Verwenden Sie AdminInitiateAuthmit einem AWS SDK oder CLI
- Verwenden Sie AdminRespondToAuthChallengemit einem AWS SDK oder CLI
- Verwenden Sie AdminSetUserPasswordmit einem AWS SDK oder CLI
- Verwenden Sie AssociateSoftwareTokenmit einem AWS SDK oder CLI
- Verwenden Sie ConfirmDevicemit einem AWS SDK oder CLI
- Verwenden Sie mit ConfirmForgotPasswordeinem AWS SDK oder CLI
- Verwenden Sie ConfirmSignUpmit einem AWS SDK oder CLI
- Verwenden Sie CreateUserPoolmit einem AWS SDK oder CLI
- Verwenden Sie CreateUserPoolClientmit einem AWS SDK oder CLI
- Verwenden Sie DeleteUsermit einem AWS SDK oder CLI
- Verwenden Sie ForgotPasswordmit einem AWS SDK oder CLI
- Verwenden Sie InitiateAuthmit einem AWS SDK oder CLI
- Verwenden Sie mit ListUserPoolseinem AWS SDK oder CLI
- Verwenden Sie ListUsersmit einem AWS SDK oder CLI
- Verwenden Sie ResendConfirmationCodemit einem AWS SDK oder CLI
- Verwenden Sie RespondToAuthChallengemit einem AWS SDK oder CLI
- Verwenden Sie mit SignUpeinem AWS SDK oder CLI
- Verwenden Sie UpdateUserPoolmit einem AWS SDK oder CLI
- Verwenden Sie VerifySoftwareTokenmit einem AWS SDK oder CLI

Verwenden Sie **`AdminCreateUser`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AdminCreateUser` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Benutzerpools
- Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

CLI

AWS CLI

Um einen Benutzer zu erstellen

Im folgenden Beispiel für `admin-create-user` wird ein Benutzer mit den angegebenen Einstellungen für E-Mail-Adresse und Telefonnummer erstellt.

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

Ausgabe:

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- Einzelheiten zur API finden Sie AdminCreateUserin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
```

```
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie AdminCreateUserin der AWS SDK für Go API-Referenz.

Verwenden Sie **`AdminGetUser`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AdminGetUser` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
 administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
```

```
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie einen Benutzer ab

In diesem Beispiel werden Informationen zum Benutzernamen jane@example.com abgerufen.

Befehl:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --
username jane@example.com
```

Ausgabe:

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
```

```
        {
            "Name": "email_verified",
            "Value": "true"
        },
        {
            "Name": "phone_number_verified",
            "Value": "true"
        },
        {
            "Name": "phone_number",
            "Value": "+01115551212"
        },
        {
            "Name": "email",
            "Value": "jane@example.com"
        }
    ],
    "UserLastModifiedDate": 1548108509.537
}
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();
```

```
            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- Einzelheiten zur API finden Sie AdminGetUserin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
 Cognito
        to send an email to the specified email address. The email contains a
 code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
```

```
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
 response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- Einzelheiten zur API finden Sie AdminGetUserin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
```

```
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }
```

- Einzelheiten zur API finden Sie AdminGetUserin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`AdminInitiateAuth`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AdminInitiateAuth` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So melden Sie einen Benutzer als Administrator an

Im folgenden Beispiel für `admin-initiate-auth` wird der Benutzer diego@example.com angemeldet. Dieses Beispiel enthält auch Metadaten für den Bedrohungsschutz und ClientMetadata für Lambda-Trigger. Der Benutzer ist für TOTP MFA konfiguriert und erhält eine Aufforderung, einen Code von seiner Authenticator-App einzugeben, bevor er die Authentifizierung abschließen kann.

```
aws cognito-idp admin-initiate-auth \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --auth-flow ADMIN_USER_PASSWORD_AUTH \
    --auth-parameters USERNAME=diego@example.com,PASSWORD="My@Example
$Password3!",SECRET_HASH=ExampleEncodedClientIdSecretAndUsername= \
    --context-data="{\"EncodedData\":\"abc123example\",\"HttpHeaders\":
[{\"headerName\":\"UserAgent\",\"headerValue\":\"Mozilla/5.0 (Windows NT
 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\"}],\"IpAddress\":
\"192.0.2.1\",\"ServerName\":\"example.com\",\"ServerPath\":\"/login\"}" \
    --client-metadata="{\"MyExampleKey\": \"MyExampleValue\"}"
```

Ausgabe:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeExample...",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "MyAuthenticatorApp",
        "USER_ID_FOR_SRP": "diego@example.com"
    }
}
```

Weitere Informationen finden Sie unter Ablauf der Administratorauthentifizierung im Entwicklerhandbuch für Amazon-Cognito.

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
```

```
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
```

```
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Set up authentication parameters
        DATA(lt_auth_params) = VALUE /aws1/
cl_cgpauthparamstype_w=>tt_authparameterstype(
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'PASSWORD'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_password ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpauthparamstype_w( iv_secret_hash )
          ) INTO TABLE lt_auth_params.
        ENDIF.

        oo_result = lo_cgp->admininitiateauth(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_authflow = 'ADMIN_USER_PASSWORD_AUTH'
          it_authparameters = lt_auth_params
        ).

        DATA(lv_challenge) = oo_result->get_challengename( ).

        IF lv_challenge IS INITIAL.
          MESSAGE 'User successfully signed in.' TYPE 'I'.
        ELSE.
```

```
          MESSAGE |Authentication challenge required: { lv_challenge }.| TYPE
 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpusernotfoundex INTO DATA(lo_user_ex).
        MESSAGE |User { iv_user_name } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check credentials.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
```

```
                        userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`AdminRespondToAuthChallenge`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AdminRespondToAuthChallenge` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Respond to an admin authentication challenge.
/// </summary>
/// <param name="userName">The name of the user.</param>
/// <param name="clientId">The client ID.</param>
/// <param name="mfaCode">The multi-factor authentication code.</param>
/// <param name="session">The current application session.</param>
/// <param name="clientId">The user pool ID.</param>
/// <returns>The result of the authentication response.</returns>
public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
    string userName,
    string clientId,
    string mfaCode,
    string session,
    string userPoolId)
{
    Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

    var challengeResponses = new Dictionary<string, string>();
    challengeResponses.Add("USERNAME", userName);
    challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
```

```
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So reagieren Sie auf eine Authentifizierungsaufforderung

Abhängig von Ihrem Authentifizierungsablauf, der Konfiguration des Benutzerpools und den Benutzereinstellungen gibt es viele Möglichkeiten, auf unterschiedliche

Authentifizierungsaufforderungen zu reagieren. Im folgenden Beispiel für `admin-respond-to-auth-challenge` wird ein TOTP-MFA-Code für diego@example.com bereitgestellt und die Anmeldung abgeschlossen. In diesem Benutzerpool ist die Geräteerkennung aktiviert, sodass das Authentifizierungsergebnis auch einen neuen Geräteschlüssel zurückgibt.

```
aws cognito-idp admin-respond-to-auth-challenge \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-
responses USERNAME=diego@example.com,SOFTWARE_TOKEN_MFA_CODE=000000 \
    --session AYABeExample...
```

Ausgabe:

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-ExAmPlE1"
        }
    }
}
```

Weitere Informationen finden Sie unter Ablauf der Administratorauthentifizierung im Entwicklerhandbuch für Amazon-Cognito.

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
           String userName, String clientId, String mfaCode, String session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
               .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
               .clientId(clientId)
               .challengeResponses(challengeResponses)
               .session(session)
               .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
               + respondToAuthChallengeResult.authenticationResult());
   }
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Reagieren Sie auf eine MFA-Herausforderung, indem Sie einen Code bereitstellen, der von einer zugehörigen MFA-Anwendung generiert wurde.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
 of
        a two-factor sign-in. When sign-in is successful, it returns an access
 token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
```

```
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Build challenge responses
        DATA(lt_challenge_responses) = VALUE /aws1/
cl_cgpchallengerspstyp00=>tt_challengeresponsestype(
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'SOFTWARE_TOKEN_MFA_CODE'
              value = NEW /aws1/cl_cgpchallengerspstyp00( iv_mfa_code ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_secret_hash )
          ) INTO TABLE lt_challenge_responses.
        ENDIF.

        DATA(lo_result) = lo_cgp->adminrespondtoauthchallenge(
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_challengename = 'SOFTWARE_TOKEN_MFA'
          it_challengeresponses = lt_challenge_responses
```

```
          iv_session = iv_session
        ).

        oo_auth_result = lo_result->get_authenticationresult( ).

        IF oo_auth_result IS BOUND.
          MESSAGE 'MFA challenge completed successfully.' TYPE 'I'.
        ELSE.
          " Another challenge might be required
          DATA(lv_next_challenge) = lo_result->get_challengename( ).
          MESSAGE |Additional challenge required: { lv_next_challenge }.| TYPE
 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpexpiredcodeex INTO DATA(lo_expired_ex).
        MESSAGE 'MFA code has expired.' TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check MFA configuration.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider
```

```
    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
```

```
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
 "<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`AdminSetUserPassword`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AdminSetUserPassword` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

CLI

AWS CLI

So legen Sie ein Benutzerpasswort als Administrator fest

Im folgenden Beispiel für `admin-set-user-password` wird das Passwort für diego@example.com dauerhaft festgelegt.

```
aws cognito-idp admin-set-user-password \
    --user-pool-id us-west-2_EXAMPLE \
    --username diego@example.com \
    --password MyExamplePassword1! \
    --permanent
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Passwörter, Passwortwiederherstellung und Passwortrichtlinien im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie AdminSetUserPasswordin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie AdminSetUserPasswordin der AWS SDK für Go API-Referenz.

Verwenden Sie **`AssociateSoftwareToken`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AssociateSoftwareToken` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
```

```
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So generieren Sie einen geheimen Schlüssel für eine MFA-Authenticator-App

Im folgenden Beispiel für `associate-software-token` wird ein privater TOTP-Schlüssel für einen Benutzer generiert, der sich angemeldet und ein Zugriffstoken erhalten hat. Der resultierende private Schlüssel kann manuell in eine Authenticator-App eingegeben werden oder Anwendungen können ihn als QR-Code rendern, den der Benutzer scannen kann.

```
aws cognito-idp associate-software-token \
    --access-token eyJra456defEXAMPLE
```

Ausgabe:

```
{
    "SecretCode": "QWERTYUIOP123456EXAMPLE"
}
```

Weitere Informationen finden Sie unter TOTP-Software-Token-MFA im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const associateSoftwareToken = (session) => {
```

```
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
 user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
```

```
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_cgp->associatesoftwaretoken(
          iv_session = iv_session
        ).

        ov_secret_code = lo_result->get_secretcode( ).

        MESSAGE 'MFA secret code generated successfully.' TYPE 'I'.

      CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE 'Session not found or expired.' TYPE 'E'.
```

```
      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized to associate software token.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
 authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )
```

```
            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ConfirmDevice`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmDevice` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- Einzelheiten zur API finden Sie ConfirmDeviceein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So bestätigen Sie ein Benutzergerät

Im folgenden Beispiel für `confirm-device` wird dem aktuellen Benutzer ein neues Gerät hinzugefügt, das erkannt werden kann.

```
aws cognito-idp confirm-device \
   --access-token eyJra456defEXAMPLE \
   --device-key us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
   --device-secret-verifier-
config PasswordVerifier=TXlWZXJpZmllclN0cmluZw,Salt=TXlTUlBTYWx0
```

Ausgabe:

```
{
     "UserConfirmationNecessary": false
}
```

Weitere Informationen finden Sie unter Verwendung von Benutzergeräten in Ihrem Benutzerpool im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie ConfirmDevicein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ConfirmDevicein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
 new
```

```
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
```

```
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- Einzelheiten zur API finden Sie ConfirmDevicein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ConfirmForgotPassword`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmForgotPassword` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

CLI

AWS CLI

So bestätigen Sie ein vergessenes Passwort

In diesem Beispiel wird ein vergessenes Passwort für den Benutzernamen diego@example.com bestätigt.

Befehl:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- Einzelheiten zur API finden Sie ConfirmForgotPasswordin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie ConfirmForgotPasswordin der AWS SDK für Go API-Referenz.

Verwenden Sie **`ConfirmSignUp`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmSignUp` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
```

```
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);
```

```
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So bestätigen Sie die Registrierung

In diesem Beispiel wird die Registrierung des Benutzernamens diego@example.com bestätigt.

Befehl:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void confirmSignUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });
```

```
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
 registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
```

```
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- Einzelheiten zur API finden Sie ConfirmSignUpin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
```

```
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateUserPool`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateUserPool` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Benutzerpools

CLI

AWS CLI

So erstellen Sie einen minimal konfigurierten Benutzerpool

In diesem Beispiel wird ein Benutzerpool erstellt, der MyUserPool mit Standardwerten benannt wird. Es gibt keine erforderlichen Attribute und keine Anwendungsclients. MFA und erweiterte Sicherheit sind deaktiviert.

Befehl:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

Ausgabe:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
```

```
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
```

```
            },
            {
                "Name": "updated_at",
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547833345.777,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {},
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "CreationDate": 1547833345.777,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

So erstellen Sie einen Benutzerpool mit zwei erforderlichen Attributen

In diesem Beispiel wird ein Benutzerpool erstellt MyUserPool. Der Pool ist so konfiguriert, dass er E-Mail-Adressen als Benutzernamensattribut akzeptiert. Außerdem wird die E-Mail-Quelladresse mit Amazon Simple Email Service auf eine validierte Adresse gesetzt.

Befehl:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

Ausgabe:

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
```

```
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
```

```
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
```

```
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
```

```
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
```

```
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547837788.189,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {
            "ReplyToEmailAddress": "jane@example.com",
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

- Einzelheiten zur API finden Sie CreateUserPoolin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
 cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateUserPoolin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateUserPoolClient`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateUserPoolClient` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Benutzerpools

CLI

AWS CLI

So erstellen Sie einen Benutzerpoolclient

Im folgenden `create-user-pool-client` Beispiel wird ein neuer Benutzerpool-Client mit einem geheimen Client-Schlüssel, expliziten Lese- und Schreibattributen, Anmeldung mit Benutzername-Passwort und SRP-Flows, Anmeldung mit drei, Zugriff auf eine Teilmenge von OAuth-Bereichen IdPs, Analysen und einer erweiterten Gültigkeitsdauer der Authentifizierungssitzung erstellt. PinPoint

```
aws cognito-idp create-user-pool-client \
    --user-pool-id us-west-2_EXAMPLE \
    --client-name MyTestClient \
    --generate-secret \
    --refresh-token-validity 10 \
    --access-token-validity 60 \
    --id-token-validity 60 \
    --token-validity-units AccessToken=minutes,IdToken=minutes,RefreshToken=days
 \
    --read-attributes email phone_number email_verified phone_number_verified \
    --write-attributes email phone_number \
    --explicit-auth-
flows ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH ALLOW_REFRESH_TOKEN_AUTH \
    --supported-identity-providers Google Facebook MyOIDC \
    --callback-urls https://www.amazon.com https://example.com http://
localhost:8001 myapp://example \
    --allowed-o-auth-flows code implicit \
    --allowed-o-auth-scopes openid profile aws.cognito.signin.user.admin solar-
system-data/asteroids.add \
    --allowed-o-auth-flows-user-pool-client \
    --analytics-configuration ApplicationArn=arn:aws:mobiletargeting:us-
west-2:767671399759:apps/thisisanexamplepinpointapplicationid,UserDataShared=TRUE
 \
    --prevent-user-existence-errors ENABLED \
    --enable-token-revocation \
```

```
    --enable-propagate-additional-user-context-data \
    --auth-session-validity 4
```

Ausgabe:

```
{
    "UserPoolClient": {
        "UserPoolId": "us-west-2_EXAMPLE",
        "ClientName": "MyTestClient",
        "ClientId": "123abc456defEXAMPLE",
        "ClientSecret": "this1234is5678my91011example1213client1415secret",
        "LastModifiedDate": 1726788459.464,
        "CreationDate": 1726788459.464,
        "RefreshTokenValidity": 10,
        "AccessTokenValidity": 60,
        "IdTokenValidity": 60,
        "TokenValidityUnits": {
            "AccessToken": "minutes",
            "IdToken": "minutes",
            "RefreshToken": "days"
        },
        "ReadAttributes": [
            "email_verified",
            "phone_number_verified",
            "phone_number",
            "email"
        ],
        "WriteAttributes": [
            "phone_number",
            "email"
        ],
        "ExplicitAuthFlows": [
            "ALLOW_USER_PASSWORD_AUTH",
            "ALLOW_USER_SRP_AUTH",
            "ALLOW_REFRESH_TOKEN_AUTH"
        ],
        "SupportedIdentityProviders": [
            "Google",
            "MyOIDC",
            "Facebook"
        ],
        "CallbackURLs": [
            "https://example.com",
```

```
            "https://www.amazon.com",
            "myapp://example",
            "http://localhost:8001"
        ],
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "AllowedOAuthScopes": [
            "aws.cognito.signin.user.admin",
            "openid",
            "profile",
            "solar-system-data/asteroids.add"
        ],
        "AllowedOAuthFlowsUserPoolClient": true,
        "AnalyticsConfiguration": {
            "ApplicationArn": "arn:aws:mobiletargeting:us-
west-2:123456789012:apps/thisisanexamplepinpointapplicationid",
            "RoleArn": "arn:aws:iam::123456789012:role/aws-service-role/cognito-
idp.amazonaws.com/AWSServiceRoleForAmazonCognitoIdp",
            "UserDataShared": true
        },
        "PreventUserExistenceErrors": "ENABLED",
        "EnableTokenRevocation": true,
        "EnablePropagateAdditionalUserContextData": true,
        "AuthSessionValidity": 4
    }
}
```

Weitere Informationen finden Sie unter Application-specific Einstellungen mit App-Clients im Amazon Cognito Developer Guide.

- Einzelheiten zur API finden Sie CreateUserPoolClientin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateUserPoolClientin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteUser`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteUser` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Benutzer löschen

In diesem Beispiel wird ein Benutzer gelöscht.

Befehl:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
```

```
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
```

```
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`ForgotPassword`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ForgotPassword` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

CLI

AWS CLI

So erzwingen Sie eine Passwortänderung

Im folgenden Beispiel für `forgot-password` wird eine Nachricht an jane@example.com gesendet, um ihr Passwort zu ändern.

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --
username jane@example.com
```

Ausgabe:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- Einzelheiten zur API finden Sie ForgotPasswordin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
```

```
 }
 return output.CodeDeliveryDetails, err
}
```

- Einzelheiten zur API finden Sie ForgotPasswordin der AWS SDK für Go API-Referenz.

Verwenden Sie **`InitiateAuth`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `InitiateAuth` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Erste Schritte mit Benutzerpools
- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert
- Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
```

```
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- Einzelheiten zur API finden Sie InitiateAuthin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So melden Sie einen Benutzer an

Im folgenden Beispiel für `initiate-auth` wird ein Benutzer mit dem grundlegenden Benutzername-Passwort-Ablauf und ohne zusätzliche Aufforderungen angemeldet.

```
aws cognito-idp initiate-auth \
    --auth-flow USER_PASSWORD_AUTH \
    --client-id 1example23456789 \
    --analytics-metadata AnalyticsEndpointId=d70b2ba36a8c4dc5a04a0451aEXAMPLE \
    --auth-parameters USERNAME=testuser,PASSWORD=[Password] --user-context-
data EncodedData=mycontextdata --client-metadata MyTestKey=MyTestValue
```

Ausgabe:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

Weitere Informationen finden Sie unter Authentifizierung im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie InitiateAuthin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- Einzelheiten zur API finden Sie InitiateAuthin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie InitiateAuthin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird veranschaulicht, wie die Authentifizierung mit einem nachverfolgten Gerät gestartet wird. Um die Anmeldung abzuschließen, muss der Client korrekt auf SRP-Abfragen (Secure Remote Password) reagieren.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
```

```
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
```

```
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
```

```
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Einzelheiten zur API finden Sie InitiateAuthin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListUserPools`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListUserPools` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
```

```
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Benutzerpools auf

Das folgende `list-user-pools` Beispiel listet 3 der verfügbaren Benutzerpools im AWS Konto mit den aktuellen CLI-Anmeldeinformationen auf.

```
aws cognito-idp list-user-pools \
    --max-results 3
```

Ausgabe:

```
{
    "NextToken": "[Pagination token]",
    "UserPools": [
        {
            "CreationDate": 1681502497.741,
            "Id": "us-west-2_EXAMPLE1",
            "LambdaConfig": {
                "CustomMessage": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreSignUp": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
```

```
                "PreTokenGeneration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGenerationConfig": {
                    "LambdaArn": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                    "LambdaVersion": "V1_0"
                }
            },
            "LastModifiedDate": 1681502497.741,
            "Name": "user pool 1"
        },
        {
            "CreationDate": 1686064178.717,
            "Id": "us-west-2_EXAMPLE2",
            "LambdaConfig": {
            },
            "LastModifiedDate": 1686064178.873,
            "Name": "user pool 2"
        },
        {
            "CreationDate": 1627681712.237,
            "Id": "us-west-2_EXAMPLE3",
            "LambdaConfig": {
                "UserMigration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction"
            },
            "LastModifiedDate": 1678486942.479,
            "Name": "user pool 3"
        }
    ]
}
```

Weitere Informationen finden Sie unter Amazon-Cognito-Benutzerpools im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie ListUserPoolsunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
```

```
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListUserPoolsin der AWS SDK for Java 2.x API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListUserPoolsin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`ListUsers`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListUsers` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Benutzerpools
- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();

    var usersPaginator = _cognitoService.Paginators.ListUsers(request);
    await foreach (var response in usersPaginator.Responses)
    {
        users.AddRange(response.Users);
    }

    return users;
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So listen Sie Benutzer mit einem serverseitigen Filter auf

Im folgenden Beispiel für `list-users` werden 3 Benutzer im angeforderten Benutzerpool aufgeführt, deren E-Mail-Adressen mit `testuser` beginnen.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --filter email^=\"testuser\" \
    --max-items 3
```

Ausgabe:

```
{
    "PaginationToken": "efgh5678EXAMPLE",
    "Users": [
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "eaad0219-2117-439f-8d46-4db20e59268f"
                },
                {
                    "Name": "email",
                    "Value": "testuser@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1682955829.578,
            "UserLastModifiedDate": 1689030181.63,
            "UserStatus": "CONFIRMED",
            "Username": "testuser"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
```

```
                    "Value": "3b994cfd-0b07-4581-be46-3c82f9a70c90"
                },
                {
                    "Name": "email",
                    "Value": "testuser2@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427979.201,
            "UserLastModifiedDate": 1684427979.201,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser2"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "5929e0d1-4c34-42d1-9b79-a5ecacfe66f7"
                },
                {
                    "Name": "email",
                    "Value": "testuser3@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427823.641,
            "UserLastModifiedDate": 1684427823.641,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser3@example.com"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten von und Suchen nach Benutzern im Entwicklerhandbuch für Amazon Cognito.

Beispiel 2: So listen Sie Benutzer mit einem clientseitigen Filter auf

Im folgenden Beispiel für `list-users` werden die Attribute von 3 Benutzern aufgelistet, deren Attribut, in diesem Fall ihre E-Mail-Adresse, die E-Mail-Domain „@example.com“ enthält. Wenn andere Attribute diese Zeichenfolge enthalten, werden sie ebenfalls angezeigt. Der zweite Benutzer hat keine Attribute, die der Abfrage entsprechen, und wird von der angezeigten Ausgabe ausgeschlossen, aber nicht von der Serverantwort.

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --max-items 3
    --query Users\[\*\].Attributes\[\?Value\.contains\(\@\,\'@example.com\'\)\]
```

Ausgabe:

```
[
    [
        {
            "Name": "email",
            "Value": "admin@example.com"
        }
    ],
    [],
    [
        {
            "Name": "email",
            "Value": "operator@example.com"
        }
    ]
]
```

Weitere Informationen finden Sie unter Verwalten von und Suchen nach Benutzern im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie ListUsersin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
 created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }
```

```
   public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
       try {
           ListUsersRequest usersRequest = ListUsersRequest.builder()
                   .userPoolId(userPoolId)
                   .build();

           ListUsersResponse response = cognitoClient.listUsers(usersRequest);
           response.users().forEach(user -> {
               System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                       + user.userCreateDate());
           });

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Shows how to list users by using a filter.
   public static void listUsersFilter(CognitoIdentityProviderClient
cognitoClient, String userPoolId) {

       try {
           String filter = "email = \"tblue@noserver.com\"";
           ListUsersRequest usersRequest = ListUsersRequest.builder()
                   .userPoolId(userPoolId)
                   .filter(filter)
                   .build();

           ListUsersResponse response = cognitoClient.listUsers(usersRequest);
           response.users().forEach(user -> {
               System.out.println("User with filter applied " + user.username()
+ " Status " + user.userStatus()
                       + " Created " + user.userCreateDate());
           });

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

```
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers(userPoolId: String) {
```

```
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
```

```
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- Einzelheiten zur API finden Sie ListUsersin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_cgp->listusers(
          iv_userpoolid = iv_user_pool_id
```

```
        ).

        ot_users = lo_result->get_users( ).

        MESSAGE |Found { lines( ot_users ) } users in the pool.| TYPE 'I'.

      CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |User pool { iv_user_pool_id } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized to list users.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        do {
            let output = try await cognitoClient.listUsers(
                input: ListUsersInput(
                    userPoolId: poolId
                )
            )

            guard let users = output.users else {
                print("No users found.")
                return
            }

            print("\(users.count) user(s) found.")
            for user in users {
                print("  \(user.username ?? "<unknown>")")
            }
```

```
        } catch _ as NotAuthorizedException {
            print("*** Please authenticate with AWS before using this command.")
            return
        } catch _ as ResourceNotFoundException {
            print("*** The specified User Pool was not found.")
            return
        } catch {
            print("*** An unexpected type of error occurred.")
            return
        }
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ResendConfirmationCode`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ResendConfirmationCode` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
```

```
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So senden Sie einen Bestätigungscode erneut

Im folgenden Beispiel für `resend-confirmation-code` wird ein Bestätigungscode an den Benutzer `jane` gesendet.

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

Ausgabe:

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

Weitere Informationen finden Sie unter Registrieren und Bestätigen von Benutzerkonten im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
 clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
```

```
                print("*** Unable to get the delivery method for the resent
code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
\(userName).")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`RespondToAuthChallenge`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RespondToAuthChallenge` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

CLI

AWS CLI

Beispiel 1: So antworten Sie auf eine NEW_PASSWORD_REQUIRED-Aufforderung

Das folgende Beispiel für `respond-to-auth-challenge` antwortet auf eine NEW_PASSWORD_REQUIRED-Aufforderung, die initiate-auth zurückgegeben hat. Es wird ein Passwort für den Benutzer `jane@example.com` festgelegt.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --challenge-name NEW_PASSWORD_REQUIRED \
    --challenge-responses USERNAME=jane@example.com,NEW_PASSWORD=[Password] \
```

```
    --session AYABeEv5HklEXAMPLE
```

Ausgabe:

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "ACCESS_TOKEN",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "REFRESH_TOKEN",
        "IdToken": "ID_TOKEN",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-wt2ha1Zd"
        }
    }
}
```

Weitere Informationen finden Sie unter Authentifizierung im Entwicklerhandbuch für Amazon Cognito.

Beispiel 2: So antworten Sie auf eine SELECT_MFA_TYPE-Aufforderung

Im folgenden Beispiel für `respond-to-auth-challenge` wird TOTP MFA als MFA-Option für den aktuellen Benutzer ausgewählt. Der Benutzer wurde aufgefordert, einen MFA-Typ auszuwählen, und wird im nächsten Schritt gebeten, seinen MFA-Code einzugeben.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789
    --session AYABeEv5HklEXAMPLE
    --challenge-name SELECT_MFA_TYPE
    --challenge-responses USERNAME=testuser,ANSWER=SOFTWARE_TOKEN_MFA
```

Ausgabe:

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeEv5HklEXAMPLE",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "transparent"
```

```
    }
}
```

Weitere Informationen finden Sie unter Hinzufügen von MFA im Entwicklerhandbuch für Amazon Cognito.

Beispiel 3: So antworten Sie auf eine SOFTWARE_TOKEN_MFA-Aufforderung

Im folgenden Beispiel für `respond-to-auth-challenge` wird ein TOTP-MFA-Code bereitgestellt und die Anmeldung abgeschlossen.

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --session AYABeEv5HklEXAMPLE \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-responses USERNAME=testuser,SOFTWARE_TOKEN_MFA_CODE=123456
```

Ausgabe:

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

Weitere Informationen finden Sie unter Hinzufügen von MFA im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie RespondToAuthChallengein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie RespondToAuthChallengein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem nachverfolgten Gerät an. Um die Anmeldung abzuschließen, muss der Client korrekt auf SRP-Abfragen (Secure Remote Password) reagieren.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
```

```
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
```

```
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- Einzelheiten zur API finden Sie RespondToAuthChallengein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **SignUp**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SignUp` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Erste Schritte mit Benutzerpools
- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
```

```
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
```

```
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So registrieren einen Benutzer

In diesem Beispiel wird jane@example.com registriert.

Befehl:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

Ausgabe:

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- Einzelheiten zur API finden Sie SignUpin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
```

```
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void signUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String userName,
            String password, String email) {
```

```
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
```

```
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Einzelheiten zur API finden Sie SignUpin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
```

```
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        return confirmed
```

- Einzelheiten zur API finden Sie SignUpin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
```

```
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- Einzelheiten zur API finden Sie SignUpin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`UpdateUserPool`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateUserPool` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion
- Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

- Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

CLI

AWS CLI

So aktualisieren Sie einen Benutzerpool

Im folgenden Beispiel für `update-user-pool` wird ein Benutzerpool mit einer Beispielsyntax für jede der verfügbaren Konfigurationsoptionen geändert. Wenn Sie einen Benutzerpool aktualisieren möchten, müssen Sie alle zuvor konfigurierten Optionen angeben. Andernfalls werden sie auf einen Standardwert zurückgesetzt.

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_EXAMPLE \
    --policies PasswordPolicy=
\{MinimumLength=6,RequireUppercase=true,RequireLowercase=true,RequireNumbers=true,Require
 \
    --deletion-protection ACTIVE \
    --lambda-config PreSignUp="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-presignup-
function",PreTokenGeneration="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-pretoken-function" \
    --auto-verified-attributes "phone_number" "email" \
    --verification-message-template \{\"SmsMessage\":\""Your code is
 {####}"\",\"EmailMessage\":\""Your code is {####}"\",\"EmailSubject\":\""Your
 verification code"\",\"EmailMessageByLink\":\""Click {##here##} to verify
 your email address."\",\"EmailSubjectByLink\":\""Your verification link"\",
\"DefaultEmailOption\":\"CONFIRM_WITH_LINK\"\} \
    --sms-authentication-message "Your code is {####}" \
    --user-attribute-update-settings
 AttributesRequireVerificationBeforeUpdate="email","phone_number" \
    --mfa-configuration "OPTIONAL" \
    --device-
configuration ChallengeRequiredOnNewDevice=true,DeviceOnlyRememberedOnUserPrompt=true
 \
    --email-configuration SourceArn="arn:aws:ses:us-
west-2:123456789012:identity/admin@example.com",ReplyToEmailAddress="amdin
+noreply@example.com",EmailSendingAccount=DEVELOPER,From="admin@amazon.com",Configuration
configuration-set" \
    --sms-configuration SnsCallerArn="arn:aws:iam::123456789012:role/service-
role/SNS-SMS-Role",ExternalId="12345",SnsRegion="us-west-2" \
```

```
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-tags "Function"="MyMobileGame","Developers"="Berlin" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-add-ons AdvancedSecurityMode="AUDIT" \
    --account-recovery-setting RecoveryMechanisms=
\[\{Priority=1,Name="verified_email"\},
\{Priority=2,Name="verified_phone_number"\}\]
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Aktualisieren der Benutzerpoolkonfiguration im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie UpdateUserPoolin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
```

```
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
```

```
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie UpdateUserPoolin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
```

```
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Einzelheiten zur API finden Sie UpdateUserPoolin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`VerifySoftwareToken`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `VerifySoftwareToken` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So bestätigen Sie die Registrierung einer TOTP-Authentifizierung

Im folgenden Beispiel für `verify-software-token` wird die TOTP-Registrierung für den aktuellen Benutzer abgeschlossen.

```
aws cognito-idp verify-software-token \
    --access-token eyJra456defEXAMPLE \
    --user-code 123456
```

Ausgabe:

```
{
    "Status": "SUCCESS"
}
```

Weitere Informationen finden Sie unter Hinzufügen der MFA zu einem Benutzerpool im Entwicklerhandbuch für Amazon Cognito.

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_cgp->verifysoftwaretoken(
          iv_session = iv_session
          iv_usercode = iv_user_code
        ).

        ov_status = lo_result->get_status( ).

        IF ov_status = 'SUCCESS'.
          MESSAGE 'MFA token verified successfully.' TYPE 'I'.
        ELSE.
          MESSAGE |MFA verification status: { ov_status }.| TYPE 'I'.
        ENDIF.
```

```
      CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
        MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

      CATCH /aws1/cx_cgpenbsoftwaretokmf00 INTO DATA(lo_enabled_ex).
        MESSAGE 'Software token MFA is already enabled.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
```

```
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der API-Referenz zum AWS SDK für Swift.

Szenarien für die Verwendung von Amazon Cognito Identity Provider AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Cognito Identity Provider mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Cognito Identity Provider oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Erste Schritte mit Benutzerpools
- Registrieren Sie einen Benutzer mit einem Amazon Cognito Cognito-Benutzerpool, für den MFA erforderlich ist, mithilfe eines AWS SDK
- Verwenden Sie Amazon Cognito Cognito-Identitätspools und Authentifizierungsabläufe
- Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
```

```
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
```

```
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")
```

```
 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
```

```
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
```

```
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
```

```
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
```

```
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
```

```
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
```

```
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
```

```
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
```

```
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
```

```
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
```

```
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
```

```
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
```

```
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}
```

```
// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
```

```
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie einen interaktiven „Szenario“-Lauf. Die JavaScript (v3) -Beispiele verwenden gemeinsam einen Szenario-Runner, um komplexe Beispiele zu vereinfachen. Der komplette Quellcode ist aktiviert GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
```

```
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Dieses Szenario demonstriert die automatische Bestätigung eines bekannten Benutzers. Es orchestriert die Beispielschritte.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
```

```
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */
```

```
const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);
```

```
const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
```

```
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
```

```
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
```

```
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }
```

```
    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
```

```
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Dies sind Schritte, die mit anderen Szenarien geteilt werden.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);
```

```
export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Ein Handler für den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
 '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
```

```
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName
 '${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Modul zur CloudWatch Protokollierung von Aktionen.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
```

```
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für Amazon-Cognito-Aktionen.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
```

```
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
```

```
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
```

```
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für DynamoDB-Aktionen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - DeleteUser
  - InitiateAuth

- SignUp
- UpdateUserPool

Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch migriert werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `MigrateUser`-Trigger aufzurufen.
- Melden Sie sich bei Amazon Cognito mit einem Benutzernamen und einer E-Mail-Adresse an, die sich nicht im Benutzerpool befinden.
- Die Lambda-Funktion durchsucht eine DynamoDB-Tabelle und migriert bekannte Benutzer automatisch in den Benutzerpool.
- Führen Sie den Ablauf „Passwort vergessen“ aus, um das Passwort für den migrierten Benutzer zurückzusetzen.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
```

```
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
```

```
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
```

```
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
 user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to
 Cognito, you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.",
 *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
```

```
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))
```

```
 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `MigrateUser`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)
```

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
```

```
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
```

```
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
```

```
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
```

```
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
```

```
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
```

```
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
```

```
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
```

```
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
```

```
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
```

```
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
```

```
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}
```

```
// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}
```

```
func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
```

```
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Erste Schritte mit Benutzerpools

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Benutzerpools
- Erstellen eines App-Clients
- Richten Sie eine Domain für Ihren Benutzerpool ein
- Erstellen Sie einen Benutzer als Administrator
- Aktivieren Sie die Selbstregistrierung
- Benutzer im Benutzerpool auflisten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Cognito User Pools Getting Started Script
# This script creates and configures an Amazon Cognito user pool with an app
 client

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cognito-user-pool-setup.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Cognito User Pool setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -qi "error\|failed"; then
    echo "ERROR: Command failed: $cmd" >&2
    echo "Output: $output" >&2
    cleanup_on_error
    exit 1
  fi
```

```
}

# Function to check AWS CLI return code
check_aws_error() {
  local exit_code=$?
  local cmd=$1

  if [ $exit_code -ne 0 ]; then
    echo "ERROR: AWS CLI command failed with exit code $exit_code: $cmd" >&2
    cleanup_on_error
    exit "$exit_code"
  fi
}

# Function to clean up resources on error
cleanup_on_error() {
  echo "Error encountered. Attempting to clean up resources..." >&2

  if [ -n "${DOMAIN_NAME:-}" ] && [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool domain: $DOMAIN_NAME" >&2
    aws cognito-idp delete-user-pool-domain \
      --user-pool-id "$USER_POOL_ID" \
      --domain "$DOMAIN_NAME" 2>/dev/null || true
  fi

  if [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool: $USER_POOL_ID" >&2
    aws cognito-idp delete-user-pool \
      --user-pool-id "$USER_POOL_ID" 2>/dev/null || true
  fi
}

# Set trap for cleanup on exit
trap cleanup_on_error EXIT ERR

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
  echo "ERROR: AWS CLI is not installed" >&2
  exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
  echo "ERROR: jq is not installed" >&2
```

```
  exit 1
fi

# Validate openssl is installed
if ! command -v openssl &> /dev/null; then
  echo "ERROR: openssl is not installed" >&2
  exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
  echo "ERROR: AWS CLI is not configured or credentials are invalid" >&2
  exit 1
fi

# Get the current AWS region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
  AWS_REGION="us-east-1" # Default region if not configured
fi
echo "Using AWS Region: $AWS_REGION"

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
  echo "ERROR: Invalid AWS region format: $AWS_REGION" >&2
  exit 1
fi

# Generate random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
  echo "ERROR: Failed to generate random identifier" >&2
  exit 1
fi

USER_POOL_NAME="MyUserPool-${RANDOM_ID}"
APP_CLIENT_NAME="MyAppClient-${RANDOM_ID}"
DOMAIN_NAME="my-auth-domain-${RANDOM_ID}"

# Validate resource names don't exceed limits
if [ ${#USER_POOL_NAME} -gt 128 ]; then
  echo "ERROR: User pool name exceeds maximum length of 128 characters" >&2
  exit 1
fi
```

```
if [ ${#APP_CLIENT_NAME} -gt 128 ]; then
  echo "ERROR: App client name exceeds maximum length of 128 characters" >&2
  exit 1
fi

echo "Using random identifier: $RANDOM_ID"
echo "User pool name: $USER_POOL_NAME"
echo "App client name: $APP_CLIENT_NAME"
echo "Domain name: $DOMAIN_NAME"

# Step 1: Create a User Pool
echo "Creating user pool..."
USER_POOL_OUTPUT=$(aws cognito-idp create-user-pool \
  --pool-name "$USER_POOL_NAME" \
  --auto-verified-attributes email \
  --username-attributes email \
  --policies '{"PasswordPolicy":
{"MinimumLength":12,"RequireUppercase":true,"RequireLowercase":true,"RequireNumbers":true
 \
  --schema '[{"Name":"email","Required":true,"Mutable":true}]' \
  --mfa-configuration OFF \
  --user-attribute-update-settings '{"AttributesRequireVerificationBeforeUpdate":
["email"]}' \
  --account-recovery-setting
 'RecoveryMechanisms=[{Name=verified_email,Priority=1}]' \
  --deletion-protection INACTIVE \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool"

# Extract the User Pool ID using jq for safety
USER_POOL_ID=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Id // empty')
if [ -z "$USER_POOL_ID" ]; then
  echo "ERROR: Failed to extract User Pool ID" >&2
  exit 1
fi

# Validate User Pool ID format
if ! [[ "$USER_POOL_ID" =~ ^[a-z]{2}-[a-z]+-[0-9]+_[a-zA-Z0-9]+$ ]]; then
  echo "ERROR: Invalid User Pool ID format: $USER_POOL_ID" >&2
  exit 1
fi

echo "User Pool created with ID: $USER_POOL_ID"
```

```
USER_POOL_ARN=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Arn // empty')
aws cognito-idp tag-resource \
  --resource-arn "$USER_POOL_ARN" \
  --tags project=doc-smith,tutorial=amazon-cognito-gs

# Wait for user pool to be ready
echo "Waiting for user pool to be ready..."
sleep 5

# Step 2: Create an App Client with enhanced security
echo "Creating app client..."
APP_CLIENT_OUTPUT=$(aws cognito-idp create-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-name "$APP_CLIENT_NAME" \
  --no-generate-secret \
  --explicit-auth-flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH \
  --callback-urls '["https://localhost:3000/callback"]' \
  --allowed-o-auth-flows 'code' \
  --allowed-o-auth-scopes 'openid' 'email' 'profile' \
  --allowed-o-auth-flows-user-pool-client \
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --access-token-validity 1 \
  --id-token-validity 1 \
  --refresh-token-validity 30 \
  --token-validity-units 'AccessToken=hours,IdToken=hours,RefreshToken=days' \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-client"

# Extract the Client ID using jq for safety
CLIENT_ID=$(echo "$APP_CLIENT_OUTPUT" | jq -r '.UserPoolClient.ClientId //
 empty')
if [ -z "$CLIENT_ID" ]; then
  echo "ERROR: Failed to extract Client ID" >&2
  cleanup_on_error
  exit 1
fi

# Validate Client ID format
if ! [[ "$CLIENT_ID" =~ ^[a-z0-9]{26}$ ]]; then
  echo "ERROR: Invalid Client ID format: $CLIENT_ID" >&2
  cleanup_on_error
```

```
  exit 1
fi

echo "App Client created with ID: $CLIENT_ID"

# Step 3: Set Up a Domain for Your User Pool
echo "Setting up user pool domain..."
DOMAIN_OUTPUT=$(aws cognito-idp create-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-domain"
echo "Domain created: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"

# Step 4: View User Pool Details
echo "Retrieving user pool details..."
USER_POOL_DETAILS=$(aws cognito-idp describe-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool"
echo "User Pool details retrieved successfully"

# Step 5: View App Client Details
echo "Retrieving app client details..."
APP_CLIENT_DETAILS=$(aws cognito-idp describe-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-id "$CLIENT_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool-client"
echo "App Client details retrieved successfully"

# Step 6: Create a User (Admin)
echo "Creating admin user..."
ADMIN_USER_EMAIL="admin@example.com"
TEMP_PASSWORD="$(openssl rand -base64 12 | tr -d '\n')!@#"
if [ -z "$TEMP_PASSWORD" ]; then
  echo "ERROR: Failed to generate temporary password" >&2
  cleanup_on_error
  exit 1
fi
```

```
ADMIN_USER_OUTPUT=$(aws cognito-idp admin-create-user \
  --user-pool-id "$USER_POOL_ID" \
  --username "$ADMIN_USER_EMAIL" \
  --user-attributes Name=email,Value="$ADMIN_USER_EMAIL"
 Name=email_verified,Value=true \
  --temporary-password "$TEMP_PASSWORD" \
  --message-action SUPPRESS \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-create-user"
echo "Admin user created: $ADMIN_USER_EMAIL"

# Securely clear temporary password from memory
unset TEMP_PASSWORD

# Step 7: Self-Registration
echo "Demonstrating self-registration..."
USER_EMAIL="user@example.com"
USER_PASSWORD="SecurePassword123!"
SIGNUP_OUTPUT=$(aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" \
  --username "$USER_EMAIL" \
  --password "$USER_PASSWORD" \
  --user-attributes Name=email,Value="$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "sign-up"
echo "User signed up: $USER_EMAIL"
echo "A confirmation code would be sent to the user's email in a real scenario"

# Securely clear password from memory
unset USER_PASSWORD

echo ""
echo "====================================================="
echo "IMPORTANT: In a real scenario, the user would receive"
echo "a confirmation code via email. For this demo, we'll"
echo "use admin-confirm-sign-up instead."
echo "====================================================="
echo ""

# Step 8: Confirm User Registration (using admin privileges for demo)
echo "Confirming user registration (admin method)..."
CONFIRM_OUTPUT=$(aws cognito-idp admin-confirm-sign-up \
```

```
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-confirm-sign-up"
echo "User confirmed: $USER_EMAIL"

# Step 9: Set permanent password for user
echo "Setting permanent password for user..."
SET_PASSWORD="SecureUserPassword123!"
SET_PASS_OUTPUT=$(aws cognito-idp admin-set-user-password \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --password "$SET_PASSWORD" \
  --permanent \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-set-user-password"
echo "Permanent password set for user"

unset SET_PASSWORD

# Step 10: Authenticate a User
echo "Authenticating user..."
AUTH_PASSWORD="SecureUserPassword123!"
AUTH_OUTPUT=$(aws cognito-idp initiate-auth \
  --client-id "$CLIENT_ID" \
  --auth-flow USER_PASSWORD_AUTH \
  --auth-parameters "USERNAME=$USER_EMAIL,PASSWORD=$AUTH_PASSWORD" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "initiate-auth"
echo "User authenticated successfully"

unset AUTH_PASSWORD

# Extract auth tokens securely
ID_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.IdToken // empty')
ACCESS_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.AccessToken //
 empty')

# Validate tokens exist
if [ -z "$ID_TOKEN" ] || [ -z "$ACCESS_TOKEN" ]; then
  echo "WARNING: Failed to extract authentication tokens" >&2
```

```
else
  echo "Authentication tokens obtained successfully"
fi

# Securely clear tokens from memory
unset ID_TOKEN
unset ACCESS_TOKEN

# Step 11: List Users in the User Pool
echo "Listing users in the user pool..."
USERS_OUTPUT=$(aws cognito-idp list-users \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "list-users"
echo "Users listed successfully"

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "User Pool ID: $USER_POOL_ID"
echo "User Pool Name: $USER_POOL_NAME"
echo "App Client ID: $CLIENT_ID"
echo "App Client Name: $APP_CLIENT_NAME"
echo "Domain: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
echo "Admin User: $ADMIN_USER_EMAIL"
echo "Regular User: $USER_EMAIL"
echo "==================================================="
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting cleanup process..."

# Step 12: Clean Up Resources
echo "Deleting user pool domain..."
DELETE_DOMAIN_OUTPUT=$(aws cognito-idp delete-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
```

```
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool-domain"
echo "Domain deleted successfully"

# Wait for domain deletion to complete
echo "Waiting for domain deletion to complete..."
sleep 5

echo "Deleting user pool (this will also delete the app client)..."
DELETE_POOL_OUTPUT=$(aws cognito-idp delete-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool"
echo "User pool deleted successfully"

echo "All resources have been cleaned up"

echo "Script completed at $(date)"

# Remove trap to prevent cleanup on successful exit
trap - EXIT ERR
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AdminConfirmSignUp
  - AdminCreateUser
  - CreateUserPool
  - CreateUserPoolClient
  - CreateUserPoolDomain
  - DeleteUserPool
  - DeleteUserPoolDomain
  - DescribeUserPool
  - DescribeUserPoolClient
  - InitiateAuth
  - ListUsers
  - SignUp

Registrieren Sie einen Benutzer mit einem Amazon Cognito Cognito-Benutzerpool, für den MFA erforderlich ist, mithilfe eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
```

```
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");
```

```
        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
```

```
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
 pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}
```

```
using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
```

```
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
```

```
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
 _cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
 authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
```

```
    }

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
 string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
```

```
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
```

```
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
```

```
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge

- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
```

```
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
    bool userExists = false;
    do {
        // 1. Add a user with a username, password, and email address.
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
```

```
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (userExists);

    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

```
    printAsterisksLine();

    {
        // 4. Send the confirmation code that's received in the email.
 (ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
 session, client)) {
        return false;
    }
```

```
    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
```

```
   {
       Aws::String mfaCode = askQuestion(
               "Re-enter the 6 digit code displayed in the authenticator app:
");

       // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }

       std::cout << "You have successfully added a user to Amazon Cognito."
                 << std::endl;
   }
```

```
    if (askYesNoQuestion("Would you like to delete the user that you just added?
 (y/n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<
```

```
 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
```

```
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
```

```
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
 AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
 script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
```

```
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
```

```
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }

   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
```

```
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
```

```
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um die beste Erfahrung zu erzielen, klonen Sie das GitHub Repository und führen Sie dieses Beispiel aus. Der folgende Code ist ein Teil der vollständigen Beispielanwendung.

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";
```

```
const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;
```

```
  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
```

```
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};
```

```
const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });
```

```
  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
```

```
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
```

```
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
 cognito_scenario_user_pool_with_mfa.
```

```
 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
```

```
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
 new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
```

```
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
```

```
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
```

```
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AdminGetUser

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon-Cognito-Funktionen einschließt, die im Szenario verwendet werden.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
```

```
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
```

```
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
 response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
```

```
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.
```

```
        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
```

```
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
```

```
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
```

```
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
```

```
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
```

```
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
```

```
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
```

```
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

Erstellen Sie eine Klasse, die das Szenario ausführt. Dieses Beispiel registriert auch ein MFA-Gerät für die Nachverfolgung durch Amazon Cognito und zeigt Ihnen, wie Sie sich mithilfe eines Passworts und der Informationen des nachverfolgten Geräts anmelden. Dadurch wird die Eingabe eines neuen MFA-Codes unnötig.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
 q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
 q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
 confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)
```

```
    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
```

```
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
 q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
```

```
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).

- AdminGetUser
- AdminInitiateAuth
- AdminRespondToAuthChallenge
- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "cognito-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
```

```
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
            dependencies: [
                .product(name: "AWSCognitoIdentityProvider", package: "aws-sdk-
swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei.

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
```

```
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
```

```
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
```

```
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]
```

```
        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
```

```
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
                print("*** Unable to get the delivery method for the resent
code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user
\(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
```

```
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )
```

```
            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }
```

```
            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
```

```
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String,
mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
```

```
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that
user in the
              specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
```

```
             user by associating an authenticator application (such as Google
Authenticator
             or a password manager that supports TOTP). Then, the user uses a
code from their
             authenticator application to sign in.

             """)

       let userName = stringRequest("Please enter a new username: ")
       let password = stringRequest("Enter a password: ")
       let email = stringRequest("Enter your email address: ", minLength: 5)

       // Submit the sign-up request to AWS.

       print("==> Signing up user \(userName)...")
       if await signUp(cipClient: cipClient, clientId: clientId,
                       userName: userName, password: password,
                       email: email) == false {
           return
       }

       // Check the user's status. This time, it should come back "unconfirmed".

       print("==> Getting the status of user \(userName) from the user pool
(should be 'unconfirmed')...")
       if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
           return
       }

       // Ask the user if they want a replacement code sent, such as if the
       // code hasn't arrived yet. If the user responds with a "yes," send a
       // new code.

       if yesNoRequest("==> A confirmation code was sent to \(userName). Would
you like to send a new code (Y/N)? ") {
           print("==> Sending a new confirmation code...")
           if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
               return
           }
       }

       // Ask the user to enter the confirmation code, then send it to Amazon
```

```
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId,
userName: userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should
be 'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient,
authSession: authSession)
        if newSession == nil {
            return
        }
```

```
        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in
 your authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
 mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")

        let session2 = await initiateAuth(cipClient: cipClient, clientId:
 clientId,
                                     userName: userName, password: password,
 userPoolId: poolId)
        guard let session2 else {
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a
 new 6-digit code, then enter it: ",
                                     minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName:
 userName,
                                          clientId: clientId, userPoolId: poolId,
                                          mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Verwenden Sie Amazon Cognito Cognito-Identitätspools und Authentifizierungsabläufe

Das folgende Codebeispiel zeigt, wie Sie eine webbasierte Demo-Anwendung erstellen, die Authentifizierungsabläufe für Identitätspools demonstriert.

Python

SDK für Python (Boto3)

Zeigt eine webbasierte Demo-Anwendung, die Authentifizierungsabläufe von Amazon Cognito Identity Pools demonstriert, sodass Benutzer interaktiv sowohl erweiterte als auch

grundlegende Authentifizierungsabläufe mit verschiedenen Identitätsanbietern erkunden können.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity Provider

Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon-Cognito-Benutzerauthentifizierung geschrieben werden.

- Verwenden Sie Administratorfunktionen, um einen Benutzer zu einem Benutzerpool hinzuzufügen.
- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PostAuthentication`-Trigger aufzurufen.
- Melden Sie den neuen Benutzer bei Amazon Cognito an.
- Die Lambda-Funktion schreibt benutzerdefinierte Informationen in CloudWatch Logs und in eine DynamoDB-Tabelle.
- Ermitteln Sie benutzerdefinierte Daten aus der DynamoDB-Tabelle, zeigen Sie diese an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
```

```
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
```

```
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
```

```
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
```

```
}
```

Behandeln Sie den `PostAuthentication`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
```

```
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
```

```
    log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
   } else {
    log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
   }
  }

  return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
```

```
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
```

```
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
```

```
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
```

```
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
```

```
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
```

```
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
```

```
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
```

```
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}
```

```
// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
```

```
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Codebeispiele für Amazon Cognito Sync mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Cognito Sync mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch von Amazon Cognito – Weitere Informationen zu Amazon Cognito Sync.
- API-Referenz für Amazon Cognito Sync – Details zu allen verfügbaren Aktionen von Amazon Cognito Sync.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Cognito Sync AWS SDKs
  - Aktionen für Amazon Cognito Sync mit AWS SDKs
    - Verwenden Sie ListIdentityPoolUsagemit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Cognito Sync AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Cognito Sync mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Cognito Sync mit AWS SDKs
  - Verwenden Sie ListIdentityPoolUsagemit einem AWS SDK

Aktionen für Amazon Cognito Sync mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Cognito Sync-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Cognito Sync-API-Referenz.

Beispiele

- Verwenden Sie ListIdentityPoolUsagemit einem AWS SDK

Verwenden Sie **`ListIdentityPoolUsage`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListIdentityPoolUsage`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
```

```
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListIdentityPoolUsagein der API-Referenz zum AWS SDK für Rust.

Codebeispiele für Amazon Comprehend mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Comprehend mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Comprehend-Entwicklerhandbuch – Weitere Informationen zu Amazon Comprehend.
- API-Referenz für Amazon Comprehend – Details zu allen verfügbaren Aktionen von Amazon Comprehend.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Comprehend AWS SDKs

- Aktionen für Amazon Comprehend mit AWS SDKs
  - Verwenden Sie CreateDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDocumentClassificationJobmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DescribeTopicsDetectionJobmit einem AWS SDK oder CLI
  - Verwenden Sie DetectDominantLanguagemit einem AWS SDK oder CLI
  - Verwenden Sie DetectEntitiesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectKeyPhrasesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectPiiEntitiesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectSentimentmit einem AWS SDK oder CLI
  - Verwenden Sie DetectSyntaxmit einem AWS SDK oder CLI
  - Verwenden Sie ListDocumentClassificationJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListDocumentClassifiersmit einem AWS SDK oder CLI
  - Verwenden Sie ListTopicsDetectionJobsmit einem AWS SDK oder CLI
  - Verwenden Sie StartDocumentClassificationJobmit einem AWS SDK oder CLI
  - Verwenden Sie StartTopicsDetectionJobmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Comprehend AWS SDKs
  - Erstellen einer Amazon-Transcribe-Streaming-App
  - Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher
  - Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft
  - Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
  - Erkennen Sie Dokumentelemente mit Amazon Comprehend und einem AWS SDK
  - Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mit einem AWS SDK
  - Führen Sie einen Amazon Comprehend-Themenmodellierungsjob für Beispieldaten mit einem aus AWS SDK
  - Trainieren Sie einen benutzerdefinierten Amazon Comprehend-Klassifikator und klassifizieren Sie Dokumente mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Comprehend AWS SDKs

Die folgenden Codebeispiele veranschaulichen die grundlegende Verwendung von Amazon Comprehend mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Comprehend mit AWS SDKs
  - Verwenden Sie CreateDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDocumentClassificationJobmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDocumentClassifiermit einem AWS SDK oder CLI
  - Verwenden Sie DescribeTopicsDetectionJobmit einem AWS SDK oder CLI
  - Verwenden Sie DetectDominantLanguagemit einem AWS SDK oder CLI
  - Verwenden Sie DetectEntitiesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectKeyPhrasesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectPiiEntitiesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectSentimentmit einem AWS SDK oder CLI
  - Verwenden Sie DetectSyntaxmit einem AWS SDK oder CLI
  - Verwenden Sie ListDocumentClassificationJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListDocumentClassifiersmit einem AWS SDK oder CLI
  - Verwenden Sie ListTopicsDetectionJobsmit einem AWS SDK oder CLI
  - Verwenden Sie StartDocumentClassificationJobmit einem AWS SDK oder CLI
  - Verwenden Sie StartTopicsDetectionJobmit einem AWS SDK oder CLI

Aktionen für Amazon Comprehend mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Comprehend Comprehend-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Comprehend-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Comprehend AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Comprehend.

Beispiele

- Verwenden Sie CreateDocumentClassifiermit einem AWS SDK oder CLI
- Verwenden Sie DeleteDocumentClassifiermit einem AWS SDK oder CLI
- Verwenden Sie DescribeDocumentClassificationJobmit einem AWS SDK oder CLI
- Verwenden Sie DescribeDocumentClassifiermit einem AWS SDK oder CLI
- Verwenden Sie DescribeTopicsDetectionJobmit einem AWS SDK oder CLI
- Verwenden Sie DetectDominantLanguagemit einem AWS SDK oder CLI
- Verwenden Sie DetectEntitiesmit einem AWS SDK oder CLI
- Verwenden Sie DetectKeyPhrasesmit einem AWS SDK oder CLI
- Verwenden Sie DetectPiiEntitiesmit einem AWS SDK oder CLI
- Verwenden Sie DetectSentimentmit einem AWS SDK oder CLI
- Verwenden Sie DetectSyntaxmit einem AWS SDK oder CLI
- Verwenden Sie ListDocumentClassificationJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListDocumentClassifiersmit einem AWS SDK oder CLI
- Verwenden Sie ListTopicsDetectionJobsmit einem AWS SDK oder CLI
- Verwenden Sie StartDocumentClassificationJobmit einem AWS SDK oder CLI
- Verwenden Sie StartTopicsDetectionJobmit einem AWS SDK oder CLI

Verwenden Sie **`CreateDocumentClassifier`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDocumentClassifier` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So erstellen Sie einen Dokumentenklassifikator zur Kategorisierung von Dokumenten

Im folgenden Beispiel für `create-document-classifier` wird der Trainingsprozess für ein Dokumentenklassifikatormodell gestartet. Die Trainingsdatendatei, `training.csv`, befindet sich im Tag `--input-data-config`. `training.csv` ist ein zweispaltiges Dokument, in dem die Bezeichnungen oder Klassifizierungen in der ersten Spalte und die Dokumente in der zweiten Spalte angegeben sind.

```
aws comprehend create-document-classifier \
    --document-classifier-name example-classifier \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --language-code en
```

Ausgabe:

```
{
    "DocumentClassifierArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classifier/example-classifier"
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Klassifizierung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie CreateDocumentClassifierin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
```

```
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-
using-amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];

        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
```

```
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
 documentClassifierName);
        comClient.close();
    }

    public static void createDocumentClassifier(ComprehendClient comClient,
 String dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
 DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
 CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
 comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
 createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " +
 documentClassifierArn);

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateDocumentClassifierin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
 immediately
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier`
 function
        can be used to get training status and returns a status of TRAINED when
 the
        classifier is ready to use.

        :param name: The name of the classifier.
```

```
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the
 training
                             bucket. If multiple objects have the same prefix,
 all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
                                     grants Comprehend permission to read from
 the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
 self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- Einzelheiten zur API finden Sie CreateDocumentClassifierin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->createdocumentclassifier(
      iv_documentclassifiername = iv_classifier_name
      iv_languagecode = iv_language_code
      io_inputdataconfig = NEW /aws1/cl_cpddocclifierinpdat00(
        iv_s3uri = iv_training_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
      iv_mode = iv_mode
    ).
    MESSAGE 'Document classifier creation started.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanytagsex.
    MESSAGE 'Too many tags.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDocumentClassifierin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteDocumentClassifier`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDocumentClassifier` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So löschen Sie einen benutzerdefinierten Dokumentenklassifikator

Im folgenden Beispiel für `delete-document-classifier` wird ein benutzerdefiniertes Dokumentenklassifikatormodell gelöscht.

```
aws comprehend delete-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwalten von Amazon-Comprehend-Endpunkten im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DeleteDocumentClassifierin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
```

```
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise
```

- Einzelheiten zur API finden Sie DeleteDocumentClassifierin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->deletedocumentclassifier(
          iv_documentclassifierarn = iv_classifier_arn
        ).
        MESSAGE 'Document classifier deleted.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
```

```
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourceinuseex.
        MESSAGE 'Resource in use.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDocumentClassifierin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeDocumentClassificationJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDocumentClassificationJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So beschreiben Sie einen Auftrag zur Dokumentenklassifizierung

Im folgenden Beispiel für `describe-document-classification-job` werden die Eigenschaften eines asynchronen Dokumentenklassifizierungsauftrags abgerufen.

```
aws comprehend describe-document-classification-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

Ausgabe:

```
{
    "DocumentClassificationJobProperties": {
```

```
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "exampleclassificationjob",
        "JobStatus": "COMPLETED",
        "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
        "EndTime": "2023-06-14T17:15:58.582000+00:00",
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/mymodel/version/1",
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-CLN-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-servicerole"
    }
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Klassifizierung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DescribeDocumentClassificationJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
```

```
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- Einzelheiten zur API finden Sie DescribeDocumentClassificationJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->describedocclassificationjob(
          iv_jobid = iv_job_id
        ).
        MESSAGE 'Document classification job described.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdjobnotfoundex.
        MESSAGE 'Job not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDocumentClassificationJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeDocumentClassifier`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDocumentClassifier` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So beschreiben Sie einen Dokumentenklassifikator

Im folgenden Beispiel für `describe-document-classifier` werden die Eigenschaften eines benutzerdefinierten Dokumentenklassifikatormodells abgerufen.

```
aws comprehend describe-document-classifier \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-classifier/example-classifier-1
```

Ausgabe:

```
{
    "DocumentClassifierProperties": {
        "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/example-classifier-1",
        "LanguageCode": "en",
        "Status": "TRAINED",
        "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
        "EndTime": "2023-06-13T19:42:31.752000+00:00",
        "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
        "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
        "InputDataConfig": {
            "DataFormat": "COMPREHEND_CSV",
            "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
        },
        "OutputDataConfig": {},
        "ClassifierMetadata": {
            "NumberOfLabels": 3,
            "NumberOfTrainedDocuments": 5016,
            "NumberOfTestDocuments": 557,
            "EvaluationMetrics": {
                "Accuracy": 0.9856,
                "Precision": 0.9919,
                "Recall": 0.9459,
                "F1Score": 0.9673,
                "MicroPrecision": 0.9856,
                "MicroRecall": 0.9856,
                "MicroF1Score": 0.9856,
                "HammingLoss": 0.0144
            }
        },
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role",
        "Mode": "MULTI_CLASS"
    }
}
```

Weitere Informationen finden Sie unter Erstellen und Verwalten von benutzerdefinierten Modellen im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DescribeDocumentClassifierin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- Einzelheiten zur API finden Sie DescribeDocumentClassifierin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->describedocumentclassifier(
      iv_documentclassifierarn = iv_classifier_arn
    ).
    MESSAGE 'Document classifier described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDocumentClassifierin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeTopicsDetectionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTopicsDetectionJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten

CLI

AWS CLI

So beschreiben Sie einen Auftrag zur Themenerkennung

Im folgenden Beispiel für `describe-topics-detection-job` werden die Eigenschaften eines asynchronen Auftrags zur Themenerkennung abgerufen.

```
aws comprehend describe-topics-detection-job \
    --job-id 123456abcdeb0e11022f22a11EXAMPLE
```

Ausgabe:

```
{
    "TopicsDetectionJobProperties": {
        "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
        "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
        "JobName": "example_topics_detection",
        "JobStatus": "IN_PROGRESS",
        "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
        "InputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-bucket",
            "InputFormat": "ONE_DOC_PER_LINE"
        },
        "OutputDataConfig": {
            "S3Uri": "s3://amzn-s3-demo-destination-bucket/
testfolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
        },
        "NumberOfTopics": 10,
        "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-examplerole"
    }
}
```

Weitere Informationen finden Sie unter Asynchrone Analyse für Amazon-Comprehend-Erkenntnisse im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DescribeTopicsDetectionJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- Einzelheiten zur API finden Sie DescribeTopicsDetectionJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->describetopicsdetectionjob(
      iv_jobid = iv_job_id
    ).
    MESSAGE 'Topics detection job described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdjobnotfoundex.
    MESSAGE 'Job not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeTopicsDetectionJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectDominantLanguage`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectDominantLanguage` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example calls the Amazon Comprehend service to determine the
    /// dominant language.
    /// </summary>
    public static class DetectDominantLanguage
    {
        /// <summary>
        /// Calls Amazon Comprehend to determine the dominant language used in
        /// the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle.";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            Console.WriteLine("Calling DetectDominantLanguage\n");
            var detectDominantLanguageRequest = new
DetectDominantLanguageRequest()
            {
```

```
                Text = text,
            };

            var detectDominantLanguageResponse = await
 comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
            foreach (var dl in detectDominantLanguageResponse.Languages)
            {
                Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
 {dl.Score}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Einzelheiten zur API finden Sie DetectDominantLanguagein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie die dominante Sprache eines Eingabetexts

Im folgenden Beispiel für `detect-dominant-language` wird der Eingabetext analysiert und die dominante Sprache identifiziert. Der Konfidenzwert des vortrainierten Modells wird ebenfalls ausgegeben.

```
aws comprehend detect-dominant-language \
    --text "It is a beautiful day in Seattle."
```

Ausgabe:

```
{
    "Languages": [
        {
            "LanguageCode": "en",
            "Score": 0.9877256155014038
        }
```

```
    ]
}
```

Weitere Informationen finden Sie unter Dominante Sprache im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectDominantLanguagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLanguage {
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;
```

```
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient,
 String text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectDominantLanguagein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- Einzelheiten zur API finden Sie DetectDominantLanguagein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->detectdominantlanguage( iv_text = iv_text ).
    MESSAGE 'Languages detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DetectDominantLanguagein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectEntities`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectEntities` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the AmazonComprehend service detect any
/// entities in submitted text.
/// </summary>
public static class DetectEntities
{
    /// <summary>
    /// The main method calls the DetectEntitiesAsync method to find any
    /// entities in the sample code.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        Console.WriteLine("Calling DetectEntities\n");
        var detectEntitiesRequest = new DetectEntitiesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

        foreach (var e in detectEntitiesResponse.Entities)
        {
```

```
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
 {e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Einzelheiten zur API finden Sie DetectEntitiesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie benannte Entitäten im Eingabetext

Im folgenden Beispiel für `detect-entities` wird der Eingabetext analysiert und die benannten Entitäten werden zurückgegeben. Der Konfidenzwert des vortrainierten Modells wird ebenfalls für jede Prognose ausgegeben.

```
aws comprehend detect-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
    account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due by
 July 31st. Based on your autopay settings, \
    we will withdraw your payment on the due date from your bank account number
 XXXXXX1111 with the routing number XXXXX0000. \
    Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments to
 Alice at AnySpa@example.com."
```

Ausgabe:

```
{
    "Entities": [
        {
            "Score": 0.9994556307792664,
            "Type": "PERSON",
            "Text": "Zhang Wei",
            "BeginOffset": 6,
```

```
            "EndOffset": 15
        },
        {
            "Score": 0.9981022477149963,
            "Type": "PERSON",
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9986887574195862,
            "Type": "ORGANIZATION",
            "Text": "AnyCompany Financial Services, LLC",
            "BeginOffset": 33,
            "EndOffset": 67
        },
        {
            "Score": 0.9959119558334351,
            "Type": "OTHER",
            "Text": "1111-XXXX-1111-XXXX",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9708039164543152,
            "Type": "QUANTITY",
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9987268447875977,
            "Type": "DATE",
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9858865737915039,
            "Type": "OTHER",
            "Text": "XXXXXX1111",
            "BeginOffset": 271,
            "EndOffset": 281
        },
```

```
        {
            "Score": 0.9700471758842468,
            "Type": "OTHER",
            "Text": "XXXXX0000",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.9591118693351746,
            "Type": "ORGANIZATION",
            "Text": "Sunshine Spa",
            "BeginOffset": 340,
            "EndOffset": 352
        },
        {
            "Score": 0.9797496795654297,
            "Type": "LOCATION",
            "Text": "123 Main St",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.994929313659668,
            "Type": "PERSON",
            "Text": "Alice",
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9949769377708435,
            "Type": "OTHER",
            "Text": "AnySpa@example.com",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

Weitere Informationen finden Sie unter Entitäten im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectEntitiesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }
```

```
    public static void detectAllEntities(ComprehendClient comClient, String text)
 {
        try {
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectEntitiesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- Einzelheiten zur API finden Sie DetectEntitiesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->detectentities(
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'Entities detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
        MESSAGE 'Unsupported language.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetectEntitiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectKeyPhrases`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectKeyPhrases` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
```

```
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to
    /// search text for key phrases.
    /// </summary>
    public static class DetectKeyPhrase
    {
        /// <summary>
        /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
        /// to detect any key phrases in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
            foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score},
BeginOffset: {kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Einzelheiten zur API finden Sie DetectKeyPhrasesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie Schlüsselphrasen im Eingabetext

Im folgenden Beispiel für `detect-key-phrases` wird der Eingabetext analysiert und die wichtigsten Nominalphrasen werden zurückgegeben. Der Konfidenzwert des vortrainierten Modells wird ebenfalls für jede Prognose ausgegeben.

```
aws comprehend detect-key-phrases \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

Ausgabe:

```
{
    "KeyPhrases": [
        {
            "Score": 0.8996376395225525,
            "Text": "Zhang Wei",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9992469549179077,
            "Text": "John",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.988385021686554,
            "Text": "Your AnyCompany Financial Services",
            "BeginOffset": 28,
            "EndOffset": 62
        },
```

```
        {
            "Score": 0.8740853071212769,
            "Text": "LLC credit card account 1111-XXXX-1111-XXXX",
            "BeginOffset": 64,
            "EndOffset": 107
        },
        {
            "Score": 0.9999437928199768,
            "Text": "a minimum payment",
            "BeginOffset": 112,
            "EndOffset": 129
        },
        {
            "Score": 0.9998900890350342,
            "Text": ".53",
            "BeginOffset": 133,
            "EndOffset": 136
        },
        {
            "Score": 0.9979453086853027,
            "Text": "July 31st",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9983011484146118,
            "Text": "your autopay settings",
            "BeginOffset": 172,
            "EndOffset": 193
        },
        {
            "Score": 0.9996572136878967,
            "Text": "your payment",
            "BeginOffset": 211,
            "EndOffset": 223
        },
        {
            "Score": 0.9995037317276001,
            "Text": "the due date",
            "BeginOffset": 227,
            "EndOffset": 239
        },
        {
            "Score": 0.9702621698379517,
```

```
            "Text": "your bank account number XXXXXX1111",
            "BeginOffset": 245,
            "EndOffset": 280
        },
        {
            "Score": 0.9179925918579102,
            "Text": "the routing number XXXXX0000.Customer feedback",
            "BeginOffset": 286,
            "EndOffset": 332
        },
        {
            "Score": 0.9978160858154297,
            "Text": "Sunshine Spa",
            "BeginOffset": 337,
            "EndOffset": 349
        },
        {
            "Score": 0.9706913232803345,
            "Text": "123 Main St",
            "BeginOffset": 351,
            "EndOffset": 362
        },
        {
            "Score": 0.9941995143890381,
            "Text": "comments",
            "BeginOffset": 379,
            "EndOffset": 387
        },
        {
            "Score": 0.9759287238121033,
            "Text": "Alice",
            "BeginOffset": 391,
            "EndOffset": 396
        },
        {
            "Score": 0.8376792669296265,
            "Text": "AnySpa@example.com",
            "BeginOffset": 400,
            "EndOffset": 415
        }
    ]
}
```

Weitere Informationen finden Sie unter Schlüsselphrasen im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectKeyPhrasesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();
```

```
        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String
 text) {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectKeyPhrasesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
 its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases
```

- Einzelheiten zur API finden Sie DetectKeyPhrasesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->detectkeyphrases(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Key phrases detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DetectKeyPhrasesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectPiiEntities`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectPiiEntities` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to find
    /// personally identifiable information (PII) within text submitted to the
    /// DetectPiiEntitiesAsync method.
    /// </summary>
    public class DetectingPII
    {
        /// <summary>
        /// This method calls the DetectPiiEntitiesAsync method to locate any
        /// personally dientifiable information within the supplied text.
        /// </summary>
        public static async Task Main()
        {
            var comprehendClient = new AmazonComprehendClient();
            var text = @"Hello Paul Santos. The latest statement for your
                        credit card account 1111-0000-1111-0000 was
                        mailed to 123 Any Street, Seattle, WA 98109.";

            var request = new DetectPiiEntitiesRequest
            {
                Text = text,
                LanguageCode = "EN",
            };

            var response = await
comprehendClient.DetectPiiEntitiesAsync(request);
```

```
            if (response.Entities.Count > 0)
            {
                foreach (var entity in response.Entities)
                {
                    var entityValue = text.Substring(entity.BeginOffset,
 entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectPiiEntitiesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie PII-Entitäten im Eingabetext

Im folgenden Beispiel für `detect-pii-entities` wird der Eingabetext analysiert und es werden Entitäten identifiziert, die persönlich identifizierbare Informationen (PII) enthalten. Der Konfidenzwert des vortrainierten Modells wird ebenfalls für jede Prognose ausgegeben.

```
aws comprehend detect-pii-entities \
    --language-code en \
    --text "Hello Zhang Wei, I am John. Your AnyCompany Financial Services, LLC
 credit card \
        account 1111-XXXX-1111-XXXX has a minimum payment of $24.53 that is due
 by July 31st. Based on your autopay settings, \
        we will withdraw your payment on the due date from your bank account
 number XXXXXX1111 with the routing number XXXXX0000. \
        Customer feedback for Sunshine Spa, 123 Main St, Anywhere. Send comments
 to Alice at AnySpa@example.com."
```

Ausgabe:

```
{
    "Entities": [
        {
```

```
            "Score": 0.9998322129249573,
            "Type": "NAME",
            "BeginOffset": 6,
            "EndOffset": 15
        },
        {
            "Score": 0.9998878240585327,
            "Type": "NAME",
            "BeginOffset": 22,
            "EndOffset": 26
        },
        {
            "Score": 0.9994089603424072,
            "Type": "CREDIT_DEBIT_NUMBER",
            "BeginOffset": 88,
            "EndOffset": 107
        },
        {
            "Score": 0.9999760985374451,
            "Type": "DATE_TIME",
            "BeginOffset": 152,
            "EndOffset": 161
        },
        {
            "Score": 0.9999449253082275,
            "Type": "BANK_ACCOUNT_NUMBER",
            "BeginOffset": 271,
            "EndOffset": 281
        },
        {
            "Score": 0.9999847412109375,
            "Type": "BANK_ROUTING",
            "BeginOffset": 306,
            "EndOffset": 315
        },
        {
            "Score": 0.999925434589386,
            "Type": "ADDRESS",
            "BeginOffset": 354,
            "EndOffset": 365
        },
        {
            "Score": 0.9989161491394043,
            "Type": "NAME",
```

```
            "BeginOffset": 394,
            "EndOffset": 399
        },
        {
            "Score": 0.9994171857833862,
            "Type": "EMAIL",
            "BeginOffset": 403,
            "EndOffset": 418
        }
    ]
}
```

Weitere Informationen Sie unter Persönlich identifizierbare Informationen (PII) im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectPiiEntitiesin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
 be
        things like names, account numbers, or addresses.
```

```
        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- Einzelheiten zur API finden Sie DetectPiiEntitiesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->detectpiientities(
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'PII entities detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
```

```
        MESSAGE 'Unsupported language.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetectPiiEntitiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectSentiment`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectSentiment` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
```

```
    {
        /// <summary>
        /// This method calls the DetetectSentimentAsync method to analyze the
        /// supplied text and determine the overal sentiment.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
 AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            var detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectSentimentResponse = await
 comprehendClient.DetectSentimentAsync(detectSentimentRequest);
            Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
            Console.WriteLine("Done");
        }
    }
```

- Einzelheiten zur API finden Sie DetectSentimentin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie die Stimmung eines Eingabetextes

Im folgenden Beispiel für `detect-sentiment` wird der Eingabetext analysiert und eine Inferenz der vorherrschenden Stimmung (`POSITIVE`, `NEUTRAL`, `MIXED` oder `NEGATIVE`) wird zurückgegeben.

```
aws comprehend detect-sentiment \
    --language-code en \
```

```
    --text "It is a beautiful day in Seattle"
```

Ausgabe:

```
{
    "Sentiment": "POSITIVE",
    "SentimentScore": {
        "Positive": 0.9976957440376282,
        "Negative": 9.653854067437351e-05,
        "Neutral": 0.002169104292988777,
        "Mixed": 3.857641786453314e-05
    }
}
```

Weitere Informationen finden Sie unter Stimmung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectSentimentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text)
 {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectSentimentin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DetectSentimentin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->detectsentiment(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Sentiment detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DetectSentimentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectSyntax`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectSyntax` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erkennen von Dokumentenelementen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use Amazon Comprehend to detect syntax
    /// elements by calling the DetectSyntaxAsync method.
    /// </summary>
    public class DetectingSyntax
    {
        /// <summary>
        /// This method calls DetectSynaxAsync to identify the syntax elements
        /// in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntaxAsync\n");
            var detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
```

```
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
                Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
 {s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- Einzelheiten zur API finden Sie DetectSyntaxin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie die Wortarten in einem Eingabetext

Im folgenden Beispiel für `detect-syntax` wird die Syntax des Eingabetextes analysiert und die verschiedenen Wortarten werden zurückgegeben. Der Konfidenzwert des vortrainierten Modells wird ebenfalls für jede Prognose ausgegeben.

```
aws comprehend detect-syntax \
    --language-code en \
    --text "It is a beautiful day in Seattle."
```

Ausgabe:

```
{
    "SyntaxTokens": [
        {
            "TokenId": 1,
            "Text": "It",
            "BeginOffset": 0,
            "EndOffset": 2,
            "PartOfSpeech": {
                "Tag": "PRON",
                "Score": 0.9999740719795227
            }
        },
```

```
        {
            "TokenId": 2,
            "Text": "is",
            "BeginOffset": 3,
            "EndOffset": 5,
            "PartOfSpeech": {
                "Tag": "VERB",
                "Score": 0.999901294708252
            }
        },
        {
            "TokenId": 3,
            "Text": "a",
            "BeginOffset": 6,
            "EndOffset": 7,
            "PartOfSpeech": {
                "Tag": "DET",
                "Score": 0.9999938607215881
            }
        },
        {
            "TokenId": 4,
            "Text": "beautiful",
            "BeginOffset": 8,
            "EndOffset": 17,
            "PartOfSpeech": {
                "Tag": "ADJ",
                "Score": 0.9987351894378662
            }
        },
        {
            "TokenId": 5,
            "Text": "day",
            "BeginOffset": 18,
            "EndOffset": 21,
            "PartOfSpeech": {
                "Tag": "NOUN",
                "Score": 0.9999796748161316
            }
        },
        {
            "TokenId": 6,
            "Text": "in",
            "BeginOffset": 22,
```

```
            "EndOffset": 24,
            "PartOfSpeech": {
                "Tag": "ADP",
                "Score": 0.9998047947883606
            }
        },
        {
            "TokenId": 7,
            "Text": "Seattle",
            "BeginOffset": 25,
            "EndOffset": 32,
            "PartOfSpeech": {
                "Tag": "PROPN",
                "Score": 0.9940530061721802
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Syntaxanalyse im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie DetectSyntaxin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest =
 DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectSyntaxin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
```

```
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

- Einzelheiten zur API finden Sie DetectSyntaxin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->detectsyntax(
          iv_text = iv_text
          iv_languagecode = iv_language_code
        ).
        MESSAGE 'Syntax tokens detected.' TYPE 'I'.
      CATCH /aws1/cx_cpdtextsizelmtexcdex.
        MESSAGE 'Text size exceeds limit.' TYPE 'E'.
      CATCH /aws1/cx_cpdunsuppedlanguageex.
        MESSAGE 'Unsupported language.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetectSyntaxin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListDocumentClassificationJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDocumentClassificationJobs` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So listen Sie alle Aufträge zur Dokumentenklassifizierung auf

Im folgenden Beispiel für `list-document-classification-jobs` werden alle Aufträge zur Dokumentenklassifizierung aufgelistet.

```
aws comprehend list-document-classification-jobs
```

Ausgabe:

```
{
    "DocumentClassificationJobPropertiesList": [
        {
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName": "exampleclassificationjob",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:09:51.788000+00:00",
            "EndTime": "2023-06-14T17:15:58.582000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
```

```
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-e758dd56b824aa717ceab551f11749fb/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:1234567890101:document-
classification-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "exampleclassificationjob2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-14T17:22:39.829000+00:00",
            "EndTime": "2023-06-14T17:28:46.107000+00:00",
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:1234567890101:document-classifier/mymodel/version/12",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket/jobdata/",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/1234567890101-CLN-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "DataAccessRoleArn": "arn:aws:iam::1234567890101:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Klassifizierung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie ListDocumentClassificationJobsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- Einzelheiten zur API finden Sie ListDocumentClassificationJobsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->listdocclassificationjobs( ).
    MESSAGE 'Document classification jobs listed.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidfilterex.
    MESSAGE 'Invalid filter.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListDocumentClassificationJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListDocumentClassifiers`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDocumentClassifiers` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So listen Sie alle Dokumentenklassifikatoren auf

Im folgenden Beispiel für `list-document-classifiers` werden alle trainierten und im Training befindlichen Dokumentenklassifikatorenmodelle aufgelistet.

```
aws comprehend list-document-classifiers
```

Ausgabe:

```
{
    "DocumentClassifierPropertiesList": [
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier1",
            "LanguageCode": "en",
            "Status": "TRAINED",
            "SubmitTime": "2023-06-13T19:04:15.735000+00:00",
            "EndTime": "2023-06-13T19:42:31.752000+00:00",
            "TrainingStartTime": "2023-06-13T19:08:20.114000+00:00",
            "TrainingEndTime": "2023-06-13T19:41:35.080000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "ClassifierMetadata": {
                "NumberOfLabels": 3,
                "NumberOfTrainedDocuments": 5016,
                "NumberOfTestDocuments": 557,
                "EvaluationMetrics": {
                    "Accuracy": 0.9856,
                    "Precision": 0.9919,
                    "Recall": 0.9459,
                    "F1Score": 0.9673,
                    "MicroPrecision": 0.9856,
                    "MicroRecall": 0.9856,
                    "MicroF1Score": 0.9856,
                    "HammingLoss": 0.0144
                }
```

```
            },
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        },
        {
            "DocumentClassifierArn": "arn:aws:comprehend:us-
west-2:111122223333:document-classifier/exampleclassifier2",
            "LanguageCode": "en",
            "Status": "TRAINING",
            "SubmitTime": "2023-06-13T21:20:28.690000+00:00",
            "InputDataConfig": {
                "DataFormat": "COMPREHEND_CSV",
                "S3Uri": "s3://amzn-s3-demo-bucket/trainingdata"
            },
            "OutputDataConfig": {},
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-testorle",
            "Mode": "MULTI_CLASS"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen und Verwalten von benutzerdefinierten Modellen im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie ListDocumentClassifiersin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
```

```
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- Einzelheiten zur API finden Sie ListDocumentClassifiersin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_cpd->listdocumentclassifiers( ).
        MESSAGE 'Document classifiers listed.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidfilterex.
        MESSAGE 'Invalid filter.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListDocumentClassifiersin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTopicsDetectionJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTopicsDetectionJobs` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten

CLI

AWS CLI

So listen Sie alle Aufträge zur Themenerkennung auf

Im folgenden Beispiel für `list-topics-detection-jobs` werden alle laufenden und abgeschlossenen asynchronen Aufträge zur Themenerkennung aufgelistet.

```
aws comprehend list-topics-detection-jobs
```

Ausgabe:

```
{
    "TopicsDetectionJobPropertiesList": [
        {
```

```
            "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a11EXAMPLE",
            "JobName" "topic-analysis-1"
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:40:35.384000+00:00",
            "EndTime": "2023-06-09T18:46:41.936000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a11EXAMPLE/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE2",
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE2",
            "JobName": "topic-analysis-2",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2023-06-09T18:44:43.414000+00:00",
            "EndTime": "2023-06-09T18:50:50.872000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE2/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        },
        {
            "JobId": "123456abcdeb0e11022f22a1EXAMPLE3",
```

```
            "JobArn": "arn:aws:comprehend:us-west-2:111122223333:topics-
detection-job/123456abcdeb0e11022f22a1EXAMPLE3",
            "JobName": "topic-analysis-2",
            "JobStatus": "IN_PROGRESS",
            "SubmitTime": "2023-06-09T18:50:56.737000+00:00",
            "InputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-bucket",
                "InputFormat": "ONE_DOC_PER_LINE"
            },
            "OutputDataConfig": {
                "S3Uri": "s3://amzn-s3-demo-destination-bucket/
thefolder/111122223333-TOPICS-123456abcdeb0e11022f22a1EXAMPLE3/output/
output.tar.gz"
            },
            "NumberOfTopics": 10,
            "DataAccessRoleArn": "arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role"
        }
    ]
}
```

Weitere Informationen finden Sie unter Asynchrone Analyse für Amazon-Comprehend-Erkenntnisse im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie ListTopicsDetectionJobsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
```

```
        """
        self.comprehend_client = comprehend_client


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- Einzelheiten zur API finden Sie ListTopicsDetectionJobsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->listtopicsdetectionjobs( ).
        MESSAGE 'Topics detection jobs listed.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
```

```
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdinvalidfilterex.
        MESSAGE 'Invalid filter.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTopicsDetectionJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartDocumentClassificationJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartDocumentClassificationJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

CLI

AWS CLI

So starten Sie einen Auftrag zur Dokumentenklassifizierung

Im folgenden Beispiel für `start-document-classification-job` wird ein Auftrag zur Dokumentenklassifizierung mit einem benutzerdefinierten Modell für alle Dateien an der durch das Tag `--input-data-config` angegebenen Adresse gestartet. In diesem Beispiel enthält der S3-Eingabe-Bucket `SampleSMStext1.txt`, `SampleSMStext2.txt` und `SampleSMStext3.txt`. Das Modell wurde zuvor anhand von Dokumentenklassifizierungen von Spam- und Nicht-Spam-SMS-Nachrichten bzw. „betrügerischen“ SMS-Nachrichten trainiert. Wenn der Auftrag abgeschlossen ist, wird `output.tar.gz` an der durch das `--output-data-config` Tag angegebenen Stelle platziert. `output.tar.gz` enthält die Datei `predictions.jsonl`, in der die Klassifikation der einzelnen Dokumente aufgeführt ist. Die JSON-Ausgabe wird in einer Zeile pro Datei gedruckt, ist aber hier zur besseren Lesbarkeit formatiert.

```
aws comprehend start-document-classification-job \
```

```
    --job-name exampleclassificationjob \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket-INPUT/jobdata/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --document-classifier-arn arn:aws:comprehend:us-west-2:111122223333:document-
classifier/mymodel/version/12
```

Inhalt von `SampleSMStext1.txt`:

```
"CONGRATULATIONS! TXT 2155550100 to win $5000"
```

Inhalt von `SampleSMStext2.txt`:

```
"Hi, when do you want me to pick you up from practice?"
```

Inhalt von `SampleSMStext3.txt`:

```
"Plz send bank account # to 2155550100 to claim prize!!"
```

Ausgabe:

```
{
    "JobId": "e758dd56b824aa717ceab551fEXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:document-classification-
job/e758dd56b824aa717ceab551fEXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

Inhalt von `predictions.jsonl`:

```
{"File": "SampleSMSText1.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
{"File": "SampleSMStext2.txt", "Line": "0", "Classes": [{"Name": "ham", "Score":
 0.9994}, {"Name": "spam", "Score": 0.0006}]}
{"File": "SampleSMSText3.txt", "Line": "0", "Classes": [{"Name": "spam", "Score":
 0.9999}, {"Name": "ham", "Score": 0.0001}]}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Klassifizierung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie StartDocumentClassificationJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
 can
        call `describe_document_classification_job` to get job status until it
```

```
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie StartDocumentClassificationJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->startdocclassificationjob(
          iv_jobname = iv_job_name
          iv_documentclassifierarn = iv_classifier_arn
          io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
            iv_s3uri = iv_input_s3_uri
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
        ).
        MESSAGE 'Document classification job started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourceunavailex.
        MESSAGE 'Resource unavailable.' TYPE 'E'.
      CATCH /aws1/cx_cpdkmskeyvalidationex.
        MESSAGE 'KMS key validation error.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
```

```
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie StartDocumentClassificationJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartTopicsDetectionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartTopicsDetectionJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NEt
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
```

```
        /// <summary>
        /// This methos calls a topic detection job by calling the Amazon
        /// Comprehend StartTopicsDetectionJobRequest.
        /// </summary>
        public static async Task Main()
        {
            var comprehendClient = new AmazonComprehendClient();

            string inputS3Uri = "s3://input bucket/input path";
            InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
            string outputS3Uri = "s3://output bucket/output path";
            string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
            int numberOfTopics = 10;

            var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
            {
                InputDataConfig = new InputDataConfig()
                {
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };
```

```
            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
        private static void PrintJobProperties(TopicsDetectionJobProperties
props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics:
{props.NumberOfTopics}\nInputS3Uri: {props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- Einzelheiten zur API finden Sie StartTopicsDetectionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So starten Sie einen Analyseauftrag zur Themenerkennung

Im folgenden Beispiel für `start-topics-detection-job` wird ein asynchroner Auftrag zur Themenerkennung für alle Dateien gestartet, die sich an der durch das Tag `--input-data-config` angegebenen Adresse befinden. Wenn der Auftrag abgeschlossen ist, wird der Ordner `output` an dem durch das Tag `--ouput-data-config` angegebenen Ort platziert. `output` enthält die Dateien topic-terms.csv und doc-topics.csv. Die erste Ausgabedatei, topic-terms.csv, ist eine Liste von Themen in der Sammlung. Für jedes Thema enthält die Liste standardmäßig die wichtigsten Begriffe, sortiert nach Themen, entsprechend ihrer Gewichtung. In der zweiten Datei `doc-topics.csv` werden die Dokumente aufgeführt, die einem Thema zugeordnet sind, sowie der Anteil des Dokuments, der sich mit dem Thema befasst.

```
aws comprehend start-topics-detection-job \
    --job-name example_topics_detection_job \
    --language-code en \
    --input-data-config "S3Uri=s3://amzn-s3-demo-bucket/" \
    --output-data-config "S3Uri=s3://amzn-s3-demo-destination-bucket/testfolder/"
 \
    --data-access-role-arn arn:aws:iam::111122223333:role/service-role/
AmazonComprehendServiceRole-example-role \
    --language-code en
```

Ausgabe:

```
{
    "JobId": "123456abcdeb0e11022f22a11EXAMPLE",
    "JobArn": "arn:aws:comprehend:us-west-2:111122223333:key-phrases-detection-
job/123456abcdeb0e11022f22a11EXAMPLE",
    "JobStatus": "SUBMITTED"
}
```

Weitere Informationen finden Sie unter Themenmodellierung im Entwicklerhandbuch für Amazon Comprehend.

- Einzelheiten zur API finden Sie StartTopicsDetectionJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
 so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
```

```
                                bucket. If multiple objects have the same prefix,
all
                                of them are used.
        :param input_format: The format of the input data, either one document
per
                                file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                        grants Comprehend permission to read from
the
                                        input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie StartTopicsDetectionJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->starttopicsdetectionjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
            iv_s3uri = iv_input_s3_uri
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
        ).
        MESSAGE 'Topics detection job started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdkmskeyvalidationex.
        MESSAGE 'KMS key validation error.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartTopicsDetectionJobin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Comprehend AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Comprehend mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Comprehend oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen einer Amazon-Transcribe-Streaming-App
- Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher
- Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erkennen Sie Dokumentelemente mit Amazon Comprehend und einem AWS SDK
- Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mit einem AWS SDK
- Führen Sie einen Amazon Comprehend-Themenmodellierungsjob für Beispieldaten mit einem aus AWS SDK
- Trainieren Sie einen benutzerdefinierten Amazon Comprehend-Klassifikator und klassifizieren Sie Dokumente mithilfe eines AWS SDK

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher

Die folgenden Codebeispiele zeigen, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

Java

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel „Einen Amazon Lex-Chatbot erstellen“ im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

Java

SDK für Java 2.x

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK für Kotlin

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

Die folgenden Codebeispiele zeigen, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus ihrer Originalsprache übersetzt, ihre Stimmung ermittelt und aus dem übersetzten Text eine Audiodatei generiert.

.NET

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.

- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie der innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
```

```
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";
```

```
/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
```

```
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.

- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Erkennen Sie Dokumentelemente mit Amazon Comprehend und einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erkennen von Sprachen, Entitäten und Schlüsselbegriffen in einem Dokument
- Erkennen von persönlich identifizierbaren Informationen (PII) in einem Dokument
- Erkennen der Stimmung eines Dokuments
- Erkennen von Syntaxelementen in einem Dokument

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon-Comprehend-Aktionen umschließt.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
 places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
```

```
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and
its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can
be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
```

```
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
```

```
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

Rufen Sie Funktionen der Wrapper-Klasse auf, um Entitäten, Formulierungen und mehr in einem Dokument zu erkennen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])
```

```
    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

Python

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter TextractAndComprehendNotebook.ipynb.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Führen Sie einen Amazon Comprehend-Themenmodellierungsjob für Beispieldaten mit einem aus AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten
- Rufen Sie von Informationen über den Auftrag ab.
- Extrahieren von Auftragsausgabedaten aus Amazon S3

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Wrapper-Klasse, um Amazon-Comprehend-Themenmodellierungsaktionen aufzurufen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously,
 so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.
```

```
        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix,
all
                            of them are used.
        :param input_format: The format of the input data, either one document
per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                    grants Comprehend permission to read from
the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
```

```
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

Verwenden Sie die Wrapper-Klasse, um einen Themenmodellierungsauftrag auszuführen und Auftragsdaten abzurufen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    input_prefix = "input/"
    output_prefix = "output/"
```

```
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
    print(f"First {lines} lines of document topics output:")
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
```

```
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeTopicsDetectionJob
  - ListTopicsDetectionJobs
  - StartTopicsDetectionJob

Trainieren Sie einen benutzerdefinierten Amazon Comprehend-Klassifikator und klassifizieren Sie Dokumente mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Amazon-Comprehend-Klassifikators mit mehreren Labels
- Trainieren des Klassifikators anhand von Beispieldaten
- Ausführen eines Klassifizierungsauftrags für einen zweiten Datensatz
- Extrahieren der Auftragsausgabedaten aus Amazon S3

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Wrapper-Klasse, um die Aktionen des Amazon-Comprehend-Dokumentenklassifikators aufzurufen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it
 immediately
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier`
 function
        can be used to get training status and returns a status of TRAINED when
 the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the
 training
                             bucket. If multiple objects have the same prefix,
 all
                             of them are used.
```

```
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
 that
                                     grants Comprehend permission to read from
 the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/
{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
 self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
```

```
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.",
self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
```

```
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you
can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document
per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role
that
                                     grants Comprehend permission to read from
the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
```

```
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response =
self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
```

```
            )
            raise
        else:
            return jobs
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """

    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
```

```
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub
 API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
                    "labels": {label["name"] for label in issue["labels"]},
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
```

```
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
        """
        issues = []
        per_label_count = 5
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
        Uploads issue data to an Amazon S3 bucket, either for training or for
input.
        The data is first put into the format expected by Comprehend. For
training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
```

```
                    f"{'|'.join(issue['labels'])},{issue['title']}
{issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
            )
            raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the
classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
```

```
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

Trainieren Sie einen Klassifizierer anhand einer Reihe von GitHub Problemen mit bekannten Labels und senden Sie dann eine zweite Gruppe von GitHub Problemen an den Klassifizierer, damit sie gekennzeichnet werden können.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
```

```
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
```

```
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/
n)? ")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDocumentClassifier
  - DeleteDocumentClassifier
  - DescribeDocumentClassificationJob
  - DescribeDocumentClassifier
  - ListDocumentClassificationJobs
  - ListDocumentClassifiers
  - StartDocumentClassificationJob

Codebeispiele für AWS Config verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Config mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS Config Entwicklerhandbuch — Weitere Informationen zu AWS Config.
- AWS Config API-Referenz — Details zu allen verfügbaren AWS Config Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Config verwenden AWS SDKs
  - Aktionen für AWS Config verwenden AWS SDKs
    - Verwenden Sie mit DeleteConfigRuleeinem AWS SDK oder CLI
    - Verwendung von DescribeComplianceByConfigRule mit einer CLI
    - Verwendung von DescribeComplianceByResource mit einer CLI
    - Verwendung von DescribeConfigRuleEvaluationStatus mit einer CLI
    - Verwenden Sie mit einem DescribeConfigRules AWS SDK oder CLI
    - Verwendung von DescribeConfigurationRecorderStatus mit einer CLI
    - Verwendung von DescribeConfigurationRecorders mit einer CLI
    - Verwendung von DescribeDeliveryChannels mit einer CLI
    - Verwendung von GetComplianceDetailsByConfigRule mit einer CLI
    - Verwendung von GetComplianceDetailsByResource mit einer CLI
    - Verwendung von GetComplianceSummaryByConfigRule mit einer CLI
    - Verwendung von GetComplianceSummaryByResourceType mit einer CLI
    - Verwenden Sie mit einem PutConfigRule AWS SDK oder CLI
    - Verwendung von PutDeliveryChannel mit einer CLI
- Szenarien für AWS Config verwenden AWS SDKs

- Erste Schritte mit dem Konfigurationsmanagement

Grundlegende Beispiele für AWS Config verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Config mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS Config verwenden AWS SDKs
  - Verwenden Sie mit DeleteConfigRuleeinem AWS SDK oder CLI
  - Verwendung von DescribeComplianceByConfigRule mit einer CLI
  - Verwendung von DescribeComplianceByResource mit einer CLI
  - Verwendung von DescribeConfigRuleEvaluationStatus mit einer CLI
  - Verwenden Sie mit einem DescribeConfigRules AWS SDK oder CLI
  - Verwendung von DescribeConfigurationRecorderStatus mit einer CLI
  - Verwendung von DescribeConfigurationRecorders mit einer CLI
  - Verwendung von DescribeDeliveryChannels mit einer CLI
  - Verwendung von GetComplianceDetailsByConfigRule mit einer CLI
  - Verwendung von GetComplianceDetailsByResource mit einer CLI
  - Verwendung von GetComplianceSummaryByConfigRule mit einer CLI
  - Verwendung von GetComplianceSummaryByResourceType mit einer CLI
  - Verwenden Sie mit einem PutConfigRule AWS SDK oder CLI
  - Verwendung von PutDeliveryChannel mit einer CLI

Aktionen für AWS Config verwenden AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne AWS Config Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS Config API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS Config verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Config -API-Referenz.

Beispiele

- Verwenden Sie mit DeleteConfigRuleeinem AWS SDK oder CLI
- Verwendung von DescribeComplianceByConfigRule mit einer CLI
- Verwendung von DescribeComplianceByResource mit einer CLI
- Verwendung von DescribeConfigRuleEvaluationStatus mit einer CLI
- Verwenden Sie mit einem DescribeConfigRules AWS SDK oder CLI
- Verwendung von DescribeConfigurationRecorderStatus mit einer CLI
- Verwendung von DescribeConfigurationRecorders mit einer CLI
- Verwendung von DescribeDeliveryChannels mit einer CLI
- Verwendung von GetComplianceDetailsByConfigRule mit einer CLI
- Verwendung von GetComplianceDetailsByResource mit einer CLI
- Verwendung von GetComplianceSummaryByConfigRule mit einer CLI
- Verwendung von GetComplianceSummaryByResourceType mit einer CLI
- Verwenden Sie mit einem PutConfigRule AWS SDK oder CLI
- Verwendung von PutDeliveryChannel mit einer CLI

Verwenden Sie mit **`DeleteConfigRule`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteConfigRule` verwendet wird.

CLI

AWS CLI

Um eine AWS Config-Regel zu löschen

Der folgende Befehl löscht eine AWS Config-Regel mit dem Namen`MyConfigRule`:

```
aws configservice delete-config-rule --config-rule-name MyConfigRule
```

- Einzelheiten zur API finden Sie DeleteConfigRulein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def delete_config_rule(self, rule_name):
        """
        Delete the specified rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.config_client.delete_config_rule(ConfigRuleName=rule_name)
            logger.info("Deleted rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't delete rule %s.", rule_name)
            raise
```

- Einzelheiten zur API finden Sie DeleteConfigRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
lo_cfs->deleteconfigrule( iv_rule_name ).
MESSAGE 'Deleted AWS Config rule.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie DeleteConfigRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeComplianceByConfigRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeComplianceByConfigRule` verwendet wird.

CLI

AWS CLI

Um Compliance-Informationen für Ihre AWS Config-Regeln zu erhalten

Der folgende Befehl gibt Konformitätsinformationen für jede AWS Config-Regel zurück, gegen die eine oder mehrere AWS Ressourcen verstoßen:

```
aws configservice describe-compliance-by-config-rule --compliance-
types NON_COMPLIANT
```

In der Ausgabe gibt der Wert für jedes `CappedCount`-Attribut an, wie viele Ressourcen der zugehörigen Regel nicht entsprechen. Die folgende Ausgabe gibt beispielsweise an, dass 3 Ressourcen der Regel mit dem Namen `InstanceTypesAreT2micro` nicht entsprechen.

Ausgabe:

```
{
```

```
    "ComplianceByConfigRules": [
        {
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            },
            "ConfigRuleName": "InstanceTypesAreT2micro"
        },
        {
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 10,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            },
            "ConfigRuleName": "RequiredTagsForVolumes"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeComplianceByConfigRuleunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Konformitätsdetails für die Regel „ebs-optimized-instance“ abgerufen, für die derzeit keine Bewertungsergebnisse vorliegen. Daher wird „INSUFFICIENT_DATA“ zurückgegeben

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Ausgabe:

```
ComplianceContributorCount ComplianceType
------------------------- --------------
```

```
                              INSUFFICIENT_DATA
```

Beispiel 2: In diesem Beispiel wird die Anzahl der nicht konformen Ressourcen für die Regel „ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK“ zurückgegeben.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName
 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -ComplianceType
 NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Einzelheiten zur API finden Sie unter DescribeComplianceByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Konformitätsdetails für die Regel „ebs-optimized-instance“ abgerufen, für die derzeit keine Bewertungsergebnisse vorliegen. Daher wird „INSUFFICIENT_DATA“ zurückgegeben

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Ausgabe:

```
ComplianceContributorCount ComplianceType
-------------------------- --------------
                           INSUFFICIENT_DATA
```

Beispiel 2: In diesem Beispiel wird die Anzahl der nicht konformen Ressourcen für die Regel „ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK“ zurückgegeben.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName
 ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -ComplianceType
 NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- Einzelheiten zur API finden Sie unter DescribeComplianceByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeComplianceByResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeComplianceByResource` verwendet wird.

CLI

AWS CLI

Um Compliance-Informationen für Ihre AWS Ressourcen zu erhalten

Der folgende Befehl gibt Konformitätsinformationen für jede EC2-Instance zurück, die von AWS Config aufgezeichnet wurde und gegen eine oder mehrere Regeln verstößt:

```
aws configservice describe-compliance-by-resource --resource-
type AWS::EC2::Instance --compliance-types NON_COMPLIANT
```

In der Ausgabe gibt der Wert für jedes `CappedCount`-Attribut an, gegen wie viele Regeln die Ressource verstößt. Die folgende Ausgabe gibt beispielsweise an, dass die Instance `i-1a2b3c4d` gegen 2 Regeln verstößt.

Ausgabe:

```
{
    "ComplianceByResources": [
        {
            "ResourceType": "AWS::EC2::Instance",
            "ResourceId": "i-1a2b3c4d",
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 2,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
```

```
            }
        },
        {
            "ResourceType": "AWS::EC2::Instance",
            "ResourceId": "i-2a2b3c4d ",
            "Compliance": {
                "ComplianceContributorCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceType": "NON_COMPLIANT"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeComplianceByResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Ressourcentyp **`AWS::SSM::ManagedInstanceInventory`** auf den Konformitätstyp „COMPLIANT“ überprüft.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Ausgabe:

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Einzelheiten zur API finden Sie unter DescribeComplianceByResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Ressourcentyp **`AWS::SSM::ManagedInstanceInventory`** auf den Konformitätstyp „COMPLIANT“ überprüft.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Ausgabe:

```
Compliance                              ResourceId          ResourceType
----------                              ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- Einzelheiten zur API finden Sie unter DescribeComplianceByResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeConfigRuleEvaluationStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeConfigRuleEvaluationStatus` verwendet wird.

CLI

AWS CLI

So rufen Sie Statusinformationen für eine AWS Config-Regel ab

Der folgende Befehl gibt die Statusinformationen für eine AWS Config-Regel mit dem Namen zurück`MyConfigRule`:

```
aws configservice describe-config-rule-evaluation-status --config-rule-
names MyConfigRule
```

Ausgabe:

```
{
```

```
    "ConfigRulesEvaluationStatus": [
        {
            "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
            "FirstActivatedTime": 1450311703.844,
            "ConfigRuleId": "config-rule-abcdef",
            "LastSuccessfulInvocationTime": 1450314643.156,
            "ConfigRuleName": "MyConfigRule"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeConfigRuleEvaluationStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Statusinformationen für die angegebenen Konfigurationsregeln zurückgegeben.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Ausgabe:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
```

```
ConfigRuleId                  : config-rule-z1s23b
ConfigRuleName                : vpc-flow-logs-enabled
FirstActivatedTime            : 8/14/2019 6:23:44 AM
FirstEvaluationStarted        : True
LastErrorCode                 :
LastErrorMessage              :
LastFailedEvaluationTime      : 1/1/0001 12:00:00 AM
LastFailedInvocationTime      : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Einzelheiten zur API finden Sie unter DescribeConfigRuleEvaluationStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Statusinformationen für die angegebenen Konfigurationsregeln zurückgegeben.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Ausgabe:

```
ConfigRuleArn                 : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                  : config-rule-kvq1wk
ConfigRuleName                : root-account-mfa-enabled
FirstActivatedTime            : 8/27/2019 8:05:17 AM
FirstEvaluationStarted        : True
LastErrorCode                 :
LastErrorMessage              :
LastFailedEvaluationTime      : 1/1/0001 12:00:00 AM
LastFailedInvocationTime      : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                 : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                  : config-rule-z1s23b
ConfigRuleName                : vpc-flow-logs-enabled
FirstActivatedTime            : 8/14/2019 6:23:44 AM
FirstEvaluationStarted        : True
```

```
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- Einzelheiten zur API finden Sie unter DescribeConfigRuleEvaluationStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeConfigRules`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeConfigRules` verwendet wird.

CLI

AWS CLI

Um Details für eine AWS Config-Regel abzurufen

Der folgende Befehl gibt Details für eine AWS Config-Regel mit dem Namen zurück`InstanceTypesAreT2micro`:

```
aws configservice describe-config-rules --config-rule-
names InstanceTypesAreT2micro
```

Ausgabe:

```
{
    "ConfigRules": [
        {
            "ConfigRuleState": "ACTIVE",
            "Description": "Evaluates whether EC2 instances are the t2.micro
 type.",
            "ConfigRuleName": "InstanceTypesAreT2micro",
            "ConfigRuleArn": "arn:aws:config:us-east-1:123456789012:config-rule/
config-rule-abcdef",
            "Source": {
                "Owner": "CUSTOM_LAMBDA",
                "SourceIdentifier": "arn:aws:lambda:us-
east-1:123456789012:function:InstanceTypeCheck",
```

```
                "SourceDetails": [
                    {
                        "EventSource": "aws.config",
                        "MessageType": "ConfigurationItemChangeNotification"
                    }
                ]
            },
            "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}",
            "Scope": {
                "ComplianceResourceTypes": [
                    "AWS::EC2::Instance"
                ]
            },
            "ConfigRuleId": "config-rule-abcdef"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeConfigRulesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Konfigurationsregeln für das Konto mit ausgewählten Eigenschaften aufgeführt.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Ausgabe:

```
ConfigRuleName                                     ConfigRuleId
 ConfigRuleArn
 ConfigRuleState
--------------                                     ------------
 -------------
 ---------------
ALB_REDIRECTION_CHECK                              config-rule-12iyn3
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-12iyn3
 ACTIVE
```

```
access-keys-rotated                                 config-rule-aospfr
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-aospfr
 ACTIVE
autoscaling-group-elb-healthcheck-required          config-rule-cn1f2x
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x
 ACTIVE
```

- Einzelheiten zur API finden Sie unter DescribeConfigRules AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Konfigurationsregeln für das Konto mit ausgewählten Eigenschaften aufgeführt.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Ausgabe:

```
ConfigRuleName                                      ConfigRuleId
 ConfigRuleArn
 ConfigRuleState
--------------                                      ------------
 -------------
 ---------------
ALB_REDIRECTION_CHECK                               config-rule-12iyn3
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-12iyn3
 ACTIVE
access-keys-rotated                                 config-rule-aospfr
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-aospfr
 ACTIVE
autoscaling-group-elb-healthcheck-required          config-rule-cn1f2x
 arn:aws:config-service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x
 ACTIVE
```

- Einzelheiten zur API finden Sie unter DescribeConfigRules AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def describe_config_rule(self, rule_name):
        """
        Gets data for the specified rule.

        :param rule_name: The name of the rule to retrieve.
        :return: The rule data.
        """
        try:
            response = self.config_client.describe_config_rules(
                ConfigRuleNames=[rule_name]
            )
            rule = response["ConfigRules"]
            logger.info("Got data for rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't get data for rule %s.", rule_name)
            raise
        else:
            return rule
```

- Einzelheiten zur API finden Sie DescribeConfigRulesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA(lo_result) = lo_cfs->describeconfigrules(
      it_configrulenames = VALUE /aws1/
cl_cfsconfigrulenames_w=>tt_configrulenames(
        ( NEW /aws1/cl_cfsconfigrulenames_w( iv_rule_name ) )
      )
    ).
    ot_cfg_rules = lo_result->get_configrules( ).
    MESSAGE 'Retrieved AWS Config rule data.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie DescribeConfigRulesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeConfigurationRecorderStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeConfigurationRecorderStatus` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Konfigurationsmanagement

CLI

AWS CLI

So rufen Sie Statusinformationen für den Konfigurations-Recorder ab

Der folgende Befehl gibt den Status des Standardkonfigurations-Recorders zurück:

```
aws configservice describe-configuration-recorder-status
```

Ausgabe:

```
{
    "ConfigurationRecordersStatus": [
        {
            "name": "default",
            "lastStatus": "SUCCESS",
            "recording": true,
            "lastStatusChangeTime": 1452193834.344,
            "lastStartTime": 1441039997.819,
            "lastStopTime": 1441039992.835
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeConfigurationRecorderStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt den Status der Konfigurations-Recorder zurück.

```
Get-CFGConfigurationRecorderStatus
```

Ausgabe:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
```

```
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorderStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt den Status der Konfigurations-Recorder zurück.

```
Get-CFGConfigurationRecorderStatus
```

Ausgabe:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorderStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeConfigurationRecorders`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeConfigurationRecorders` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Konfigurationsmanagement

CLI

AWS CLI

So rufen Sie Details zum Konfigurations-Recorder ab

Der folgende Befehl gibt Details zum Standardkonfigurations-Recorder zurück:

```
aws configservice describe-configuration-recorders
```

Ausgabe:

```
{
    "ConfigurationRecorders": [
        {
            "recordingGroup": {
                "allSupported": true,
                "resourceTypes": [],
                "includeGlobalResourceTypes": true
            },
            "roleARN": "arn:aws:iam::123456789012:role/config-ConfigRole-
A1B2C3D4E5F6",
            "name": "default"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeConfigurationRecordersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details von Konfigurations-Recorders zurückgegeben.

```
Get-CFGConfigurationRecorder | Format-List
```

Ausgabe:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorders AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details von Konfigurations-Recorders zurückgegeben.

```
Get-CFGConfigurationRecorder | Format-List
```

Ausgabe:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- Einzelheiten zur API finden Sie unter DescribeConfigurationRecorders AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeDeliveryChannels`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDeliveryChannels` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Konfigurationsmanagement

CLI

AWS CLI

So rufen Sie Details über den Bereitstellungskanal ab

Der folgende Befehl gibt Details über den Bereitstellungskanal zurück:

```
aws configservice describe-delivery-channels
```

Ausgabe:

```
{
    "DeliveryChannels": [
        {
            "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
            "name": "default",
            "s3BucketName": "config-bucket-123456789012"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeDeliveryChannelsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Lieferkanal für die Region abgerufen und Details angezeigt.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Ausgabe:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Einzelheiten zur API finden Sie unter DescribeDeliveryChannels AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Lieferkanal für die Region abgerufen und Details angezeigt.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Ausgabe:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- Einzelheiten zur API finden Sie unter DescribeDeliveryChannels AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetComplianceDetailsByConfigRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetComplianceDetailsByConfigRule` verwendet wird.

CLI

AWS CLI

Um die Auswertungsergebnisse für eine AWS Config-Regel abzurufen

Der folgende Befehl gibt die Auswertungsergebnisse für alle Ressourcen zurück, die nicht einer AWS Config-Regel mit dem Namen entsprechen`InstanceTypesAreT2micro`:

```
aws configservice get-compliance-details-by-config-rule --config-rule-
name InstanceTypesAreT2micro --compliance-types NON_COMPLIANT
```

Ausgabe:

```
{
    "EvaluationResults": [
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314645.261,
            "ConfigRuleInvokedTime": 1450314642.948,
            "ComplianceType": "NON_COMPLIANT"
```

```
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-2a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314645.18,
            "ConfigRuleInvokedTime": 1450314642.902,
            "ComplianceType": "NON_COMPLIANT"
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-3a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314643.346,
            "ConfigRuleInvokedTime": 1450314643.124,
            "ComplianceType": "NON_COMPLIANT"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetComplianceDetailsByConfigRulein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Auswertungsergebnisse für die Regel access-keys-rotated abgerufen und die Ausgabe nach Compliance-Typ gruppiert zurückgegeben

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Beispiel 2: In diesem Beispiel werden Konformitätsdetails für die Regel access-keys-rotated für COMPLIANT-Ressourcen abgefragt.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Ausgabe:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Auswertungsergebnisse für die Regel access-keys-rotated abgerufen und die Ausgabe nach Compliance-Typ gruppiert zurückgegeben

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
```

```
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Beispiel 2: In diesem Beispiel werden Konformitätsdetails für die Regel access-keys-rotated für COMPLIANT-Ressourcen abgefragt.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Ausgabe:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetComplianceDetailsByResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetComplianceDetailsByResource` verwendet wird.

CLI

AWS CLI

Um die Evaluierungsergebnisse für eine AWS Ressource abzurufen

Der folgende Befehl gibt die Auswertungsergebnisse für jede Regel zurück, der die EC2-Instance `i-1a2b3c4d` nicht entspricht:

```
aws configservice get-compliance-details-by-resource --resource-
type AWS::EC2::Instance --resource-id i-1a2b3c4d --compliance-types NON_COMPLIANT
```

Ausgabe:

```
{
    "EvaluationResults": [
```

```
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "InstanceTypesAreT2micro"
                }
            },
            "ResultRecordedTime": 1450314643.288,
            "ConfigRuleInvokedTime": 1450314643.034,
            "ComplianceType": "NON_COMPLIANT"
        },
        {
            "EvaluationResultIdentifier": {
                "OrderingTimestamp": 1450314635.065,
                "EvaluationResultQualifier": {
                    "ResourceType": "AWS::EC2::Instance",
                    "ResourceId": "i-1a2b3c4d",
                    "ConfigRuleName": "RequiredTagForEC2Instances"
                }
            },
            "ResultRecordedTime": 1450314645.261,
            "ConfigRuleInvokedTime": 1450314642.948,
            "ComplianceType": "NON_COMPLIANT"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetComplianceDetailsByResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel für die Auswertungsergebnisse für die angegebene Ressource.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -
ResourceType 'AWS::IAM::User'
```

Ausgabe:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier :
 Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel für die Auswertungsergebnisse für die angegebene Ressource.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -
ResourceType 'AWS::IAM::User'
```

Ausgabe:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier :
 Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- Einzelheiten zur API finden Sie unter GetComplianceDetailsByResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetComplianceSummaryByConfigRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetComplianceSummaryByConfigRule` verwendet wird.

CLI

AWS CLI

Um die Konformitätsübersicht für Ihre AWS Config-Regeln abzurufen

Der folgende Befehl gibt die Anzahl der Regeln zurück, die konform sind, und die Anzahl derer, die nicht konform sind:

```
aws configservice get-compliance-summary-by-config-rule
```

In der Ausgabe gibt der Wert für jedes `CappedCount`-Attribut an, wie viele Regeln konform bzw. nicht konform sind.

Ausgabe:

```
{
    "ComplianceSummary": {
        "NonCompliantResourceCount": {
            "CappedCount": 3,
            "CapExceeded": false
        },
        "ComplianceSummaryTimestamp": 1452204131.493,
        "CompliantResourceCount": {
            "CappedCount": 2,
            "CapExceeded": false
        }
    }
}
```

- Einzelheiten zur API finden Sie GetComplianceSummaryByConfigRulein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt die Anzahl der Config-Regeln zurück, die nicht konform sind.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt die Anzahl der Config-Regeln zurück, die nicht konform sind.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByConfigRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetComplianceSummaryByResourceType`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetComplianceSummaryByResourceType` verwendet wird.

CLI

AWS CLI

So rufen Sie die Compliance-Zusammenfassung für alle Ressourcentypen ab

Der folgende Befehl gibt die Anzahl der AWS Ressourcen zurück, die nicht konform sind, und die Anzahl, die konform sind:

```
aws configservice get-compliance-summary-by-resource-type
```

In der Ausgabe gibt der Wert für jedes `CappedCount`-Attribut an, wie viele Ressourcen konform bzw. nicht konform sind.

Ausgabe:

```
{
    "ComplianceSummariesByResourceType": [
```

```
        {
            "ComplianceSummary": {
                "NonCompliantResourceCount": {
                    "CappedCount": 16,
                    "CapExceeded": false
                },
                "ComplianceSummaryTimestamp": 1453237464.543,
                "CompliantResourceCount": {
                    "CappedCount": 10,
                    "CapExceeded": false
                }
            }
        }
    ]
}
```

So rufen Sie die Compliance-Zusammenfassung für einen bestimmten Ressourcentyp ab

Der folgende Befehl gibt die Anzahl der EC2-Instances zurück, die nicht konform sind, und die Anzahl derer, die konform sind:

```
aws configservice get-compliance-summary-by-resource-type --resource-
types AWS::EC2::Instance
```

In der Ausgabe gibt der Wert für jedes `CappedCount`-Attribut an, wie viele Ressourcen konform bzw. nicht konform sind.

Ausgabe:

```
{
    "ComplianceSummariesByResourceType": [
        {
            "ResourceType": "AWS::EC2::Instance",
            "ComplianceSummary": {
                "NonCompliantResourceCount": {
                    "CappedCount": 3,
                    "CapExceeded": false
                },
                "ComplianceSummaryTimestamp": 1452204923.518,
                "CompliantResourceCount": {
                    "CappedCount": 7,
                    "CapExceeded": false
                }
```

```
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByResourceType AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt die Anzahl der Ressourcen zurück, die konform oder nicht konform sind, und konvertiert die Ausgabe in JSON.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByResourceType AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt die Anzahl der Ressourcen zurück, die konform oder nicht konform sind, und konvertiert die Ausgabe in JSON.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
```

```
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- Einzelheiten zur API finden Sie unter GetComplianceSummaryByResourceType AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutConfigRule`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutConfigRule` verwendet wird.

CLI

AWS CLI

So fügen Sie eine AWS verwaltete Konfigurationsregel hinzu

Der folgende Befehl stellt JSON-Code zum Hinzufügen einer AWS verwalteten Config-Regel bereit:

```
aws configservice put-config-rule --config-rule file://
RequiredTagsForEC2Instances.json
```

`RequiredTagsForEC2Instances.json` ist eine JSON-Datei, die die Regelkonfiguration enthält:

```
{
  "ConfigRuleName": "RequiredTagsForEC2Instances",
  "Description": "Checks whether the CostCenter and Owner tags are applied to EC2
 instances.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "AWS",
```

```
    "SourceIdentifier": "REQUIRED_TAGS"
  },
  "InputParameters": "{\"tag1Key\":\"CostCenter\",\"tag2Key\":\"Owner\"}"
}
```

Für das `ComplianceResourceTypes` Attribut beschränkt dieser JSON-Code den Bereich auf Ressourcen des `AWS::EC2::Instance` Typs, sodass AWS Config nur EC2-Instances anhand der Regel auswertet. Da es sich bei der Regel um eine verwaltete Regel handelt, ist das `Owner`-Attribut auf `AWS` und das `SourceIdentifier`-Attribut auf den Regelbezeichner `REQUIRED_TAGS` festgelegt. Für das `InputParameters`-Attribut werden die Tag-Schlüssel, `Owner` und `CostCenter`, die die Regel benötigt, angegeben.

Wenn der Befehl erfolgreich ist, gibt AWS Config keine Ausgabe zurück. Zur Überprüfung der Regelkonfiguration führen Sie den Befehl describe-config-rules aus und geben den Regelnamen an.

So fügen Sie eine vom Kunden verwaltete Konfigurationsregel hinzu

Der folgende Befehl stellt JSON-Code zum Hinzufügen einer vom Kunden verwalteten Konfigurationsregel bereit:

```
aws configservice put-config-rule --config-rule file://
InstanceTypesAreT2micro.json
```

`InstanceTypesAreT2micro.json` ist eine JSON-Datei, die die Regelkonfiguration enthält:

```
{
  "ConfigRuleName": "InstanceTypesAreT2micro",
  "Description": "Evaluates whether EC2 instances are the t2.micro type.",
  "Scope": {
    "ComplianceResourceTypes": [
      "AWS::EC2::Instance"
    ]
  },
  "Source": {
    "Owner": "CUSTOM_LAMBDA",
    "SourceIdentifier": "arn:aws:lambda:us-
east-1:123456789012:function:InstanceTypeCheck",
    "SourceDetails": [
      {
        "EventSource": "aws.config",
        "MessageType": "ConfigurationItemChangeNotification"
```

```
      }
    ]
  },
  "InputParameters": "{\"desiredInstanceType\":\"t2.micro\"}"
}
```

Für das `ComplianceResourceTypes` Attribut beschränkt dieser JSON-Code den Bereich auf Ressourcen des `AWS::EC2::Instance` Typs, sodass AWS Config nur EC2-Instances anhand der Regel auswertet. Da es sich bei dieser Regel um eine vom Kunden verwaltete Regel handelt`CUSTOM_LAMBDA`, ist das `Owner` Attribut auf und das `SourceIdentifier` Attribut auf den ARN der AWS Lambda-Funktion gesetzt. Das `SourceDetails`-Objekt ist erforderlich. Die für das `InputParameters` Attribut angegebenen Parameter werden an die AWS Lambda-Funktion übergeben, wenn AWS Config sie aufruft, um Ressourcen anhand der Regel auszuwerten.

Wenn der Befehl erfolgreich ist, gibt AWS Config keine Ausgabe zurück. Zur Überprüfung der Regelkonfiguration führen Sie den Befehl describe-config-rules aus und geben den Regelnamen an.

- Einzelheiten zur API finden Sie PutConfigRulein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
```

```
        self.config_client = config_client


    def put_config_rule(self, rule_name):
        """
        Sets a configuration rule that prohibits making Amazon S3 buckets
publicly
        readable.

        :param rule_name: The name to give the rule.
        """
        try:
            self.config_client.put_config_rule(
                ConfigRule={
                    "ConfigRuleName": rule_name,
                    "Description": "S3 Public Read Prohibited Bucket Rule",
                    "Scope": {
                        "ComplianceResourceTypes": [
                            "AWS::S3::Bucket",
                        ],
                    },
                    "Source": {
                        "Owner": "AWS",
                        "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED",
                    },
                    "InputParameters": "{}",
                    "ConfigRuleState": "ACTIVE",
                }
            )
            logger.info("Created configuration rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't create configuration rule %s.", rule_name)
            raise
```

- Einzelheiten zur API finden Sie PutConfigRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create a config rule for S3 bucket public read prohibition
    lo_cfs->putconfigrule(
      io_configrule = NEW /aws1/cl_cfsconfigrule(
        iv_configrulename = iv_rule_name
        iv_description = |S3 Public Read Prohibited Bucket Rule|
        io_scope = NEW /aws1/cl_cfsscope(
          it_complianceresourcetypes = VALUE /aws1/
cl_cfscplncresrctypes_w=>tt_complianceresourcetypes(
            ( NEW /aws1/cl_cfscplncresrctypes_w( |AWS::S3::Bucket| ) )
          )
        )
        io_source = NEW /aws1/cl_cfssource(
          iv_owner = |AWS|
          iv_sourceidentifier = |S3_BUCKET_PUBLIC_READ_PROHIBITED|
        )
        iv_inputparameters = '{}'
        iv_configrulestate = |ACTIVE|
      )
    ).
    MESSAGE 'Created AWS Config rule.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie PutConfigRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`PutDeliveryChannel`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutDeliveryChannel` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Konfigurationsmanagement

CLI

AWS CLI

So erstellen Sie einen Bereitstellungskanal

Der folgende Befehl stellt die Einstellungen für den Bereitstellungskanal als JSON-Code zur Verfügung:

```
aws configservice put-delivery-channel --delivery-channel file://
deliveryChannel.json
```

Die Datei `deliveryChannel.json` gibt die Attribute des Bereitstellungskanals an:

```
{
    "name": "default",
    "s3BucketName": "config-bucket-123456789012",
    "snsTopicARN": "arn:aws:sns:us-east-1:123456789012:config-topic",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Twelve_Hours"
    }
}
```

In diesem Beispiel werden die folgenden Attribute festgelegt:

`name`: Der Name des Übermittlungskanals. Standardmäßig weist AWS Config den Namen einem neuen Lieferkanal `default` zu. Sie können den Namen des Lieferkanals nicht mit dem Befehl aktualisieren. `put-delivery-channel` Die Schritte zum Ändern des Namens finden Sie unter „Umbenennen des Bereitstellungskanals“.`s3BucketName` - Der Name des Amazon S3 S3-Buckets, für den AWS Config Konfigurations-Snapshots und Konfigurationsverlaufsdateien bereitstellt. Wenn Sie einen Bucket angeben, der zu einem anderen AWS Konto gehört, muss dieser Bucket über Richtlinien verfügen, die Config Zugriffsberechtigungen gewähren. AWS Weitere Informationen finden Sie unter „Berechtigungen für den Amazon-S3-Bucket“.

`snsTopicARN`— Der Amazon-Ressourcenname (ARN) des Amazon SNS-Themas, an das AWS Config Benachrichtigungen über Konfigurationsänderungen sendet. Wenn Sie ein

Thema aus einem anderen Konto auswählen, muss das Thema über Richtlinien verfügen, die Config Zugriffsberechtigungen gewähren. AWS Weitere Informationen finden Sie unter „Berechtigungen für das Amazon-SNS-Thema“.

`configSnapshotDeliveryProperties`- Enthält das `deliveryFrequency` Attribut, das festlegt, wie oft AWS Config Konfigurations-Snapshots liefert und wie oft es Evaluierungen für periodische Config-Regeln aufruft.

Wenn der Befehl erfolgreich ist, gibt AWS Config keine Ausgabe zurück. Führen Sie den Befehl describe-delivery-channels aus, um die Einstellungen Ihres Bereitstellungskanals zu überprüfen.

- Einzelheiten zur API finden Sie PutDeliveryChannelin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die deliveryFrequency-Eigenschaft eines vorhandenen Lieferkanals geändert.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName
 amzn-s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Einzelheiten zur API finden Sie unter PutDeliveryChannel AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die deliveryFrequency-Eigenschaft eines vorhandenen Lieferkanals geändert.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName
 amzn-s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- Einzelheiten zur API finden Sie unter PutDeliveryChannel AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für AWS Config verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS Config mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS Config oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit dem Konfigurationsmanagement

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
```

```
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
```

```
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
```

```
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
```

```
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
```

```
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
```

```
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
```

```
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
```

```
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
```

```
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole

- CreateTopic
- DeleteBucket
- DeleteConfigurationRecorder
- DeleteDeliveryChannel
- DeleteRole
- DeleteRolePolicy
- DeleteTopic
- DescribeConfigurationRecorderStatus
- DescribeConfigurationRecorders
- DescribeDeliveryChannels
- DetachRolePolicy
- GetCallerIdentity
- PutConfigurationRecorder
- PutDeliveryChannel
- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Codebeispiele für Connect Customer mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Connect Customer mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Connect-Kundenadministratorhandbuch — Weitere Informationen zu Connect Customer.
- Connect-Kunden-API-Referenz — Details zu allen verfügbaren Connect-Kundenaktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Connect Customer AWS SDKs
  - Eine Contact-Center-Instanz erstellen

Szenarien für die Verwendung von Connect Customer AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Connect Customer mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Connect Customer oder in Kombination mit anderen aufrufen AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Eine Contact-Center-Instanz erstellen

Eine Contact-Center-Instanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Amazon-Connect-Instance
- Konfigurieren Sie einen Administratorbenutzer
- Konfigurieren Sie Telefonieoptionen
- Richten Sie eine Telefonnummer ein
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon Connect instance using AWS CLI
# This script follows the steps in the Amazon Connect instance creation tutorial

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="connect-instance-creation.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting Amazon Connect instance creation at $(date)" > "$LOG_FILE"

# Set default region
AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"

# Validate AWS CLI is installed and credentials are available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid" | tee -a
 "$LOG_FILE"
    exit 1
fi

# Function to log commands and their output
log_cmd() {
```

```
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    if [[ -n "${INSTANCE_ID:-}" ]]; then
        echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a
 "$LOG_FILE"
        log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --
region '$AWS_REGION'" || true
    fi

    echo "Cleanup completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
}

# Set trap to clean up on error
trap cleanup_on_error ERR EXIT

# Function to wait for instance to be fully active
wait_for_instance() {
    local instance_id="$1"
    local max_attempts=30
    local attempt=1
```

```
    echo "Waiting for instance $instance_id to become fully active..." | tee -a
 "$LOG_FILE"

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt of $max_attempts: Checking instance status..." |
 tee -a "$LOG_FILE"

        # Try to describe the instance
        local result
        result=$(log_cmd "aws connect describe-instance --instance-id
 '$instance_id' --region '$AWS_REGION' --output json" 2>&1) || true

        # Check if the command was successful and instance status is ACTIVE
        if [[ $? -eq 0 && "$result" =~ "ACTIVE" ]]; then
            echo "Instance is now fully active and ready to use." | tee -a
 "$LOG_FILE"
            return 0
        fi

        echo "Instance not fully active yet. Waiting 30 seconds before next
 check..." | tee -a "$LOG_FILE"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for instance to become fully active." | tee -a
 "$LOG_FILE"
    return 1
}

# Function to check and handle existing instances
check_existing_instances() {
    echo "Checking for existing Amazon Connect instances..." | tee -a "$LOG_FILE"

    local instances
    instances=$(log_cmd "aws connect list-instances --region '$AWS_REGION' --
output json" 2>&1) || true
    if [[ $? -ne 0 ]]; then
        echo "ERROR: Failed to list existing instances" | tee -a "$LOG_FILE"
        return 1
    fi

    # Check if there are any instances
    local instance_count
```

```
    instance_count=$(echo "$instances" | jq '.InstanceSummaryList | length' 2>/
dev/null || echo 0)

    if [[ $instance_count -gt 0 ]]; then
        echo "Found $instance_count existing Amazon Connect instance(s)" | tee -a
 "$LOG_FILE"
        echo "$instances" | jq '.InstanceSummaryList[] | {Id, Alias}' 2>/dev/null
 | tee -a "$LOG_FILE" || true

        echo ""
        echo "============================================"
        echo "EXISTING INSTANCES FOUND"
        echo "============================================"
        echo "Found $instance_count existing Amazon Connect instance(s)."
        echo "Auto-deleting existing instances to free up quota..." | tee -a
 "$LOG_FILE"

        echo "Deleting existing instances..." | tee -a "$LOG_FILE"

        # Extract instance IDs and delete each one
        local instance_ids
        instance_ids=$(echo "$instances" | jq -r '.InstanceSummaryList[].Id' 2>/
dev/null || echo "")

        while IFS= read -r id; do
            if [[ -n "$id" ]]; then
                echo "Deleting instance: $id" | tee -a "$LOG_FILE"
                log_cmd "aws connect delete-instance --instance-id '$id' --region
 '$AWS_REGION'" || {
                    echo "WARNING: Failed to delete instance $id" | tee -a
 "$LOG_FILE"
                }

                # Wait a bit between deletions
                sleep 5
            fi
        done <<< "$instance_ids"

        echo "Waiting for deletions to complete..." | tee -a "$LOG_FILE"
        sleep 30
    else
        echo "No existing Amazon Connect instances found" | tee -a "$LOG_FILE"
    fi
```

```
    return 0
}

# Check for existing instances before proceeding
check_existing_instances

# Generate a random instance alias to avoid naming conflicts
INSTANCE_ALIAS="connect-instance-$(openssl rand -hex 6 2>/dev/null || date +%s
%N)"
echo "Using instance alias: $INSTANCE_ALIAS" | tee -a "$LOG_FILE"

# Step 1: Create Amazon Connect instance
echo "Step 1: Creating Amazon Connect instance..." | tee -a "$LOG_FILE"
INSTANCE_RESULT=$(log_cmd "aws connect create-instance --identity-management-type
 CONNECT_MANAGED --instance-alias '$INSTANCE_ALIAS' --inbound-calls-enabled --
outbound-calls-enabled --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_RESULT" $? "Failed to create Amazon Connect
 instance"; then
    # Check if the error is due to quota limit
    if [[ "$INSTANCE_RESULT" =~ "ServiceQuotaExceededException" ||
 "$INSTANCE_RESULT" =~ "Quota limit reached" ]]; then
        echo "Quota limit reached for Amazon Connect instances. Please delete
 existing instances or request a quota increase." | tee -a "$LOG_FILE"
    fi
    exit 1
fi

# Extract instance ID from the result using jq
INSTANCE_ID=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceId' 2>/dev/null || echo
 "")
INSTANCE_ARN=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceArn' 2>/dev/null || echo
 "")

if [[ -z "$INSTANCE_ID" ]]; then
    echo "ERROR: Failed to extract instance ID from the result" | tee -a
 "$LOG_FILE"
    exit 1
fi

echo "Successfully created Amazon Connect instance with ID: $INSTANCE_ID" | tee -
a "$LOG_FILE"
echo "Instance ARN: $INSTANCE_ARN" | tee -a "$LOG_FILE"
```

```
# Tag the instance
aws connect tag-resource --resource-arn "$INSTANCE_ARN" \
    --tags project=doc-smith,tutorial=connect-gs \
    --region "$AWS_REGION"

# Wait for the instance to be fully created and active
if ! wait_for_instance "$INSTANCE_ID"; then
    echo "ERROR: Instance did not become fully active within the timeout period"
 | tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: Get security profiles to find the Admin profile ID
echo "Step 2: Getting security profiles..." | tee -a "$LOG_FILE"
SECURITY_PROFILES=$(log_cmd "aws connect list-security-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$SECURITY_PROFILES" $? "Failed to list security profiles"; then
    exit 1
fi

# Extract Admin security profile ID using jq
ADMIN_PROFILE_ID=$(echo "$SECURITY_PROFILES" | jq -r
 '.SecurityProfileSummaryList[] | select(.Name=="Admin") | .Id' 2>/dev/null |
 head -1 || echo "")

if [[ -z "$ADMIN_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find Admin security profile ID" | tee -a "$LOG_FILE"
    echo "Available security profiles:" | tee -a "$LOG_FILE"
    echo "$SECURITY_PROFILES" | jq '.SecurityProfileSummaryList[] | {Id,
 Name}' 2>/dev/null | tee -a "$LOG_FILE" || echo "$SECURITY_PROFILES" | tee -a
 "$LOG_FILE"
    exit 1
fi

echo "Found Admin security profile ID: $ADMIN_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 3: Get routing profiles to find a default routing profile ID
echo "Step 3: Getting routing profiles..." | tee -a "$LOG_FILE"
ROUTING_PROFILES=$(log_cmd "aws connect list-routing-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$ROUTING_PROFILES" $? "Failed to list routing profiles"; then
    exit 1
```

```
fi

# Extract the first routing profile ID using jq
ROUTING_PROFILE_ID=$(echo "$ROUTING_PROFILES" | jq -r
 '.RoutingProfileSummaryList[0].Id' 2>/dev/null || echo "")

if [[ -z "$ROUTING_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find a routing profile ID" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Found routing profile ID: $ROUTING_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 4: Create an admin user
echo "Step 4: Creating admin user..." | tee -a "$LOG_FILE"

# Generate a secure password
ADMIN_PASSWORD="Connect$(openssl rand -base64 12 2>/dev/null || head -c 12 /dev/
urandom | base64)"

USER_RESULT=$(log_cmd "aws connect create-user --instance-id '$INSTANCE_ID'
 --username admin --password '$ADMIN_PASSWORD' --identity-info
 FirstName=Admin,LastName=User,Email=admin@example.com --phone-config
 PhoneType=DESK_PHONE,AutoAccept=true,AfterContactWorkTimeLimit=30,DeskPhoneNumber=
+12065550100 --security-profile-ids '$ADMIN_PROFILE_ID' --routing-profile-id
 '$ROUTING_PROFILE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$USER_RESULT" $? "Failed to create admin user"; then
    exit 1
fi

# Extract user ID using jq
USER_ID=$(echo "$USER_RESULT" | jq -r '.UserId' 2>/dev/null || echo "")

if [[ -z "$USER_ID" ]]; then
    echo "ERROR: Failed to extract user ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created admin user with ID: $USER_ID" | tee -a "$LOG_FILE"
echo "Admin password: $ADMIN_PASSWORD" | tee -a "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Step 5: Configure telephony options
```

```
echo "Step 5: Configuring telephony options..." | tee -a "$LOG_FILE"

# Enable early media
EARLY_MEDIA_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-id
 '$INSTANCE_ID' --attribute-type EARLY_MEDIA --value true --region '$AWS_REGION'"
 2>&1) || true

if ! check_error "$EARLY_MEDIA_RESULT" $? "Failed to enable early media"; then
    exit 1
fi

# Enable multi-party calls and enhanced monitoring for voice
MULTI_PARTY_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_RESULT" $? "Failed to enable multi-party calls";
 then
    exit 1
fi

# Enable multi-party chats and enhanced monitoring for chat
MULTI_PARTY_CHAT_RESULT=$(log_cmd "aws connect update-instance-attribute --
instance-id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CHAT_CONFERENCE --value
 true --region '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_CHAT_RESULT" $? "Failed to enable multi-party
 chats"; then
    exit 1
fi

echo "Successfully configured telephony options" | tee -a "$LOG_FILE"

# Step 6: View storage configurations
echo "Step 6: Viewing storage configurations..." | tee -a "$LOG_FILE"

# List storage configurations for chat transcripts
STORAGE_CONFIGS=$(log_cmd "aws connect list-instance-storage-configs --instance-
id '$INSTANCE_ID' --resource-type CHAT_TRANSCRIPTS --region '$AWS_REGION' --
output json" 2>&1) || true

if ! check_error "$STORAGE_CONFIGS" $? "Failed to list storage configurations";
 then
    exit 1
```

```
fi

echo "Successfully retrieved storage configurations" | tee -a "$LOG_FILE"

# Step 7: Verify instance details
echo "Step 7: Verifying instance details..." | tee -a "$LOG_FILE"
INSTANCE_DETAILS=$(log_cmd "aws connect describe-instance --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_DETAILS" $? "Failed to describe instance"; then
    exit 1
fi

echo "Successfully verified instance details" | tee -a "$LOG_FILE"

# Step 8: Search for available phone numbers (optional)
echo "Step 8: Searching for available phone numbers..." | tee -a "$LOG_FILE"
PHONE_NUMBERS=$(log_cmd "aws connect search-available-phone-numbers --target-arn
 '$INSTANCE_ARN' --phone-number-type TOLL_FREE --phone-number-country-code US --
max-results 5 --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$PHONE_NUMBERS" $? "Failed to search for available phone
 numbers"; then
    exit 1
fi

# Extract the first phone number if available using jq
PHONE_NUMBER=$(echo "$PHONE_NUMBERS" | jq -r
 '.AvailableNumbersList[0].PhoneNumber' 2>/dev/null || echo "")

if [[ -n "$PHONE_NUMBER" ]]; then
    echo "Found available phone number: $PHONE_NUMBER" | tee -a "$LOG_FILE"

    echo ""
    echo "============================================"
    echo "CLAIM PHONE NUMBER"
    echo "============================================"
    echo "Auto-claiming available phone number $PHONE_NUMBER..." | tee -a
 "$LOG_FILE"

    CLAIM_RESULT=$(log_cmd "aws connect claim-phone-number --target-arn
 '$INSTANCE_ARN' --phone-number '$PHONE_NUMBER' --region '$AWS_REGION' --output
 json" 2>&1) || true
```

```
    if ! check_error "$CLAIM_RESULT" $? "Failed to claim phone number"; then
        echo "WARNING: Failed to claim phone number, but continuing with script"
 | tee -a "$LOG_FILE"
    else
        echo "Successfully claimed phone number" | tee -a "$LOG_FILE"
        # Extract the phone number ID from the claim result using jq
        PHONE_NUMBER_ID=$(echo "$CLAIM_RESULT" | jq -r '.PhoneNumberId' 2>/dev/
null || echo "")
    fi
else
    echo "No available phone numbers found" | tee -a "$LOG_FILE"
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCE SUMMARY"
echo "==========================================="
echo "Amazon Connect Instance ID: $INSTANCE_ID"
echo "Amazon Connect Instance ARN: $INSTANCE_ARN"
echo "Admin User ID: $USER_ID"
echo "Admin Username: admin"
echo "Admin Password: $ADMIN_PASSWORD"
if [[ -n "${PHONE_NUMBER:-}" ]]; then
    echo "Claimed Phone Number: $PHONE_NUMBER"
    if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
        echo "Claimed Phone Number ID: $PHONE_NUMBER_ID"
    fi
fi
echo "==========================================="
echo ""

# Auto-confirm cleanup and clean up resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Auto-starting cleanup..." | tee -a "$LOG_FILE"

# Release claimed phone number if applicable
if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
    echo "Releasing phone number: $PHONE_NUMBER_ID" | tee -a "$LOG_FILE"
    RELEASE_RESULT=$(log_cmd "aws connect release-phone-number --phone-number-id
 '$PHONE_NUMBER_ID' --region '$AWS_REGION'" 2>&1) || true
```

```
    if ! check_error "$RELEASE_RESULT" $? "Failed to release phone number"; then
        echo "WARNING: Failed to release phone number" | tee -a "$LOG_FILE"
    else
        echo "Successfully released phone number" | tee -a "$LOG_FILE"
    fi

    echo "Waiting for phone number release to complete..." | tee -a "$LOG_FILE"
    sleep 10
fi

# Delete the Amazon Connect instance (this will also delete all associated
 resources)
echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
DELETE_RESULT=$(log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID'
 --region '$AWS_REGION'" 2>&1) || true

if ! check_error "$DELETE_RESULT" $? "Failed to delete instance"; then
    echo "WARNING: Failed to delete instance" | tee -a "$LOG_FILE"
else
    echo "Successfully deleted instance" | tee -a "$LOG_FILE"
fi

echo "Cleanup completed. All resources have been deleted." | tee -a "$LOG_FILE"

echo "Script completed successfully. See $LOG_FILE for details." | tee -a
 "$LOG_FILE"

trap - ERR EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - ClaimPhoneNumber
  - CreateInstance
  - CreateUser
  - DeleteInstance
  - DescribeInstance
  - ListInstanceStorageConfigs
  - ListInstances
  - ListRoutingProfiles

- ListSecurityProfiles
- ReleasePhoneNumber
- SearchAvailablePhoneNumbers
- UpdateInstanceAttribute

Codebeispiele für AWS Control Tower verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Control Tower mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS Control Tower Benutzerhandbuch — Weitere Informationen zu AWS Control Tower.
- AWS Control Tower API-Referenz — Details zu allen verfügbaren AWS Control Tower Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Control Tower verwenden AWS SDKs
  - Hallo AWS Control Tower
  - Lernen Sie die Grundlagen von AWS Control Tower mit einem AWS SDK
  - Aktionen für AWS Control Tower verwenden AWS SDKs
    - Verwenden Sie DisableBaselinemit einem AWS SDK oder CLI
    - Verwenden Sie DisableControlmit einem AWS SDK oder CLI
    - Verwenden Sie EnableBaselinemit einem AWS SDK oder CLI
    - Verwenden Sie EnableControlmit einem AWS SDK oder CLI

- Verwenden Sie GetBaselineOperationmit einem AWS SDK oder CLI
- Verwenden Sie GetControlOperationmit einem AWS SDK oder CLI
- Verwenden Sie ListBaselinesmit einem AWS SDK oder CLI
- Verwenden Sie ListEnabledBaselinesmit einem AWS SDK oder CLI
- Verwenden Sie ListEnabledControlsmit einem AWS SDK oder CLI
- Verwenden Sie ListLandingZonesmit einem AWS SDK oder CLI
- Verwenden Sie ResetEnabledBaselinemit einem AWS SDK oder CLI

Grundlegende Beispiele für AWS Control Tower verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Control Tower mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS Control Tower
- Lernen Sie die Grundlagen von AWS Control Tower mit einem AWS SDK
- Aktionen für AWS Control Tower verwenden AWS SDKs
  - Verwenden Sie DisableBaselinemit einem AWS SDK oder CLI
  - Verwenden Sie DisableControlmit einem AWS SDK oder CLI
  - Verwenden Sie EnableBaselinemit einem AWS SDK oder CLI
  - Verwenden Sie EnableControlmit einem AWS SDK oder CLI
  - Verwenden Sie GetBaselineOperationmit einem AWS SDK oder CLI
  - Verwenden Sie GetControlOperationmit einem AWS SDK oder CLI
  - Verwenden Sie ListBaselinesmit einem AWS SDK oder CLI
  - Verwenden Sie ListEnabledBaselinesmit einem AWS SDK oder CLI
  - Verwenden Sie ListEnabledControlsmit einem AWS SDK oder CLI
  - Verwenden Sie ListLandingZonesmit einem AWS SDK oder CLI
  - Verwenden Sie ResetEnabledBaselinemit einem AWS SDK oder CLI

Hallo AWS Control Tower

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS Control Tower beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using LogLevel = Microsoft.Extensions.Logging.LogLevel;

namespace ControlTowerActions;

/// <summary>
/// A class that introduces the AWS Control Tower by listing the
/// available baselines for the account.
/// </summary>
public class HelloControlTower
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Control Tower.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
```

```
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloControlTower>();

        var amazonClient =
 host.Services.GetRequiredService<IAmazonControlTower>();

        Console.Clear();
        Console.WriteLine("Hello, AWS Control Tower! Let's list available
 baselines:");
        Console.WriteLine();

        var baselines = new List<BaselineSummary>();

        try
        {
            var baselinesPaginator = amazonClient.Paginators.ListBaselines(new
 ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            Console.WriteLine($"{baselines.Count} baseline(s) retrieved.");
            foreach (var baseline in baselines)
            {
                Console.WriteLine($"\t{baseline.Name}");
            }
        }
        catch (Amazon.ControlTower.Model.AccessDeniedException)
        {
            Console.WriteLine("Access denied. Please ensure you have the
 necessary permissions.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListBaselinesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloControlTower {

    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
 e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
     * and list all available baselines.
     * This example uses the default settings specified in your shared
 credentials
     * and config files.
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
 baselines:\n");
```

```
        ListBaselinesIterable paginator =
 controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));

            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode()))
 {
                System.out.println("Access denied. Please ensure you have the
 necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie ListBaselinesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from typing import Any, List
```

```
def hello_controltower(controltower_client: Any) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Control Tower client
    and list all available baselines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param controltower_client: A Boto3 AWS Control Tower Client object. This
 object wraps
                                the low-level AWS Control Tower service API.
    """
    print("Hello, AWS Control Tower! Let's list available baselines:\n")
    paginator = controltower_client.get_paginator("list_baselines")
    page_iterator = paginator.paginate()

    baseline_names: List[str] = []
    try:
        for page in page_iterator:
            for baseline in page["baselines"]:
                baseline_names.append(baseline["name"])

        print(f"{len(baseline_names)} baseline(s) retrieved.")
        for baseline_name in baseline_names:
            print(f"\t{baseline_name}")

    except controltower_client.exceptions.AccessDeniedException:
        print("Access denied. Please ensure you have the necessary permissions.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


if __name__ == "__main__":
    hello_controltower(boto3.client("controltower"))
```

- Einzelheiten zur API finden Sie ListBaselinesin AWS SDK for Python (Boto3) API Reference.

Lernen Sie die Grundlagen von AWS Control Tower mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Auflisten der Landing Zones
- Listen Sie Baselines auf, aktivieren Sie sie, rufen Sie sie ab, setzen Sie sie zurück und deaktivieren Sie sie.
- Listen Sie Steuerelemente auf, aktivieren Sie sie, rufen Sie sie ab und deaktivieren Sie sie.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Control Tower Funktionen demonstriert werden.

```
using Amazon.ControlCatalog;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Amazon.Organizations;
using Amazon.Organizations.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using ControlTowerActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace ControlTowerBasics;

/// <summary>
/// Scenario class for AWS Control Tower basics.
/// </summary>
public class ControlTowerBasics
{
    public static bool isInteractive = true;
    public static ILogger logger = null!;
    public static IAmazonOrganizations? orgClient = null;
```

```
    public static IAmazonSecurityTokenService? stsClient = null;
    public static ControlTowerWrapper? wrapper = null;
    private static string? ouArn;
    private static bool useLandingZone = false;

    /// <summary>
    /// Main entry point for the AWS Control Tower basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
                .AddAWSService<IAmazonControlCatalog>()
                .AddAWSService<IAmazonOrganizations>()
                .AddAWSService<IAmazonSecurityTokenService>()
                .AddTransient<ControlTowerWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ControlTowerBasics>();

        wrapper = host.Services.GetRequiredService<ControlTowerWrapper>();
        orgClient = host.Services.GetRequiredService<IAmazonOrganizations>();
        stsClient =
host.Services.GetRequiredService<IAmazonSecurityTokenService>();

        await RunScenario();
    }

    /// <summary>
    /// Runs the example scenario.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("\tWelcome to the AWS Control Tower with ControlCatalog
example scenario.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("This demo will walk you through working with AWS
Control Tower for landing zones,");
        Console.WriteLine("managing baselines, and working with controls.");
```

```
        try
        {
            var accountId = (await stsClient!.GetCallerIdentityAsync(new
 GetCallerIdentityRequest())).Account;
            Console.WriteLine($"\nAccount ID: {accountId}");

            Console.WriteLine("\nSome demo operations require the use of a
 landing zone.");
            Console.WriteLine("You can use an existing landing zone or opt out of
 these operations in the demo.");
            Console.WriteLine("For instructions on how to set up a landing
 zone,");
            Console.WriteLine("see https://docs.aws.amazon.com/controltower/
latest/userguide/getting-started-from-console.html");

            // List available landing zones
            var landingZones = await wrapper!.ListLandingZonesAsync();
            if (landingZones.Count > 0)
            {
                Console.WriteLine("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.Count; i++)
                {
                    Console.WriteLine($"{i + 1}. {landingZones[i].Arn}");
                }

                Console.Write($"\nDo you want to use the first landing zone in
 the list ({landingZones[0].Arn})? (y/n): ");
                if (GetUserConfirmation())
                {
                    useLandingZone = true;
                    Console.WriteLine($"Using landing zone:
 {landingZones[0].Arn}");
                    ouArn = await SetupOrganizationAsync();
                }
            }

            // Managing Baselines
            Console.WriteLine("\nManaging Baselines:");
            var baselines = await wrapper.ListBaselinesAsync();
            Console.WriteLine("\nListing available Baselines:");
            BaselineSummary? controlTowerBaseline = null;
            foreach (var baseline in baselines)
            {
```

```
                if (baseline.Name == "AWSControlTowerBaseline")
                    controlTowerBaseline = baseline;
                Console.WriteLine($"  - {baseline.Name}");
            }

            EnabledBaselineSummary? identityCenterBaseline = null;
            string? baselineArn = null;

            if (useLandingZone && ouArn != null)
            {
                Console.WriteLine("\nListing enabled baselines:");
                var enabledBaselines = await wrapper.ListEnabledBaselinesAsync();
                foreach (var baseline in enabledBaselines)
                {
                    if (baseline.BaselineIdentifier.Contains("baseline/
LN25R72TTG6IGPTQ"))
                        identityCenterBaseline = baseline;
                    Console.WriteLine($"  - {baseline.BaselineIdentifier}");
                }

                if (controlTowerBaseline != null)
                {
                    Console.Write("\nDo you want to enable the Control Tower
 Baseline? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine("\nEnabling Control Tower Baseline.");
                        var icBaselineArn = identityCenterBaseline?.Arn;
                        baselineArn = await wrapper.EnableBaselineAsync(ouArn,
                            controlTowerBaseline.Arn, "5.0", icBaselineArn ??
 "");
                        var alreadyEnabled = false;
                        if (baselineArn != null)
                        {
                            Console.WriteLine($"Enabled baseline ARN:
 {baselineArn}");
                        }
                        else
                        {
                            // Find the enabled baseline
                            foreach (var enabled in enabledBaselines)
                            {
                                if (enabled.BaselineIdentifier ==
 controlTowerBaseline.Arn)
```

```
                                {
                                    baselineArn = enabled.Arn;
                                    alreadyEnabled = true;
                                    Console.WriteLine("No change, the selected
baseline was already enabled.");
                                    break;
                                }
                            }
                        }

                        if (baselineArn != null)
                        {
                            Console.Write("\nDo you want to reset the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"\nResetting Control Tower
Baseline: {baselineArn}");
                                var operationId = await
wrapper.ResetEnabledBaselineAsync(baselineArn);
                                Console.WriteLine($"Reset baseline operation id:
{operationId}");
                            }

                            Console.Write("\nDo you want to disable the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"Disabling baseline ARN:
{baselineArn}");
                                var operationId = await
wrapper.DisableBaselineAsync(baselineArn);
                                Console.WriteLine($"Disabled baseline operation
id: {operationId}");
                                if (alreadyEnabled)
                                {
                                    Console.WriteLine($"\nRe-enabling Control
Tower Baseline: {baselineArn}");
                                    // Re-enable the Control Tower baseline if it
was originally enabled.
                                    await wrapper.EnableBaselineAsync(ouArn,
                                        controlTowerBaseline.Arn, "5.0",
icBaselineArn ?? "");
                                }
```

```
                            }
                        }
                    }
                }
            }

            // Managing Controls
            Console.WriteLine("\nManaging Controls:");
            var controls = await wrapper.ListControlsAsync();
            Console.WriteLine("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.Min(5, controls.Count); i++)
            {
                Console.WriteLine($"{i + 1}. {controls[i].Name} -
 {controls[i].Arn}");
            }

            if (useLandingZone && ouArn != null)
            {
                var enabledControls = await
 wrapper.ListEnabledControlsAsync(ouArn);
                Console.WriteLine("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.Count; i++)
                {
                    Console.WriteLine($"{i + 1}.
 {enabledControls[i].ControlIdentifier}");
                }

                // Find first non-enabled control
                var enabledControlArns = enabledControls.Select(c =>
 c.Arn).ToHashSet();
                var controlArn = controls.FirstOrDefault(c => !
enabledControlArns.Contains(c.Arn))?.Arn;

                if (controlArn != null)
                {
                    Console.Write($"\nDo you want to enable the control
 {controlArn}? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine($"\nEnabling control: {controlArn}");
                        var operationId = await
 wrapper.EnableControlAsync(controlArn, ouArn);
                        if (operationId != null)
                        {
```

```
                            Console.WriteLine($"Enabled control with operation
id: {operationId}");

                            Console.Write("\nDo you want to disable the control?
(y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine("\nDisabling the control...");
                                var disableOpId = await
wrapper.DisableControlAsync(controlArn, ouArn);
                                Console.WriteLine($"Disable operation ID:
{disableOpId}");
                            }
                        }
                    }
                }
            }

            Console.WriteLine("\nThis concludes the example scenario.");
            Console.WriteLine("Thanks for watching!");
            Console.WriteLine(new string('-', 88));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "An error occurred during the Control Tower
scenario.");
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Sets up AWS Organizations and creates or finds a Sandbox OU.
    /// </summary>
    /// <returns>The ARN of the Sandbox organizational unit.</returns>
    private static async Task<string> SetupOrganizationAsync()
    {
        Console.WriteLine("\nChecking organization status...");

        try
        {
            var orgResponse = await orgClient!.DescribeOrganizationAsync(new
DescribeOrganizationRequest());
            var orgId = orgResponse.Organization.Id;
            Console.WriteLine($"Account is part of organization: {orgId}");
```

```
        }
        catch (AWSOrganizationsNotInUseException)
        {
            Console.WriteLine("No organization found. Creating a new
organization...");
            var createResponse = await orgClient!.CreateOrganizationAsync(new
CreateOrganizationRequest { FeatureSet = OrganizationFeatureSet.ALL });
            var orgId = createResponse.Organization.Id;
            Console.WriteLine($"Created new organization: {orgId}");
        }

        // Look for Sandbox OU
        var roots = await orgClient.ListRootsAsync(new ListRootsRequest());
        var rootId = roots.Roots[0].Id;

        Console.WriteLine("Checking for Sandbox OU...");
        var ous = await orgClient.ListOrganizationalUnitsForParentAsync(new
ListOrganizationalUnitsForParentRequest { ParentId = rootId });
        var sandboxOu = ous.OrganizationalUnits.FirstOrDefault(ou => ou.Name ==
"Sandbox");

        if (sandboxOu == null)
        {
            Console.WriteLine("Creating Sandbox OU...");
            var createOuResponse = await
orgClient.CreateOrganizationalUnitAsync(new CreateOrganizationalUnitRequest
{ ParentId = rootId, Name = "Sandbox" });
            sandboxOu = createOuResponse.OrganizationalUnit;
            Console.WriteLine($"Created new Sandbox OU: {sandboxOu.Id}");
        }
        else
        {
            Console.WriteLine($"Found existing Sandbox OU: {sandboxOu.Id}");
        }

        return sandboxOu.Arn;
    }

    /// <summary>
    /// Gets user confirmation by waiting for input or returning true if not
interactive.
    /// </summary>
    /// <returns>True if user enters 'y' or if isInteractive is false, otherwise
false.</returns>
```

```
    private static bool GetUserConfirmation()
    {
        return Console.ReadLine()?.ToLower() == "y" || !isInteractive;
    }
}
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
using Amazon.ControlCatalog;
using Amazon.ControlCatalog.Model;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using ValidationException = Amazon.ControlTower.Model.ValidationException;

namespace ControlTowerActions;

/// <summary>
/// Methods to perform AWS Control Tower actions.
/// </summary>
public class ControlTowerWrapper
{
    private readonly IAmazonControlTower _controlTowerService;
    private readonly IAmazonControlCatalog _controlCatalogService;

    /// <summary>
    /// Constructor for the wrapper class containing AWS Control Tower actions.
    /// </summary>
    /// <param name="controlTowerService">The AWS Control Tower client object.</
param>
    /// <param name="controlCatalogService">The AWS Control Catalog client
 object.</param>
    public ControlTowerWrapper(IAmazonControlTower controlTowerService,
 IAmazonControlCatalog controlCatalogService)
    {
        _controlTowerService = controlTowerService;
        _controlCatalogService = controlCatalogService;
    }

    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
```

```
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }
```

```
            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a baseline for the specified target.
```

```
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier,
 string baselineIdentifier, string baselineVersion, string
 identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
            };

            var response = await
 _controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
 BaselineOperationStatus.FAILED)
                {
```

```
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;
```

```
            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}.
Skipping disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
```

```
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
```

```
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's
why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit
identifier.</param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>>
ListEnabledControlsAsync(string targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();
```

```
            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to enable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID or null if already enabled.</returns>
    public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            Console.WriteLine(controlArn);
            Console.WriteLine(targetIdentifier);

            var request = new EnableControlRequest
            {
                ControlIdentifier = controlArn,
```

```
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.EnableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Disable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to disable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            var request = new DisableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await
_controlTowerService.DisableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
```

```
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's
why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all controls in the Control Tower control catalog.
    /// </summary>
    /// <returns>A list of control summaries.</returns>
```

```
    public async Task<List<ControlSummary>> ListControlsAsync()
    {
        try
        {
            var controls = new List<ControlSummary>();

            var controlsPaginator =
 _controlCatalogService.Paginators.ListControls(new
 Amazon.ControlCatalog.Model.ListControlsRequest());

            await foreach (var response in controlsPaginator.Responses)
            {
                controls.AddRange(response.Controls);
            }

            return controls;
        }
        catch (AmazonControlCatalogException ex)
        {
            Console.WriteLine($"Couldn't list controls. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation

- ListBaselines
- ListEnabledBaselines
- ListEnabledControls
- ListLandingZones
- ResetEnabledBaseline

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Control Tower Funktionen demonstriert werden.

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;

    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);
```

```
        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ----------------------------
    // Utilities
    // ----------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ----------------------------
            // Step 1: Landing Zones
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in
 the demo.
        For instructions on how to set up a landing zone,
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);

            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-
dependent steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
```

```
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1,
landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the
list (" +
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing
Landing Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }

            // ----------------------------
            // Setup Organization + Sandbox OU
            // ----------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);

                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();

                System.out.println("Organization ID: " + result.orgId());
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // ----------------------------
            // Step 2: Baselines
            // ----------------------------
            System.out.println(DASHES);
```

```
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-
baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");



            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();

            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
            }

            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
```

```
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if
 (eb.baselineIdentifier().equals(controlTowerBaseline.arn())) {
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
                if (askYesNo("Do you want to enable the Control Tower Baseline?
 (y/n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");

                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
 the OU or account ARN
                                    baselineId,             // baselineIdentifier
 # the Control Tower baseline ARN
                                    "5.0"                   // baselineVersion #
 version string
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
 enabledBaselineId);
                    if (enabledBaselineId == null) {
```

```
                        enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower
Baseline? (y/n): ")) {
                        String operationId =

actions.resetEnabledBaselineAsync(enabledBaselineId).join();
                        System.out.println("Reset baseline operation ID: " +
operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower
Baseline? (y/n): ")) {
                        String operationId =

actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
operationId);

                        System.out.println("Now we will re-enable the baseline
and wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();

                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
```

```
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }

                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
                }

                waitForInputToContinue(scanner);
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
```

```
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
                        askYesNo("Do you want to disable the control? (y/n): "))
{

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
            System.out.println("Unexpected error running scenario: " +
e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
                System.out.println("Continuing...");
                break;
            }
        }
```

```
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return orgAsyncClient;
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }

    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();

            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }

    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }

    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();
```

```
        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&

"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode()))
{
                        System.out.println("No organization found. Creating
one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}"
+ createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
                                    new RuntimeException("No root found in
organization")
                            );
```

```
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);

                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU:
" + ou.id());
                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
                                        System.out.println("Sandbox OU not
found.");
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
       }).exceptionally(ex -> {
           Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
           System.out.println("Failed to setup organization: {}" +
cause.getMessage());
           throw new CompletionException(cause);
       });
   }

   /**
    * Lists all landing zones using pagination to retrieve complete results.
```

```
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request =
 ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing
 zones: " + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }


    /**
     * Lists all available baselines using pagination to retrieve complete
 results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total:
 " + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }

    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");
```

```
        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled
baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: "
+
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled
baselines: %s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
```

```
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }


    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null
if already enabled.
     */
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled
for this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN
synchronously
                            }

                            throw new RuntimeException("Error enabling baseline:
" + code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
```

```
                                BaselineOperation op =
opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED)
{
                                    return enabledBaselineArn;
                                } else if (status ==
BaselineOperationStatus.FAILED) {
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline
operation failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {

Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target
asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
```

```
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines())
{
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                &&
targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }


    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
```

```
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
```

```
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }



    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request =
GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
```

```
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target "
 + targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled
 controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +"
 enabled controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
```

```
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to
work with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled
controls", cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
```

```
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };
```

```
                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ?
e.getMessage() : "";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for
this target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default
and cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS
Control Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
```

```
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
```

```
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling:
" + e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
```

```
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }



    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
```

```
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
 %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
 operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }


    /**
     * Lists all controls in the Control Tower control catalog.
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
 getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.controls() != null && !
response.controls().isEmpty()) {
```

```
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
sdkEx.getMessage().contains("AccessDeniedException")) {
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
%s".formatted(sdkEx.getMessage()),
                                    sdkEx
                            );
                        }
                    }

                    throw new CompletionException("Failed to list controls",
cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
```

```
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request =
ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   Runnable poller = new Runnable() {
                       @Override
                       public void run() {
                           getBaselineOperationAsync(operationId)
                                   .thenAccept(statusObj -> {
                                       String status = statusObj.toString(); //
Convert enum/status to string for printing
                                       System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                       if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                           System.out.println("Baseline
operation finished with status: " + status);
                                           resultFuture.complete(operationId);
                                       } else {
                                           // Schedule next poll in 15 seconds
                                           CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                   .execute(this);
                                       }
                                   })
```

```
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
 baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code="
 + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline:
 " + cause.getMessage());
                    return null;
                });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Control Tower Funktionen demonstriert werden.

```
class ControlTowerScenario:
    IDENTITY_CENTER_BASELINE = "baseline/LN25R72TTG6IGPTQ"
    stack_name = ""

    def __init__(
        self, controltower_wrapper: ControlTowerWrapper, org_client: boto3.client
    ):
```

```
        """
        :param controltower_wrapper: An instance of the ControlTowerWrapper
 class.
        :param org_client: A Boto3 Organization client.
        """
        self.controltower_wrapper = controltower_wrapper
        self.org_client = org_client
        self.stack = None
        self.ou_id = None
        self.ou_arn = None
        self.account_id = None
        self.landing_zone_id = None
        self.use_landing_zone = False

    def run_scenario(self) -> None:
        print("-" * 88)
        print(
            "\tWelcome to the AWS Control Tower with ControlCatalog example
 scenario."
        )
        print("-" * 88)

        print(
            "This demo will walk you through working with AWS Control Tower for
 landing zones,\n"
            "managing baselines, and working with controls."
        )

        self.account_id = boto3.client("sts").get_caller_identity()["Account"]

        print(
            "Some demo operations require the use of a landing zone. "
            "\nYou can use an existing landing zone or opt out of these
 operations in the demo."
            "\nFor instructions on how to set up a landing zone, "
            "\nsee https://docs.aws.amazon.com/controltower/latest/userguide/
getting-started-from-console.html"
        )
        # List available landing zones
        landing_zones = self.controltower_wrapper.list_landing_zones()
        if landing_zones:
            print("\nAvailable Landing Zones:")
            for i, lz in enumerate(landing_zones, 1):
                print(f"{i} {lz['arn']}")
```

```
            # Ask if user wants to use the first landing zone in the list
            if q.ask(
                f"Do you want to use the first landing zone in the list
({landing_zones[0]['arn']})? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = landing_zones[0]["arn"]
                print(f"Using landing zone ID: {self.landing_zone_id})")
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id
            elif q.ask(
                f"Do you want to use a different existing Landing Zone for this
demo? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = q.ask("Enter landing zone id: ",
q.non_empty)
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id

        # List and Enable Baseline.
        print("\nManaging Baselines:")
        control_tower_baseline = None
        identity_center_baseline = None
        baselines = self.controltower_wrapper.list_baselines()
        print("\nListing available Baselines:")
        for baseline in baselines:
            if baseline["name"] == "AWSControlTowerBaseline":
                control_tower_baseline = baseline
            print(f"{baseline['name']}")

        if self.use_landing_zone:
            print("\nListing enabled baselines:")
            enabled_baselines =
self.controltower_wrapper.list_enabled_baselines()
            for baseline in enabled_baselines:
```

```
                # If the Identity Center baseline is enabled, the identifier must
be used for other baselines.
                if self.IDENTITY_CENTER_BASELINE in
baseline["baselineIdentifier"]:
                    identity_center_baseline = baseline
                print(f"{baseline['baselineIdentifier']}")

            if q.ask(
                f"Do you want to enable the Control Tower Baseline? (y/n) ",
                q.is_yesno,
            ):
                print("\nEnabling Control Tower Baseline.")
                ic_baseline_arn = (
                    identity_center_baseline["arn"]
                    if identity_center_baseline
                    else None
                )
                baseline_arn = self.controltower_wrapper.enable_baseline(
                    self.ou_arn, ic_baseline_arn, control_tower_baseline["arn"],
"5.0"
                )
                if baseline_arn:
                    print(f"Enabled baseline ARN: {baseline_arn}")
                else:
                    # Find the enabled baseline so we can reset it.
                    for enabled_baseline in enabled_baselines:
                        if (
                            enabled_baseline["baselineIdentifier"]
                            == control_tower_baseline["arn"]
                        ):
                            baseline_arn = enabled_baseline["arn"]
                    if baseline_arn:
                        print("No change, the selected baseline was already
enabled.")

                if baseline_arn and q.ask(
                    f"Do you want to reset the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"\nResetting Control Tower Baseline. {baseline_arn}")
                    operation_id =
self.controltower_wrapper.reset_enabled_baseline(
                        baseline_arn
                    )
```

```
                    print(f"\nReset baseline operation id {operation_id}.")

                if baseline_arn and q.ask(
                    f"Do you want to disable the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"Disabling baseline ARN: {baseline_arn}")
                    operation_id = self.controltower_wrapper.disable_baseline(
                        baseline_arn
                    )
                    print(f"\nDisabled baseline operation id {operation_id}.")

                    # Re-enable the baseline for the next step.
                    print("\nEnabling Control Tower Baseline.")
                    self.controltower_wrapper.enable_baseline(
                        self.ou_arn,
                        ic_baseline_arn,
                        control_tower_baseline["arn"],
                        "5.0",
                    )

        # List and Enable Controls.
        print("\nManaging Controls:")
        controls = self.controltower_wrapper.list_controls()
        print("\nListing first 5 available Controls:")
        for i, control in enumerate(controls[:5], 1):
            print(f"{i}. {control['Name']} - {control['Arn']}")

        if self.use_landing_zone:
            target_ou = self.ou_arn
            enabled_controls = self.controltower_wrapper.list_enabled_controls(
                target_ou
            )
            print("\nListing enabled controls:")
            for i, control in enumerate(enabled_controls, 1):
                print(f"{i}. {control['controlIdentifier']}")

            # Enable first non-enabled control as an example.
            enabled_control_arns = [control["arn"] for control in
enabled_controls]
            control_arn = next(
                control["Arn"]
                for control in controls
                if control["Arn"] not in enabled_control_arns
```

```
            )

            if control_arn and q.ask(
                f"Do you want to enable the control {control_arn}? (y/n) ",
                q.is_yesno,
            ):
                print(f"\nEnabling control: {control_arn}")
                operation_id = self.controltower_wrapper.enable_control(
                    control_arn, target_ou
                )

                if operation_id:
                    print(f"Enabled control with operation id {operation_id}")

            if control_arn and q.ask(
                f"Do you want to disable the control? (y/n) ",
                q.is_yesno,
            ):
                print("\nDisabling the control...")
                operation_id = self.controltower_wrapper.disable_control(
                    control_arn, target_ou
                )
                print(f"Disable operation ID: {operation_id}")

        print("\nThis concludes the example scenario.")

        print("Thanks for watching!")
        print("-" * 88)

    def setup_organization(self):
        """
        Checks if the current account is part of an organization and creates one
if needed.
        Also ensures a Sandbox OU exists and returns its ID.

        :return: The ID of the Sandbox OU
        """
        print("\nChecking organization status...")

        try:
            # Check if account is part of an organization
            org_response = self.org_client.describe_organization()
            org_id = org_response["Organization"]["Id"]
            print(f"Account is part of organization: {org_id}")
```

```
        except ClientError as error:
            if error.response["Error"]["Code"] ==
"AWSOrganizationsNotInUseException":
                print("No organization found. Creating a new organization...")
                try:
                    create_response = self.org_client.create_organization(
                        FeatureSet="ALL"
                    )
                    org_id = create_response["Organization"]["Id"]
                    print(f"Created new organization: {org_id}")

                    # Wait for organization to be available.
                    waiter = self.org_client.get_waiter("organization_active")
                    waiter.wait(
                        Organization=org_id,
                        WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                    )

                except ClientError as create_error:
                    logger.error(
                        "Couldn't create organization. Here's why: %s: %s",
                        create_error.response["Error"]["Code"],
                        create_error.response["Error"]["Message"],
                    )
                    raise
            else:
                logger.error(
                    "Couldn't describe organization. Here's why: %s: %s",
                    error.response["Error"]["Code"],
                    error.response["Error"]["Message"],
                )
                raise

        # Look for Sandbox OU.
        sandbox_ou_id = None
        paginator = self.org_client.get_paginator(
            "list_organizational_units_for_parent"
        )

        try:
            # Get root ID first.
            roots = self.org_client.list_roots()["Roots"]
            if not roots:
```

```
                raise ValueError("No root found in organization")
            root_id = roots[0]["Id"]

            # Search for existing Sandbox OU.
            print("Checking for Sandbox OU...")
            for page in paginator.paginate(ParentId=root_id):
                for ou in page["OrganizationalUnits"]:
                    if ou["Name"] == "Sandbox":
                        sandbox_ou_id = ou["Id"]
                        self.ou_arn = ou["Arn"]
                        print(f"Found existing Sandbox OU: {sandbox_ou_id}")
                        break
                if sandbox_ou_id:
                    break

            # Create Sandbox OU if it doesn't exist.
            if not sandbox_ou_id:
                print("Creating Sandbox OU...")
                create_ou_response = self.org_client.create_organizational_unit(
                    ParentId=root_id, Name="Sandbox"
                )
                sandbox_ou_id = create_ou_response["OrganizationalUnit"]["Id"]
                print(f"Created new Sandbox OU: {sandbox_ou_id}")

                # Wait for OU to be available.
                waiter = self.org_client.get_waiter("organizational_unit_active")
                waiter.wait(
                    OrganizationalUnitId=sandbox_ou_id,
                    WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                )

        except ClientError as error:
            logger.error(
                "Couldn't set up Sandbox OU. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

        return sandbox_ou_id


if __name__ == "__main__":
    try:
```

```
        org = boto3.client("organizations")
        control_tower_wrapper = ControlTowerWrapper.from_client()

        scenario = ControlTowerScenario(control_tower_wrapper, org)
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the scenario.")



class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center
baseline if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other
than it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
```

```
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception:
%s: %s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None


    def list_controls(self):
        """
        Lists all controls in the Control Tower control catalog.

        :return: List of controls.
        :raises ClientError: If the listing operation fails.
        """
```

```
        try:
            paginator = self.controlcatalog_client.get_paginator("list_controls")
            controls = []
            for page in paginator.paginate():
                controls.extend(page["Controls"])
            return controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
```

```
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
```

```
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
```

```
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
```

```
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled
baseline to reset.
        :return: The operation ID.
```

```
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
 complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
 disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
```

```
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
 self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
```

```
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return enabled_controls
            else:
                logger.error(
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Aktionen für AWS Control Tower verwenden AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne AWS Control Tower Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Control Tower -API-Referenz.

Beispiele

- Verwenden Sie DisableBaselinemit einem AWS SDK oder CLI
- Verwenden Sie DisableControlmit einem AWS SDK oder CLI
- Verwenden Sie EnableBaselinemit einem AWS SDK oder CLI
- Verwenden Sie EnableControlmit einem AWS SDK oder CLI
- Verwenden Sie GetBaselineOperationmit einem AWS SDK oder CLI
- Verwenden Sie GetControlOperationmit einem AWS SDK oder CLI
- Verwenden Sie ListBaselinesmit einem AWS SDK oder CLI
- Verwenden Sie ListEnabledBaselinesmit einem AWS SDK oder CLI
- Verwenden Sie ListEnabledControlsmit einem AWS SDK oder CLI
- Verwenden Sie ListLandingZonesmit einem AWS SDK oder CLI
- Verwenden Sie ResetEnabledBaselinemit einem AWS SDK oder CLI

Verwenden Sie **`DisableBaseline`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableBaseline` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
```

```
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}.
 Skipping disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DisableBaselinein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Control Tower Tower-Baseline zu deaktivieren

Das folgende `disable-baseline` Beispiel deaktiviert eine AWS Control Tower Tower-Baseline.

```
aws controltower disable-baseline \
    --enabled-baseline-identifier arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XOM12BEL4YD578CQ2
```

Ausgabe:

```
{
    "operationIdentifier": "b33486d7-5396-4ad0-9eae-3a57969fe8cd"
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DisableBaselinein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Disables a baseline for a specified target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting disable of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       DisableBaselineRequest request = DisableBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().disableBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Disable baseline operation ID: " +
operationId);
```

```
                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
 disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
 baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
 baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- Einzelheiten zur API finden Sie DisableBaselinein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""
```

```
    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie DisableBaselinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Disable the baseline
        DATA(lo_output) = io_ctt->disablebaseline(
          iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
        ).

        DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

        " Wait for operation to complete
        DATA lv_status TYPE /aws1/cttbaselineopstatus.
        DO 100 TIMES.
          lv_status = get_baseline_operation(
```

```
            io_ctt = io_ctt
            iv_operation_id = lv_operation_id
          ).

          DATA(lv_msg) = |Baseline operation status: { lv_status }|.
          MESSAGE lv_msg TYPE 'I'.

          IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
            EXIT.
          ENDIF.

          " Wait 30 seconds
          WAIT UP TO 30 SECONDS.
        ENDDO.

        ov_operation_id = lv_operation_id.
        MESSAGE 'Baseline disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_cttconflictexception INTO DATA(lo_conflict).
        " Log conflict but don't fail - return empty operation ID
        DATA(lv_msg2) = |Conflict disabling baseline: { lo_conflict-
>get_text( ) }. Skipping disable step.|.
        MESSAGE lv_msg2 TYPE 'I'.
        CLEAR ov_operation_id.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DisableBaselinein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DisableControl`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableControl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Disable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to disable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID.</returns>
   public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           var request = new DisableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await
_controlTowerService.DisableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetControlOperationAsync(operationId);
               Console.WriteLine($"Control operation status: {status}");
               if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
               {
                   break;
               }
```

```
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DisableControlin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Control Tower zu deaktivieren

Das folgende `disable-control` Beispiel deaktiviert eine AWS Control Tower Tower-fähige Steuerung.

```
aws controltower disable-control \
    --control-
identifier arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrxxxxxx \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-
oqxx-i5wnxxxx
```

Ausgabe:

```
{
    "operationIdentifier": "b8f0dxxx-08xx-43xx-a2xx-568e9922xxxx"
}
```

Weitere Informationen finden Sie unter Über Steuerungen im AWS Control Tower im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DisableControlin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();
```

```
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling:
" + e.getMessage());
                            return null;
```

```
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }
```

- Einzelheiten zur API finden Sie DisableControlin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
```

```
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
```

- Einzelheiten zur API finden Sie DisableControlin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Disable the control
DATA(lo_output) = io_ctt->disablecontrol(
  iv_controlidentifier = iv_control_arn
  iv_targetidentifier  = iv_target_identifier
).

DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

" Wait for operation to complete
DATA lv_status TYPE /aws1/cttcontrolopstatus.
DO 100 TIMES.
  lv_status = get_control_operation(
    io_ctt = io_ctt
    iv_operation_id = lv_operation_id
  ).

  DATA(lv_msg) = |Control operation status: { lv_status }|.
  MESSAGE lv_msg TYPE 'I'.

  IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
    EXIT.
  ENDIF.

  " Wait 30 seconds
```

```
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control disabled successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie DisableControlin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableBaseline`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableBaseline` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
```

```
   public async Task<string?> EnableBaselineAsync(string targetIdentifier,
string baselineIdentifier, string baselineVersion, string
identityCenterBaseline)
   {
       try
       {
           var parameters = new List<EnabledBaselineParameter>();
           if (!string.IsNullOrEmpty(identityCenterBaseline))
           {
               parameters.Add(
                   new EnabledBaselineParameter
                   {
                       Key = "IdentityCenterEnabledBaselineArn",
                       Value = identityCenterBaseline
                   });
           }
           var request = new EnableBaselineRequest
           {
               BaselineIdentifier = baselineIdentifier,
               BaselineVersion = baselineVersion,
               TargetIdentifier = targetIdentifier,
               Parameters = parameters
           };

           var response = await
_controlTowerService.EnableBaselineAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetBaselineOperationAsync(operationId);
               Console.WriteLine($"Baseline operation status: {status}");
               if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
               {
                   break;
               }
               await Task.Delay(30000); // Wait 30 seconds
           }

           return response.Arn;
       }
       catch (ValidationException ex)
```

```
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie EnableBaselinein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Um eine Control Tower Tower-Baseline zu aktivieren, die deaktiviert ist

Das folgende `enable-baseline` Beispiel aktiviert eine AWS Control Tower Tower-Baseline, wenn die Baseline deaktiviert `IdentityCenterBaseline` ist.

```
aws controltower enable-baseline \
    --baseline-identifier arn:aws:controltower:us-
east-1::baseline/17BSJV3IGJ2QSGA2 \
    --baseline-version 4.0 \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryixxxx/ou-
oq9f-i5wnxxxx
```

Ausgabe:

```
{
    "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XOM12BEL4YD578CQ2",
    "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

Beispiel 2: Um eine aktivierte Control Tower Tower-Baseline zu aktivieren

Das folgende `enable-baseline` Beispiel aktiviert eine AWS Control Tower Tower-Baseline, wenn die Baseline aktiviert `IdentityCenterBaseline` ist.

```
aws controltower enable-baseline \
    --baseline-identifier arn:aws:controltower:us-
east-1::baseline/17BSJV3IGJ2QSGA2 \
    --baseline-version 4.0 \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryixxxx/ou-
oqxx-i5wnxxxx \
    --parameters
 '[{"key":"IdentityCenterEnabledBaselineArn","value":"arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XAJNZNCBC1I386C7B"}]'
```

Ausgabe:

```
{
    "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XOM12BEL4YD578CQ2",
    "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie EnableBaselinein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null
if already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
               .baselineIdentifier(baselineIdentifier)
               .baselineVersion(baselineVersion)
               .targetIdentifier(targetIdentifier)
               .build();

       return getAsyncClient().enableBaseline(request)
               .handle((resp, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ControlTowerException e) {
                           String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                           String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                           if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                               System.out.println("Baseline is already enabled
for this target # fetching ARN...");
                               return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                       .join(); // fetch existing ARN
synchronously
                           }
```

```
                            throw new RuntimeException("Error enabling baseline:
" + code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op =
opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED)
{
                                    return enabledBaselineArn;
                                } else if (status ==
BaselineOperationStatus.FAILED) {
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline
operation failed (ID: " + opId + "), status: "
```

```
                                                + status + ", reason: " + reason);
                                }

                                try {

Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target
asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines())
{
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                &&
targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }
```

- Einzelheiten zur API finden Sie EnableBaselinein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
```

```
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center
baseline if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other
than it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
```

```
                    print(
                        "Unable to enable baseline due to validation exception:
 %s: %s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None
```

- Einzelheiten zur API finden Sie EnableBaselinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Prepare parameters for enabling baseline
    DATA lt_parameters TYPE /aws1/
cl_cttenbdbaselineparam=>tt_enabledbaselineparameters.

    " Add Identity Center baseline parameter if provided
    IF iv_identity_center_baseline IS NOT INITIAL.
      " Create a JSON document with the baseline ARN value
      DATA(lv_json) = |\{ "IdentityCenterEnabledBaselineArn":
 "{ iv_identity_center_baseline }" \}|.
      DATA(lo_param) = NEW /aws1/cl_cttenbdbaselineparam(
        iv_key = 'IdentityCenterEnabledBaselineArn'
        io_value = /aws1/cl_rt_document=>from_json_str( lv_json )
      ).
```

```
      APPEND lo_param TO lt_parameters.
    ENDIF.

    " Enable the baseline
    DATA(lo_output) = io_ctt->enablebaseline(
      iv_baselineidentifier = iv_baseline_identifier
      iv_baselineversion    = iv_baseline_version
      iv_targetidentifier   = iv_target_identifier
      it_parameters         = lt_parameters
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttbaselineopstatus.
    DO 100 TIMES.
      lv_status = get_baseline_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Baseline operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_enabled_baseline_arn = lo_output->get_arn( ).
    MESSAGE 'Baseline enabled successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie EnableBaselinein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableControl`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableControl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Enable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to enable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID or null if already enabled.</returns>
   public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           Console.WriteLine(controlArn);
           Console.WriteLine(targetIdentifier);

           var request = new EnableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await
_controlTowerService.EnableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
```

```
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie EnableControlin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Control Tower zu aktivieren

Das folgende `enable-control` Beispiel aktiviert eine AWS Control Tower Tower-Steuerung.

```
aws controltower enable-control \
    --control-
identifier arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrxxxxxx \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-
oqxx-i5wnxxxx
```

Ausgabe:

```
{
    "arn": "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/18J5KBJ3W3VTIRLV",
    "operationIdentifier": "7691fc5a-de87-4540-8c95-b0aabd56382c"
}
```

Weitere Informationen finden Sie unter Über Steuerungen im AWS Control Tower im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie EnableControlin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Enables a control for a specified target.
     *
```

```
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started.
Operation ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
```

```
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ?
e.getMessage() : "";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for
this target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default
and cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS
Control Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
```

```
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- Einzelheiten zur API finden Sie EnableControlin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
```

```
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
```

```
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie EnableControlin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Enable the control
DATA(lo_output) = io_ctt->enablecontrol(
  iv_controlidentifier = iv_control_arn
  iv_targetidentifier  = iv_target_identifier
).

DATA(lv_operation_id) = lo_output->get_operationidentifier( ).
```

```
    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttcontrolopstatus.
    DO 100 TIMES.
      lv_status = get_control_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Control operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control enabled successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie EnableControlin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetBaselineOperation`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBaselineOperation` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's
why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie GetBaselineOperationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen Control Tower Tower-Basisbetrieb zu starten

Im folgenden `get-baseline-operation` Beispiel werden Details zu einem AWS Control Tower Tower-Basisvorgang abgerufen.

```
aws controltower get-baseline-operation \
    --operation-identifier "51e190ac-8a37-4f6d-b63c-fb5104b5db38"
```

Ausgabe:

```
{
    "baselineOperation": {
        "endTime": "2025-04-17T23:48:46+00:00",
        "operationIdentifier": "51e190ac-8a37-4f6d-b63c-fb5104b5db38",
        "operationType": "ENABLE_BASELINE",
        "startTime": "2025-04-17T23:46:37+00:00",
        "status": "SUCCEEDED",
        "statusMessage": "AWS Control Tower completed the baseline operation
 successfully."
    }
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetBaselineOperationin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
```

```
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request =
GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }
```

```
                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- Einzelheiten zur API finden Sie GetBaselineOperationin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
```

```
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie GetBaselineOperationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_output) = io_ctt->getbaselineoperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_baselineoperation( )->get_status( ).
```

- Einzelheiten zur API finden Sie GetBaselineOperationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetControlOperation`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetControlOperation` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's
why: {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie GetControlOperationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um den Control Tower in Betrieb zu nehmen

Im folgenden `get-control-operation` Beispiel werden Details zu einem AWS Kontrollvorgang im Control Tower abgerufen.

```
aws controltower get-control-operation \
    --operation-identifier "7691fc5a-de87-4540-8c95-b0aabd56382c"
```

Ausgabe:

```
{
    "controlOperation": {
        "controlIdentifier":
 "arn:aws:controlcatalog:::control/497wrm2xnk1wxlf4obrdo7mej",
        "enabledControlIdentifier": "arn:aws:controltower:us-
east-1:123456789012:enabledcontrol/18J5KBJ3W3VTIRLV",
        "endTime": "2025-04-17T03:08:55+00:00",
        "operationIdentifier": "7691fc5a-de87-4540-8c95-b0aabd56382c",
        "operationType": "ENABLE_CONTROL",
        "startTime": "2025-04-17T03:07:52+00:00",
        "status": "SUCCEEDED",
        "statusMessage": "Operation was successful.",
        "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-s64ryixxxx/
ou-oqxx-i5wnxxxx"
    }
}
```

Weitere Informationen finden Sie unter Über Steuerungen im AWS Control Tower im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetControlOperationin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
```

```
                            );
                        }

                        throw new CompletionException("Failed to get control
 operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- Einzelheiten zur API finden Sie GetControlOperationin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
```

```
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie GetControlOperationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_output) = io_ctt->getcontroloperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_controloperation( )->get_status( ).
```

- Einzelheiten zur API finden Sie GetControlOperationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListBaselines`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListBaselines` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ListBaselinesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Control Tower Tower-Baselines aufzulisten

Das folgende `list-baselines` Beispiel listet alle verfügbaren AWS Control Tower Tower-Baselines auf.

```
aws controltower list-baselines
```

Ausgabe:

```
{
    "baselines": [
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/4T4HA1KMO10S6311",
            "description": "Sets up resources to monitor security and compliance
 of accounts in your organization.",
            "name": "AuditBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/J8HX46AHS5MIKQPD",
            "description": "Sets up a central repository for logs of API
 activities and resource configurations from accounts in your organization.",
            "name": "LogArchiveBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/LN25R72TTG6IGPTQ",
            "description": "Sets up shared resources for AWS Identity Center,
 which prepares the AWSControlTowerBaseline to set up Identity Center access for
 accounts.",
            "name": "IdentityCenterBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/17BSJV3IGJ2QSGA2",
            "description": "Sets up resources and mandatory controls for member
 accounts within the target OU, required for AWS Control Tower governance.",
            "name": "AWSControlTowerBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/3WPD0NA6TJ9AOMU2",
            "description": "Sets up a central AWS Backup vault in your
 organization.",
            "name": "BackupCentralVaultBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/H6C5JFCJJ3CPU3J5",
            "description": "Sets up AWS Backup Audit Manager.",
            "name": "BackupAdminBaseline"
        },
        {
            "arn": "arn:aws:controltower:us-east-1::baseline/APO9ATVPBKFRRGLK",
            "description": "Sets up a local AWS Backup vault and attaches
 multiple AWS Backup plans.",
```

```
            "name": "BackupBaseline"
        }
    ]
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListBaselinesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all available baselines using pagination to retrieve complete
 results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
```

```
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total:
" + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- Einzelheiten zur API finden Sie ListBaselinesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
```

```
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListBaselinesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA lt_baselines TYPE /aws1/cl_cttbaselinesummary=>tt_baselines.
DATA lv_nexttoken TYPE /aws1/cttstring.

" List all baselines using pagination
DO.
  DATA(lo_output) = io_ctt->listbaselines(
    iv_nexttoken = lv_nexttoken
```

```
      ).

      APPEND LINES OF lo_output->get_baselines( ) TO lt_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_baselines = lt_baselines.
    MESSAGE 'Listed baselines successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie ListBaselinesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListEnabledBaselines`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListEnabledBaselines` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
```

```
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ListEnabledBaselinesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Control Tower Tower-fähige Baselines aufzulisten

Das folgende `list-enabled-baselines` Beispiel listet alle aktivierten AWS Control Tower Tower-Baselines auf.

```
aws controltower list-enabled-baselines
```

Ausgabe:

```
{
    "enabledBaselines": [
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XAJNZNCBC1I386C7B",
            "baselineIdentifier": "arn:aws:controltower:us-east-1::baseline/
LN25R72TTG6IGPTQ",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/123456789012"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XAH3ZJL9DWA386CA5",
            "baselineIdentifier": "arn:aws:controltower:us-
east-1::baseline/4T4HA1KMO10S6311",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/012345098765"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledbaseline/
XALFJ9548TL386CBT",
            "baselineIdentifier": "arn:aws:controltower:us-east-1::baseline/
J8HX46AHS5MIKQPD",
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:account/
o-3onqfuxxxx/098765432109"
        }
    ]
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListEnabledBaselinesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists all enabled baselines using pagination to retrieve complete results.
    *
    * @return a list of all enabled baselines
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
       System.out.println("Starting list enabled baselines paginator…");

       ListEnabledBaselinesRequest request =
               ListEnabledBaselinesRequest.builder().build();

       ListEnabledBaselinesPublisher paginator =
               getAsyncClient().listEnabledBaselinesPaginator(request);

       List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
       return paginator.subscribe(response -> {
                   if (response.enabledBaselines() != null
                           && !response.enabledBaselines().isEmpty()) {

                       response.enabledBaselines().forEach(baseline -> {
                           enabledBaselines.add(baseline);
                       });
                   } else {
                       System.out.println("Page contained no enabled
baselines.");
                   }
               })
               .thenRun(() ->
                       System.out.println(
```

```
                                "Successfully listed enabled baselines. Total: "
+
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled
baselines: %s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

- Einzelheiten zur API finden Sie ListEnabledBaselinesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
```

```
            paginator =
 self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListEnabledBaselinesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_enabled_baselines TYPE /aws1/
cl_cttenbdbaselinesumm=>tt_enabledbaselines.
    DATA lv_nexttoken TYPE /aws1/cttlstenbdbaselinesnex00.

    " List all enabled baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledbaselines(
        iv_nexttoken = lv_nexttoken
```

```
      ).

      APPEND LINES OF lo_output->get_enabledbaselines( ) TO lt_enabled_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_baselines = lt_enabled_baselines.
    MESSAGE 'Listed enabled baselines successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie ListEnabledBaselinesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListEnabledControls`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListEnabledControls` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit
identifier.</param>
```

```
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>>
ListEnabledControlsAsync(string targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ListEnabledControlsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Control Tower Tower-fähige Steuerungen aufzulisten

Im folgenden `list-enabled-controls` Beispiel werden Details zu AWS Control Tower Tower-fähigen Steuerelementen abgerufen.

```
aws controltower list-enabled-controls \
    --target-identifier arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-oqxx-i5wnxxxx
```

Ausgabe:

```
{
    "enabledControls": [
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledcontrol/26RGJRSLXCP1KW8D",
            "controlIdentifier": "arn:aws:controltower:us-east-1::control/AWS-GR_CLOUDTRAIL_CHANGE_PROHIBITED",
            "driftStatusSummary": {
                "driftStatus": "NOT_CHECKING"
            },
            "statusSummary": {
                "status": "SUCCEEDED"
            },
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-s64ryxxxxx/ou-oqxx-i5wnxxxx"
        },
        {
            "arn": "arn:aws:controltower:us-east-1:123456789012:enabledcontrol/18AY24CWKM6IVSLU",
            "controlIdentifier": "arn:aws:controltower:us-east-1::control/AWS-GR_CLOUDTRAIL_CLOUDWATCH_LOGS_ENABLED",
            "driftStatusSummary": {
                "driftStatus": "NOT_CHECKING"
            },
            "statusSummary": {
                "status": "SUCCEEDED"
            },
```

```
            "targetIdentifier": "arn:aws:organizations::123456789012:ou/o-
s64ryxxxxx/ou-oqxx-i5wnxxxx"
        }
    ]
}
```

Weitere Informationen finden Sie unter Über Steuerungen im AWS Control Tower im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListEnabledControlsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target "
 + targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();
```

```
        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled
 controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +"
 enabled controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to
 work with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
 %s".formatted(e.getMessage()), e);
```

```
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
 %s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled
 controls", cause);
                });
    }
```

- Einzelheiten zur API finden Sie ListEnabledControlsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
```

```
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with
controls.")
                return enabled_controls
            else:
                logger.error(
```

```
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListEnabledControlsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_enabled_controls TYPE /aws1/
cl_cttenabledcontrolsumm=>tt_enabledcontrols.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all enabled controls using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledcontrols(
        iv_targetidentifier = iv_target_identifier
        iv_nexttoken        = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_enabledcontrols( ) TO lt_enabled_controls.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_controls = lt_enabled_controls.
```

```
    MESSAGE 'Listed enabled controls successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie ListEnabledControlsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListLandingZones`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListLandingZones` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
```

```
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ListLandingZonesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Landezonenkennung für den Control Tower aufzulisten

Das folgende `list-landing-zones` Beispiel listet eine landing zone im Control Tower aufidentifier.

```
aws controltower list-landing-zones
```

Ausgabe:

```
{
    "landingZones": [
        {
            "arn": "arn:aws:controltower:us-
east-1:123456789012:landingzone/13CJG46WZKXXX4X5"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte mit AWS Control Tower im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListLandingZonesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request =
 ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
```

```
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing
zones: " + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
   }
```

- Einzelheiten zur API finden Sie ListLandingZonesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
 self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
```

```
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListLandingZonesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_landing_zones TYPE /aws1/
cl_cttlandingzonesummary=>tt_landingzonesummaries.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all landing zones using pagination
    DO.
      DATA(lo_output) = io_ctt->listlandingzones(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_landingzones( ) TO lt_landing_zones.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.
```

```
    ot_landing_zones = lt_landing_zones.
    MESSAGE 'Listed landing zones successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie ListLandingZonesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ResetEnabledBaseline`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ResetEnabledBaseline` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
```

```
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ResetEnabledBaselinein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So setzen Sie eine für Control Tower aktivierte Baseline zurück

Im folgenden `reset-enabled-baseline` Beispiel wird eine für AWS Control Tower aktivierte Baseline zurückgesetzt.

```
aws controltower reset-enabled-baseline \
    --enabled-baseline-identifier arn:aws:controltower:us-
east-1:123456789012:enabledbaseline/XOM12BEL4YD578CQ2
```

Ausgabe:

```
{
    "operationIdentifier": "214cde95-5c39-46b9-b429-4fad550a7096"
}
```

Weitere Informationen finden Sie unter Typen von Baselines im AWS Control Tower Tower-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ResetEnabledBaselinein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
 reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
```

```
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        ResetEnabledBaselineRequest request =
ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);

                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status)
|| "FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline
operation finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
```

```
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking
baseline operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code="
+ errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline:
" + cause.getMessage());
                    return null;
                });
    }
```

- Einzelheiten zur API finden Sie ResetEnabledBaselinein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client:
 boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.
```

```
        :param enabled_baseline_identifier: The identifier of the enabled
baseline to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ResetEnabledBaselinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Reset the enabled baseline
    DATA(lo_output) = io_ctt->resetenabledbaseline(
      iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttbaselineopstatus.
    DO 100 TIMES.
      lv_status = get_baseline_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Baseline operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Baseline reset successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie ResetEnabledBaselinein der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Firehose mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Data Firehose mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Firehose-Benutzerhandbuch – Weitere Informationen zu Firehose.
- Firehose-API-Referenz – Details zu allen verfügbaren Firehose-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Firehose AWS SDKs
  - Aktionen für die Verwendung von Firehose AWS SDKs
    - Verwenden Sie PutRecordmit einem AWS SDK oder CLI
    - Verwenden Sie PutRecordBatchmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Firehose AWS SDKs
  - Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Grundlegende Beispiele für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Data Firehose mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für die Verwendung von Firehose AWS SDKs
  - Verwenden Sie PutRecordmit einem AWS SDK oder CLI
  - Verwenden Sie PutRecordBatchmit einem AWS SDK oder CLI

Aktionen für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Firehose-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Firehouse-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Firehose AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Data Firehose.

Beispiele

- Verwenden Sie PutRecordmit einem AWS SDK oder CLI
- Verwenden Sie PutRecordBatchmit einem AWS SDK oder CLI

Verwenden Sie **PutRecord**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecord` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Datensätze in Firehose einfügen

CLI

AWS CLI

So schreiben Sie einen Datensatz in einen Stream

Im folgenden Beispiel für `put-record` werden Daten in einen Stream geschrieben. Die Daten sind im Base64-Format codiert.

```
aws firehose put-record \
    --delivery-stream-name my-stream \
    --record '{"Data":"SGVsbG8gd29ybGQ="}'
```

Ausgabe:

```
{
    "RecordId": "RjB5K/nnoGFHqwTsZlNd/
TTqvjE8V5dsyXZTQn2JXrdpMTOwssyEb6nfC8fwf1whhwnItt4mvrn+gsqeK5jB7QjuLg283+Ps4Sz/
j1Xujv31iDhnPdaLw4BOyM9Amv7PcCuB2079RuM0NhoakbyUymlwY8yt20G8X2420wu1jlFafhci4erAt7QhDEvpw
    "Encrypted": false
```

```
}
```

Weitere Informationen finden Sie unter Senden von Daten an einen Bereitstellungsstream von Amazon Kinesis Data Firehose im Entwicklerhandbuch für Amazon Kinesis Data Firehose.

- Einzelheiten zur API finden Sie PutRecordin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
```

```
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
 e);
        }
    }
```

- Einzelheiten zur API finden Sie PutRecordin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
```

```
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- Einzelheiten zur API finden Sie PutRecordin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

    DATA(lo_result) = lo_frh->putrecord(
      iv_deliverystreamname = iv_deliv_stream_name
      io_record             = lo_record ).

    MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
  CATCH /aws1/cx_frhresourcenotfoundex.
    MESSAGE 'Delivery stream not found.' TYPE 'E'.
  CATCH /aws1/cx_frhinvalidargumentex.
    MESSAGE 'Invalid argument provided.' TYPE 'E'.
  CATCH /aws1/cx_frhserviceunavailex.
    MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutRecordin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutRecordBatch`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecordBatch` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Datensätze in Firehose einfügen

CLI

AWS CLI

So schreiben Sie mehrere Datensätze in einen Stream

Im folgenden Beispiel für `put-record-batch` werden drei Datensätze in einen Stream geschrieben. Die Daten sind im Base64-Format codiert.

```
aws firehose put-record-batch \
    --delivery-stream-name my-stream \
    --records file://records.json
```

Inhalt von `myfile.json`:

```
[
    {"Data": "Rmlyc3QgdGhpbmc="},
    {"Data": "U2Vjb25kIHRoaW5n"},
    {"Data": "VGhpcmQgdGhpbmc="}
]
```

Ausgabe:

```
{
    "FailedPutCount": 0,
    "Encrypted": false,
    "RequestResponses": [
        {
            "RecordId": "9D2OJ6t2EqCTZTXwGzeSv/EVHxRoRCw89xd+o3+sXg8DhYOaWKPSmZy/
CGlRVEys1u1xbeKh6VofEYKkoeiDrcjrxhQp9iF7sUW7pujiMEQ5LzlrzCkGosxQn
+3boDnURDEaD42V7GiixpOyLJkYZcae1i7HzlCEoy9LJhMr8EjDSi4Om/9Vc2uhwwuAtGE0XKpxJ2WD7ZRWtAnYlK
        },
        {
            "RecordId": "jFirejqxCLlK5xjH/UNmlMVcjktEN76I7916X9PaZ
+PVaOSXDfU1WGOqEZhxq2js7xcZ552eoeDxsuTU1MSq9nZTbVfb6cQTIXnm/
GsuF37Uhg67GKmR5z90l6XKJ+/
+pDloFv7Hh9a3oUS6wYm3DcNRLTHHAimANp1PhkQvWpvLRfzbuCUkBphR2QVzhP9OiHLbzGwy8/
DfH8sqWEUYASNJKS8GXP5s"
        },
        {
            "RecordId":
 "oy0amQ40o5Y2YV4vxzufdcMOOw6n3EPr3tpPJGoYVNKH4APPVqNcbUgefo1stEFRg4hTLrf2k6eliHu/9+YJ5R3
DTBt3qBlmTj7Xq8SKVbO1S7YvMTpWkMKA86f8JfmT8BMKoMb4XZS/sOkQLe+qh0sYKXWl"
```

```
        }
    ]
}
```

Weitere Informationen finden Sie unter Senden von Daten an einen Bereitstellungsstream von Amazon Kinesis Data Firehose im Entwicklerhandbuch für Amazon Kinesis Data Firehose.

- Einzelheiten zur API finden Sie PutRecordBatchin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();
```

```
        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord =
objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose
record", e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- Einzelheiten zur API finden Sie PutRecordBatchin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
 region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
```

```
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")
```

- Einzelheiten zur API finden Sie PutRecordBatchin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- Einzelheiten zur API finden Sie PutRecordBatchin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).

        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.'
 TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
```

```
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutRecordBatchin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Firehose AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Firehose mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Firehouse oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Verwenden von Amazon Data Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze

Die folgenden Codebeispiele zeigen, wie Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze verwendet werden kann.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
```

```
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
```

```
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery
stream.
     *
     * @param record The record to be put to the delivery stream. The record must
be a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream
name is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

```
   /**
    * Puts a batch of records to an Amazon Kinesis Data Firehose delivery
stream.
    *
    * @param records           a list of maps representing the records to be
sent
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null
or empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or
delivery stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord =
objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose
record", e);
                   }
               }).collect(Collectors.toList());
```

```
                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " +
batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime,
Instant endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).bui
                .startTime(startTime)
                .endTime(endTime)
```

```
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8))
 {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutRecord
  - PutRecordBatch

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Skript werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
```

```
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and
region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose
delivery stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
```

```
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records.
Error: {e}")


    def get_metric_statistics(
        self,
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
```

```
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data
points.
            statistics (list): A list of statistics to retrieve. Default is
['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value":
self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords',
and 'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())

        metrics = {
```

```
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5
minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
```

```
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
```

```
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

Diese Datei enthält die Konfiguration für das oben genannte Skript.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - PutRecord
  - PutRecordBatch

Codebeispiele für Device Farm mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Device Farm mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Device-Farm-Entwicklerhandbuch – Weitere Informationen zu Device Farm.
- API-Referenz für Device Farm – Details zu allen verfügbaren Device-Farm-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Device Farm AWS SDKs
  - Aktionen für die Verwendung von Device Farm AWS SDKs
    - Verwendung von CreateUpload mit einer CLI
- Szenarien für die Verwendung von Device Farm AWS SDKs
  - Führen Sie Browsertests mit Device Farm durch und machen Sie Screenshots mit einem AWS SDK
  - Laden Sie Pakete für mobile Geräte mit Device Farm hoch und testen Sie sie mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Device Farm AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Device Farm mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für die Verwendung von Device Farm AWS SDKs
  - Verwendung von CreateUpload mit einer CLI

Aktionen für die Verwendung von Device Farm AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Device Farm Farm-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Device-Farm-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Device Farm AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Device Farm -API-Referenz.

Beispiele

- Verwendung von CreateUpload mit einer CLI

Verwendung von **`CreateUpload`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateUpload` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Hochladen und Testen von Gerätepaketen

CLI

AWS CLI

So erstellen Sie einen Upload

Der folgende Befehl erstellt einen Upload für eine Android-App:

```
aws devicefarm create-upload --project-arn "arn:aws:devicefarm:us-west-2:123456789012:project:070fc3ca-7ec1-4741-9c1f-d3e044efc506" --name app.apk
 --type ANDROID_APP
```

Sie können den Projekt-ARN aus der Ausgabe von create-project oder list-projects abrufen.

Ausgabe:

```
{
    "upload": {
        "status": "INITIALIZED",
        "name": "app.apk",
        "created": 1503614408.769,
```

```
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEPBUMBC3GA%2F20170824%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f",
        "type": "ANDROID_APP",
        "arn": "arn:aws:devicefarm:us-
west-2:123456789012:upload:070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514"
    }
}
```

Verwenden Sie die signierte URL in der Ausgabe, um eine Datei auf Device Farm hochzuladen:

```
curl -T app.apk "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
arn%3Aaws%3Adevicefarm%3Aus-west-2%3A123456789012%3Aproject%3A070fc3ca-
c7e1-4471-91cf-d3e4efc50604/uploads/arn%3Aaws%3Adevicefarm%3Aus-
west-2%3A123456789012%3Aupload%3A070fc3ca-7ec1-4741-9c1f-d3e044efc506/dd72723a-
ae9e-4087-09e6-f4cea3599514/app.apk?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20170824T224008Z&X-Amz-SignedHeaders=host&X-Amz-Expires=86400&X-Amz-
Credential=AKIAEXAMPLEPBUMBC3GA%2F20170824%2Fus-west-2%2Fs3%2Faws4_request&X-Amz-
Signature=05050370c38894ef5bd09f5d009f36fc8f96fa4bb04e1bba9aca71b8dbe49a0f"
```

- Einzelheiten zur API finden Sie CreateUploadin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein AWS Device Farm Farm-Upload für eine Android-App erstellt. Sie können den Projekt-ARN aus der Ausgabe von New-DFProject oder abrufen Get-DFProjectList. Verwenden Sie die signierte URL in der New-DFUpload Ausgabe, um eine Datei auf Device Farm hochzuladen.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Einzelheiten zur API finden Sie unter CreateUpload AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein AWS Device Farm Farm-Upload für eine Android-App erstellt. Sie können den Projekt-ARN aus der Ausgabe von New-DFProject oder abrufen Get-DFProjectList. Verwenden Sie die signierte URL in der New-DFUpload Ausgabe, um eine Datei auf Device Farm hochzuladen.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- Einzelheiten zur API finden Sie unter CreateUpload AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Device Farm AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Device Farm mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Device Farm oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Führen Sie Browsertests mit Device Farm durch und machen Sie Screenshots mit einem AWS SDK
- Laden Sie Pakete für mobile Geräte mit Device Farm hoch und testen Sie sie mithilfe eines AWS SDK

Führen Sie Browsertests mit Device Farm durch und machen Sie Screenshots mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit Device-Farm-Browsertests durchführen und Screenshots erstellen können.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie PyTest und Selenium, um bestimmte Websites aufzurufen, Screenshots zu machen und den tatsächlichen Inhalt der Website mit dem erwarteten Inhalt zu vergleichen.

```
import datetime
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait


def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return (
            subprocess.check_output(["git", "rev-parse", "--short", "HEAD"])
            .decode("utf-8")
            .strip()
        )
```

```
    except:
        return "norepo"


class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.

    """

    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))

    def setup_method(self, method):
        """
        Set up a test.

        This makes sure that the session for an individual test is ready.

        The AWS credentials are read from the default ~/.aws/credentials or from
 the
        command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY
 environment
        variables.

        The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
        environment variable.
        """
        devicefarm_client = boto3.client("devicefarm")
        project_arn = os.environ.get("PROJECT_ARN", None)
        if project_arn is None:
            raise ValueError("Must set PROJECT_ARN")
        # Request a driver hub URL for the Selenium client
        testgrid_url_response = devicefarm_client.create_test_grid_url(
            projectArn=project_arn, expiresInSeconds=300
        )

        # We want a directory to save our files into. We're going to make a
 directory
        # in the current directory that holds our results.
        self.screenshot_path = os.path.join(
```

```
            ".", "results", get_git_hash() + "-" +
 (datetime.date.today().isoformat())
        )
        if not os.path.exists(self.screenshot_path):
            os.makedirs(self.screenshot_path, exist_ok=True)

        # We want a Firefox instance on Windows
        desired_cap = DesiredCapabilities.FIREFOX
        desired_cap["platform"] = "windows"
        desired_cap["BrowserVersion"] = "latest"

        # Configure the webdriver with the appropriate remote endpoint.
        self.driver = webdriver.Remote(testgrid_url_response["url"], desired_cap)

        #
        # Auto-Tagging
        #

        # In order to get the Session ARN, we need to look up the session by the
        # Project ARN and session ID (from the driver).
        testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
            projectArn=project_arn, sessionId=self.driver.session_id
        )

        # Save the session's ARN so we can tag the session.
        self.session_arn = testgrid_session_arn_response["testGridSession"]
["arn"]

        # In order to tag it, we're going to use the resourcegroupstaggingapi
 client to
        # add a tag to the session ARN that we just got.
        tag_client = boto3.client("resourcegroupstaggingapi")
        tag_client.tag_resources(
            ResourceARNList=[self.session_arn],
            Tags={"TestSuite": f"testsuite {method.__name__}", "GitId":
 get_git_hash()},
        )

    def teardown_method(self, method):
        """
        Clean up resources used by each method.
        """
        # End the Selenium session so we're off the clock.
        self.driver.quit()
```

```
    @pytest.mark.parametrize(
        "query,leading",
        [
            pytest.param(
                "Seattle",
                "Seattle (/si#ætəl/ (listen) see-AT-əl) is a seaport city on the
West Coast of the United States.",
            ),
            pytest.param(
                "Selenium",
                "Selenium is a chemical element with the symbol Se and atomic
number 34.",
            ),
            pytest.param(
                "Amazon Locker",
                "Amazon Locker is a self-service package delivery service offered
by online retailer Amazon.",
            ),
            pytest.param(
                "Kootenai Falls",
                "Kootenai Falls is a waterfall on the Kootenay River located in
Lincoln County, Montana, just off U.S. Route 2.",
            ),
            pytest.param(
                "Dorayaki",
                "Dorayaki (####, ####, ####, ####) is a type of Japanese
confection.",
            ),
            pytest.param("Robot Face", "<|°_°|> (also known as Robot Face or
Robot)"),
        ],
    )
    def test_first_paragraph_text(self, query, leading):
        """
        This test looks at the first paragraph of a page on Wikipedia, comparing
it to
        a known leading sentence.

        If the leading sentence matches, the test passes. A screenshot is taken
before
        the final assertion is made, letting us debug if something isn't right.
        """
        # Open the main page of Wikipedia
```

```
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter a query, and press enter
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # Wait for the search box to go stale -- This means we've navigated
fully.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the leading paragraph of the article.
        lead = leading.lower()
        # Find the element...
        lead_para = self.driver.find_element(
            By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]"
        )
        # ... and copy out its text.
        our_text = lead_para.text.lower()
        our_text = our_text[: len(lead)]
        # Take a screenshot and compare the strings.
        self.save_screenshot(f"leadingpara_{query}.png")
        assert our_text.startswith(lead)

    @pytest.mark.parametrize(
        "query,expected",
        [
            pytest.param("Automation Testing", "Test Automation"),
            pytest.param("DevOps", "DevOps"),
            pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
            pytest.param("EarthBound", "EarthBound"),
            pytest.param("Covered Bridges Today", "Covered Bridges Today"),
            pytest.param("Kurt Godel", "Kurt Gödel"),
            pytest.param("N//ng language", "N#ng language"),
            pytest.param(
                "Who the Frick Is Jackson Pollock?", "Who the $&% Is Jackson
Pollock?"
            ),
        ],
    )
    def test_redirect_titles(self, query, expected):
        """
        A test comparing pages we expect to (or not to) redirect on Wikipedia.
```

```
        This test checks to see that the page ("query") redirects (or doesn't) to
the
        "expected" page title. Several of these are common synonyms
("Jackdaws...")
        while others are because of characters untypable by most keyboards ("N#ng
language")

        A screenshot is taken just before the final assertion is made to aid in
        debugging and verification.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter some text into it, and send an enter key.
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # wait until the page has rolled over -- once the search input handle is
stale,
        # the browser has navigated.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the first heading & take a screenshot
        our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
        self.save_screenshot(f"redirect_{query}.png")
        # did it match?
        assert our_text == expected.lower()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTestGridUrl
  - GetTestGridSession

Laden Sie Pakete für mobile Geräte mit Device Farm hoch und testen Sie sie mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Pakete für mobile Geräte mit Device Farm hochladen und testen können.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie die kompilierte Android-Anwendung und die Testpakete auf Device Farm hoch, starten Sie einen Test, warten Sie auf den Abschluss des Tests und melden Sie die Ergebnisse.

```
import boto3
import os
import requests
import string
import random
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
    # This is our app under test.
    "appFilePath": "app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
    "poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
    "namePrefix": "MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage": "tests.zip",
```

```
}

client = boto3.client("devicefarm")

unique = (
    config["namePrefix"]
    + "-"
    + (datetime.date.today().isoformat())
    + ("".join(random.sample(string.ascii_letters, 8)))
)

print(
    f"The unique identifier for this run is '{unique}'. All uploads will be
 prefixed "
    f"with this."
)


def upload_df_file(filename, type_, mime="application/octet-stream"):
    upload_response = client.create_upload(
        projectArn=config["projectArn"],
        name=unique + "_" + os.path.basename(filename),
        type=type_,
        contentType=mime,
    )
    upload_arn = upload_response["upload"]["arn"]
    # Extract the URL of the upload and use Requests to upload it.
    upload_url = upload_response["upload"]["url"]
    with open(filename, "rb") as file_stream:
        print(
            f"Uploading {filename} to Device Farm as "
            f"{upload_response['upload']['name']}... ",
            end="",
        )
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime}
        )
        print(" done")
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(
```

```
            f"Upload of {filename} in state {upload_response['upload']['status']}
 "
            f"after " + str(datetime.datetime.now() - started)
        )
        if upload_response["upload"]["status"] == "FAILED":
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is:
 \n"
                f"{+upload_response['upload']['message']}"
            )
        if upload_response["upload"]["status"] == "SUCCEEDED":
            break
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn


our_upload_arn = upload_df_file(config["appFilePath"], "ANDROID_APP")
our_test_package_arn = upload_df_file(
    config["testPackage"], "APPIUM_PYTHON_TEST_PACKAGE"
)
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],
    appArn=our_upload_arn,
    devicePoolArn=config["poolArn"],
    name=unique,
    test={
        "type": "APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn,
    },
)
run_arn = response["run"]["arn"]
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = "UNKNOWN"
try:
    while True:
        response = client.get_run(arn=run_arn)
        state = response["run"]["status"]
```

```
        if state == "COMPLETED" or state == "ERRORED":
            break
        else:
            print(
                f" Run {unique} in state {state}, total "
                f"time {datetime.datetime.now() - start_time}"
            )
            time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)

print(f"Tests finished in state {state} after {datetime.datetime.now() -
 start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), "results", unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response["jobs"]:
    job_name = job["name"]
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job["arn"])["suites"]
    for suite in suites:
        for test in client.list_tests(arn=suite["arn"])["tests"]:
            # Get the artifacts.
            for artifact_type in ["FILE", "SCREENSHOT", "LOG"]:
                artifacts = client.list_artifacts(type=artifact_type,
 arn=test["arn"])[
                    "artifacts"
                ]
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows &
 macOS.
                    path_to = os.path.join(
                        save_path,
                        job_name,
                        suite["name"],
                        test["name"].replace(":", "_"),
                    )
                    os.makedirs(path_to, exist_ok=True)
                    filename = (
```

```
                        artifact["type"]
                        + "_"
                        + artifact["name"]
                        + "."
                        + artifact["extension"]
                    )
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, "wb") as fn:
                        with requests.get(
                            artifact["url"], allow_redirects=True
                        ) as request:
                            fn.write(request.content)
print("Finished")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateUpload
  - GetRun
  - GetUpload
  - ListArtifacts
  - ListJobs
  - ListSuites
  - ListTests
  - ScheduleRun
  - StopRun

Codebeispiele für Direct Connect verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Direct Connect mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Direct Connect Benutzerhandbuch — Weitere Informationen zu Direct Connect.
- Direct Connect API-Referenz — Details zu allen verfügbaren Direct Connect Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für Direct Connect verwenden AWS SDKs
  - Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI

Szenarien für Direct Connect verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien Direct Connect mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb Direct Connect oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI

Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 CreateVpnGateway
- Verwenden Sie ec2 DeleteVpnGateway
- Verwenden Sie ec2 DescribeVpnGateways
- Verwenden Sie Directconnect CreateConnection
- Verwenden Sie Directconnect CreatePrivateVirtualInterface

- Verwenden Sie Directconnect DeleteConnection

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect
 connections using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
```

```
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
```

```
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is
 $CONNECTION_STATE, waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
```

```
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
```

```
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID"
 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"'
 -f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual
 interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
```

```
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o
 '"virtualInterfaceId": "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
```

```
fi

# Step 8: Check connection state and provide guidance for public virtual
 interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
```

```
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is
 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
```

```
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways
  - UpdateConnection

Codebeispiele für Directory Service verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Directory Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Directory Service Administratorhandbuch — Weitere Informationen zu Directory Service.

- Directory Service API-Referenz — Details zu allen verfügbaren Directory Service Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Directory Service verwenden AWS SDKs
  - Aktionen für Directory Service verwenden AWS SDKs
    - Verwendung von DescribeDirectories mit einer CLI
    - Verwendung von DescribeTrusts mit einer CLI

Grundlegende Beispiele für Directory Service verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Directory Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Directory Service verwenden AWS SDKs
  - Verwendung von DescribeDirectories mit einer CLI
  - Verwendung von DescribeTrusts mit einer CLI

Aktionen für Directory Service verwenden AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne Directory Service Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Directory Service -API-Referenz.

Beispiele

- Verwendung von DescribeDirectories mit einer CLI
- Verwendung von DescribeTrusts mit einer CLI

Verwendung von **`DescribeDirectories`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDirectories` verwendet wird.

CLI

AWS CLI

So rufen Sie Details zu Ihren Verzeichnissen ab

Im folgenden Beispiel für `describe-directories` werden die Details für das angegebene Verzeichnis angezeigt.

```
aws ds describe-directories \
   --directory-id d-a1b2c3d4e5
```

Ausgabe:

```
{
    "DirectoryDescriptions": [
        {
            "DirectoryId": "d-a1b2c3d4e5",
            "Name": "mydirectory.example.com",
            "ShortName": "mydirectory",
            "Size": "Small",
            "Edition": "Standard",
            "Alias": "d-a1b2c3d4e5",
            "AccessUrl": "d-a1b2c3d4e5.awsapps.com",
            "Stage": "Active",
            "ShareStatus": "Shared",
            "ShareMethod": "HANDSHAKE",
            "ShareNotes": "These are my share notes",
            "LaunchTime": "2019-07-08T15:33:46.327000-07:00",
            "StageLastUpdatedDateTime": "2019-07-08T15:59:12.307000-07:00",
            "Type": "SharedMicrosoftAD",
            "SsoEnabled": false,
            "DesiredNumberOfDomainControllers": 0,
            "OwnerDirectoryDescription": {
                "DirectoryId": "d-b2c3d4e5f6",
                "AccountId": "123456789111",
                "DnsIpAddrs": [
                    "203.113.0.248",
                    "203.113.0.253"
```

```
                ],
                "VpcSettings": {
                    "VpcId": "vpc-a1b2c3d4",
                    "SubnetIds": [
                        "subnet-a1b2c3d4",
                        "subnet-d4c3b2a1"
                    ],
                    "AvailabilityZones": [
                        "us-west-2a",
                        "us-west-2c"
                    ]
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeDirectoriesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft Informationen über die Verzeichnisse ab, die zu diesem Konto gehören.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Ausgabe:

```
DirectoryId  Name           DnsIpAddrs                     Type
-----------  ----           ----------                     ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Einzelheiten zur API finden Sie unter DescribeDirectories AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft Informationen über die Verzeichnisse ab, die zu diesem Konto gehören.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Ausgabe:

```
DirectoryId  Name           DnsIpAddrs                      Type
-----------  ----           ----------                      ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- Einzelheiten zur API finden Sie unter DescribeDirectories AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTrusts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrusts` verwendet wird.

CLI

AWS CLI

So rufen Sie Details zu Ihren Vertrauensbeziehungen ab

Im folgenden Beispiel für `describe-trusts` werden die Details zu den Vertrauensbeziehungen des angegebenen Verzeichnisses angezeigt.

```
aws ds describe-trusts \
   --directory-id d-a1b2c3d4e5
```

Ausgabe:

```
{
    "Trusts": [
        {
```

```
            "DirectoryId": "d-a1b2c3d4e5",
            "TrustId": "t-9a8b7c6d5e",
            "RemoteDomainName": "other.example.com",
            "TrustType": "Forest",
            "TrustDirection": "Two-Way",
            "TrustState": "Verified",
            "CreatedDateTime": "2017-06-20T18:08:45.614000-07:00",
            "LastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
            "StateLastUpdatedDateTime": "2019-06-04T10:52:12.410000-07:00",
            "SelectiveAuth": "Disabled"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeTrustsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft die Informationen zu den Vertrauensbeziehungen ab, die für die angegebene directory-id erstellt wurden.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Ausgabe:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- Einzelheiten zur API finden Sie unter DescribeTrusts AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft die Informationen zu den Vertrauensbeziehungen ab, die für die angegebene directory-id erstellt wurden.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Ausgabe:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- Einzelheiten zur API finden Sie unter DescribeTrusts AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für AWS DMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Database Migration Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS DMS Entwicklerhandbuch — Weitere Informationen zu AWS DMS.
- AWS DMS API-Referenz — Details zu allen verfügbaren AWS DMS Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS DMS verwenden AWS SDKs
  - Aktionen für AWS DMS verwenden AWS SDKs
    - Verwendung von CreateReplicationTask mit einer CLI

Grundlegende Beispiele für AWS DMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Database Migration Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS DMS verwenden AWS SDKs
  - Verwendung von CreateReplicationTask mit einer CLI

Aktionen für AWS DMS verwenden AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne AWS DMS Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Database Migration Service -API-Referenz.

Beispiele

- Verwendung von CreateReplicationTask mit einer CLI

Verwendung von **`CreateReplicationTask`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateReplicationTask` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Replikationsaufgabe

Im folgenden Beispiel für `create-replication-task` wird eine Replikationsaufgabe erstellt.

```
aws dms create-replication-task \
    --replication-task-identifier movedata \
    --source-endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA \
    --target-endpoint-arn arn:aws:dms:us-
east-1:123456789012:endpoint:EOM4SFKCZEYHZBFGAGZT3QEC5U \
    --replication-instance-arn $RI_ARN \
    --migration-type full-load \
    --table-mappings file://table-mappings.json
```

Inhalt von `table-mappings.json`:

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "prodrep",
                "table-name": "%"
            },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

Ausgabe:

```
{
    "ReplicationTask": {
        "ReplicationTaskIdentifier": "moveit2",
        "SourceEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:6GGI6YPWWGAYUVLKIB732KEVWA",
        "TargetEndpointArn": "arn:aws:dms:us-
east-1:123456789012:endpoint:EOM4SFKCZEYHZBFGAGZT3QEC5U",
        "ReplicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:T3OM7OUB5NM2LCVZF7JPGJRNUE",
        "MigrationType": "full-load",
        "TableMappings": ...output omitted... ,
        "ReplicationTaskSettings": ...output omitted... ,
```

```
        "Status": "creating",
        "ReplicationTaskCreationDate": 1590524772.505,
        "ReplicationTaskArn": "arn:aws:dms:us-
east-1:123456789012:task:K55IUCGBASJS5VHZJIINA45FII"
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit AWS DMS-Aufgaben im AWS Database Migration Service Service-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter CreateReplicationTask AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Replikationsaufgabe des AWS Datenbankmigrationsdienstes erstellt, die CdcStartTime anstelle von verwendet CdcStartPosition. Der MigrationType ist auf „full-load-and-cdc“ gesetzt, was bedeutet, dass die Zieltabelle leer sein muss. Die neue Aufgabe wird mit einem Tag versehen, dessen Schlüssel „Stufe“ lautet und dessen Schlüsselwert „Test“ ist. Weitere Informationen zu den von diesem Cmdlet verwendeten Werten finden Sie unter Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) im AWS Database Migration Service Service-Benutzerhandbuch.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Einzelheiten zur API finden Sie unter CreateReplicationTask AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Replikationsaufgabe des AWS Datenbankmigrationsdienstes erstellt, die CdcStartTime anstelle von verwendet CdcStartPosition. Der MigrationType ist auf „full-load-and-cdc“ gesetzt, was bedeutet, dass die Zieltabelle leer sein muss. Die neue Aufgabe wird mit einem Tag versehen, dessen Schlüssel „Stufe“ lautet und dessen Schlüsselwert „Test“ ist. Weitere Informationen zu den von diesem Cmdlet verwendeten Werten finden Sie unter Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) im AWS Database Migration Service Service-Benutzerhandbuch.

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- Einzelheiten zur API finden Sie unter CreateReplicationTask AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Amazon DocumentDB mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon DocumentDB (mit MongoDB-Kompatibilität) mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-DocumentDB-Entwicklerhandbuch – Weitere Informationen zu Amazon DocumentDB.

- API-Referenz zu Amazon DocumentDB – Details zu allen verfügbaren Amazon DocumentDB-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Amazon DocumentDB AWS SDKs
  - Erste Schritte mit Dokumentendatenbanken
- Serverless-Beispiele für Amazon DocumentDB
  - Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Szenarien für die Verwendung von Amazon DocumentDB AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon DocumentDB mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon DocumentDB oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Dokumentendatenbanken

Erste Schritte mit Dokumentendatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

################################################################################
# Logging
################################################################################
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Log file: $LOG_FILE"
echo ""

################################################################################
# Region pre-check
################################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

################################################################################
# Error handler
################################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

################################################################################
# Wait function
################################################################################
```

```
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}
```

```
##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}
```

```
##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
```

```
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""
```

```
if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "============================================"
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
```

```
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)
```

```
if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "============================================"
echo "Step 4: Create DocumentDB cluster"
echo "============================================"
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
```

```
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "============================================"
echo "Step 5: Create DocumentDB instance"
echo "============================================"
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
```

```
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
```

```
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
```

```
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
```

```
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Serverless-Beispiele für Amazon DocumentDB

Die folgenden Codebeispiele zeigen, wie Amazon DocumentDB mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaDocDb;

public class Function
{
```

```
     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });

        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
    }



    public class Event
    {
        [JsonPropertyName("eventSourceArn")]
        public string EventSourceArn { get; set; }

        [JsonPropertyName("events")]
        public List<DocumentDBEventRecord> Events { get; set; }
```

```
        [JsonPropertyName("eventSource")]
        public string EventSource { get; set; }
    }

    public class DocumentDBEventRecord
    {
        [JsonPropertyName("event")]
        public EventData Event { get; set; }
    }

    public class EventData
    {
        [JsonPropertyName("_id")]
        public IdData Id { get; set; }

        [JsonPropertyName("clusterTime")]
        public ClusterTime ClusterTime { get; set; }

        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
```

```
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
```

```
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
```

```
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verwenden eines Amazon DocumentDB DocumentDB-Ereignisses mit Lambda. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
```

```
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Ein Amazon DocumentDB DocumentDB-Ereignis mit Lambda verwenden TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };
```

```
// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(),
 Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Codebeispiele für DynamoDB mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon DynamoDB mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden. AWS Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Weitere -Quellen

- DynamoDB-Entwicklerhandbuch – Weitere Informationen zu DynamoDB.
- DynamoDB-API-Referenz – Details zu allen verfügbaren DynamoDB-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von DynamoDB AWS SDKs
  - Hello DynamoDB
  - Erlernen Sie die Grundlagen von DynamoDB mit einem AWS SDK
  - Aktionen für DynamoDB mit AWS SDKs
    - Verwenden Sie BatchExecuteStatementmit einem AWS SDK
    - Verwenden Sie BatchGetItemmit einem AWS SDK oder CLI
    - Verwenden Sie BatchWriteItemmit einem AWS SDK oder CLI
    - Verwenden Sie CreateTablemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteItemmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteTablemit einem AWS SDK oder CLI
    - Verwenden Sie DescribeTablemit einem AWS SDK oder CLI
    - Verwenden Sie DescribeTimeToLivemit einem AWS SDK oder CLI
    - Verwenden Sie mit ExecuteStatementeinem AWS SDK
    - Verwenden Sie GetItemmit einem AWS SDK oder CLI
    - Verwenden Sie ListTablesmit einem AWS SDK oder CLI
    - Verwenden Sie PutItemmit einem AWS SDK oder CLI
    - Verwenden Sie Query mit einem AWS SDK oder CLI
    - Verwenden Sie Scan mit einem AWS SDK oder CLI
    - Verwenden Sie UpdateItemmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateTablemit einem AWS SDK oder CLI
    - Verwenden Sie mit einem UpdateTimeToLive AWS SDK oder CLI
- Szenarien für die Verwendung von DynamoDB AWS SDKs
  - Beschleunigen Sie DynamoDB-Lesevorgänge mit DAX mithilfe eines AWS SDK
  - Arbeiten Sie mit erweiterten DynamoDB Global Secondary Index-Szenarien unter Verwendung von AWS Command Line Interface v2
  - Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
  - Vergleichen Sie mehrere Werte mit einem einzelnen Attribut in DynamoDB mit einem AWS SDK
  - Bedingtes Aktualisieren eines DynamoDB-Elements mit einer TTL unter Verwendung eines AWS SDK
  - Stellen Sie mithilfe eines eine Connect zu einer lokalen DynamoDB-Instanz her AWS SDK

- Zählen Sie Ausdrucksoperatoren in DynamoDB mit einem AWS SDK
- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Erstellen einer Messenger-Anwendung mit Step Functions
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine DynamoDB-Tabelle mit einem globalen sekundären Index mithilfe der AWS SDK
- Erstellen Sie eine DynamoDB-Tabelle mit einer Warmdurchsatzeinstellung AWS SDK
- Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten
- Erstellen einer Websocket-Chat-Anwendung mit API Gateway
- Erstellen Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK
- Erstellen und verwalten Sie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency mithilfe eines AWS SDK
- Erstellen und verwalten Sie globale DynamoDB-Tabellen, die MREC demonstrieren, mit einem AWS SDK
- Löschen Sie DynamoDB-Daten mithilfe von PartiQL DELETE-Anweisungen mit einem AWS SDK
- Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erste Schritte mit NoSQL-Datenbanken
- Fügen Sie DynamoDB-Daten mithilfe von PartiQL INSERT-Anweisungen mit einem AWS SDK
- Aufrufen einer Lambda-Funktion von einem Browser aus
- Globale Sekundärindizes von DynamoDB verwalten mit AWS Command Line Interface v2
- Verwalten Sie ressourcenbasierte DynamoDB-Richtlinien mit AWS Command Line Interface v2
- Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK
- Führen Sie erweiterte DynamoDB-Abfrageoperationen mit einem AWS SDK
- Führen Sie Listenoperationen in DynamoDB mit einem AWS SDK
- Führen Sie Kartenoperationen in DynamoDB mit einem AWS SDK
- Führen Sie Mengenoperationen in DynamoDB mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mithilfe von Batches von PartiQL-Anweisungen und AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mit PartiQL ab und AWS SDK

- Abfragen einer DynamoDB-Tabelle mithilfe eines globalen sekundären Indexes mit einem AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mithilfe einer Begins_with-Bedingung mit einem ab AWS SDK
- Fragen Sie eine DynamoDB-Tabelle unter Verwendung eines Datumsbereichs im Sortierschlüssel mit einem ab AWS SDK
- Abfragen einer DynamoDB-Tabelle mit einem komplexen Filterausdruck mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit einem dynamischen Filterausdruck mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit einem Filterausdruck und einem Limit mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit verschachtelten Attributen unter Verwendung eines AWS SDK
- Abfragen einer DynamoDB-Tabelle mit Paginierung mithilfe eines AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mit stark konsistenten Lesevorgängen ab, indem Sie eine AWS SDK
- Abfragen von DynamoDB-Daten mithilfe von PartiQL SELECT-Anweisungen mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle nach TTL-Elementen mithilfe eines AWS SDK
- Abfragen von DynamoDB-Tabellen mithilfe von Datums- und Uhrzeitmustern mit einem AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK
- Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2
- Verstehen Sie die Reihenfolge der Aktualisierungsausdrücke in DynamoDB mit einem AWS SDK
- Aktualisieren Sie eine DynamoDB-Tabelleneinstellung mit warmem Durchsatz mithilfe eines AWS SDK
- Aktualisieren Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK
- Aktualisieren Sie DynamoDB-Daten mithilfe von PartiQL UPDATE-Anweisungen mit einem AWS SDK
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Verwenden Sie ein Dokumentmodell für DynamoDB mit einem AWS SDK

- Verwenden Sie ein Objektpersistenzmodell auf hoher Ebene für DynamoDB mit einem AWS SDK
- Verwenden Sie atomare Zähleroperationen in DynamoDB mit einem AWS SDK
- Verwenden Sie bedingte Operationen in DynamoDB mit einem AWS SDK
- Verwenden Sie Ausdrucksattributnamen in DynamoDB mit einem AWS SDK
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion
- Arbeiten Sie mit lokalen sekundären DynamoDB-Indizes unter Verwendung von AWS Command Line Interface v2
- Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2
- Arbeiten Sie mit globalen DynamoDB-Tabellen und Multi-Region-Replikation mit Eventual Consistency (MREC) unter Verwendung von AWS Command Line Interface v2
- Arbeiten Sie mit DynamoDB-Ressourcen-Tagging mit AWS Command Line Interface v2
- Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

- Serverless-Beispiele für DynamoDB
  - Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser
  - Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser
- AWS Community-Beiträge für DynamoDB
  - Erstellen und Testen einer Serverless-Anwendung

Grundlegende Beispiele für die Verwendung von DynamoDB AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon DynamoDB mit AWS SDKs verwendet werden.

Beispiele

- Hello DynamoDB
- Erlernen Sie die Grundlagen von DynamoDB mit einem AWS SDK
- Aktionen für DynamoDB mit AWS SDKs
  - Verwenden Sie BatchExecuteStatementmit einem AWS SDK
  - Verwenden Sie BatchGetItemmit einem AWS SDK oder CLI

- Verwenden Sie BatchWriteItemmit einem AWS SDK oder CLI
- Verwenden Sie CreateTablemit einem AWS SDK oder CLI
- Verwenden Sie DeleteItemmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTablemit einem AWS SDK oder CLI
- Verwenden Sie DescribeTablemit einem AWS SDK oder CLI
- Verwenden Sie DescribeTimeToLivemit einem AWS SDK oder CLI
- Verwenden Sie mit ExecuteStatementeinem AWS SDK
- Verwenden Sie GetItemmit einem AWS SDK oder CLI
- Verwenden Sie ListTablesmit einem AWS SDK oder CLI
- Verwenden Sie PutItemmit einem AWS SDK oder CLI
- Verwenden Sie Query mit einem AWS SDK oder CLI
- Verwenden Sie Scan mit einem AWS SDK oder CLI
- Verwenden Sie UpdateItemmit einem AWS SDK oder CLI
- Verwenden Sie UpdateTablemit einem AWS SDK oder CLI
- Verwenden Sie mit einem UpdateTimeToLive AWS SDK oder CLI

Hello DynamoDB

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von DynamoDB beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Microsoft.Extensions.DependencyInjection;
```

```
namespace DynamoDBActions;

/// <summary>
/// A simple example that demonstrates basic DynamoDB operations.
/// </summary>
public class HelloDynamoDB
{
    /// <summary>
    /// HelloDynamoDB lists the existing DynamoDB tables for the default user.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
            )
            .Build();

        // Now the client is available for injection.
        var dynamoDbClient = host.Services.GetRequiredService<IAmazonDynamoDB>();

        try
        {
            var request = new ListTablesRequest();
            var tableNames = new List<string>();

            var paginatorForTables =
 dynamoDbClient.Paginators.ListTables(request);

            await foreach (var tableName in paginatorForTables.TableNames)
            {
                tableNames.Add(tableName);
            }

            Console.WriteLine("Welcome to the DynamoDB Hello Service example. " +
                              "\nLet's list your DynamoDB tables:");
            tableNames.ForEach(table =>
            {
                Console.WriteLine($"Table: {table}");
            });
```

```
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB service error occurred while
 listing tables. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing tables.
 {ex.Message}");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS dynamodb)

# Set this project's name.
project("hello_dynamodb")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_dynamodb.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 (DynamoDB) client and lists the
 *  DynamoDB tables.
 *
 *  main function
 *
 *  Usage: 'hello_dynamodb'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do {
            const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamodbClient.ListTables(
                    listTablesRequest);
            if (!outcome.IsSuccess()) {
                std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
                result = 1;
                break;
            }

            for (const auto &tableName: outcome.GetResult().GetTableNames()) {
                std::cout << tableName << std::endl;
            }

            listTablesRequest.SetExclusiveStartTableName(
                    outcome.GetResult().GetLastEvaluatedTableName());

        } while (!listTablesRequest.GetExclusiveStartTableName().empty());
```

```
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListTablesunter AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
```

```
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request =
ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
```

```
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Weitere Informationen zur Arbeit mit DynamoDB finden Sie unter DynamoDB programmieren mit. AWS SDK für JavaScript JavaScript

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response.TableNames.join("\n"));
  return response;
};
```

- Einzelheiten zur API finden Sie unter ListTablesAPI-Referenz.AWS SDK für JavaScript

Python

SDK für Python

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
This is a test output. Examples for AWS SDK for Python
for AWS DynamoDB will be available here soon.
```

- Einzelheiten zur API finden Sie ListTablesunter AWS SDK for Python API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3

# Create a DynamoDB client using the default credentials and region
dynamodb = boto3.client("dynamodb")

# Initialize a paginator for the list_tables operation
paginator = dynamodb.get_paginator("list_tables")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(Limit=10)

# List the tables in the current AWS account
print("Here are the DynamoDB tables in your account:")

# Use pagination to list all tables
```

```
table_names = []

for page in page_iterator:
    for table_name in page.get("TableNames", []):
        print(f"- {table_name}")
        table_names.append(table_name)

if not table_names:
    print("You don't have any DynamoDB tables in your account.")
else:
    print(f"\nFound {len(table_names)} tables.")
```

- Einzelheiten zur API finden Sie ListTablesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-dynamodb'
require 'logger'

# DynamoDBManager is a class responsible for managing DynamoDB operations
# such as listing all tables in the current AWS account.
class DynamoDBManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all DynamoDB tables in the current AWS account.
  def list_tables
    @logger.info('Here are the DynamoDB tables in your account:')
```

```
    paginator = @client.list_tables(limit: 10)
    table_names = []

    paginator.each_page do |page|
      page.table_names.each do |table_name|
        @logger.info("- #{table_name}")
        table_names << table_name
      end
    end

    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  dynamodb_client = Aws::DynamoDB::Client.new
  manager = DynamoDBManager.new(dynamodb_client)
  manager.list_tables
end
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für Ruby API-Referenz.

Erlernen Sie die Grundlagen von DynamoDB mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// This example application performs the following basic Amazon DynamoDB
/// functions:
///     CreateTableAsync
///     PutItemAsync
///     UpdateItemAsync
///     BatchWriteItemAsync
///     GetItemAsync
///     DeleteItemAsync
///     Query
///     Scan
///     DeleteItemAsync.
/// </summary>
public class DynamoDbBasics
{
    public static bool IsInteractive = true;

    // Separator for the console display.
    private static readonly string SepBar = new string('-', 80);

    /// <summary>
    /// The main entry point for the DynamoDB Basics example application.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonDynamoDB>()
                    .AddTransient<DynamoDbWrapper>())
            .Build();

        // Now the wrapper is available for injection.
        var dynamoDbWrapper =
host.Services.GetRequiredService<DynamoDbWrapper>();

        var tableName = "movie_table";

        var movieFileName = @"movies.json";

        DisplayInstructions();

        // Create a new table and wait for it to be active.
        Console.WriteLine($"Creating the new table: {tableName}");

        var success = await dynamoDbWrapper.CreateMovieTableAsync(tableName);

        Console.WriteLine(success
            ? $"\nTable: {tableName} successfully created."
            : $"\nCould not create {tableName}.");

        WaitForEnter();

        // Add a single new movie to the table.
        var newMovie = new Movie
        {
            Year = 2021,
            Title = "Spider-Man: No Way Home",
        };

        success = await dynamoDbWrapper.PutItemAsync(newMovie, tableName);
        if (success)
        {
            Console.WriteLine($"Added {newMovie.Title} to the table.");
        }
        else
        {
            Console.WriteLine("Could not add movie to table.");
        }

        WaitForEnter();
```

```
        // Update the new movie by adding a plot and rank.
        var newInfo = new MovieInfo
        {
            Plot = "With Spider-Man's identity now revealed, Peter asks" +
                   "Doctor Strange for help. When a spell goes wrong, dangerous"
+
                   "foes from other worlds start to appear, forcing Peter to" +
                   "discover what it truly means to be Spider-Man.",
            Rank = 9,
        };

        success = await dynamoDbWrapper.UpdateItemAsync(newMovie, newInfo,
tableName);
        if (success)
        {
            Console.WriteLine($"Successfully updated the movie:
{newMovie.Title}");
        }
        else
        {
            Console.WriteLine("Could not update the movie.");
        }

        WaitForEnter();

        // Add a batch of movies to the DynamoDB table from a list of
        // movies in a JSON file.
        var itemCount = await dynamoDbWrapper.BatchWriteItemsAsync(movieFileName,
tableName);
        Console.WriteLine($"Added {itemCount} movies to the table.");

        WaitForEnter();

        // Get a movie by key. (partition + sort)
        var lookupMovie = new Movie
        {
            Title = "Jurassic Park",
            Year = 1993,
        };

        Console.WriteLine("Looking for the movie \"Jurassic Park\".");
        var item = await dynamoDbWrapper.GetItemAsync(lookupMovie, tableName);
        if (item?.Count > 0)
        {
```

```
            dynamoDbWrapper.DisplayItem(item);
        }
        else
        {
            Console.WriteLine($"Couldn't find {lookupMovie.Title}");
        }

        WaitForEnter();

        // Delete a movie.
        var movieToDelete = new Movie
        {
            Title = "The Town",
            Year = 2010,
        };

        success = await dynamoDbWrapper.DeleteItemAsync(tableName,
 movieToDelete);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {movieToDelete.Title}.");
        }

        WaitForEnter();

        // Use Query to find all the movies released in 2010.
        int findYear = 2010;
        Console.WriteLine($"Movies released in {findYear}");
        var queryCount = await dynamoDbWrapper.QueryMoviesAsync(tableName,
 findYear);
        Console.WriteLine($"Found {queryCount} movies released in {findYear}");

        WaitForEnter();

        // Use Scan to get a list of movies from 2001 to 2011.
        int startYear = 2001;
        int endYear = 2011;
        var scanCount = await dynamoDbWrapper.ScanTableAsync(tableName,
 startYear, endYear);
```

```
        Console.WriteLine($"Found {scanCount} movies released between {startYear}
and {endYear}");

        WaitForEnter();

        // Delete the table.
        success = await dynamoDbWrapper.DeleteTableAsync(tableName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {tableName}");
        }
        else
        {
            Console.WriteLine($"Could not delete {tableName}");
        }

        Console.WriteLine("The DynamoDB Basics example application is
complete.");

        WaitForEnter();
    }

    /// <summary>
    /// Displays the description of the application on the console.
    /// </summary>
    private static void DisplayInstructions()
    {
        if (!IsInteractive)
        {
            return;
        }

        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using
DynamoDB with the AWS SDK.");
        Console.WriteLine(SepBar);
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
sort:title.");
```

```
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that was
 just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition + sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released in
 a given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released
 within a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
 created.");
        WaitForEnter();
    }

    /// <summary>
    /// Simple method to wait for the Enter key to be pressed.
    /// </summary>
    private static void WaitForEnter()
    {
        if (IsInteractive)
        {
            Console.WriteLine("\nPress <Enter> to continue.");
            Console.WriteLine(SepBar);
            _ = Console.ReadLine();
        }
    }
}
```

Verwenden Sie den injizierten Client für Tabellenoperationen.

```
using System.Text.Json;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.DynamoDBv2.Model;

namespace DynamoDBActions;

/// <summary>
/// Methods of this class perform Amazon DynamoDB operations.
```

```
/// </summary>
public class DynamoDbWrapper
{
    private readonly IAmazonDynamoDB _amazonDynamoDB;

    /// <summary>
    /// Constructor for the DynamoDbWrapper class.
    /// </summary>
    /// <param name="amazonDynamoDB">The injected DynamoDB client.</param>
    public DynamoDbWrapper(IAmazonDynamoDB amazonDynamoDB)
    {
        _amazonDynamoDB = amazonDynamoDB;
    }
```

Erstellt eine Tabelle, die Filmdaten enthält.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
```

```
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
 _amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
```

```
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table
 {tableName}. {ex.Message}");
            throw;
        }
    }
```

Fügt der Tabelle einen einzelnen Film hinzu.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
```

```
            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
 {ex.Message}");
            throw;
        }
    }
```

Aktualisiert ein einzelnes Element in einer Tabelle.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
 param>
```

```
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
 {ex.Message}");
            return false;
```

```
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
 {ex.Message}");
            throw;
        }
    }
```

Ruft ein einzelnes Element aus der Filmtabelle ab.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</
param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
 newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
```

```
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
{ex.Message}");
            throw;
        }
    }
```

Schreibt einen Stapel von Elementen in die Filmtabelle.

```
    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The name of the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public List<Movie> ImportMovies(string movieFileName)
    {
        var moviesList = new List<Movie>();
        if (!File.Exists(movieFileName))
        {
            return moviesList;
        }
```

```
        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonSerializer.Deserialize<List<Movie>>(
            json,
            new JsonSerializerOptions
            {
                PropertyNameCaseInsensitive = true
            });

        // Now return the first 250 entries.
        if (allMovies != null && allMovies.Any())
        {
            moviesList = allMovies.GetRange(0, 250);
        }
        return moviesList;
    }

    /// <summary>
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie
data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of
IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();
```

```
            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch
write operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

Löscht ein einzelnes Element aus der Tabelle.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
```

```
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N =
movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

Fragt die Tabelle nach Filmen ab, die in einem bestimmten Jahr veröffentlicht wurden.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };

        // Value used to track how many movies match the
        // supplied criteria.
        var moviesFound = 0;

        var search = movieTable.Query(config);
        do
        {
```

```
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
 movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
 {ex.Message}");
            throw;
        }
    }
```

Scannt die Tabelle nach Filmen, die in einem bestimmten Zeitraum veröffentlicht wurden.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</
returns>
```

```
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string,
AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0],
info.directors, info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the
LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
```

```
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
 table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
 {ex.Message}");
            throw;
        }
    }
```

Löscht die Filmtabelle.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
 successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
```

```
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table
{tableName}. {ex.Message}");
            return false;
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das DynamoDB-Szenario „Erste Schritte“.

```
###############################################################################
# function dynamodb_getting_started_movies
#
# Scenario to create an Amazon DynamoDB table and perform a series of operations
 on the table.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function dynamodb_getting_started_movies() {

  source ./dynamodb_operations.sh

  key_schema_json_file="dynamodb_key_schema.json"
  attribute_definitions_json_file="dynamodb_attr_def.json"
  item_json_file="movie_item.json"
  key_json_file="movie_key.json"
  batch_json_file="batch.json"
  attribute_names_json_file="attribute_names.json"
  attributes_values_json_file="attribute_values.json"

  echo_repeat "*" 88
  echo
  echo "Welcome to the Amazon DynamoDB getting started demo."
  echo
  echo_repeat "*" 88
  echo

  local table_name
  echo -n "Enter a name for a new DynamoDB table: "
```

```
  get_input
  table_name=$get_input_result

  echo '[
  {"AttributeName": "year", "KeyType": "HASH"},
   {"AttributeName": "title", "KeyType": "RANGE"}
  ]' >"$key_schema_json_file"

  echo '[
  {"AttributeName": "year", "AttributeType": "N"},
   {"AttributeName": "title", "AttributeType": "S"}
  ]' >"$attribute_definitions_json_file"

  if dynamodb_create_table -n "$table_name" -a "$attribute_definitions_json_file"
 \
    -k "$key_schema_json_file" 1>/dev/null; then
    echo "Created a DynamoDB table named $table_name"
  else
    errecho "The table failed to create. This demo will exit."
    clean_up
    return 1
  fi

  echo "Waiting for the table to become active...."

  if dynamodb_wait_table_active -n "$table_name"; then
    echo "The table is now active."
  else
    errecho "The table failed to become active. This demo will exit."
    cleanup "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo -n "Enter the title of a movie you want to add to the table: "
  get_input
  local added_title
  added_title=$get_input_result

  local added_year
  get_int_input "What year was it released? "
```

```
  added_year=$get_input_result

  local rating
  get_float_input "On a scale of 1 - 10, how do you rate it? " "1" "10"
  rating=$get_input_result

  local plot
  echo -n "Summarize the plot for me: "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"},
    "info": {"M" : {"plot": {"S" : "'"$plot"'"}, "rating":
{"N" :"'"$rating"'"} } }
  }' >"$item_json_file"

  if dynamodb_put_item -n "$table_name" -i "$item_json_file"; then
    echo "The movie '$added_title' was successfully added to the table
'$table_name'."
  else
    errecho "Put item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "Let's update your movie '$added_title'."
  get_float_input "You rated it $rating, what new rating would you give it? " "1"
"10"
  rating=$get_input_result

  echo -n "You summarized the plot as '$plot'."
  echo "What would you say now? "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"}
```

```
    }' >"$key_json_file"

  echo '{
    ":r": {"N" :"'"$rating"'"},
    ":p": {"S" : "'"$plot"'"}
   }' >"$item_json_file"

  local update_expression="SET info.rating = :r, info.plot = :p"

  if dynamodb_update_item -n "$table_name" -k "$key_json_file" -e
"$update_expression" -v "$item_json_file"; then
    echo "Updated '$added_title' with new attributes."
  else
    errecho "Update item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "We will now use batch write to upload 150 movie entries into the table."

  local batch_json
  for batch_json in movie_files/movies_*.json; do
    echo "{ \"$table_name\" : $(<"$batch_json") }" >"$batch_json_file"
    if dynamodb_batch_write_item -i "$batch_json_file" 1>/dev/null; then
      echo "Entries in $batch_json added to table."
    else
      errecho "Batch write failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  done

  local title="The Lord of the Rings: The Fellowship of the Ring"
  local year="2001"

  if get_yes_no_input "Let's move on...do you want to get info about '$title'?
(y/n) "; then
    echo '{
  "year": {"N" :"'"$year"'"},
  "title": {"S" :  "'"$title"'"}
```

```
  }' >"$key_json_file"
    local info
    info=$(dynamodb_get_item -n "$table_name" -k "$key_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Get item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$info"
  fi

  local ask_for_year=true
  while [[ "$ask_for_year" == true ]]; do
    echo "Let's get a list of movies released in a given year."
    get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
    year=$get_input_result
    echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

    echo '{
    ":v": {"N" :"'"$year"'"}
    }' >"$attributes_values_json_file"

    response=$(dynamodb_query -n "$table_name" -k "#n=:v" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Query table failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$response"

    if ! get_yes_no_input "Try another year? (y/n) "; then
      ask_for_year=false
    fi
```

```
  done

  echo "Now let's scan for movies released in a range of years. Enter a year: "
  get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
  local start=$get_input_result

  get_int_input "Enter another year: " "1972" "2018"
  local end=$get_input_result

  echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

  echo '{
    ":v1": {"N" : "'"$start"'"},
    ":v2": {"N" : "'"$end"'"}
    }' >"$attributes_values_json_file"

  response=$(dynamodb_scan -n "$table_name" -f "#n BETWEEN :v1 AND :v2" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "Scan table failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo "Here is what I found:"
  echo "$response"

  echo
  echo_repeat "*" 88
  echo

  echo "Let's remove your movie '$added_title' from the table."

  if get_yes_no_input "Do you want to remove '$added_title'? (y/n) "; then
    echo '{
  "year": {"N" :"'"$added_year"'"},
  "title": {"S" :  "'"$added_title"'"}
  }' >"$key_json_file"

    if ! dynamodb_delete_item -n "$table_name" -k "$key_json_file"; then
```

```
      errecho "Delete item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  fi

  if get_yes_no_input "Do you want to delete the table '$table_name'? (y/n) ";
 then
    if ! clean_up "$table_name"; then
      return 1
    fi
  else
    if ! clean_up; then
      return 1
    fi
  fi

  return 0
}
```

Die in diesem Szenario verwendeten „DynamoDB“-Funktionen.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and
 their types.
#       -k key_schema -- JSON file path of a list of attributes and their key
 types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.
```

```
 ######################################
 # Function usage explanation
 ######################################
 function usage() {
   echo "function dynamodb_create_table"
   echo "Creates an Amazon DynamoDB table with on-demand billing."
   echo " -n table_name  -- The name of the table to create."
   echo " -a attribute_definitions -- JSON file path of a list of attributes and
their types."
   echo " -k key_schema -- JSON file path of a list of attributes and their key
types."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "n:a:k:h" option; do
   case "${option}" in
     n) table_name="${OPTARG}" ;;
     a) attribute_definitions="${OPTARG}" ;;
     k) key_schema="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$table_name" ]]; then
   errecho "ERROR: You must provide a table name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$attribute_definitions" ]]; then
   errecho "ERROR: You must provide an attribute definitions json file path the
-a parameter."
   usage
   return 1
```

```
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k
 parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}

###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  local table_status
    table_status=$(
      aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
```

```
        --query 'Table.TableStatus'
    )

   local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}

###############################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }
```

```
while getopts "n:i:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    i) item="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$item" ]]; then
  errecho "ERROR: You must provide an item with the -i parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    item:   $item"
iecho ""
iecho ""

response=$(aws dynamodb put-item \
  --table-name "$table_name" \
  --item file://"$item")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports put-item operation failed.$response"
```

```
    return 1
  fi

  return 0

}

##############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to update.
#       -e update expression  -- An expression that defines one or more
 attributes to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to update."
    echo " -e update expression  -- An expression that defines one or more
 attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }
```

```
while getopts "n:k:e:v:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    e) update_expression="${OPTARG}" ;;
    v) values="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi
if [[ -z "$update_expression" ]]; then
  errecho "ERROR: You must provide an update expression with the -e parameter."
  usage
  return 1
fi

if [[ -z "$values" ]]; then
  errecho "ERROR: You must provide a values json file path the -v parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
```

```
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
```

```
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}
```

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

if [[ -n "$query" ]]; then
  response=$(aws dynamodb get-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --output text \
    --query "$query")
else
  response=$(
    aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text
  )
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports get-item operation failed.$response"
  return 1
fi

if [[ -n "$query" ]]; then
```

```
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}

##############################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute
 names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
```

```
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
 parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
```

```
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names.
```

```
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
 parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
```

```
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to delete."
```

```
  echo ""
}
while getopts "n:k:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    keys:   $keys"
iecho ""

response=$(aws dynamodb delete-item \
  --table-name "$table_name" \
  --key file://"$keys")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Szenario verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}
```

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table with partition: year (N) and sort: title (S).
(CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

            // 9. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
```

```
        }
    }

//! Scenario to modify and query a DynamoDB table.
/*!
  \sa dynamodbGettingStartedScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
 std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
 it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,

 Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,

 Aws::DynamoDB::Model::AttributeValue().SetS(title));
```

```
        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
                putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    // 3. Update the rating and plot of the movie by using an update expression.
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);
        plot = askQuestion(Aws::String("You summarized the plot as '") + plot +
                           "'.\nWhat would you say now? ");

        Aws::DynamoDB::Model::UpdateItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
Aws::DynamoDB::Model::AttributeValue().SetS(title));
```

```
        request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
        std::stringstream expressionStream;
        expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
                         << INFO_KEY << "." << PLOT_KEY << " =:p";
        request.SetUpdateExpression(expressionStream.str());
        request.SetExpressionAttributeValues({
                                                    {":r",
Aws::DynamoDB::Model::AttributeValue().SetN(
                                                            rating)},
                                                    {":p",
Aws::DynamoDB::Model::AttributeValue().SetS(
                                                            plot)}
                                                });

        request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

        const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
                request);
        if (!result.IsSuccess()) {
            std::cerr << "Error updating movie " + result.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    // 4. Put 250 movies in the table from moviedata.json.
    {
        std::cout << "Adding movies from a json file to the database." <<
std::endl;
        const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
        const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
        Aws::String jsonString = getMovieJSON();
        if (!jsonString.empty()) {
            Aws::Utils::Json::JsonValue json(jsonString);
            Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
json.View().AsArray();
            Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

            // To add movies with a cross-section of years, use an appropriate
increment
```

```
            // value for iterating through the database.
            size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
            for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
                writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
putItems = movieJsonViewToAttributeMap(
                        movieJsons[i]);
                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(putItems);
                writeRequests.back().SetPutRequest(putRequest);
                if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                    Aws::DynamoDB::Model::BatchWriteItemRequest request;
                    request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
                    const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                            request);
                    if (!outcome.IsSuccess()) {
                        std::cerr << "Unable to batch write movie data: "
                                  << outcome.GetError().GetMessage()
                                  << std::endl;
                        writeRequests.clear();
                        break;
                    }
                    else {
                        std::cout << "Added batch of " << writeRequests.size()
                                  << " movies to the database."
                                  << std::endl;
                    }
                    writeRequests.clear();
                }
            }
        }
    }

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    // 5. Get a movie by Key (partition + sort).
    {
        Aws::String titleToGet("King Kong");
        Aws::String answer = askQuestion(Aws::String(
                "Let's move on...Would you like to get info about '" + titleToGet
+
                "'? (y/n) "));
```

```
        if (answer == "y") {
            Aws::DynamoDB::Model::GetItemRequest request;
            request.SetTableName(MOVIE_TABLE_NAME);
            request.AddKey(TITLE_KEY,

Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(1933));

            const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
                    request);
            if (!result.IsSuccess()) {
                std::cerr << "Error " << result.GetError().GetMessage();
            }
            else {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
                if (!item.empty()) {
                    std::cout << "\nHere's what I found:" << std::endl;
                    printMovieInfo(item);
                }
                else {
                    std::cout << "\nThe movie was not found in the database."
                              << std::endl;
                }
            }
        }
    }

    // 6. Use Query with a key condition expression to return all movies
    //    released in a given year.
    Aws::String doAgain = "n";
    do {
        Aws::DynamoDB::Model::QueryRequest req;

        req.SetTableName(MOVIE_TABLE_NAME);

        // "year" is a DynamoDB reserved keyword and must be replaced with an
        // expression attribute name.
        req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
        req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

        int yearToMatch = askQuestionForIntRange(
```

```
                "\nLet's get a list of movies released in"
                " a given year. Enter a year between 1972 and 2018 ",
                1972, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
        attributeValues.emplace(":valueToMatch",
                                Aws::DynamoDB::Model::AttributeValue().SetN(
                                        yearToMatch));
        req.SetExpressionAttributeValues(attributeValues);

        const Aws::DynamoDB::Model::QueryOutcome &result =
dynamoClient.Query(req);
        if (result.IsSuccess()) {
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "\nThere were " << items.size()
                          << " movies in the database from "
                          << yearToMatch << "." << std::endl;
                for (const auto &item: items) {
                    printMovieInfo(item);
                }
                doAgain = "n";
            }
            else {
                std::cout << "\nNo movies from " << yearToMatch
                          << " were found in the database"
                          << std::endl;
                doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
            }
        }
        else {
            std::cerr << "Failed to Query items: " <<
result.GetError().GetMessage()
                      << std::endl;
        }

    } while (doAgain == "y");

    //  7. Use Scan to return movies released within a range of years.
    //     Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
    {
```

```
        int startYear = askQuestionForIntRange("\nNow let's scan a range of years
"
                                               "for movies in the database. Enter
a start year: ",
                                               1972, 2018);
        int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                             startYear, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
        do {
            Aws::DynamoDB::Model::ScanRequest scanRequest;
            scanRequest.SetTableName(MOVIE_TABLE_NAME);
            scanRequest.SetFilterExpression(
                    "#dynobase_year >= :startYear AND #dynobase_year
<= :endYear");
            scanRequest.SetExpressionAttributeNames({{"#dynobase_year",
YEAR_KEY}});

            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
            attributeValues.emplace(":startYear",
                                    Aws::DynamoDB::Model::AttributeValue().SetN(
                                            startYear));
            attributeValues.emplace(":endYear",
                                    Aws::DynamoDB::Model::AttributeValue().SetN(
                                            endYear));
            scanRequest.SetExpressionAttributeValues(attributeValues);

            if (!exclusiveStartKey.empty()) {
                scanRequest.SetExclusiveStartKey(exclusiveStartKey);
            }

            const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
                    scanRequest);
            if (result.IsSuccess()) {
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
                if (!items.empty()) {
                    std::stringstream stringStream;
                    stringStream << "\nFound " << items.size() << " movies in one
scan."
                                 << " How many would you like to see? ";
                    size_t count = askQuestionForInt(stringStream.str());
                    for (size_t i = 0; i < count && i < items.size(); ++i) {
```

```
                        printMovieInfo(items[i]);
                    }
                }
                else {
                    std::cout << "\nNo movies in the database between " <<
startYear <<
                              " and " << endYear << "." << std::endl;
                }

                exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
                if (!exclusiveStartKey.empty()) {
                    std::cout << "Not all movies were retrieved. Scanning for
more."
                              << std::endl;
                }
                else {
                    std::cout << "All movies were retrieved with this scan."
                              << std::endl;
                }
            }
            else {
                std::cerr << "Failed to Scan movies: "
                          << result.GetError().GetMessage() << std::endl;
            }
        } while (!exclusiveStartKey.empty());
    }

    // 8. Delete a movie. (DeleteItem)
    {
        std::stringstream stringStream;
        stringStream << "\nWould you like to delete the movie " << title
                     << " from the database? (y/n) ";
        Aws::String answer = askQuestion(stringStream.str());
        if (answer == "y") {
            Aws::DynamoDB::Model::DeleteItemRequest request;
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(title));
            request.SetTableName(MOVIE_TABLE_NAME);

            const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(
                    request);
```

```
            if (result.IsSuccess()) {
                std::cout << "\nRemoved \"" << title << "\" from the database."
                          << std::endl;
            }
            else {
                std::cerr << "Failed to delete the movie: "
                          << result.GetError().GetMessage()
                          << std::endl;
            }
        }
    }

    return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
  \sa movieJsonViewToAttributeMap()
  \param jsonView: Json view object.
  \return map: Map that can be used in a DynamoDB request.
 */
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
        const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
 std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue
 = std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
```

```
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue
 = std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }

        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);
```

```
        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
```

```
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
```

```
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunMovieScenario is an interactive example that shows you how to use the AWS
 SDK for Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
//  1. Create a table that can hold movie data.
//  2. Put, get, and update a single movie in the table.
//  3. Write movie data to the table from a sample JSON file.
//  4. Query for movies that were released in a given year.
//  5. Scan for movies that were released in a range of years.
//  6. Delete a movie from the table.
//  7. Delete the table.
//
```

```
// This example creates a DynamoDB service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is
 loaded
// into the named table.
func RunMovieScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner,
 tableName string,
 movieSampler actions.IMovieSampler) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB getting started demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{TableName: tableName,
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
 } else {
  log.Printf("Table %v already exists.\n", tableName)
 }
```

```
var customMovie actions.Movie
customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
 demotools.NotEmpty{})
customMovie.Year = questioner.AskInt("What year was it released?",
 demotools.NotEmpty{}, demotools.InIntRange{Lower: 1900, Upper: 2030})
customMovie.Info = map[string]interface{}{}
customMovie.Info["rating"] = questioner.AskFloat64(
 "Enter a rating between 1 and 10:",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
 demotools.NotEmpty{})
err = tableBasics.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
 "What new rating would you give it?",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
 demotools.NotEmpty{})
attributes, err := tableBasics.UpdateMovie(ctx, customMovie)
if err == nil {
 log.Printf("Updated %v with new values.\n", customMovie.Title)
 for _, attVal := range attributes {
  for valKey, val := range attVal {
   log.Printf("\t%v: %v\n", valKey, val)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",
 movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(ctx, movies, 250)
if err != nil {
 panic(err)
} else {
 log.Printf("Added %v movies to the table.\n", written)
```

```
}

show := 10
if show > written {
 show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
 log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
 "Enter the number of a movie to get info about it: ",
 demotools.InIntRange{Lower: 1, Upper: show},
)
movie, err := tableBasics.GetMovie(ctx, movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018},
)
releases, err := tableBasics.Query(ctx, releaseYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
 } else {
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
endYear := questioner.AskInt("Enter another year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
releases, err = tableBasics.Scan(ctx, startYear, endYear)
if err == nil {
```

```
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released between %v and %v!\n",
startYear, endYear)
 } else {
  log.Printf("Found %v movies. In this list, the plot is <nil> because "+
   "we used a projection expression when scanning for items to return only "+
   "the title, year, and rating.\n", len(releases))
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

var tables []string
if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
 tables, err = tableBasics.ListTables(ctx)
 if err == nil {
  log.Printf("Found %v tables:", len(tables))
  for _, table := range tables {
   log.Printf("\t%v", table)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
 err = tableBasics.DeleteMovie(ctx, customMovie)
}
if err == nil {
 log.Printf("Deleted %v.\n", customMovie.Title)
}

if questioner.AskBool("Delete the table, too? (y/n)", "y") {
 err = tableBasics.DeleteTable(ctx)
} else {
 log.Println("Don't forget to delete the table when you're done or you might " +
  "incur charges on your account.")
}
if err == nil {
 log.Printf("Deleted table %v.\n", tableBasics.TableName)
}
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
```

```
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die DynamoDB-Aktionen aufrufen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
```

```
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined
 as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
```

```
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
```

```
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}



// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the
 update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx,
 &dynamodb.UpdateItemInput{
   TableName:                 aws.String(basics.TableName),
   Key:                       movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   UpdateExpression:          expr.Update(),
   ReturnValues:              types.ReturnValueUpdatedNew,
  })
  if err != nil {
   log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
  } else {
   err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
   if err != nil {
    log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
```

```
    }
   }
  }
  return attributeMap, err
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx,
 &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
```

```
  }
  start = end
  end += batchSize
 }

 return written, err
}



// GetMovie gets movie data from the DynamoDB table by using the primary
 composite key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// Query gets all movies in the DynamoDB table that were released in the
 specified year.
// The function uses the `expression` package to build the key condition
 expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
```

```
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// Scan gets all movies in the DynamoDB table that were released in a range of
 years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
```

```
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"),
 expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:         expr.Filter(),
   ProjectionExpression:     expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why:
 %v\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
```

```
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);
```

```
        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }
```

Rufen Sie ein Element aus einer Tabelle ab.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();
```

```
        try {
            Map<String, AttributeValue> returnedItem =
 ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Vollständiges Beispiel.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
```

```
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named
 year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB
 table.");
        putRecord(ddb);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();
```

```
        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();
```

```
            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                    .partitionValue(2013)
                    .build());

            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " +
rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
 fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
```

```
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper
\",\"Ernest B. Schoedsack\"]")
                .build())
            .action(AttributeAction.PUT)
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
```

```
            Map<String, AttributeValue> returnedItem =
 ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { readFileSync } from "node:fs";
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";

/**
 * This module is a convenience library. It abstracts Amazon DynamoDB's data type
 * descriptors (such as S, N, B, and BOOL) by marshalling JavaScript objects into
 * AttributeValue shapes.
 */
import {
  BatchWriteCommand,
  DeleteCommand,
  DynamoDBDocumentClient,
  GetCommand,
  PutCommand,
  UpdateCommand,
  paginateQuery,
  paginateScan,
} from "@aws-sdk/lib-dynamodb";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";
```

```
const dirname = dirnameFromMetaUrl(import.meta.url);
const tableName = getUniqueName("Movies");
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);

export const main = async () => {
  /**
   * Create a table.
   */

  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "year",
        // 'N' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "N",
      },
      { AttributeName: "title", AttributeType: "S" },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [
      // The way your data is accessed determines how you structure your keys.
      // The movies table will be queried for movies by year. It makes sense
      // to make year our partition (HASH) key.
      { AttributeName: "year", KeyType: "HASH" },
      { AttributeName: "title", KeyType: "RANGE" },
    ],
  });
```

```
  log("Creating a table.");
  const createTableResponse = await client.send(createTableCommand);
  log(`Table created: ${JSON.stringify(createTableResponse.TableDescription)}`);

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Add a movie to the table.
   */

  log("Adding a single movie to the table.");
  // PutCommand is the first example usage of 'lib-dynamodb'.
  const putCommand = new PutCommand({
    TableName: tableName,
    Item: {
      // In 'client-dynamodb', the AttributeValue would be required (`year: { N:
1981 }`)
      // 'lib-dynamodb' simplifies the usage ( `year: 1981` )
      year: 1981,
      // The preceding KeySchema defines 'title' as our sort (RANGE) key, so
'title'
      // is required.
      title: "The Evil Dead",
      // Every other attribute is optional.
      info: {
        genres: ["Horror"],
      },
    },
  });
  await docClient.send(putCommand);
  log("The movie was added.");

  /**
   * Get a movie from the table.
   */

  log("Getting a single movie from the table.");
  const getCommand = new GetCommand({
    TableName: tableName,
    // Requires the complete primary key. For the movies table, the primary key
```

```
    // is only the id (partition key).
    Key: {
      year: 1981,
      title: "The Evil Dead",
    },
    // Set this to make sure that recent writes are reflected.
    // For more information, see https://docs.aws.amazon.com/amazondynamodb/
latest/developerguide/HowItWorks.ReadConsistency.html.
    ConsistentRead: true,
  });
  const getResponse = await docClient.send(getCommand);
  log(`Got the movie: ${JSON.stringify(getResponse.Item)}`);

  /**
   * Update a movie in the table.
   */

  log("Updating a single movie in the table.");
  const updateCommand = new UpdateCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
    // This update expression appends "Comedy" to the list of genres.
    // For more information on update expressions, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.UpdateExpressions.html
    UpdateExpression: "set #i.#g = list_append(#i.#g, :vals)",
    ExpressionAttributeNames: { "#i": "info", "#g": "genres" },
    ExpressionAttributeValues: {
      ":vals": ["Comedy"],
    },
    ReturnValues: "ALL_NEW",
  });
  const updateResponse = await docClient.send(updateCommand);
  log(`Movie updated: ${JSON.stringify(updateResponse.Attributes)}`);

  /**
   * Delete a movie from the table.
   */

  log("Deleting a single movie from the table.");
  const deleteCommand = new DeleteCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
  });
```

```
  await docClient.send(deleteCommand);
  log("Movie deleted.");

  /**
   * Upload a batch of movies.
   */

  log("Adding movies from local JSON file.");
  const file = readFileSync(
    `${dirname}../../../../resources/sample_files/movies.json`,
  );
  const movies = JSON.parse(file.toString());
  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);
  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        [tableName]: putRequests,
      },
    });

    await docClient.send(command);
  }
  log("Movies added.");

  /**
   * Query for movies by year.
   */

  log("Querying for all movies from 1981.");
  const paginatedQuery = paginateQuery(
    { client: docClient },
    {
      TableName: tableName,
      //For more information about query expressions, see
```

```
      // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Query.html#Query.KeyConditionExpressions
      KeyConditionExpression: "#y = :y",
      // 'year' is a reserved word in DynamoDB. Indicate that it's an attribute
      // name by using an expression attribute name.
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y": 1981 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1981 = [];
  for await (const page of paginatedQuery) {
    movies1981.push(...page.Items);
  }
  log(`Movies: ${movies1981.map((m) => m.title).join(", ")}`);

  /**
   * Scan the table for movies between 1980 and 1990.
   */

  log("Scan for movies released between 1980 and 1990");
  // A 'Scan' operation always reads every item in the table. If your design
 requires
  // the use of 'Scan', consider indexing your table or changing your design.
  // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-
scan.html
  const paginatedScan = paginateScan(
    { client: docClient },
    {
      TableName: tableName,
      // Scan uses a filter expression instead of a key condition expression.
 Scan will
      // read the entire table and then apply the filter.
      FilterExpression: "#y between :y1 and :y2",
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y1": 1980, ":y2": 1990 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
```

```
   */
  const movies1980to1990 = [];
  for await (const page of paginatedScan) {
    movies1980to1990.push(...page.Items);
  }
  log(
    `Movies: ${movies1980to1990
      .map((m) => `${m.title} (${m.year})`)
      .join(", ")}`,
  );

  /**
   * Delete the table.
   */

  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  log(`Deleting table ${tableName}.`);
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
```

```
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
```

```
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Rufen Sie ein Element aus einer Tabelle ab.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
```

```
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Vollständiges Beispiel.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id
 and a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
```

```
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
```

```
    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
```

```
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
```

```
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable

- GetItem
- PutItem
- Query
- Scan
- UpdateItem

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Da in diesem Beispiel Hilfsdateien verwendet werden, sollten Sie unbedingt die Anleitungen in der README.md PHP-Beispieldatei lesen.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!
\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
```

```
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
```

```
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
 {$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?
\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
```

```
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

        $movie = $service->getItemByKey($tableName, $key);
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a
{$movie['Item']['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
```

```
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die eine DynamoDB-Tabelle enthält.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
```

```
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists
```

```
    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition
key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
```

```
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts
sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
least
                       the keys required by the schema that was specified when
the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p":
plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
```

```
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
```

```
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies


    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
def get_sample_movie_data(movie_file_name):
    """
    Gets sample movie data, either from a local file or by first downloading it
 from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in
 JSON format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(
            "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip"
        )
        movie_zip = ZipFile(BytesIO(movie_content.content))
        movie_zip.extractall()

    try:
        with open(movie_file_name) as movie_file:
            movie_data = json.load(movie_file, parse_float=Decimal)
    except FileNotFoundError:
        print(
            f"File {movie_file_name} not found. You must first download the file
 to "
            "run this demo. See the README for instructions."
        )
        raise
    else:
        # The sample file lists over 4000 movies, return only the first 250.
        return movie_data[:250]
```

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print("-" * 88)

    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")

    my_movie = Question.ask_questions(
        [
            Question(
                "title", "Enter the title of a movie you want to add to the
 table: "
            ),
            Question("year", "What year was it released? ", Question.is_int),
            Question(
                "rating",
                "On a scale of 1 - 10, how do you rate it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question("plot", "Summarize the plot for me: "),
        ]
    )
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'.")
    print("-" * 88)

    movie_update = Question.ask_questions(
        [
            Question(
                "rating",
                f"\nLet's update your movie.\nYou rated it {my_movie['rating']},
 what new "
```

```
                f"rating would you give it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question(
                "plot",
                f"You summarized the plot as '{my_movie['plot']}'.\nWhat would
you say now? ",
            ),
        ]
    )
    my_movie.update(movie_update)
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print("-" * 88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
        movies.write_batch(movie_data)
        print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
    print("-" * 88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
    if Question.ask_question(
        f"Let's move on...do you want to get info about '{title}'? (y/n) ",
        Question.is_yesno,
    ):
        movie = movies.get_movie(title, 2001)
        print("\nHere's what I found:")
        pprint(movie)
    print("-" * 88)

    ask_for_year = True
    while ask_for_year:
        release_year = Question.ask_question(
            f"\nLet's get a list of movies released in a given year. Enter a year
between "
            f"1972 and 2018: ",
            Question.is_int,
            Question.in_range(1972, 2018),
        )
        releases = movies.query_movies(release_year)
```

```
        if releases:
            print(f"There were {len(releases)} movies released in
{release_year}:")
            for release in releases:
                print(f"\t{release['title']}")
            ask_for_year = False
        else:
            print(f"I don't know about any movies released in {release_year}!")
            ask_for_year = Question.ask_question(
                "Try another year? (y/n) ", Question.is_yesno
            )
    print("-" * 88)

    years = Question.ask_questions(
        [
            Question(
                "first",
                f"\nNow let's scan for movies released in a range of years. Enter
a year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
            Question(
                "second",
                "Now enter another year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
        ]
    )
    releases = movies.scan_movies(years)
    if releases:
        count = Question.ask_question(
            f"\nFound {len(releases)} movies. How many do you want to see? ",
            Question.is_int,
            Question.in_range(1, len(releases)),
        )
        print(f"\nHere are your {count} movies:\n")
        pprint(releases[:count])
    else:
        print(
            f"I don't know about any movies released between {years['first']} "
            f"and {years['second']}."
        )
```

```
    print("-" * 88)

    if Question.ask_question(
        f"\nLet's remove your movie from the table. Do you want to remove "
        f"'{my_movie['title']}'? (y/n)",
        Question.is_yesno,
    ):
        movies.delete_movie(my_movie["title"], my_movie["year"])
        print(f"\nRemoved '{my_movie['title']}' from the table.")
    print("-" * 88)

    if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
        movies.delete_table()
        print(f"Deleted {table_name}.")
    else:
        print(
            "Don't forget to delete the table when you're done or you might incur
 "
            "charges on your account."
        )

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            "doc-example-table-movies", "moviedata.json",
 boto3.resource("dynamodb")
        )
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

In diesem Szenario wird die folgende Helper-Klasse verwendet, um Fragen an einer Eingabeaufforderung zu stellen.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """
```

```
    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
                    multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators
until
                           one fails or they all pass. Validators may also
convert the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators
            )
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the
question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
```

```
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
                if answer is None:
                    print(complaint)
                    break
        return answer

    @staticmethod
    def non_empty(answer):
        """
        Validates that the answer is not empty.
        :return: The non-empty answer, or None.
        """
        return answer if answer != "" else None, "I need an answer. Please?"

    @staticmethod
    def is_yesno(answer):
        """
        Validates a yes/no answer.
        :return: True when the answer is 'y'; otherwise, False.
        """
        return answer.lower() == "y", ""

    @staticmethod
    def is_int(answer):
        """
        Validates that the answer can be converted to an int.
        :return: The int answer; otherwise, None.
        """
        try:
            int_answer = int(answer)
        except ValueError:
            int_answer = None
        return int_answer, f"{answer} must be a valid integer."

    @staticmethod
    def is_letter(answer):
        """
        Validates that the answer is a letter.
        :return The letter answer, converted to uppercase; otherwise, None.
        """
        return (
```

```
            answer.upper() if answer.isalpha() else None,
            f"{answer} must be a single letter.",
        )

    @staticmethod
    def is_float(answer):
        """
        Validate that the answer can be converted to a float.
        :return The float answer; otherwise, None.
        """
        try:
            float_answer = float(answer)
        except ValueError:
            float_answer = None
        return float_answer, f"{answer} must be a valid float."

    @staticmethod
    def in_range(lower, upper):
        """
        Validate that the answer is within a range. The answer must be of a type
that can
        be compared to the lower and upper bounds.
        :return: The answer, if it is within the range; otherwise, None.
        """

        def _validate(answer):
            return (
                answer if lower <= answer <= upper else None,
                f"{answer} must be between {lower} and {upper}.",
            )

        return _validate
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable

- DescribeTable
- GetItem
- PutItem
- Query
- Scan
- UpdateItem

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die eine DynamoDB-Tabelle enthält.

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
  @table = @dynamo_resource.create_table(
    table_name: table_name,
    key_schema: [
      { attribute_name: 'year', key_type: 'HASH' }, # Partition key
      { attribute_name: 'title', key_type: 'RANGE' } # Sort key
    ],
    attribute_definitions: [
      { attribute_name: 'year', attribute_type: 'N' },
      { attribute_name: 'title', attribute_type: 'S' }
    ],
    billing_mode: 'PAY_PER_REQUEST'
  )
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
```

```
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
  # Gets sample movie data, either from a local file or by first downloading it
 from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip'
      )
      movie_json = ''
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
 table_name = "doc-example-table-movies-#{rand(10**4)}"
 scaffold = Scaffold.new(table_name)
 dynamodb_wrapper = DynamoDBBasics.new(table_name)

 new_step(1, 'Create a new DynamoDB table if none already exists.')
 unless scaffold.exists?(table_name)
   puts("\nNo such table: #{table_name}. Creating it...")
   scaffold.create_table(table_name)
   print "Done!\n".green
 end

 new_step(2, 'Add a new record to the DynamoDB table.')
 my_movie = {}
 my_movie[:title] = CLI::UI::Prompt.ask('Enter the title of a movie to add to
the table. E.g. The Matrix')
 my_movie[:year] = CLI::UI::Prompt.ask('What year was it released? E.g.
1989').to_i
 my_movie[:rating] = CLI::UI::Prompt.ask('On a scale of 1 - 10, how do you rate
it? E.g. 7').to_i
 my_movie[:plot] = CLI::UI::Prompt.ask('Enter a brief summary of the plot. E.g.
A man awakens to a new reality.')
 dynamodb_wrapper.add_item(my_movie)
 puts("\nNew record added:")
 puts JSON.pretty_generate(my_movie).green
 print "Done!\n".green

 new_step(3, 'Update a record in the DynamoDB table.')
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with
a new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, 'Get a record from the DynamoDB table.')
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green
```

```
 new_step(5, 'Write a batch of items into the DynamoDB table.')
 download_file = 'moviedata.json'
 puts("Downloading movie database to #{download_file}...")
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
 print "Done!\n".green

 new_step(5, 'Query for a batch of items by key.')
 loop do
   release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie['title']}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in
#{release_year}! Try another year? (y/n)")
     break unless continue.eql?('y')
   end
 end
 print "\nDone!\n".green

 new_step(6, 'Scan for a batch of items using a filter expression.')
 years = {}
 years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
2018:')
 years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and
2018:')
 releases = dynamodb_wrapper.scan_items(years)
 if !releases.empty?
   puts("Found #{releases.length} movies.")
   count = Question.ask(
     'How many do you want to see? ', method(:is_int), in_range(1,
releases.length)
   )
   puts("Here are your #{count} movies:")
   releases.take(count).each do |release|
     puts("\t#{release['title']}")
```

```
    end
  else
    puts("I don't know about any movies released between #{years[:start]} "\
         "and #{years[:end]}.")
  end
  print "\nDone!\n".green

  new_step(7, 'Delete an item from the DynamoDB table.')
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/
n) ")
  if answer.eql?('y')
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, 'Delete the DynamoDB table.')
  answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
  if answer.eql?('y')
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
  true
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create an Amazon Dynamo DB table.

    TRY.
        DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
        DATA(lo_dyn) = /aws1/cl_dyn_factory=>create( lo_session ).
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        DATA(oo_result) = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
```

```
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " It throws exception if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

    " Describe table
    TRY.
        DATA(lo_table) = lo_dyn->describetable( iv_tablename = iv_table_name ).
        DATA(lv_tablename) = lo_table->get_table( )->ask_tablename( ).
        MESSAGE 'The table name is ' && lv_tablename TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table does not exist' TYPE 'E'.
    ENDTRY.

    " Put items into the table.
    TRY.
        DATA(lo_resp_putitem) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1975' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.5' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
```

```
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s = 'Star
 Wars' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1978' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '8.1' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Speed' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1994' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.9' }| ) ) )
          ) ).
        " TYPE REF TO /AWSEX/CL_AWS1_dyn_PUT_ITEM_OUTPUT
        MESSAGE '3 rows inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Get item from table.
    TRY.
        DATA(lo_resp_getitem) = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = VALUE /aws1/cl_dynattributevalue=>tt_key(
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
```

```
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ) ).
        DATA(lt_attr) = lo_resp_getitem->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'year' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Query item from table.
    TRY.
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
              ( NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) ).
        DATA(lt_keyconditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
            it_attributevaluelist = lt_attributelist
            iv_comparisonoperator = |EQ|
          ) ) ) ).
        DATA(lo_query_result) = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_keyconditions ).
        DATA(lt_items) = lo_query_result->get_items( ).
        READ TABLE lo_query_result->get_items( ) INTO DATA(lt_item) INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Scan items from table.
    TRY.
        DATA(lo_scan_result) = lo_dyn->scan( iv_tablename = iv_table_name ).
```

```
        lt_items = lo_scan_result->get_items( ).
        " Read the first item and display the attributes.
        READ TABLE lo_query_result->get_items( ) INTO lt_item INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Update items from table.
    TRY.
        DATA(lt_attributeupdates) = VALUE /aws1/
cl_dynattrvalueupdate=>tt_attributeupdates(
          ( VALUE /aws1/cl_dynattrvalueupdate=>ts_attributeupdates_maprow(
          key = 'rating' value = NEW /aws1/cl_dynattrvalueupdate(
            io_value  = NEW /aws1/cl_dynattributevalue( iv_n = '7.6' )
            iv_action = |PUT| ) ) ) ).
        DATA(lt_key) = VALUE /aws1/cl_dynattributevalue=>tt_key(
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 '1980' ) ) ) ).
        DATA(lo_resp) = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = lt_key
          it_attributeupdates = lt_attributeupdates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Delete table.
    TRY.
```

```
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted as it is in use' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Swift-Klasse, die DynamoDB-Aufrufe an das SDK für Swift verarbeitet

```
import AWSDynamoDB
import Foundation

/// An enumeration of error codes representing issues that can arise when using
/// the `MovieTable` class.
enum MoviesError: Error {
    /// The specified table wasn't found or couldn't be created.
    case TableNotFound
    /// The specified item wasn't found or couldn't be created.
    case ItemNotFound
    /// The Amazon DynamoDB client is not properly initialized.
    case UninitializedClient
    /// The table status reported by Amazon DynamoDB is not recognized.
    case StatusUnknown
    /// One or more specified attribute values are invalid or missing.
    case InvalidAttributes
}

/// A class representing an Amazon DynamoDB table containing movie
/// information.
public class MovieTable {
    var ddbClient: DynamoDBClient?
    let tableName: String

    /// Create an object representing a movie table in an Amazon DynamoDB
    /// database.
    ///
    /// - Parameters:
    ///   - region: The optional Amazon Region to create the database in.
    ///   - tableName: The name to assign to the table. If not specified, a
    ///     random table name is generated automatically.
    ///
    /// > Note: The table is not necessarily available when this function
    /// returns. Use `tableExists()` to check for its availability, or
    /// `awaitTableActive()` to wait until the table's status is reported as
    /// ready to use by Amazon DynamoDB.
    ///
    init(region: String? = nil, tableName: String) async throws {
        do {
            let config = try await DynamoDBClient.DynamoDBClientConfiguration()
            if let region = region {
                config.region = region
            }
```

```
            self.ddbClient = DynamoDBClient(config: config)
            self.tableName = tableName

            try await self.createTable()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing Amazon
DynamoDBClient client"))
            throw error
        }
    }

    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"year", attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
"title", attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
```

```
        }
    }


    /// Check to see if the table exists online yet.
    ///
    /// - Returns: `true` if the table exists, or `false` if not.
    ///
    func tableExists() async throws -> Bool {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }

            return description.tableName == self.tableName
        } catch {
            print("ERROR: tableExists:", dump(error))
            throw error
        }
    }


    ///
    /// Waits for the table to exist and for its status to be active.
    ///
    func awaitTableActive() async throws {
        while try (await self.tableExists() == false) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25
seconds to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }
```

```
        while try (await self.getTableStatus() != .active) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25
seconds to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }
    }


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
        }
    }


    /// Get the table's status.
    ///
    /// - Returns: The table status, as defined by the
    ///   `DynamoDBClientTypes.TableStatus` enum.
    ///
    func getTableStatus() async throws -> DynamoDBClientTypes.TableStatus {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
```

```
                tableName: self.tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }
            guard let status = description.tableStatus else {
                throw MoviesError.StatusUnknown
            }
            return status
        } catch {
            print("ERROR: getTableStatus:", dump(error))
            throw error
        }
    }


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from:
jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
```

```
            // 25-entry chunks, which is the maximum size of a batch item
request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and
populate
            // them with `PutRequest` requests, each specifying one movie from
the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
```

```
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    /// Given a movie's details, add a movie to the Amazon DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title as a `String`.
    ///   - year: The release year of the movie (`Int`).
    ///   - rating: The movie's rating if available (`Double`; default is
    ///     `nil`).
    ///   - plot: A summary of the movie's plot (`String`; default is `nil`,
    ///     indicating no plot summary is available).
    ///
    func add(title: String, year: Int, rating: Double? = nil,
             plot: String? = nil) async throws
    {
        do {
            let movie = Movie(title: title, year: year, rating: rating, plot:
plot)
            try await self.add(movie: movie)
        } catch {
            print("ERROR: add with fields:", dump(error))
            throw error
```

```
        }
    }


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
```

```
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
"QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an
array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
```

```
            throw error
        }
    }


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]?
= nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y`
instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
```

```
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
 nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
```

```
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] =
[:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the
release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String:
DynamoDBClientTypes.AttributeValue] = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
```

```
            throw error
        }
    }


    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
}
```

Die Strukturen, die von der MovieTable Klasse verwendet werden, um Filme darzustellen.

```
import Foundation
import AWSDynamoDB

/// The optional details about a movie.
public struct Details: Codable {
    /// The movie's rating, if available.
    var rating: Double?
```

```
    /// The movie's plot, if available.
    var plot: String?
}

/// A structure describing a movie. The `year` and `title` properties are
/// required and are used as the key for Amazon DynamoDB operations. The
/// `info` sub-structure's two properties, `rating` and `plot`, are optional.
public struct Movie: Codable {
    /// The year in which the movie was released.
    var year: Int
    /// The movie's title.
    var title: String
    /// A `Details` object providing the optional movie rating and plot
    /// information.
    var info: Details

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - rating: The movie's rating (optional `Double`).
    ///   - plot: The movie's plot (optional `String`)
    init(title: String, year: Int, rating: Double? = nil, plot: String? = nil) {
        self.title = title
        self.year = year

        self.info = Details(rating: rating, plot: plot)
    }

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - info: The optional rating and plot information for the movie in a
    ///     `Details` object.
    init(title: String, year: Int, info: Details?){
        self.title = title
        self.year = year

        if info != nil {
```

```
            self.info = info!
        } else {
            self.info = Details(rating: nil, plot: nil)
        }
    }

    ///
    /// Return a new `MovieTable` object, given an array mapping string to Amazon
    /// DynamoDB attribute values.
    ///
    /// - Parameter item: The item information provided to the form used by
    ///   DynamoDB. This is an array of strings mapped to
    ///   `DynamoDBClientTypes.AttributeValue` values.
    init(withItem item: [Swift.String:DynamoDBClientTypes.AttributeValue]) throws
  {
        // Read the attributes.

        guard let titleAttr = item["title"],
              let yearAttr = item["year"] else {
            throw MoviesError.ItemNotFound
        }
        let infoAttr = item["info"] ?? nil

        // Extract the values of the title and year attributes.

        if case .s(let titleVal) = titleAttr {
            self.title = titleVal
        } else {
            throw MoviesError.InvalidAttributes
        }

        if case .n(let yearVal) = yearAttr {
            self.year = Int(yearVal)!
        } else {
            throw MoviesError.InvalidAttributes
        }

        // Extract the rating and/or plot from the `info` attribute, if
        // they're present.

        var rating: Double? = nil
        var plot: String? = nil

        if infoAttr != nil, case .m(let infoVal) = infoAttr {
```

```
            let ratingAttr = infoVal["rating"] ?? nil
            let plotAttr = infoVal["plot"] ?? nil

            if ratingAttr != nil, case .n(let ratingVal) = ratingAttr {
                rating = Double(ratingVal) ?? nil
            }
            if plotAttr != nil, case .s(let plotVal) = plotAttr {
                plot = plotVal
            }
        }

        self.info = Details(rating: rating, plot: plot)
    }

    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
```

```
        item["info"] = .m(details)

        return item
    }
 }
```

Ein Programm, das die MovieTable Klasse verwendet, um auf eine DynamoDB-Datenbank zuzugreifen.

```
import ArgumentParser
import ClientRuntime
import Foundation

import AWSDynamoDB


@testable import MovieList

extension String {
    // Get the directory if the string is a file path.
    func directory() -> String {
        guard let lastIndex = lastIndex(of: "/") else {
            print("Error: String directory separator not found.")
            return ""
        }
        return String(self[...lastIndex])
    }
}

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The path of the sample movie data JSON file.")
    var jsonPath: String = #file.directory() + "../../../../../resources/
sample_files/movies.json"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion: String?

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
```

```
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    /// Configuration details for the command.
    static var configuration = CommandConfiguration(
        commandName: "basics",
        abstract: "A basic scenario demonstrating the usage of Amazon DynamoDB.",
        discussion: """
        An example showing how to use Amazon DynamoDB to perform a series of
        common database activities on a simple movie database.
        """
    )

    /// Called by ``main()`` to asynchronously run the AWS example.
    func runAsync() async throws {
        print("Welcome to the AWS SDK for Swift basic scenario for Amazon
DynamoDB!")

        //======================================================================
        // 1. Create the table. The Amazon DynamoDB table is represented by
        //    the `MovieTable` class.
        //======================================================================

        let tableName = "ddb-movies-sample-\(Int.random(in: 1 ... Int.max))"

        print("Creating table \"\(tableName)\"...")

        let movieDatabase = try await MovieTable(region: awsRegion,
                                                 tableName: tableName)

        print("\nWaiting for table to be ready to use...")
        try await movieDatabase.awaitTableActive()

        //======================================================================
        // 2. Add a movie to the table.
        //======================================================================

        print("\nAdding a movie...")
```

```
        try await movieDatabase.add(title: "Avatar: The Way of Water", year:
2022)
        try await movieDatabase.add(title: "Not a Real Movie", year: 2023)

        //=====================================================================
        // 3. Update the plot and rating of the movie using an update
        //    expression.
        //=====================================================================

        print("\nAdding details to the added movie...")
        _ = try await movieDatabase.update(title: "Avatar: The Way of Water",
year: 2022,
                                           rating: 9.2, plot: "It's a sequel.")

        //=====================================================================
        // 4. Populate the table from the JSON file.
        //=====================================================================

        print("\nPopulating the movie database from JSON...")
        try await movieDatabase.populate(jsonPath: jsonPath)

        //=====================================================================
        // 5. Get a specific movie by key. In this example, the key is a
        //    combination of `title` and `year`.
        //=====================================================================

        print("\nLooking for a movie in the table...")
        let gotMovie = try await movieDatabase.get(title: "This Is the End",
year: 2013)

        print("Found the movie \"\(gotMovie.title)\", released in
\(gotMovie.year).")
        print("Rating: \(gotMovie.info.rating ?? 0.0).")
        print("Plot summary: \(gotMovie.info.plot ?? "None.")")

        //=====================================================================
        // 6. Delete a movie.
        //=====================================================================

        print("\nDeleting the added movie...")
        try await movieDatabase.delete(title: "Avatar: The Way of Water", year:
2022)

        //=====================================================================
```

```
        // 7. Use a query with a key condition expression to return all movies
        //    released in a given year.
        //===========================================================================

        print("\nGetting movies released in 1994...")
        let movieList = try await movieDatabase.getMovies(fromYear: 1994)
        for movie in movieList {
            print("    \(movie.title)")
        }

        //===========================================================================
        // 8. Use `scan()` to return movies released in a range of years.
        //===========================================================================

        print("\nGetting movies released between 1993 and 1997...")
        let scannedMovies = try await movieDatabase.getMovies(firstYear: 1993,
 lastYear: 1997)
        for movie in scannedMovies {
            print("    \(movie.title) (\(movie.year))")
        }

        //===========================================================================
        // 9. Delete the table.
        //===========================================================================

        print("\nDeleting the table...")
        try await movieDatabase.deleteTable()
    }
}

@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen für DynamoDB mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne DynamoDB-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die DynamoDB-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von DynamoDB AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon DynamoDB.

Beispiele

- Verwenden Sie BatchExecuteStatementmit einem AWS SDK
- Verwenden Sie BatchGetItemmit einem AWS SDK oder CLI
- Verwenden Sie BatchWriteItemmit einem AWS SDK oder CLI
- Verwenden Sie CreateTablemit einem AWS SDK oder CLI
- Verwenden Sie DeleteItemmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTablemit einem AWS SDK oder CLI

- Verwenden Sie DescribeTablemit einem AWS SDK oder CLI
- Verwenden Sie DescribeTimeToLivemit einem AWS SDK oder CLI
- Verwenden Sie mit ExecuteStatementeinem AWS SDK
- Verwenden Sie GetItemmit einem AWS SDK oder CLI
- Verwenden Sie ListTablesmit einem AWS SDK oder CLI
- Verwenden Sie PutItemmit einem AWS SDK oder CLI
- Verwenden Sie Query mit einem AWS SDK oder CLI
- Verwenden Sie Scan mit einem AWS SDK oder CLI
- Verwenden Sie UpdateItemmit einem AWS SDK oder CLI
- Verwenden Sie UpdateTablemit einem AWS SDK oder CLI
- Verwenden Sie mit einem UpdateTimeToLive AWS SDK oder CLI

Verwenden Sie **`BatchExecuteStatement`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `BatchExecuteStatement` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Löschen von Daten mit PartiQL DELETE
- Einfügen von Daten mithilfe von PartiQL INSERT
- Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen
- Abfragen von Daten mit PartiQL SELECT
- Aktualisieren von Daten mit PartiQL UPDATE

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
```

```
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
```

```
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
```

```
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {

Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</
param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</
param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title
 = ? AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },
```

```
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {
```

```
            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND
year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/
n) "));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
```

```
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute
= Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();
```

```
            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";
```

```
        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
    // 6. Delete multiple movies using "Delete" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());
```

```
        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Funktionsempfängerstruktur für das Beispiel.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
 to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year, movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
```

```
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
 err)
 }
 return err
}
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
```

```
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
  }
 }
 return outMovies, err
}
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the
 rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
```

```
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}
```

```
// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const breakfastFoods = ["Eggs", "Bacon", "Sausage"];
  const command = new BatchExecuteStatementCommand({
    Statements: breakfastFoods.map((food) => ({
      Statement: `INSERT INTO BreakfastFoods value {'Name':?}`,
      Parameters: [food],
    })),
```

```
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Rufen Sie mithilfe von PartiQL einen Stapel von Elementen ab.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Teaspoons"],
        ConsistentRead: true,
      },
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Grams"],
        ConsistentRead: true,
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Aktualisieren Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const eggUpdates = [
    ["duck", "fried"],
    ["chicken", "omelette"],
  ];
  const command = new BatchExecuteStatementCommand({
    Statements: eggUpdates.map((change) => ({
      Statement: "UPDATE Eggs SET Style=? where Variety=?",
      Parameters: [change[1], change[0]],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Löschen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
```

```
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Grape"],
      },
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Strawberry"],
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }
```

```
        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated
 with
```

```
                            each statement. This list must be in the same order as
the
                            statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the
table "
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why:
%s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lesen Sie einen Stapel von Elementen mithilfe von PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

Löschen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
```

```
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`BatchGetItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetItem` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
```

```
namespace LowLevelBatchGet
{
    public class LowLevelBatchGet
    {
        private static readonly string _table1Name = "Forum";
        private static readonly string _table2Name = "Thread";

        public static async void
 RetrieveMultipleItemsBatchGet(AmazonDynamoDBClient client)
        {
            var request = new BatchGetItemRequest
            {
                RequestItems = new Dictionary<string, KeysAndAttributes>()
            {
                { _table1Name,
                  new KeysAndAttributes
                  {
                      Keys = new List<Dictionary<string, AttributeValue> >()
                      {
                          new Dictionary<string, AttributeValue>()
                          {
                              { "Name", new AttributeValue {
                            S = "Amazon DynamoDB"
                        } }
                          },
                          new Dictionary<string, AttributeValue>()
                          {
                              { "Name", new AttributeValue {
                            S = "Amazon S3"
                        } }
                          }
                      }
                  }},
                {
                    _table2Name,
                    new KeysAndAttributes
                    {
                        Keys = new List<Dictionary<string, AttributeValue> >()
                        {
                            new Dictionary<string, AttributeValue>()
                            {
                                { "ForumName", new AttributeValue {
                                      S = "Amazon DynamoDB"
```

```
                                    } },
                                { "Subject", new AttributeValue {
                                        S = "DynamoDB Thread 1"
                                    } }
                            },
                            new Dictionary<string, AttributeValue>()
                            {
                                { "ForumName", new AttributeValue {
                                        S = "Amazon DynamoDB"
                                    } },
                                { "Subject", new AttributeValue {
                                        S = "DynamoDB Thread 2"
                                    } }
                            },
                            new Dictionary<string, AttributeValue>()
                            {
                                { "ForumName", new AttributeValue {
                                        S = "Amazon S3"
                                    } },
                                { "Subject", new AttributeValue {
                                        S = "S3 Thread 1"
                                    } }
                            }
                        }
                    }
                }
            }
            };

            BatchGetItemResponse response;
            do
            {
                Console.WriteLine("Making request");
                response = await client.BatchGetItemAsync(request);

                // Check the response.
                var responses = response.Responses; // Attribute list in the
response.

                foreach (var tableResponse in responses)
                {
                    var tableResults = tableResponse.Value;
                    Console.WriteLine("Items retrieved from table {0}",
tableResponse.Key);
```

```
                    foreach (var item1 in tableResults)
                    {
                        PrintItem(item1);
                    }
                }

                // Any unprocessed keys? could happen if you exceed
ProvisionedThroughput or some other error.
                Dictionary<string, KeysAndAttributes> unprocessedKeys =
response.UnprocessedKeys;
                foreach (var unprocessedTableKeys in unprocessedKeys)
                {
                    // Print table name.
                    Console.WriteLine(unprocessedTableKeys.Key);
                    // Print unprocessed primary keys.
                    foreach (var key in unprocessedTableKeys.Value.Keys)
                    {
                        PrintItem(key);
                    }
                }

                request.RequestItems = unprocessedKeys;
            } while (response.UnprocessedKeys.Count > 0);
        }

        private static void PrintItem(Dictionary<string, AttributeValue>
attributeList)
        {
            foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
            {
                string attributeName = kvp.Key;
                AttributeValue value = kvp.Value;

                Console.WriteLine(
                    attributeName + " " +
                    (value.S == null ? "" : "S=[" + value.S + "]") +
                    (value.N == null ? "" : "N=[" + value.N + "]") +
                    (value.SS == null ? "" : "SS=[" + string.Join(",",
value.SS.ToArray()) + "]") +
                    (value.NS == null ? "" : "NS=[" + string.Join(",",
value.NS.ToArray()) + "]")
                    );
            }
```

```
 Console.WriteLine("*************************************************");
        }

        static void Main()
        {
            var client = new AmazonDynamoDBClient();

            RetrieveMultipleItemsBatchGet(client);
        }
    }
}
```

- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_batch_get_item
#
# This function gets a batch of items from a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the keys of the items to get.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_batch_get_item() {
  local item response
```

```
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_get_item"
    echo "Get a batch of items from a DynamoDB table."
    echo " -i item  -- Path to json file containing the keys of the items to
 get."
    echo ""
  }

  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  response=$(aws dynamodb batch-get-item \
    --request-items file://"$item")
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-get-item operation failed.$response"
    return 1
  fi
```

```
  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie BatchGetItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Batch get items from different Amazon DynamoDB tables.
/*!
  \sa batchGetItem()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::batchGetItem(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
```

```
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages,
#v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");

    // Table2: Set key name, type, and value to search.
    std::vector<Aws::String> idValues = {"102", "103", "201"};
    for (const Aws::String &id: idValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetN(id);
        keys.emplace("Id", key);
        table2KeysAndAttributes.AddKeys(keys);
    }

    requestItems.emplace(table2Name, table2KeysAndAttributes);

    bool result = true;
    do {  // Use a do loop to handle pagination.
        request.SetRequestItems(requestItems);
        const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
```

```
                request);

        if (outcome.IsSuccess()) {
            for (const auto &responsesMapEntry:
outcome.GetResult().GetResponses()) {
                Aws::String tableName = responsesMapEntry.first;
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
                std::cout << "Retrieved " << tableResults.size()
                          << " responses for table '" << tableName << "'.\n"
                          << std::endl;
                if (tableName == "Forum") {

                    std::cout << "Name | Category | Message | Views" <<
std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Name").GetS() << " | ";
                        std::cout << item.at("Category").GetS() << " | ";
                        std::cout << (item.count("Message") == 0 ? "" : item.at(
                                "Messages").GetN()) << " | ";
                        std::cout << (item.count("Views") == 0 ? "" : item.at(
                                "Views").GetN()) << std::endl;
                    }
                }
                else {
                    std::cout << "Title | Price | Color" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Title").GetS() << " | ";
                        std::cout << (item.count("Price") == 0 ? "" : item.at(
                                "Price").GetN());
                        if (item.count("Color")) {
                            std::cout << " | ";
                            for (const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
                                    "Color").GetL())
                                std::cout << listItem->GetS() << " ";
                        }
                        std::cout << std::endl;
                    }
                }
                std::cout << std::endl;
            }
```

```
            // If necessary, repeat request for remaining items.
            requestItems = outcome.GetResult().GetUnprocessedKeys();
        }
        else {
            std::cerr << "Batch get item failed: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;
        }
    } while (!requestItems.empty());

    return result;
}
```

- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie mehrere Elemente aus einer Tabelle ab

Im folgenden Beispiel für `batch-get-items` werden mithilfe eines Batches von drei `GetItem`-Anfragen mehrere Elemente aus der `MusicCollection`-Tabelle gelesen und die Anzahl der durch den Vorgang verbrauchten Lesekapazitätseinheiten wird abgefragt. Der Befehl gibt nur das `AlbumTitle`-Attribut zurück.

```
aws dynamodb batch-get-item \
    --request-items file://request-items.json \
    --return-consumed-capacity TOTAL
```

Inhalt von `request-items.json`:

```
{
    "MusicCollection": {
        "Keys": [
            {
                "Artist": {"S": "No One You Know"},
                "SongTitle": {"S": "Call Me Today"}
```

```
            },
            {
                "Artist": {"S": "Acme Band"},
                "SongTitle": {"S": "Happy Day"}
            },
            {
                "Artist": {"S": "No One You Know"},
                "SongTitle": {"S": "Scared of My Shadow"}
            }
        ],
        "ProjectionExpression":"AlbumTitle"
    }
}
```

Ausgabe:

```
{
    "Responses": {
        "MusicCollection": [
            {
                "AlbumTitle": {
                    "S": "Somewhat Famous"
                }
            },
            {
                "AlbumTitle": {
                    "S": "Blue Sky Blues"
                }
            },
            {
                "AlbumTitle": {
                    "S": "Louder Than Ever"
                }
            }
        ]
    },
    "UnprocessedKeys": {},
    "ConsumedCapacity": [
        {
            "TableName": "MusicCollection",
            "CapacityUnits": 1.5
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Batch-Vorgänge im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie BatchGetItemin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt, wie Batch-Elemente mithilfe des Service-Clients abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = """
```

```
                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
dynamoDbClient.batchGetItem(batchGetItemRequest);

        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
```

```
            List<Map<String, AttributeValue>> musicItems =
 responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
        }
    }
}
```

Zeigt, wie Batch-Elemente mithilfe des Service-Clients und eines Paginators abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();
```

```
        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient,
 String tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
 Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter BatchGet.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { BatchGetCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchGetCommand({
    // Each key in this object is the name of a table. This example refers
    // to a Books table.
    RequestItems: {
      Books: {
        // Each entry in Keys is an object that specifies a primary key.
        Keys: [
          {
            Title: "How to AWS",
          },
          {
            Title: "DynamoDB for DBAs",
          },
        ],
        // Only return the "Title" and "PageCount" attributes.
        ProjectionExpression: "Title, PageCount",
      },
    },
  });

  const response = await docClient.send(command);
  console.log(response.Responses.Books);
```

```
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: {
      Keys: [
        { KEY_NAME: { N: "KEY_VALUE_1" } },
        { KEY_NAME: { N: "KEY_VALUE_2" } },
        { KEY_NAME: { N: "KEY_VALUE_3" } },
      ],
      ProjectionExpression: "KEY_NAME, ATTRIBUTE",
    },
  },
};

ddb.batchGetItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function (element, index, array) {
      console.log(element);
```

```
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft das Element mit dem Namen SongTitle „Somewhere Down The Road“ aus den DynamoDB-Tabellen „Music“ und „Songs“ ab.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-
DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
```

```
AlbumTitle                      Somewhat Famous
CriticRating                    10
Genre                           Country
Price                           1.94
Artist                          No One You Know
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
CriticRating                    10
Genre                           Country
Price                           1.94
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (BatchGetItemV4).

Tools für V5 PowerShell

Beispiel 1: Ruft das Element mit dem Namen SongTitle „Somewhere Down The Road“ aus den DynamoDB-Tabellen „Music“ und „Songs“ ab.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-
DDBItem
```

Ausgabe:

```
Name                            Value
```

```
----                            -----
Artist                          No One You Know
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
CriticRating                    10
Genre                           Country
Price                           1.94
Artist                          No One You Know
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
CriticRating                    10
Genre                           Country
Price                           1.94
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (BatchGetItemV5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import decimal
import json
import logging
import os
import pprint
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource("dynamodb")

MAX_GET_SIZE = 100  # Amazon DynamoDB rejects a get batch larger than 100 items.
```

```
def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.

    When Amazon DynamoDB cannot process all items in a batch, a set of
 unprocessed
    keys is returned. This function uses an exponential backoff algorithm to
 retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most
 100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1  # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get("Responses", []):
            retrieved[key] += response["Responses"][key]
        unprocessed = response["UnprocessedKeys"]
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key["Keys"]) for batch_key in batch_keys.values()]
            )
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.",
 unprocessed_count
            )
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
        else:
```

```
            break

    return retrieved
```

- Einzelheiten zur API finden Sie BatchGetItemin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Gets an array of `Movie` objects describing all the movies in the
    /// specified list. Any movies that aren't found in the list have no
    /// corresponding entry in the resulting array.
    ///
    /// - Parameters
    ///     - keys: An array of tuples, each of which specifies the title and
    ///       release year of a movie to fetch from the table.
    ///
    /// - Returns:
    ///     - An array of `Movie` objects describing each match found in the
    ///     table.
    ///
    /// - Throws:
    ///     - `MovieError.ClientUninitialized` if the DynamoDB client has not
    ///     been initialized.
    ///     - DynamoDB errors are thrown without change.
    func batchGet(keys: [(title: String, year: Int)]) async throws -> [Movie] {
        do {
```

```
            guard let client = self.ddbClient else {
                throw MovieError.ClientUninitialized
            }

            var movieList: [Movie] = []
            var keyItems: [[Swift.String: DynamoDBClientTypes.AttributeValue]] =
[]

            // Convert the list of keys into the form used by DynamoDB.

            for key in keys {
                let item: [Swift.String: DynamoDBClientTypes.AttributeValue] = [
                    "title": .s(key.title),
                    "year": .n(String(key.year))
                ]
                keyItems.append(item)
            }

            // Create the input record for `batchGetItem()`. The list of
requested
            // items is in the `requestItems` property. This array contains one
            // entry for each table from which items are to be fetched. In this
            // example, there's only one table containing the movie data.
            //
            // If we wanted this program to also support searching for matches
            // in a table of book data, we could add a second `requestItem`
            // mapping the name of the book table to the list of items we want to
            // find in it.
            let input = BatchGetItemInput(
                requestItems: [
                    self.tableName: .init(
                        consistentRead: true,
                        keys: keyItems
                    )
                ]
            )

            // Fetch the matching movies from the table.

            let output = try await client.batchGetItem(input: input)

            // Get the set of responses. If there aren't any, return the empty
            // movie list.
```

```
            guard let responses = output.responses else {
                return movieList
            }

            // Get the list of matching items for the table with the name
            // `tableName`.

            guard let responseList = responses[self.tableName] else {
                return movieList
            }

            // Create `Movie` items for each of the matching movies in the table
            // and add them to the `MovieList` array.

            for response in responseList {
                try movieList.append(Movie(withItem: response))
            }

            return movieList
        } catch {
            print("ERROR: batchGet", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie BatchGetItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`BatchWriteItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `BatchWriteItem` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Schreibt einen Stapel von Elementen in die Filmtabelle.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The name of the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public List<Movie> ImportMovies(string movieFileName)
{
    var moviesList = new List<Movie>();
    if (!File.Exists(movieFileName))
    {
        return moviesList;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonSerializer.Deserialize<List<Movie>>(
        json,
        new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        });

    // Now return the first 250 entries.
    if (allMovies != null && allMovies.Any())
    {
        moviesList = allMovies.GetRange(0, 250);
    }
    return moviesList;
}
```

```
    /// <summary>
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie
data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of
IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();

            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
```

```
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch
 write operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
 {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
```

```
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Batch write items from a JSON file.
```

```
/*!
  \sa batchWriteItem()
  \param jsonFilePath: JSON file path.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

/*
 * The input for this routine is a JSON file that you can download from the
 following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 generate
 * JSON strings when constructing the BatchWriteItem request, essentially
 outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
 jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
```

```
        //  key - table name
        //  value - list of request objects
        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                      << tableName << "' is not a list." << std::endl;
            continue;
        }

        Aws::Utils::Array<Aws::Utils::Json::JsonView> entries =
 entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
                                               writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
 dynamoClient.BatchWriteItem(
            batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
  \sa addWriteRequests()
  \param tableName: Name of the table for the write operations.
  \param requestsJson: Request data in JSON format.
  \param writeRequests: Vector to receive the WriteRequest objects.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                        const
 Aws::Utils::Array<Aws::Utils::Json::JsonView> &requestsJson,

 Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonView &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                      << requestsEntry.WriteReadable() << std::endl;
            return false;
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonView> requestsMap =
 requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonView &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                                            attributes)) {
                    std::cerr << "Error getting attributes "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }

                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(attributes);
                writeRequests.push_back(
                        Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                                putRequest));
            }
            else {
                std::cerr << "Error: unimplemented request type '" << requestType
```

```
                            << "'." << std::endl;
            }
        }
    }

    return true;
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
  \sa getAttributeObjectsMap()
  \param jsonView: JSONView of attribute records.
  \param writeRequests: Map to receive the AttributeValue objects.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView
 &jsonView,
                                          Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
 jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                      << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,

 Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie mehrere Elemente einer Tabelle hinzu

Im folgenden Beispiel für `batch-write-item` werden der `MusicCollection`-Tabelle drei neue Elemente hinzugefügt, wobei ein Batch von drei `PutItem`-Anforderungen verwendet wird. Außerdem werden Informationen zur Anzahl der Schreibkapazitätseinheiten abgefragt, die von dem Vorgang verbraucht werden, sowie zu allen Elementsammlungen, die durch den Vorgang geändert werden.

```
aws dynamodb batch-write-item \
    --request-items file://request-items.json \
    --return-consumed-capacity INDEXES \
    --return-item-collection-metrics SIZE
```

Inhalt von `request-items.json`:

```
{
    "MusicCollection": [
        {
            "PutRequest": {
                "Item": {
                    "Artist": {"S": "No One You Know"},
                    "SongTitle": {"S": "Call Me Today"},
                    "AlbumTitle": {"S": "Somewhat Famous"}
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
                    "Artist": {"S": "Acme Band"},
                    "SongTitle": {"S": "Happy Day"},
                    "AlbumTitle": {"S": "Songs About Life"}
                }
            }
        },
        {
            "PutRequest": {
                "Item": {
```

```
                    "Artist": {"S": "No One You Know"},
                    "SongTitle": {"S": "Scared of My Shadow"},
                    "AlbumTitle": {"S": "Blue Sky Blues"}
                }
            }
        }
    ]
}
```

Ausgabe:

```
{
    "UnprocessedItems": {},
    "ItemCollectionMetrics": {
        "MusicCollection": [
            {
                "ItemCollectionKey": {
                    "Artist": {
                        "S": "No One You Know"
                    }
                },
                "SizeEstimateRangeGB": [
                    0.0,
                    1.0
                ]
            },
            {
                "ItemCollectionKey": {
                    "Artist": {
                        "S": "Acme Band"
                    }
                },
                "SizeEstimateRangeGB": [
                    0.0,
                    1.0
                ]
            }
        ]
    },
    "ConsumedCapacity": [
        {
            "TableName": "MusicCollection",
            "CapacityUnits": 6.0,
```

```
            "Table": {
                "CapacityUnits": 3.0
            },
            "LocalSecondaryIndexes": {
                "AlbumTitleIndex": {
                    "CapacityUnits": 3.0
                }
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Batch-Vorgänge im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx,
 &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
```

```
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }

 return written, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
```

```
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügt mithilfe des Service-Clients viele Elemente in eine Tabelle ein.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
 AttributeValue.builder().s("Artist1").build());
```

```
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
song!").build());
        item1Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle1").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attri

        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
melody.").build());
        item2Attributes.put("SongTitle",
AttributeValue.builder().s("SongTitle2").build());

writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attri

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " +
batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Fügt mithilfe des erweiterten Clients viele Elemente in eine Tabelle ein.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import
 software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named
 Customer with these columns:
 *   - id - the id of the record that is the key
 *   - custName - the customer name
 *   - email - the email value
 *   - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnhancedBatchWriteItems {
        public static void main(String[] args) {
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
                DynamoDbEnhancedClient enhancedClient =
 DynamoDbEnhancedClient.builder()
                                .dynamoDbClient(ddb)
```

```
                                .build();
                putBatchRecords(enhancedClient);
                ddb.close();
        }

        public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient)
{
                try {
                        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                                        TableSchema.fromBean(Customer.class));
                        DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                                        TableSchema.fromBean(Music.class));
                        LocalDate localDate = LocalDate.parse("2020-04-07");
                        LocalDateTime localDateTime = localDate.atStartOfDay();
                        Instant instant =
localDateTime.toInstant(ZoneOffset.UTC);

                        Customer record2 = new Customer();
                        record2.setCustName("Fred Pink");
                        record2.setId("id110");
                        record2.setEmail("fredp@noserver.com");
                        record2.setRegistrationDate(instant);

                        Customer record3 = new Customer();
                        record3.setCustName("Susan Pink");
                        record3.setId("id120");
                        record3.setEmail("spink@noserver.com");
                        record3.setRegistrationDate(instant);

                        Customer record4 = new Customer();
                        record4.setCustName("Jerry orange");
                        record4.setId("id101");
                        record4.setEmail("jorange@noserver.com");
                        record4.setRegistrationDate(instant);

                        BatchWriteItemEnhancedRequest
batchWriteItemEnhancedRequest = BatchWriteItemEnhancedRequest
                                        .builder()
                                        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer
```

```
         // table
 .mappedTableResource(customerMappedTable)
 .addPutItem(builder -> builder.item(record2))
 .addPutItem(builder -> builder.item(record3))
 .addPutItem(builder -> builder.item(record4))
                                                                    .build(),
 WriteBatch.builder(Music.class) // delete an item from the Music
       // table
 .mappedTableResource(musicMappedTable)
 .addDeleteItem(builder -> builder.key(
       Key.builder().partitionValue(
                        "Famous Band")
                        .build()))
                                                                    .build())
                                        .build();
                        // Add three items to the Customer table and delete one
 item from the Music
                        // table.
 enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
                        System.out.println("done");
                } catch (DynamoDbException e) {
                        System.err.println(e.getMessage());
                        System.exit(1);
                }
        }
}
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter BatchWrite.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";
import { readFileSync } from "node:fs";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const file = readFileSync(
    `${dirname}../../../../../resources/sample_files/movies.json`,
  );

  const movies = JSON.parse(file.toString());

  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);
```

```
  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        // An existing table is required. A composite key of 'title' and 'year'
 is recommended
        // to account for duplicate titles.
        BatchWriteMoviesTable: putRequests,
      },
    });

    await docClient.send(command);
  }
};
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
```

```
    TABLE_NAME: [
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
    ],
  },
};

ddb.batchWriteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
 items or increase depth.");
        }

        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' =>
 ['Item' => $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
 ($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
                echo $e->getMessage();
                die();
            }
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.
\n";
                break;
            }
        }
    }
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes durch ein neues Element in den DynamoDB-Tabellen „Music“ und „Songs“.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Einzelheiten zur API finden Sie unter BatchWriteItem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes durch ein neues Element in den DynamoDB-Tabellen „Music“ und „Songs“.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
```

```
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- Einzelheiten zur API finden Sie unter BatchWriteItem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
```

```
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts
sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
least
                       the keys required by the schema that was specified when
the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Einzelheiten zur API finden Sie BatchWriteItemin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must
 contain at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
      movies[index, slice_size].each do |movie|
        movie_items.append({ put_request: { item: movie } })
```

```
      end
      @dynamo_resource.client.batch_write_item({ request_items: { @table.name =>
 movie_items } })
      index += slice_size
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:"
    )
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für Ruby API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.
```

```
            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from:
jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item
request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and
populate
            // them with `PutRequest` requests, each specifying one movie from
the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
```

```
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie BatchWriteItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateTable`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Erweiterte Szenarien für globale sekundäre Indizes
- Erstellen einer Tabelle mit einem globalen sekundären Index
- Erstellen einer Tabelle mit aktiviertem Warmdurchsatz
- Erstellen und Verwalten globaler Tabellen zur Demonstration von MREC
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Verwalten globaler sekundärer Indizes
- Verwalten ressourcenbasierter Richtlinien
- Richten Sie die Zugriffskontrolle ein Attribute-Based
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)
- Arbeiten mit lokalen sekundären Indizes
- Arbeiten mit Ressourcen-Tagging
- Arbeiten Sie mit Streams und Time-to-Live
- Arbeiten mit der Tabellenverschlüsselung

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
```

```
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
```

```
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table
 {tableName}. {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and
 their types.
#       -k key_schema -- JSON file path of a list of attributes and their key
 types.
#
#  Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$attribute_definitions" ]]; then
    errecho "ERROR: You must provide an attribute definitions json file path the
 -a parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k
 parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
```

```
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie CreateTablein der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon DynamoDB table.
/*!
  \sa createTable()
  \param tableName: Name for the DynamoDB table.
  \param primaryKey: Primary key for the DynamoDB table.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
              " with a simple primary key: \"" << primaryKey << "\"." <<
 std::endl;
```

```
    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
 dynamoClient.CreateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
                  << outcome.GetResult().GetTableDescription().GetTableName() <<
                  " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " <<
 outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
```

```
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Tabelle mit Tags

Im folgenden Beispiel für `create-table` werden die angegebenen Attribute und das angegebene Schlüsselschema verwendet, um eine Tabelle mit dem Namen `MusicCollection` zu erstellen. Diese Tabelle verwendet den bereitgestellten Durchsatz und wird im Ruhezustand mit dem standardmäßigen AWS eigenen CMK verschlüsselt. Der Befehl weist der Tabelle außerdem ein Tag mit dem Schlüssel `Owner` und dem Wert `blueTeam` zu.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
    --tags Key=Owner,Value=blueTeam
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 5
        },
        "TableSizeBytes": 0,
```

```
        "TableName": "MusicCollection",
        "TableStatus": "CREATING",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": "2020-05-26T16:04:41.627000-07:00",
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So erstellen Sie eine Tabelle im Modus On-Demand

Im folgenden Beispiel wird mit dem On-Demand-Modus anstelle des bereitgestellten Durchsatzmodus eine Tabelle mit dem Namen `MusicCollection` erstellt. Dies ist nützlich für Tabellen mit unvorhersehbaren Workloads.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
```

```
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T11:44:10.807000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 0,
            "WriteCapacityUnits": 0
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "BillingModeSummary": {
            "BillingMode": "PAY_PER_REQUEST"
        }
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 3: So erstellen Sie eine Tabelle und verschlüsseln sie mit einem kundenseitig verwalteten Schlüssel (Customer Managed Key; CMK).

Im folgenden Beispiel wird eine Tabelle mit dem Namen `MusicCollection` erstellt und mithilfe eines kundenseitig verwalteten Schlüssels (Customer Managed Key; CMK) verschlüsselt.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=abcd1234-
abcd-1234-a123-ab1234a1b234
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T11:12:16.431000-07:00",
        "ProvisionedThroughput": {
```

```
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "SSEDescription": {
            "Status": "ENABLED",
            "SSEType": "KMS",
            "KMSMasterKeyArn": "arn:aws:kms:us-west-2:123456789012:key/abcd1234-
abcd-1234-a123-ab1234a1b234"
        }
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Bespiel 4: So erstellen Sie eine Tabelle mit einem lokalen sekundären Index

Im folgenden Beispiel werden die angegebenen Attribute und das angegebene Schlüsselschema verwendet, um eine Tabelle namens `MusicCollection` mit dem lokalen sekundären Index `AlbumTitleIndex` zu erstellen.

```
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-
definitions AttributeName=Artist,AttributeType=S AttributeName=SongTitle,AttributeType=S
 \
    --key-
schema AttributeName=Artist,KeyType=HASH AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumTitleIndex\",
                \"KeySchema\": [
                    {\"AttributeName\": \"Artist\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\": \"AlbumTitle\",\"KeyType\":\"RANGE\"}
                ],
```

```
                \"Projection\": {
                    \"ProjectionType\": \"INCLUDE\",
                    \"NonKeyAttributes\": [\"Genre\", \"Year\"]
                }
            }
        ]"
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
```

```
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Genre",
                        "Year"
                    ]
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 5: So erstellen Sie eine Tabelle mit einem globalen sekundären Index

Im folgenden Beispiel wird eine Tabelle mit dem Namen `GameScores` mit einem globalen sekundären Index namens `GameTitleIndex` erstellt. Die Basistabelle hat einen Partitionsschlüssel von `UserId` und einen Sortierschlüssel von `GameTitle`, mit dem Sie effizient die beste Punktzahl eines einzelnen Benutzers für ein bestimmtes

Spiel finden können, während die GSI einen Partitionsschlüssel von `GameTitle` und einen Sortierschlüssel von `TopScore` hat, mit dem Sie finden Sie schnell die höchste Gesamtpunktzahl für ein bestimmtes Spiel.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-schema AttributeName=UserId,KeyType=HASH \
                 AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GameTitleIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"GameTitle\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"TopScore\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {
                    \"ProjectionType\":\"INCLUDE\",
                    \"NonKeyAttributes\":[\"UserId\"]
                },
                \"ProvisionedThroughput\": {
                    \"ReadCapacityUnits\": 10,
                    \"WriteCapacityUnits\": 5
                }
            }
        ]"
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "TopScore",
                "AttributeType": "N"
```

```
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-26T17:28:15.602000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "GameTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "TopScore",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
```

```
                        "UserId"
                    ]
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 5
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameTitleIndex"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 6: So erstellen Sie eine Tabelle mit mehreren globalen sekundären Indizes gleichzeitig

Im folgenden Beispiel wird eine Tabelle namens `GameScores` mit zwei globalen sekundären Indizes erstellt. Die GSI-Schemas werden über eine Datei und nicht über die Befehlszeile übergeben.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --global-secondary-indexes file://gsi.json
```

Inhalt von `gsi.json`:

```
[
    {
        "IndexName": "GameTitleIndex",
```

```
        "KeySchema": [
            {
                "AttributeName": "GameTitle",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "TopScore",
                "KeyType": "RANGE"
            }
        ],
        "Projection": {
            "ProjectionType": "ALL"
        },
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        }
    },
    {
        "IndexName": "GameDateIndex",
        "KeySchema": [
            {
                "AttributeName": "GameTitle",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "Date",
                "KeyType": "RANGE"
            }
        ],
        "Projection": {
            "ProjectionType": "ALL"
        },
        "ProvisionedThroughput": {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        }
    }
]
```

Ausgabe:

```
{
```

```
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Date",
                "AttributeType": "S"
            },
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "TopScore",
                "AttributeType": "N"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-08-04T16:40:55.524000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "GameTitleIndex",
```

```
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "TopScore",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 5
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameTitleIndex"
            },
            {
                "IndexName": "GameDateIndex",
                "KeySchema": [
                    {
                        "AttributeName": "GameTitle",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "Date",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 5,
                    "WriteCapacityUnits": 5
```

```
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/index/GameDateIndex"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 7: So erstellen Sie eine Tabelle mit aktivierten Streams

Im folgenden Beispiel wird die Tabelle `GameScores` mit aktiviertem DynamoDB Streams aufgerufen. Sowohl neue als auch alte Images von jedem Element werden in den Stream geschrieben.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --stream-specification StreamEnabled=TRUE,StreamViewType=NEW_AND_OLD_IMAGES
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
```

```
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2020-05-27T10:49:34.056000-07:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_AND_OLD_IMAGES"
        },
        "LatestStreamLabel": "2020-05-27T17:49:34.056",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2020-05-27T17:49:34.056"
    }
}
```

Weitere Informationen finden Sie unter Grundlegende Operationen für Tabellen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 8: Um eine Tabelle mit aktiviertem Keys-Only Stream zu erstellen

Im folgenden Beispiel wird die Tabelle `GameScores` mit aktiviertem DynamoDB Streams aufgerufen. Nur die Schlüsselattribute von geänderten Elementen werden in den Stream geschrieben.

```
aws dynamodb create-table \
```

```
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --stream-specification StreamEnabled=TRUE,StreamViewType=KEYS_ONLY
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T18:45:34.140000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
```

```
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "KEYS_ONLY"
        },
        "LatestStreamLabel": "2023-05-25T18:45:34.140",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
GameScores/stream/2023-05-25T18:45:34.140",
        "DeletionProtectionEnabled": false
    }
}
```

Weitere Informationen finden Sie unter Ändern der Datenerfassung für DynamoDB Streams im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 9: So erstellen Sie eine Tabelle mit der Klasse Standard Infrequent Access

Im folgenden Beispiel wird eine Tabelle mit dem Namen Standard-Infrequent Access (DynamoDB Standard-IA) erstellt `GameScores` und ihr zugewiesen. Diese Tabellenklasse ist dafür optimiert, dass Speicher der Hauptkostenfaktor ist.

```
aws dynamodb create-table \
    --table-name GameScores \
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --table-class STANDARD_INFREQUENT_ACCESS
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
```

```
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T18:33:07.581000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "TableClassSummary": {
            "TableClass": "STANDARD_INFREQUENT_ACCESS"
        },
        "DeletionProtectionEnabled": false
    }
}
```

Weitere Informationen finden Sie unter Tabellenklassen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 10: So erstellen Sie eine Tabelle mit aktiviertem Löschschutz

Im folgenden Beispiel wird eine Tabelle mit dem Namen `GameScores` erstellt und der Löschschutz aktiviert.

```
aws dynamodb create-table \
    --table-name GameScores \
```

```
    --attribute-
definitions AttributeName=UserId,AttributeType=S AttributeName=GameTitle,AttributeType=S
 \
    --key-
schema AttributeName=UserId,KeyType=HASH AttributeName=GameTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=10,WriteCapacityUnits=5 \
    --deletion-protection-enabled
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "GameTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "UserId",
                "AttributeType": "S"
            }
        ],
        "TableName": "GameScores",
        "KeySchema": [
            {
                "AttributeName": "UserId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "GameTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "CreationDateTime": "2023-05-25T23:02:17.093000+00:00",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 10,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/GameScores",
```

```
        "TableId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "DeletionProtectionEnabled": true
    }
}
```

Weitere Informationen finden Sie unter Verwenden des Löschschutzes im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie unter Befehlsreferenz CreateTable.AWS CLI

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// CreateMovieTable creates a DynamoDB table with a composite primary key defined
 as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.OnDemandThroughput;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTable {
```

```
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + "
with a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
 scales based on traffic.
            .tableName(tableName)
            .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
const client = new DynamoDBClient({});

export const main = async () => {
  const command = new CreateTableCommand({
    TableName: "EspressoDrinks",
    // For more information about data types,
    // see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes and
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
    AttributeDefinitions: [
      {
        AttributeName: "DrinkName",
        AttributeType: "S",
      },
    ],
    KeySchema: [
      {
        AttributeName: "DrinkName",
        KeyType: "HASH",
      },
    ],
    BillingMode: "PAY_PER_REQUEST",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  AttributeDefinitions: [
    {
      AttributeName: "CUSTOMER_ID",
      AttributeType: "N",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      AttributeType: "S",
    },
  ],
  KeySchema: [
    {
      AttributeName: "CUSTOMER_ID",
      KeyType: "HASH",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      KeyType: "RANGE",
    },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1,
  },
  TableName: "CUSTOMER_LIST",
  StreamSpecification: {
    StreamEnabled: false,
  },
};

// Call DynamoDB to create the table
ddb.createTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Table Created", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
```

```
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine -Tabelle.

```
        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
```

```
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
 'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName,
 'AttributeType' => $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
 'WriteCapacityUnits' => 10],
        ]);
    }
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
```

```
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Beispiel 2: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Ein lokaler sekundärer Index wird ebenfalls definiert. Der Schlüssel des lokalen sekundären Indexes wird automatisch anhand des primären Hashschlüssels in der Tabelle (ForumName) festgelegt. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Beispiel 3: Dieses Beispiel zeigt, wie eine einzelne Pipeline verwendet wird, um eine Tabelle mit dem Namen Thread zu erstellen, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp Hash) und 'Subject' (Schlüsseltypbereich) und einem lokalen sekundären Index besteht. Add-DDBKeySchema Und Add-DDBIndexSchema erstellen Sie ein neues TableSchema Objekt für Sie, falls keines über die Pipeline oder den Parameter -Schema bereitgestellt wird.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
```

```
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Einzelheiten zur API finden Sie unter CreateTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
```

```
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Beispiel 2: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht. Ein lokaler sekundärer Index wird ebenfalls definiert. Der Schlüssel des lokalen sekundären Indexes wird automatisch anhand des primären Hashschlüssels in der Tabelle (ForumName) festgelegt. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Beispiel 3: Dieses Beispiel zeigt, wie eine einzelne Pipeline verwendet wird, um eine Tabelle mit dem Namen Thread zu erstellen, deren Primärschlüssel aus 'ForumName' (Schlüsseltyp Hash) und 'Subject' (Schlüsseltypbereich) und einem lokalen sekundären Index besteht. Add-DDBKeySchema Und Add-DDBIndexSchema erstellen Sie ein neues TableSchema Objekt für Sie, falls keines über die Pipeline oder den Parameter -Schema bereitgestellt wird.

```
New-DDBTableSchema |
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
```

```
                      -RangeKeyDataType "S" `
                      -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- Einzelheiten zur API finden Sie unter CreateTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Tabelle zum Speichern von Filmdaten.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
```

```
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition
key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
```

```
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table
```

- Einzelheiten zur API finden Sie CreateTablein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Creates an Amazon DynamoDB table that can be used to store movie data.
```

```
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_table(
    client: &Client,
    table: &str,
    key: &str,
```

```
) -> Result<CreateTableOutput, Error> {
    let a_name: String = key.into();
    let table_name: String = table.into();

    let ad = AttributeDefinition::builder()
        .attribute_name(&a_name)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .map_err(Error::BuildError)?;

    let ks = KeySchemaElement::builder()
        .attribute_name(&a_name)
        .key_type(KeyType::Hash)
        .build()
        .map_err(Error::BuildError)?;

    let create_table_response = client
        .create_table()
        .table_name(table_name)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await;

    match create_table_response {
        Ok(out) => {
            println!("Added table {} with key {}", table, key);
            Ok(out)
        }
        Err(e) => {
            eprintln!("Got an error creating table:");
            eprintln!("{}", e);
            Err(Error::unhandled(e))
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        oo_result = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " This exception can happen if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
 "year", attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName:
 "title", attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
 keyType: .hash),
```

```
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteItem` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Verwenden von bedingten Operationen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N =
movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
```

```
        }
    }
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
```

```
    echo " -k keys  -- Path to json file containing the keys that identify the
item to delete."
    echo ""
  }
  while getopts "n:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho ""

  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
```

```
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an item from an Amazon DynamoDB table.
```

```
/*!
  \sa deleteItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
 dynamoClient.DeleteItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
```

```
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So löschen Sie ein Element

Im folgenden Beispiel für `delete-item` wird ein Element aus der `MusicCollection`-Tabelle gelöscht und es werden Details zu dem gelöschten Element und der von der Anforderung verwendeten Kapazität angefordert.

```
aws dynamodb delete-item \
    --table-name MusicCollection \
    --key file://key.json \
    --return-values ALL_OLD \
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Inhalt von `key.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Scared of My Shadow"}
}
```

Ausgabe:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Blue Sky Blues"
        },
        "Artist": {
            "S": "No One You Know"
        },
        "SongTitle": {
            "S": "Scared of My Shadow"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 2.0
    },
    "ItemCollectionMetrics": {
```

```
        "ItemCollectionKey": {
            "Artist": {
                "S": "No One You Know"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So löschen Sie ein Element bedingungsabhängig

Im folgenden Beispiel wird ein Element nur dann aus der `ProductCatalog`-Tabelle gelöscht, wenn `ProductCategory` entweder `Sporting Goods` oder `Gardening Supplies` ist und der Preis zwischen 500 und 600 liegt. Es werden Details zu dem gelöschten Element zurückgegeben.

```
aws dynamodb delete-item \
    --table-name ProductCatalog \
    --key '{"Id":{"N":"456"}}' \
    --condition-expression "(ProductCategory IN (:cat1, :cat2)) and (#P
 between :lo and :hi)" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-values ALL_OLD
```

Inhalt von `names.json`:

```
{
    "#P": "Price"
}
```

Inhalt von `values.json`:

```
{
    ":cat1": {"S": "Sporting Goods"},
    ":cat2": {"S": "Gardening Supplies"},
```

```
    ":lo": {"N": "500"},
    ":hi": {"N": "600"}
}
```

Ausgabe:

```
{
    "Attributes": {
        "Id": {
            "N": "456"
        },
        "Price": {
            "N": "550"
        },
        "ProductCategory": {
            "S": "Sporting Goods"
        }
    }
}
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie DeleteItemin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyval>

                Where:
                    tableName - The Amazon DynamoDB table to delete the item from
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to delete
 (for example, Famous Band).
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
 String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(keyToGet)
                .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter DeleteCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, DeleteCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new DeleteCommand({
    TableName: "Sodas",
    Key: {
      Flavor: "Cola",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Element aus einer Tabelle.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "VALUE" },
  },
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Löschen Sie ein Element mithilfe des DynamoDB-Dokument-Clients aus einer Tabelle.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  Key: {
    HASH_KEY: VALUE,
  },
  TableName: "TABLE",
};

docClient.delete(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
```

```
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- Einzelheiten zur API finden Sie DeleteItemin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
```

```
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Entfernt das DynamoDB-Element, das dem angegebenen Schlüssel entspricht.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Einzelheiten zur API finden Sie unter DeleteItem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Entfernt das DynamoDB-Element, das dem angegebenen Schlüssel entspricht.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- Einzelheiten zur API finden Sie unter DeleteItem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Sie können eine Bedingung angeben, damit ein Element nur gelöscht wird, wenn es bestimmte Kriterien erfüllt.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item
 is
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={"year": year, "title": title},
```

```
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))},
            )
        except ClientError as err:
            if err.response["Error"]["Code"] ==
 "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.",
                    title,
                    rating,
                )
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie DeleteItemin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end
```

```
  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_item(
    client: &Client,
    table: &str,
    key: &str,
    value: &str,
) -> Result<DeleteItemOutput, Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(out) => {
            println!("Deleted item from table");
            Ok(out)
```

```
        }
        Err(e) => Err(Error::unhandled(e)),
    }
}
```

- Einzelheiten zur API finden Sie DeleteItemin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lo_resp) = lo_dyn->deleteitem(
         iv_tablename                = iv_table_name
         it_key                      = it_key_input ).
       MESSAGE 'Deleted one item.' TYPE 'I'.
     CATCH /aws1/cx_dyncondalcheckfaile00.
       MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
     CATCH /aws1/cx_dynresourcenotfoundex.
       MESSAGE 'The table or index does not exist' TYPE 'E'.
     CATCH /aws1/cx_dyntransactconflictex.
       MESSAGE 'Another transaction is using the item' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteItemin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteTable`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Verwalten globaler sekundärer Indizes
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };
```

```
            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table
{tableName}. {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
```

```
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon DynamoDB table.
/*!
  \sa deleteTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine Tabelle

Im folgenden Beispiel für `delete-table` wird die Tabelle `MusicCollection` gelöscht.

```
aws dynamodb delete-table \
    --table-name MusicCollection
```

Ausgabe:

```
{
    "TableDescription": {
        "TableStatus": "DELETING",
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableName": "MusicCollection",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 5
        }
    }
}
```

Weitere Informationen finden Sie unter Löschen einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie DeleteTablein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
```

```
 return err
}
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to delete (for example,
 Music3).
```

```
                **Warning** This program will delete the table that you specify!
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n",
 tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DeleteTableCommand({
    TableName: "DecafCoffees",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
```

```
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function (err, data) {
  if (err && err.code === "ResourceNotFoundException") {
    console.log("Error: Table not found");
  } else if (err && err.code === "ResourceInUseException") {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
```

```
}
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteTable(string $TableName)
    {
        $this->customWaiter(function () use ($TableName) {
            return $this->dynamoDbClient->deleteTable([
                'TableName' => $TableName,
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht die angegebene Tabelle. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable"
```

Beispiel 2: Löscht die angegebene Tabelle. Sie werden nicht zum Bestätigen aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Einzelheiten zur API finden Sie unter DeleteTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht die angegebene Tabelle. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable"
```

Beispiel 2: Löscht die angegebene Tabelle. Sie werden nicht zum Bestätigen aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable" -Force
```

- Einzelheiten zur API finden Sie unter DeleteTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
```

```
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteTablein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_table(client: &Client, table: &str) ->
 Result<DeleteTableOutput, Error> {
    let resp = client.delete_table().table_name(table).send().await;

    match resp {
        Ok(out) => {
            println!("Deleted table");
            Ok(out)
        }
        Err(e) => Err(Error::Unhandled(e.into())),
    }
}
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        " Wait till the table is actually deleted.
        lo_dyn->get_waiter( )->tablenotexists(
```

```
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'Table ' && iv_table_name && ' deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table ' && iv_table_name && ' does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted since it is in use' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
```

```
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeTable`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten globaler Tabellen zur Demonstration von MREC
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)
- Arbeiten Sie mit Streams und Time-to-Live
- Arbeiten mit der Tabellenverschlüsselung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    private static async Task GetTableInformation()
    {
        Console.WriteLine("\n*** Retrieving table information ***");
```

```
        var response = await Client.DescribeTableAsync(new DescribeTableRequest
        {
            TableName = ExampleTableName
        });

        var table = response.Table;
        Console.WriteLine($"Name: {table.TableName}");
        Console.WriteLine($"# of items: {table.ItemCount}");

    }
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
###############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.
```

```
#######################################
# Function usage explanation
#######################################
function usage() {
  echo "function dynamodb_describe_table"
  echo "Describe the status of a DynamoDB table."
  echo "  -n table_name  -- The name of the table."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

local table_status
  table_status=$(
    aws dynamodb describe-table \
      --table-name "$table_name" \
      --output text \
      --query 'Table.TableStatus'
  )

 local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an Amazon DynamoDB table.
/*!
  \sa describeTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
```

```
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
 dynamoClient.DescribeTable(
            request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
 outcome.GetResult().GetTable();
        std::cout << "Table name  : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN   : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
                  <<
 Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                        td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
 td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
 std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
 std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
 td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

 Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
                             a.GetAttributeType()) <<
                     ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
 outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie eine Tabelle

Im folgenden Beispiel für `describe-table` wird die `MusicCollection`-Tabelle beschrieben.

```
aws dynamodb describe-table \
    --table-name MusicCollection
```

Ausgabe:

```
{
    "Table": {
        "AttributeDefinitions": [
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 5,
            "ReadCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "TableName": "MusicCollection",
        "TableStatus": "ACTIVE",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "Artist"
```

```
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "SongTitle"
            }
        ],
        "ItemCount": 0,
        "CreationDateTime": 1421866952.062
    }
}
```

Weitere Informationen finden Sie unter Beschreiben einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie DescribeTablein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
```

```
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information
 about (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
```

```
        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String
 tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n",
 tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
 tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
 throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
 throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
 tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
 a.attributeType());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DescribeTableCommand({
    TableName: "Pastries",
  });

  const response = await client.send(command);
  console.log(`TABLE NAME: ${response.Table.TableName}`);
  console.log(`TABLE ITEM COUNT: ${response.Table.ItemCount}`);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details der angegebenen Tabelle zurück.

```
Get-DDBTable -TableName "myTable"
```

- Einzelheiten zur API finden Sie unter DescribeTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details der angegebenen Tabelle zurück.

```
Get-DDBTable -TableName "myTable"
```

- Einzelheiten zur API finden Sie unter DescribeTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
```

```
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists
```

- Einzelheiten zur API finden Sie DescribeTablein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_dyn->describetable( iv_tablename = iv_table_name ).
    DATA(lv_tablename) = oo_result->get_table( )->ask_tablename( ).
    DATA(lv_tablearn) = oo_result->get_table( )->ask_tablearn( ).
    DATA(lv_tablestatus) = oo_result->get_table( )->ask_tablestatus( ).
    DATA(lv_itemcount) = oo_result->get_table( )->ask_itemcount( ).
    MESSAGE 'The table name is ' && lv_tablename
        && '. The table ARN is ' && lv_tablearn
        && '. The tablestatus is ' && lv_tablestatus
        && '. Item count is ' && lv_itemcount TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table ' && lv_tablename && ' does not exist' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeTablein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeTimeToLive`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTimeToLive` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten Sie mit Streams und Time-to-Live

CLI

AWS CLI

So zeigen Sie Time-to-Live-Einstellungen für eine Tabelle an

Im folgenden Beispiel für `describe-time-to-live` werden die Time-to-Live-Einstellungen für die `MusicCollection`-Tabelle angezeigt.

```
aws dynamodb describe-time-to-live \
    --table-name MusicCollection
```

Ausgabe:

```
{
    "TimeToLiveDescription": {
        "TimeToLiveStatus": "ENABLED",
        "AttributeName": "ttl"
    }
}
```

Weitere Informationen finden Sie unter Time to Live im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie DescribeTimeToLivein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeTimeToLivein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK für JavaScript.

```
import { DynamoDBClient, DescribeTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const describeTTL = async (tableName, region) => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    try {
        const ttlDescription = await client.send(new
 DescribeTimeToLiveCommand({ TableName: tableName }));
```

```
        if (ttlDescription.TimeToLiveDescription.TimeToLiveStatus === 'ENABLED')
 {
            console.log("TTL is enabled for table %s.", tableName);
        } else {
            console.log("TTL is not enabled for table %s.", tableName);
        }

        return ttlDescription;
    } catch (e) {
        console.error(`Error describing table: ${e}`);
        throw e;
    }
}

// Example usage (commented out for testing)
// describeTTL('your-table-name', 'us-east-1');
```

- Einzelheiten zur API finden Sie DescribeTimeToLivein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK für Python (Boto3).

```
import boto3


def describe_ttl(table_name, region):
    """
    Describes TTL on an existing table, as well as a region.

    :param table_name: String representing the name of the table
    :param region: AWS Region of the table - example `us-east-1`
    :return: Time to live description.
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        ttl_description = dynamodb.describe_time_to_live(TableName=table_name)
```

```
        print(
            f"TimeToLive for table {table_name} is status
 {ttl_description['TimeToLiveDescription']['TimeToLiveStatus']}"
        )

        return ttl_description
    except Exception as e:
        print(f"Error describing table: {e}")
        raise


# Enter your own table name and AWS region
describe_ttl("your-table-name", "us-east-1")
```

- Einzelheiten zur API finden Sie DescribeTimeToLivein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ExecuteStatement`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteStatement` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Löschen von Daten mit PartiQL DELETE
- Einfügen von Daten mithilfe von PartiQL INSERT
- Abfragen einer Tabelle mit PartiQL
- Abfragen von Daten mit PartiQL SELECT
- Aktualisieren von Daten mit PartiQL UPDATE

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
 movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
 'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
 ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
```

```
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Verwenden einer SELECT-Anweisung, um ein Liste an Elementen abzurufen.

```
        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
```

```
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden einer DELETE-Anweisung, um einen einzelnen Film zu löschen.

```
        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;
```

```
        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
    //  3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);
```

```
        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
```

```
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

       request.SetStatement(sqlStream.str());

       Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

       request.SetParameters(attributes);

       Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
               request);

       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to update a movie: "
                     << outcome.GetError().GetMessage();
           return false;
       }
   }
```

Verwenden Sie eine DELETE-Anweisung, um ein Element zu löschen.

```
   // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
   {
       Aws::DynamoDB::Model::ExecuteStatementRequest request;
       std::stringstream sqlStream;
       sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       request.SetStatement(sqlStream.str());

       Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
       request.SetParameters(attributes);

       Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
```

```
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Funktionsempfängerstruktur für das Beispiel.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
```

```
}
```

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB
 table by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
```

```
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Verwenden Sie eine SELECT-Anweisung, um eine Liste der Elemente abzurufen und die Ergebnisse zu projizieren.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the
 DynamoDB table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
```

```
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie
 that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
```

```
 }
 return err
}
```

Verwenden Sie eine DELETE-Anweisung, um ein Element zu löschen.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the
 DynamoDB table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: `INSERT INTO Flowers value {'Name':?}`,
    Parameters: ["Rose"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Rufen Sie ein Element mithilfe von PartiQL ab.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";
```

```
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "SELECT * FROM CloudTypes WHERE IsStorm=?",
    Parameters: [false],
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Aktualisieren Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "UPDATE EyeColors SET IsRecessive=? where Color=?",
    Parameters: [true, "blue"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Löschen Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "DELETE FROM PaintColors where Name=?",
    Parameters: ["Purple"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
```

```
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
 exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- Einzelheiten zur API finden Sie ExecuteStatementin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wählen Sie ein einzelnes Element mithilfe von PartiQL aus.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
```

```
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Aktualisieren Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Fügen Sie ein einzelnes Element mithilfe von PartiQL hinzu.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
```

```
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, { 'plot': plot, 'rating': rating }]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Löschen Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **GetItem**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetItem` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</
param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
 newMovie, string tableName)
```

```
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
 item to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
```

```
}
query=""
while getopts "n:k:q:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    q) query="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

if [[ -n "$query" ]]; then
  response=$(aws dynamodb get-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --output text \
    --query "$query")
else
  response=$(
    aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text
```

```
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
```

```
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie GetItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get an item from an Amazon DynamoDB table.
/*!
```

```
  \sa getItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
 dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
 outcome.GetResult().GetItem();
        if (!item.empty()) {
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << "Values: " << i.first << ": " << i.second.GetS()
                          << std::endl;
        }
        else {
            std::cout << "No item found with the key " << partitionKey <<
 std::endl;
        }
    }
    else {
        std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
```

```
}
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So lesen Sie ein Element in einer Tabelle

Das folgende Beispiel für `get-item` ruft ein Element aus der `MusicCollection`-Tabelle ab. Die Tabelle verwendet einen Hash- und Bereichs-Primärschlüssel (`Artist` und `SongTitle`), daher müssen Sie diese Attribute beide angeben. Der Befehl fragt auch Informationen über die Lesekapazität ab, die von der Operation verbraucht wird.

```
aws dynamodb get-item \
    --table-name MusicCollection \
    --key file://key.json \
    --return-consumed-capacity TOTAL
```

Inhalt von `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Ausgabe:

```
{
    "Item": {
        "AlbumTitle": {
            "S": "Songs About Life"
        },
        "SongTitle": {
            "S": "Happy Day"
        },
        "Artist": {
            "S": "Acme Band"
        }
```

```
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5
    }
}
```

Weitere Informationen finden Sie unter Lesen eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So lesen Sie ein Element mit einem konsistenten Lesevorgang

Das folgende Beispiel ruft mithilfe von strikt konsistenten Lesevorgängen ein Element aus der `MusicCollection`-Tabelle ab.

```
aws dynamodb get-item \
    --table-name MusicCollection \
    --key file://key.json \
    --consistent-read \
    --return-consumed-capacity TOTAL
```

Inhalt von `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Ausgabe:

```
{
    "Item": {
        "AlbumTitle": {
            "S": "Songs About Life"
        },
        "SongTitle": {
            "S": "Happy Day"
        },
        "Artist": {
            "S": "Acme Band"
        }
    },
```

```
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
    }
}
```

Weitere Informationen finden Sie unter Lesen eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 3: So rufen Sie bestimmte Attribute eines Elements ab

Im folgenden Beispiel wird ein Projektionsausdruck verwendet, um nur drei Attribute des gewünschten Elements abzurufen.

```
aws dynamodb get-item \
    --table-name ProductCatalog \
    --key '{"Id": {"N": "102"}}' \
    --projection-expression "#T, #C, #P" \
    --expression-attribute-names file://names.json
```

Inhalt von `names.json`:

```
{
    "#T": "Title",
    "#C": "ProductCategory",
    "#P": "Price"
}
```

Ausgabe:

```
{
    "Item": {
        "Price": {
            "N": "20"
        },
        "Title": {
            "S": "Book 102 Title"
        },
        "ProductCategory": {
            "S": "Book"
        }
    }
```

```
}
```

Weitere Informationen finden Sie unter Lesen eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie GetItemin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// GetMovie gets movie data from the DynamoDB table by using the primary
 composite key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
```

```
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ruft mithilfe von ein Element aus einer Tabelle ab DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal>

                Where:
                    tableName - The Amazon DynamoDB table from which an item is
 retrieved (for example, Music3).\s
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            // If there is no matching item, GetItem does not return any data.
            Map<String, AttributeValue> returnedItem =
ddb.getItem(request).item();
            if (returnedItem.isEmpty())
                System.out.format("No item found with the key %s!\n", key);
            else {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");
                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie GetItemunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter GetCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new GetCommand({
    TableName: "AngryAnimals",
    Key: {
      CommonName: "Shoebill",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Element aus einer Tabelle ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "001" },
  },
  ProjectionExpression: "ATTRIBUTE_NAME",
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Rufen Sie ein Element mithilfe des DynamoDB-Dokument-Clients aus einer Tabelle ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "EPISODES_TABLE",
  Key: { KEY_NAME: VALUE },
};

docClient.get(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetItemin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
```

```
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- Einzelheiten zur API finden Sie GetItemin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist zurück.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- Einzelheiten zur API finden Sie unter GetItem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist zurück.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                            Value
----                            -----
Genre                           Country
SongTitle                       Somewhere Down The Road
Price                           1.94
Artist                          No One You Know
CriticRating                    9
AlbumTitle                      Somewhat Famous
```

- Einzelheiten zur API finden Sie unter GetItem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
```

```
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]
```

- Einzelheiten zur API finden Sie GetItemin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_item = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = it_key ).
        DATA(lt_attr) = oo_item->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( )
          && 'Movie year is: ' && lo_year->get_n( )
          && 'Moving rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetItemin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB
```

```
    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie GetItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListTables`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTables` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Richten Sie die Zugriffskontrolle ein Attribute-Based

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
private static async Task ListMyTables()
{
    Console.WriteLine("\n*** Listing tables ***");

    string lastTableNameEvaluated = null;
    do
    {
        var response = await Client.ListTablesAsync(new ListTablesRequest
        {
            Limit = 2,
            ExclusiveStartTableName = lastTableNameEvaluated
        });

        foreach (var name in response.TableNames)
        {
            Console.WriteLine(name);
        }

        lastTableNameEvaluated = response.LastEvaluatedTableName;
    } while (lastTableNameEvaluated != null);
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_list_tables
#
# This function lists all the tables in a DynamoDB.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function dynamodb_list_tables() {
  response=$(aws dynamodb list-tables \
    --output text \
    --query "TableNames")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  echo "$response" | tr -s "[:space:]" "\n"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the Amazon DynamoDB tables for the current AWS account.
/*!
  \sa listTables()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamoClient.ListTables(
                listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
```

```
                outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So listen Sie Tabellen auf

Das folgende `list-tables` Beispiel listet alle Tabellen auf, die dem AWS Girokonto und der Region zugeordnet sind.

```
aws dynamodb list-tables
```

Ausgabe:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog",
        "Reply",
        "Thread"
    ]
}
```

Weitere Informationen finden Sie unter Auflisten von Tabellennamen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So begrenzen Sie die Seitengröße

Das folgende Beispiel gibt eine Liste aller vorhandenen Tabellen zurück, ruft jedoch bei jedem Aufruf nur ein Element ab und führt gegebenenfalls mehrere Aufrufe durch, um die gesamte Liste abzurufen. Die Begrenzung der Seitengröße ist nützlich, wenn Listenbefehle für eine

große Anzahl von Ressourcen ausgeführt werden, was zu einem Timeout-Fehler führen kann, wenn die Standardseitengröße von 1 000 verwendet wird.

```
aws dynamodb list-tables \
    --page-size 1
```

Ausgabe:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog",
        "Reply",
        "Thread"
    ]
}
```

Weitere Informationen finden Sie unter Auflisten von Tabellennamen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 3: So begrenzen Sie die Anzahl der zurückgegebenen Elemente

Im folgenden Beispiel wird die Anzahl der zurückgegebenen Elemente auf 2 begrenzt. Die Antwort enthält einen `NextToken`-Wert, mit dem die nächste Ergebnisseite abgerufen werden kann.

```
aws dynamodb list-tables \
    --max-items 2
```

Ausgabe:

```
{
    "TableNames": [
        "Forum",
        "ProductCatalog"
    ],
    "NextToken":
 "abCDeFGhiJKlmnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9"
}
```

Weitere Informationen finden Sie unter Auflisten von Tabellennamen im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 4: So rufen Sie die nächste Ergebnisseite ab

Der folgende Befehl verwendet den `NextToken`-Wert eines vorherigen Aufrufs des `list-tables`-Befehls, um eine weitere Ergebnisseite abzurufen. Da die Antwort in diesem Fall keinen `NextToken`-Wert enthält, wissen wir, dass wir das Ende der Ergebnisse erreicht haben.

```
aws dynamodb list-tables \
    --starting-
token abCDeFGhiJKlmnOPqrSTuvwxYZ1aBCdEFghijK7LM51nOpqRSTuv3WxY3ZabC5dEFGhI2Jk3LmnoPQ6RST9
```

Ausgabe:

```
{
    "TableNames": [
        "Reply",
        "Thread"
    ]
}
```

Weitere Informationen finden Sie unter Auflisten von Tabellennamen im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie ListTablesunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
```

```
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
   tableNames = append(tableNames, output.TableNames...)
  }
 }
 return tableNames, err
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
```

```
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request =
 ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
```

```
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

// Call DynamoDB to retrieve the list of tables
ddb.listTables({ Limit: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
            'ExclusiveStartTableName' => $exclusiveStartTableName,
            'Limit' => $limit,
        ]);
    }
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details aller Tabellen zurück und iteriert automatisch, bis der Service anzeigt, dass keine weiteren Tabellen vorhanden sind.

```
Get-DDBTableList
```

- Einzelheiten zur API finden Sie unter ListTables AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details aller Tabellen zurück und iteriert automatisch, bis der Service anzeigt, dass keine weiteren Tabellen vorhanden sind.

```
Get-DDBTableList
```

- Einzelheiten zur API finden Sie unter ListTables AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
```

```
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables
```

- Einzelheiten zur API finden Sie ListTablesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Finden Sie heraus, ob eine Tabelle vorhanden ist.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_tables(client: &Client) -> Result<Vec<String>, Error> {
    let paginator = client.list_tables().into_paginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;

    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(table_names)
}
```

Finden Sie heraus, ob eine Tabelle vorhanden ist.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;

    match table_list {
        Ok(list) => Ok(list.table_names().contains(&table.into())),
        Err(e) => Err(e.into()),
    }
}
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_dyn->listtables( ).
        " You can loop over the oo_result to get table properties like this.
        LOOP AT oo_result->get_tablenames( ) INTO DATA(lo_table_name).
          DATA(lv_tablename) = lo_table_name->get_value( ).
        ENDLOOP.
        DATA(lv_tablecount) = lines( oo_result->get_tablenames( ) ).
        MESSAGE 'Found ' && lv_tablecount && ' tables' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Get a list of the DynamoDB tables available in the specified Region.
    ///
    /// - Returns: An array of strings listing all of the tables available
    ///   in the Region specified when the session was created.
    public func getTableList() async throws -> [String] {
        let input = ListTablesInput(
        )
        return try await session.listTables(input: input)
    }
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`PutItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutItem` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Erweiterte Szenarien für globale sekundäre Indizes
- Erstellen eines Elements mit einer TTL
- Erstellen und Verwalten globaler Tabellen zur Demonstration von MREC
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Verwenden von bedingten Operationen
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)
- Arbeiten Sie mit Streams und Time-to-Live

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Adds a new item to the table.
/// </summary>
/// <param name="newMovie">A Movie object containing informtation for
/// the movie to add to the table.</param>
```

```
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
 item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
 {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }

  while getopts "n:i:h" option; do
    case "${option}" in
```

```
    n) table_name="${OPTARG}" ;;
    i) item="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$item" ]]; then
  errecho "ERROR: You must provide an item with the -i parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    item:   $item"
iecho ""
iecho ""

response=$(aws dynamodb put-item \
  --table-name "$table_name" \
  --item file://"$item")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports put-item operation failed.$response"
  return 1
fi
```

```
  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
```

```
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Put an item in an Amazon DynamoDB table.
/*!
  \sa putItem()
  \param tableName: The table name.
  \param artistKey: The artist key. This is the partition key for the table.
  \param artistValue: The artist value.
```

```
  \param albumTitleKey: The album title key.
  \param albumTitleValue: The album title value.
  \param awardsKey: The awards key.
  \param awardsValue: The awards value.
  \param songTitleKey: The song title key.
  \param songTitleValue: The song title value.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
                               const Aws::String &awardsKey,
                               const Aws::String &awardsValue,
                               const Aws::String &songTitleKey,
                               const Aws::String &songTitleValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            albumTitleValue));
    putItemRequest.AddItem(awardsKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
```

```
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
```

```
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So fügen Sie ein Element zu einer Tabelle hinzu

Das folgende `put-item` Beispiel fügt der MusicCollectionTabelle ein neues Element hinzu.

```
aws dynamodb put-item \
    --table-name MusicCollection \
    --item file://item.json \
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Inhalt von `item.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"},
    "AlbumTitle": {"S": "Greatest Hits"}
}
```

Ausgabe:

```
{
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
```

```
    },
    "ItemCollectionMetrics": {
        "ItemCollectionKey": {
            "Artist": {
                "S": "No One You Know"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So überschreiben Sie ein Element in einer Tabelle bedingungsabhängig

Im folgenden Beispiel für `put-item` wird ein vorhandenes Element in der `MusicCollection`-Tabelle nur dann überschrieben, wenn dieses vorhandene Element ein `AlbumTitle`-Attribut mit dem Wert `Greatest Hits` hat. Der Befehl gibt den vorherigen Wert des Elements zurück.

```
aws dynamodb put-item \
    --table-name MusicCollection \
    --item file://item.json \
    --condition-expression "#A = :A" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-values ALL_OLD
```

Inhalt von `item.json`:

```
{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"},
    "AlbumTitle": {"S": "Somewhat Famous"}
}
```

Inhalt von `names.json`:

```
{
    "#A": "AlbumTitle"
}
```

Inhalt von `values.json`:

```
{
    ":A": {"S": "Greatest Hits"}
}
```

Ausgabe:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Greatest Hits"
        },
        "Artist": {
            "S": "No One You Know"
        },
        "SongTitle": {
            "S": "Call Me Today"
        }
    }
}
```

Wenn der Schlüssel bereits existiert, sollten Sie die folgende Ausgabe sehen:

```
A client error (ConditionalCheckFailedException) occurred when calling the
 PutItem operation: The conditional request failed.
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie PutItemunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
```

```
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
```

```
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügt ein Element in eine Tabelle ein mit DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval>
 <awards> <awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is
 placed (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
 About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example,
 Happy Day).
                **Warning** This program will  place an item that you specify
 into a table!
                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];
        String awards = args[5];
        String awardVal = args[6];
        String songTitle = args[7];
        String songTitleVal = args[8];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
                songTitleVal);
        System.out.println("Done!");
        ddb.close();
    }

    public static void putItemInTable(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String albumTitle,
            String albumTitleValue,
            String awards,
            String awardVal,
            String songTitle,
            String songTitleVal) {

        HashMap<String, AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle,
AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();
```

```
        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The
 request id is "
                    + response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its
 name correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutItemunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter PutCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { PutCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);
```

```
export const main = async () => {
  const command = new PutCommand({
    TableName: "HappyAnimals",
    Item: {
      CommonName: "Shiba Inu",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Element in eine Tabelle ein.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "CUSTOMER_LIST",
  Item: {
    CUSTOMER_ID: { N: "001" },
    CUSTOMER_NAME: { S: "Richard Roe" },
  },
};

// Call DynamoDB to add the item to the table
```

```
ddb.putItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Fügen Sie ein Element mithilfe des DynamoDB-Dokument-Clients in eine Tabelle ein.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Item: {
    HASHKEY: VALUE,
    ATTRIBUTE_1: "STRING_VALUE",
    ATTRIBUTE_2: VALUE_2,
  },
};

docClient.put(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutItemin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- Einzelheiten zur API finden Sie PutItemin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
    }

    $service->putItem([
        'Item' => [
            'year' => [
                'N' => "$movieYear",
            ],
            'title' => [
                'S' => $movieName,
            ],
        ],
        'TableName' => $tableName,
    ]);

public function putItem(array $array)
{
    $this->dynamoDbClient->putItem($array);
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes Element durch ein neues.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Einzelheiten zur API finden Sie unter PutItem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes Element durch ein neues.

```
$item = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- Einzelheiten zur API finden Sie unter PutItem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie PutItemin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        'year' => movie[:year],
        'title' => movie[:title],
        'info' => { 'plot' => movie[:plot], 'rating' => movie[:rating] }
      }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn add_item(client: &Client, item: Item, table: &String) ->
 Result<ItemOut, Error> {
```

```
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");

    let resp = request.send().await?;

    let attributes = resp.attributes().unwrap();

    let username = attributes.get("username").cloned();
    let first_name = attributes.get("first_name").cloned();
    let last_name = attributes.get("last_name").cloned();
    let age = attributes.get("age").cloned();
    let p_type = attributes.get("p_type").cloned();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        username, first_name, last_name, age, p_type
    );

    Ok(ItemOut {
        p_type,
        age,
        username,
        first_name,
        last_name,
    })
}
```

- Einzelheiten zur API finden Sie PutItemin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_resp) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = it_item ).
        MESSAGE '1 row inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutItemin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB
```

```
    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
```

```
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
```

- Einzelheiten zur API finden Sie PutItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`Query`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Query` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Erweiterte Szenarien für globale sekundäre Indizes
- Vergleichen mehrerer Werte mit einem einzigen Attribut
- Erste Schritte mit NoSQL-Datenbanken
- Verwalten globaler sekundärer Indizes
- Ausführen erweiterter Abfrageoperationen
- Abfragen einer Tabelle mithilfe einer „begins_with“-Bedingung
- Abfragen einer Tabelle mit einem Datumsbereich
- Abfragen einer Tabelle mit einem globalen sekundären Index

- Abfragen einer Tabelle mit einem komplexen Filterausdruck
- Abfragen einer Tabelle mit einem dynamischen Filterausdruck
- Abfragen einer Tabelle mit einem Filterausdruck und einem Limit
- Abfragen einer Tabelle mit verschachtelten Attributen
- Abfragen einer Tabelle mit Paginierung
- Abfragen einer Tabelle mit strikt konsistenten Lesevorgängen
- Abfragen nach TTL-Elementen
- Abfragen von Tabellen mithilfe von Datums- und Uhrzeitmustern
- Verwenden von Ausdrucksattributnamen
- Arbeiten mit lokalen sekundären Indizes

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();
```

```
            var filter = new QueryFilter("year", QueryOperator.Equal, year);

            Console.WriteLine("\nFind movies released in: {year}:");

            var config = new QueryOperationConfig()
            {
                Limit = 10, // 10 items per page.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
                {
                    "title",
                    "year",
                },
                ConsistentRead = true,
                Filter = filter,
            };

            // Value used to track how many movies match the
            // supplied criteria.
            var moviesFound = 0;

            var search = movieTable.Query(config);
            do
            {
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
```

```
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
 movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
 {ex.Message}");
            throw;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für .NET -API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
```

```
#  And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute
 names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
 parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS CLI -Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Perform a query on an Amazon DynamoDB Table and retrieve items.
/*!
  \sa queryItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param projectionExpression: The projections expression, which is ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
```

```
 * The partition key attribute is searched with the specified value. By default,
 all fields and values
 * contained in the item are returned. If an optional projection expression is
 * specified on the command line, only the specified fields and values are
 * returned.
 */

bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &projectionExpression,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);

    if (!projectionExpression.empty()) {
        request.SetProjectionExpression(projectionExpression);
    }

    // Set query key condition expression.
    request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

    // Set Expression AttributeValues.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch", partitionValue);

    request.SetExpressionAttributeValues(attributeValues);

    bool result = true;

    // "exclusiveStartKey" is used for pagination.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 exclusiveStartKey;
    do {
        if (!exclusiveStartKey.empty()) {
            request.SetExclusiveStartKey(exclusiveStartKey);
            exclusiveStartKey.clear();
        }
        // Perform Query operation.
        const Aws::DynamoDB::Model::QueryOutcome &outcome =
 dynamoClient.Query(request);
```

```
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "Number of items retrieved from Query: " <<
 items.size()
                          << std::endl;
                // Iterate each item and print.
                for (const auto &item: items) {
                    std::cout
                            <<
 "*****************************************************"
                            << std::endl;
                    // Output each retrieved field and its value.
                    for (const auto &i: item)
                        std::cout << i.first << ": " << i.second.GetS() <<
 std::endl;
                }
            }
            else {
                std::cout << "No item found in table: " << tableName <<
 std::endl;
            }

            exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Query items: " <<
 outcome.GetError().GetMessage();
            result = false;
            break;
        }
    } while (!exclusiveStartKey.empty());

    return result;
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So fragen Sie eine Tabelle ab

Im folgenden Beispiel für `query` werden Elemente in der Tabelle `MusicCollection` abgefragt. Die Tabelle verwendet einen Hash-und-Bereichs-Primärschlüssel (`Artist` und `SongTitle`), aber in dieser Abfrage wird nur der Hash-Schlüsselwert angegeben. Es werden Songtitel des Künstlers mit dem Namen „No One You Know“ zurückgegeben.

```
aws dynamodb query \
    --table-name MusicCollection \
    --projection-expression "SongTitle" \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --return-consumed-capacity TOTAL
```

Inhalt von `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Ausgabe:

```
{
    "Items": [
        {
            "SongTitle": {
                "S": "Call Me Today"
            },
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
```

```
        "CapacityUnits": 0.5
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Abfragen in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So fragen Sie eine Tabelle mit strikt konsistenten Lesevorgängen ab und durchlaufen den Index in absteigender Reihenfolge

Im folgenden Beispiel wird dieselbe Abfrage wie im ersten Beispiel ausgeführt, die Ergebnisse werden jedoch in umgekehrter Reihenfolge zurückgegeben und es werden strikt konsistente Lesevorgänge verwendet.

```
aws dynamodb query \
    --table-name MusicCollection \
    --projection-expression "SongTitle" \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --consistent-read \
    --no-scan-index-forward \
    --return-consumed-capacity TOTAL
```

Inhalt von `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Ausgabe:

```
{
    "Items": [
        {
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        },
        {
            "SongTitle": {
                "S": "Call Me Today"
```

```
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 1.0
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Abfragen in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 3: So filtern Sie bestimmte Ergebnisse heraus

Im folgende Beispiel wird `MusicCollection` abgefragt, Ergebnisse mit bestimmten Werten im `AlbumTitle`-Attribut werden jedoch ausgeschlossen. Beachten Sie, dass sich dies nicht auf `ScannedCount` oder `ConsumedCapacity` auswirkt, da der Filter angewendet wird, nachdem die Elemente gelesen wurden.

```
aws dynamodb query \
    --table-name MusicCollection \
    --key-condition-expression "#n1 = :v1" \
    --filter-expression "NOT (#n2 IN (:v2, :v3))" \
    --expression-attribute-names file://names.json \
    --expression-attribute-values file://values.json \
    --return-consumed-capacity TOTAL
```

Inhalt von `values.json`:

```
{
    ":v1": {"S": "No One You Know"},
    ":v2": {"S": "Blue Sky Blues"},
    ":v3": {"S": "Greatest Hits"}
}
```

Inhalt von `names.json`:

```
{
    "#n1": "Artist",
```

```
    "#n2": "AlbumTitle"
}
```

Ausgabe:

```
{
    "Items": [
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Call Me Today"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Abfragen in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 4: So rufen Sie nur die Elementanzahl ab

Das folgende Beispiel ruft eine Anzahl von Elementen ab, die der Abfrage entsprechen, ruft jedoch keines der Elemente selbst ab.

```
aws dynamodb query \
    --table-name MusicCollection \
    --select COUNT \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json
```

Inhalt von `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Ausgabe:

```
{
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

Weitere Informationen finden Sie unter Arbeiten mit Abfragen in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 5: So rufen Sie einen Index ab

Im folgenden Beispiel wird der lokale sekundäre Index `AlbumTitleIndex` abgefragt. Die Abfrage gibt alle Attribute aus der Basistabelle zurück, die in den lokalen sekundären Index projiziert wurden. Beachten Sie, dass Sie bei der Abfrage eines lokalen sekundären Index oder eines globalen sekundären Index auch den Namen der Basistabelle mit dem `table-name`-Parameter angeben müssen.

```
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumTitleIndex \
    --key-condition-expression "Artist = :v1" \
    --expression-attribute-values file://expression-attributes.json \
    --select ALL_PROJECTED_ATTRIBUTES \
    --return-consumed-capacity INDEXES
```

Inhalt von `expression-attributes.json`:

```
{
    ":v1": {"S": "No One You Know"}
}
```

Ausgabe:

```
{
```

```
    "Items": [
        {
            "AlbumTitle": {
                "S": "Blue Sky Blues"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Scared of My Shadow"
            }
        },
        {
            "AlbumTitle": {
                "S": "Somewhat Famous"
            },
            "Artist": {
                "S": "No One You Know"
            },
            "SongTitle": {
                "S": "Call Me Today"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 0.5,
        "Table": {
            "CapacityUnits": 0.0
        },
        "LocalSecondaryIndexes": {
            "AlbumTitleIndex": {
                "CapacityUnits": 0.5
            }
        }
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Abfragen in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

- Weitere API-Informationen finden Sie unter Abfragen in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Query gets all movies in the DynamoDB table that were released in the
 specified year.
// The function uses the `expression` package to build the key condition
 expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
```

```
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
```

```
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Weitere API-Informationen finden Sie unter Query in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fragt eine Tabelle ab mithilfe von DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
                    tableName - The Amazon DynamoDB table to put the item in (for
 example, Music3).
                    partitionKeyName - The partition key name of the Amazon
 DynamoDB table (for example, Artist).
                    partitionKeyVal - The value of the partition key that should
 match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
 partitionAlias);
        System.out.println("There were " + count + "  record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
 partitionKeyName, String partitionKeyVal,
            String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved
 word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
```

```
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
                .s(partitionKeyVal)
                .build());

        QueryRequest queryReq = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(partitionAlias + " = :" +
 partitionKeyName)
                .expressionAttributeNames(attrNameAlias)
                .expressionAttributeValues(attrValues)
                .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}
```

Fragt eine Tabelle mithilfe von `DynamoDbClient` und eines sekundären Index ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * **year-index**. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
            expressionAttributeValues.put(":yearValue",
 AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                    .tableName(tableName)
                    .indexName("year-index")
                    .keyConditionExpression("#year = :yearValue")
                    .expressionAttributeNames(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
```

```
            response.items()
                    .forEach(movie ->
 System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter QueryCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { QueryCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new QueryCommand({
    TableName: "CoffeeCrop",
    KeyConditionExpression:
      "OriginCountry = :originCountry AND RoastDate > :roastDate",
    ExpressionAttributeValues: {
      ":originCountry": "Ethiopia",
```

```
      ":roastDate": "2023-05-01",
    },
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  ExpressionAttributeValues: {
    ":s": 2,
    ":e": 9,
    ":topic": "PHRASE",
  },
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  TableName: "EPISODES_TABLE",
};
```

```
docClient.query(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Items);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
```

```
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der API–Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);

    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
```

```
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für PHP -API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft eine Abfrage auf, die DynamoDB-Elemente mit dem angegebenen SongTitle Wert und Artist zurückgibt.

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
```

```
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft eine Abfrage auf, die DynamoDB-Elemente mit dem angegebenen SongTitle Wert und Artist zurückgibt.

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fragen Sie Elemente mithilfe eines Schlüsselbedingungsausdrucks ab.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
 self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]
```

Fragen Sie Elemente ab und projizieren Sie sie, um eine Teilmenge von Daten zurückzugeben.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def query_and_project_movies(self, year, title_bounds):
        """
        Query for movies that were released in a specified year and that have
 titles
        that start within a range of letters. A projection expression is used
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
```

```
        """
        try:
            response = self.table.query(
                ProjectionExpression="#yr, title, info.genres, info.actors[0]",
                ExpressionAttributeNames={"#yr": "year"},
                KeyConditionExpression=(
                    Key("year").eq(year)
                    & Key("title").between(
                        title_bounds["first"], title_bounds["second"]
                    )
                ),
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                logger.warning(
                    "There's a validation error. Here's the message: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't query for movies. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return response["Items"]
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: { ':year' => year }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Weitere API-Informationen finden Sie unter Query in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Finden Sie die Filme, die im angegebenen Jahr gedreht wurden.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy",
 AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Query movies for a given year .
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_year }| ) ) ).
        DATA(lt_key_conditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |EQ|
          ) ) ) ).
        oo_result = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_key_conditions ).
        DATA(lt_items) = oo_result->get_items( ).
        "You can loop over the results to get item attributes.
        LOOP AT lt_items INTO DATA(lt_item).
          DATA(lo_title) = lt_item[ key = 'title' ]-value.
          DATA(lo_year) = lt_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_result->get_count( ).
        MESSAGE 'Item count is: ' && lv_count TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz für das AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
 "QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
```

```
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an
array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }
```

- Detaillierte API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`Scan`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Scan` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Beschleunigen von Lesevorgängen mit DAX
- Vergleichen mehrerer Werte mit einem einzigen Attribut

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</
returns>
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string,
 AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0],
 info.directors, info.running_time_secs",
```

```
                Limit = 10 // Set a limit to demonstrate using the
LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für .NET -API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
```

```
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
```

```
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
###############################################################################
```

```
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS CLI -Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Scan an Amazon DynamoDB table.
/*!
  \sa scanTable()
  \param tableName: Name for the DynamoDB table.
  \param projectionExpression: An optional projection expression, ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                 const Aws::String &projectionExpression,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
 all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
```

```
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
 dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            all_items.insert(all_items.end(), items.begin(), items.end());

            last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Scan items: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!last_evaluated_key.empty());

    if (!all_items.empty()) {
        std::cout << "Number of items retrieved from scan: " << all_items.size()
                  << std::endl;
        // Iterate each item and print.
        for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 &itemMap: all_items) {
            std::cout << "*****************************************************"
                      << std::endl;
            // Output each retrieved field and its value.
            for (const auto &itemEntry: itemMap)
                std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                          << std::endl;
        }
    }

    else {
        std::cout << "No items found in table: " << tableName << std::endl;
    }

    return true;
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So scannen Sie eine Tabelle

Im folgenden Beispiel für `scan` wird die gesamte `MusicCollection`-Tabelle gescannt und die Ergebnisse werden dann auf Songs des Künstlers „No One You Know“ eingegrenzt. Für jedes Element werden nur der Albumtitel und der Songtitel zurückgegeben.

```
aws dynamodb scan \
    --table-name MusicCollection \
    --filter-expression "Artist = :a" \
    --projection-expression "#ST, #AT" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json
```

Inhalt von `expression-attribute-names.json`:

```
{
    "#ST": "SongTitle",
    "#AT":"AlbumTitle"
}
```

Inhalt von `expression-attribute-values.json`:

```
{
    ":a": {"S": "No One You Know"}
}
```

Ausgabe:

```
{
    "Count": 2,
    "Items": [
        {
            "SongTitle": {
                "S": "Call Me Today"
            },
```

```
            "AlbumTitle": {
                "S": "Somewhat Famous"
            }
        },
        {
            "SongTitle": {
                "S": "Scared of My Shadow"
            },
            "AlbumTitle": {
                "S": "Blue Sky Blues"
            }
        }
    ],
    "ScannedCount": 3,
    "ConsumedCapacity": null
}
```

Weitere Informationen finden Sie unter Arbeiten mit Scans in DynamoDB im Entwicklerhandbuch zu Amazon DynamoDB.

- Weitere API-Informationen finden Sie unter Scan in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
```

```
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Scan gets all movies in the DynamoDB table that were released in a range of
 years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"),
 expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:         expr.Filter(),
   ProjectionExpression:     expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
```

```
    response, err = scanPaginator.NextPage(ctx)
    if err != nil {
     log.Printf("Couldn't scan for movies released between %v and %v. Here's why:
 %v\n",
      startYear, endYear, err)
     break
    } else {
     var moviePage []Movie
     err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
     if err != nil {
      log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
      break
     } else {
      movies = append(movies, moviePage...)
     }
    }
   }
  }
  return movies, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
```

```
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Scannt eine Amazon DynamoDB-Tabelle mit. DynamoDbClient

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information from
 (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
```

```
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                    .tableName(tableName)
                    .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }

        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter ScanCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, ScanCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ScanCommand({
    ProjectionExpression: "#Name, Color, AvgLifeSpan",
    ExpressionAttributeNames: { "#Name": "Name" },
    TableName: "Birds",
  });

  const response = await docClient.send(command);
  for (const bird of response.Items) {
    console.log(`${bird.Name} - (${bird.Color}, ${bird.AvgLifeSpan})`);
  }
  return response;
};
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für JavaScript -API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
  // Specify which items in the results are returned.
  FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
  // Define the expression attribute value, which are substitutes for the values
 you want to compare.
  ExpressionAttributeValues: {
    ":topic": { S: "SubTitle2" },
    ":s": { N: 1 },
    ":e": { N: 2 },
  },
  // Set the projection expression, which are the attributes that you want.
  ProjectionExpression: "Season, Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
    data.Items.forEach(function (element, index, array) {
      console.log(
        "printing",
        element.Title.S + " (" + element.Subtitle.S + ")"
      );
    });
```

```
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    $yearsKey = [
        'Key' => [
            'year' => [
                'N' => [
                    'minRange' => 1990,
                    'maxRange' => 1999,
                ],
            ],
        ],
    ];
    $filter = "year between 1990 and 1999";
    echo "\nHere's a list of all the movies released in the 90s:\n";
    $result = $service->scan($tableName, $yearsKey, $filter);
    foreach ($result['Items'] as $movie) {
        $movie = $marshal->unmarshalItem($movie);
        echo $movie['title'] . "\n";
    }

public function scan(string $tableName, array $key, string $filters)
{
    $query = [
        'ExpressionAttributeNames' => ['#year' => 'year'],
        'ExpressionAttributeValues' => [
            ":min" => ['N' => '1990'],
            ":max" => ['N' => '1999'],
        ],
        'FilterExpression' => "#year between :min and :max",
        'TableName' => $tableName,
    ];
    return $this->dynamoDbClient->scan($query);
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für PHP -API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle Elemente in der Tabelle „Music“ zurück.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
Genre                          Country
Artist                         No One You Know
Price                          1.98
CriticRating                   8.4
SongTitle                      My Dog Spot
AlbumTitle                     Hey Now
```

Beispiel 2: Gibt Elemente in der Tabelle Musik zurück, deren Wert CriticRating größer oder gleich neun ist.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-
DDBItem
```

Ausgabe:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Scan in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: Gibt alle Elemente in der Tabelle „Music“ zurück.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
Genre                           Country
Artist                          No One You Know
Price                           1.98
CriticRating                    8.4
SongTitle                       My Dog Spot
AlbumTitle                      Hey Now
```

Beispiel 2: Gibt Elemente in der Tabelle Musik zurück, deren Wert CriticRating größer oder gleich neun ist.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
```

```
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-
DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Scan in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
```

```
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
```

```
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
```

```
# @return [Array] The list of movies released in the specified years.
def scan_items(year_range)
  movies = []
  scan_hash = {
    filter_expression: '#yr between :start_yr and :end_yr',
    projection_expression: '#yr, title, info.rating',
    expression_attribute_names: { '#yr' => 'year' },
    expression_attribute_values: {
      ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
    }
  }
  done = false
  start_key = nil
  until done
    scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
    response = @table.scan(scan_hash)
    movies.concat(response.items) unless response.items.empty?
    start_key = response.last_evaluated_key
    done = start_key.nil?
  end
rescue Aws::DynamoDB::Errors::ServiceError => e
  puts("Couldn't scan for movies. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  movies
end
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_items(client: &Client, table: &str, page_size: Option<i32>) ->
 Result<(), Error> {
    let page_size = page_size.unwrap_or(10);
    let items: Result<Vec<_>, _> = client
        .scan()
        .table_name(table)
        .limit(page_size)
        .into_paginator()
        .items()
        .send()
        .collect()
        .await;

    println!("Items in table (up to {page_size}):");
    for item in items? {
        println!("   {:?}", item);
    }

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Scan movies for rating greater than or equal to the rating specified
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_rating }| ) ) ).
```

```
        DATA(lt_filter_conditions) = VALUE /aws1/
cl_dyncondition=>tt_filterconditionmap(
          ( VALUE /aws1/cl_dyncondition=>ts_filterconditionmap_maprow(
          key = 'rating'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |GE|
          ) ) ) ).
        oo_scan_result = lo_dyn->scan( iv_tablename = iv_table_name
          it_scanfilter = lt_filter_conditions ).
        DATA(lt_items) = oo_scan_result->get_items( ).
        LOOP AT lt_items INTO DATA(lo_item).
          " You can loop over to get individual attributes.
          DATA(lo_title) = lo_item[ key = 'title' ]-value.
          DATA(lo_year) = lo_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_scan_result->get_count( ).
        MESSAGE 'Found ' && lv_count && ' items' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz für das AWS -SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
```

```
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]?
= nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y`
instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
```

```
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }
```

- Detaillierte API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Swift.

Verwenden Sie **`UpdateItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateItem` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Bedingtes Aktualisieren der TTL eines Elements
- Zählen von Ausdrucksoperatoren
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Erste Schritte mit NoSQL-Datenbanken
- Ausführen von Listenoperationen
- Ausführen von Map-Operationen
- Ausführen von Set-Operationen
- Verstehen der Reihenfolge der Aktualisierungsausdrücke

- Aktualisieren der TTL eines Elements
- Verwenden von Operationen mit unteilbaren Zählern
- Verwenden von bedingten Operationen
- Verwenden von Ausdrucksattributnamen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
```

```
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item
 to update.
#       -e update expression  -- An expression that defines one or more
 attributes to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
```

```
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the
item to update."
    echo " -e update expression  -- An expression that defines one or more
attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }

  while getopts "n:k:e:v:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      e) update_expression="${OPTARG}" ;;
      v) values="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi
  if [[ -z "$update_expression" ]]; then
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
```

```
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-
help-return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an Amazon DynamoDB table item.
/*!
  \sa updateItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param attributeKey: The key for the attribute to be updated.
  \param attributeValue: The value for the attribute to be updated.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
 *  The example code only sets/updates an attribute value. It processes
 *  the attribute value as a string, even if the value could be interpreted
 *  as a number. Also, the example code does not remove an existing attribute
 *  from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &attributeKey,
                                  const Aws::String &attributeValue,
```

```
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);

    // Update the item.
    const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
 dynamoClient.UpdateItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item was updated" << std::endl;
    } else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
```

```
    return false;
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie ein Element in einer Tabelle

Im folgenden Beispiel für `update-item` wird ein Element in der Tabelle `MusicCollection` aktualisiert. Es wird ein neues Attribut (`Year`) hinzugefügt und das `AlbumTitle`-Attribut geändert. Alle Attribute im Element, wie sie nach dem Update erscheinen, werden in der Antwort zurückgegeben.

```
aws dynamodb update-item \
    --table-name MusicCollection \
    --key file://key.json \
    --update-expression "SET #Y = :y, #AT = :t" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json  \
    --return-values ALL_NEW \
    --return-consumed-capacity TOTAL \
    --return-item-collection-metrics SIZE
```

Inhalt von `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Inhalt von `expression-attribute-names.json`:

```
{
    "#Y":"Year", "#AT":"AlbumTitle"
}
```

Inhalt von `expression-attribute-values.json`:

```
{
    ":y":{"N": "2015"},
    ":t":{"S": "Louder Than Ever"}
}
```

Ausgabe:

```
{
    "Attributes": {
        "AlbumTitle": {
            "S": "Louder Than Ever"
        },
        "Awards": {
            "N": "10"
        },
        "Artist": {
            "S": "Acme Band"
        },
        "Year": {
            "N": "2015"
        },
        "SongTitle": {
            "S": "Happy Day"
        }
    },
    "ConsumedCapacity": {
        "TableName": "MusicCollection",
        "CapacityUnits": 3.0
    },
    "ItemCollectionMetrics": {
        "ItemCollectionKey": {
            "Artist": {
                "S": "Acme Band"
            }
        },
        "SizeEstimateRangeGB": [
            0.0,
            1.0
        ]
    }
}
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So aktualisieren Sie ein Element bedingungsabhängig

Im folgenden Beispiel wird ein Element in der `MusicCollection`-Tabelle aktualisiert, jedoch nur, wenn das vorhandene Element noch kein `Year`-Attribut besitzt.

```
aws dynamodb update-item \
    --table-name MusicCollection \
    --key file://key.json \
    --update-expression "SET #Y = :y, #AT = :t" \
    --expression-attribute-names file://expression-attribute-names.json \
    --expression-attribute-values file://expression-attribute-values.json  \
    --condition-expression "attribute_not_exists(#Y)"
```

Inhalt von `key.json`:

```
{
    "Artist": {"S": "Acme Band"},
    "SongTitle": {"S": "Happy Day"}
}
```

Inhalt von `expression-attribute-names.json`:

```
{
    "#Y":"Year",
    "#AT":"AlbumTitle"
}
```

Inhalt von `expression-attribute-values.json`:

```
{
    ":y":{"N": "2015"},
    ":t":{"S": "Louder Than Ever"}
}
```

Wenn das Element bereits über ein `Year`-Attribut verfügt, gibt DynamoDB die folgende Ausgabe zurück.

```
An error occurred (ConditionalCheckFailedException) when calling the UpdateItem
 operation: The conditional request failed
```

Weitere Informationen finden Sie unter Schreiben eines Elements im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie UpdateItemin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified
 table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the
 update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx,
 &dynamodb.UpdateItemInput{
   TableName:                aws.String(basics.TableName),
   Key:                      movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   UpdateExpression:         expr.Update(),
   ReturnValues:             types.ReturnValueUpdatedNew,
  })
  if err != nil {
   log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
  } else {
   err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
   if err != nil {
    log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
   }
  }
 }
 return attributeMap, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
```

```
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisiert ein Element in einer Tabelle mit DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <name> <updateVal>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music3).
                    key - The name of the key in the table (for example, Artist).
                    keyVal - The value of the key (for example, Famous Band).
                    name - The name of the column where the value is updated (for
example, Awards).
                    updateVal - The value used to update an item (for example,
14).
                 Example:
                    UpdateItem Music3 Artist Famous Band Awards 14
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
            String name,
            String updateVal) {
```

```
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
}
```

- Einzelheiten zur API finden Sie UpdateItemunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. Einzelheiten zur API finden Sie unter UpdateCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, UpdateCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new UpdateCommand({
    TableName: "Dogs",
    Key: {
      Breed: "Labrador",
    },
    UpdateExpression: "set Color = :color",
    ExpressionAttributeValues: {
      ":color": "black",
    },
    ReturnValues: "ALL_NEW",
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
```

```
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- Einzelheiten zur API finden Sie UpdateItemin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?
\n";
        $rating = 0;
```

```
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
 $rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Setzt das Genre-Attribut auf 'Rap' für das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
```

```
$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Ausgabe:

```
Name                            Value
----                            -----
Genre                           Rap
```

- Einzelheiten zur API finden Sie unter UpdateItem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Setzt das Genre-Attribut auf 'Rap' für das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Ausgabe:

```
Name                            Value
```

```
----                            -----
Genre                           Rap
```

- Einzelheiten zur API finden Sie unter UpdateItem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie ein Element mithilfe eines Aktualisierungsausdrucks.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
```

```
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p":
plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Aktualisieren Sie ein Element mithilfe eines Aktualisierungsausdrucks, der eine arithmetische Operation enthält.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic
 operation,
        you can adjust a value in a single request, rather than first getting its
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the
 movie.
        :return: The updated rating.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={":val": Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Aktualisieren Sie ein Element nur, wenn es bestimmte Bedingungen erfüllt.

```
class UpdateQueryWrapper:
```

```
    def __init__(self, table):
        self.table = table


    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is
greater
        than a specified threshold. If the movie does not list more than the
threshold,
        no actors are removed.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param actor_threshold: The threshold of actors to check.
        :return: The movie data after the update.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="remove info.actors[0]",
                ConditionExpression="size(info.actors) > :num",
                ExpressionAttributeValues={":num": actor_threshold},
                ReturnValues="ALL_NEW",
            )
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"ConditionalCheckFailedException":
                logger.warning(
                    "Didn't update %s because it has fewer than %s actors.",
                    title,
                    actor_threshold + 1,
                )
            else:
                logger.error(
                    "Couldn't update movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["Attributes"]
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)
    response = @table.update_item(
      key: { 'year' => movie[:year], 'title' => movie[:title] },
      update_expression: 'set info.rating=:r',
      expression_attribute_values: { ':r' => movie[:rating] },
      return_values: 'UPDATED_NEW'
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
 #{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.attributes
  end
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_output = lo_dyn->updateitem(
         iv_tablename        = iv_table_name
         it_key              = it_item_key
         it_attributeupdates = it_attribute_updates ).
       MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
     CATCH /aws1/cx_dyncondalcheckfaile00.
       MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
     CATCH /aws1/cx_dynresourcenotfoundex.
       MESSAGE 'The table or index does not exist' TYPE 'E'.
     CATCH /aws1/cx_dyntransactconflictex.
       MESSAGE 'Another transaction is using the item' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateItemin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
 nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] =
 [:]

            if rating != nil {
```

```
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the
release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String:
DynamoDBClientTypes.AttributeValue] = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`UpdateTable`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten globaler Tabellen zur Demonstration von MREC
- Erstellen und Verwalten von globalen MRSC-Tabellen
- Verwalten globaler sekundärer Indizes
- Aktualisieren der Einstellung für den Warmdurchsatz einer Tabelle
- Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)
- Arbeiten mit der Tabellenverschlüsselung

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update a DynamoDB table.
/*!
  \sa updateTable()
  \param tableName: Name for the DynamoDB table.
  \param readCapacity: Provisioned read capacity.
  \param writeCapacity: Provisioned write capacity.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long
 writeCapacity,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput
 values"
              << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;

 provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
            writeCapacity);

 request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome =
 dynamoClient.UpdateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    } else {
        const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
        if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
            error.GetMessage().find("The provisioned throughput for the table
 will not change") != std::string::npos) {
            std::cout << "The provisioned throughput for the table will not
 change." << std::endl;
        } else {
            std::cerr << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
```

```
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So ändern Sie den Fakturierungsmodus einer Tabelle

Im folgenden Beispiel für `update-table` wird die bereitgestellte Lese- und Schreibkapazität für die `MusicCollection`-Tabelle erhöht.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --billing-mode PROVISIONED \
    --provisioned-throughput ReadCapacityUnits=15,WriteCapacityUnits=10
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
```

```
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T13:18:18.921000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        }
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 2: So erstellen Sie einen globalen sekundären Index

Das folgende Beispiel fügt der `MusicCollection`-Tabelle einen globalen sekundären Index hinzu.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --attribute-definitions AttributeName=AlbumTitle,AttributeType=S \
    --global-secondary-index-updates file://gsi-updates.json
```

Inhalt von `gsi-updates.json`:

```
[
    {
        "Create": {
            "IndexName": "AlbumTitle-index",
            "KeySchema": [
                {
```

```
                    "AttributeName": "AlbumTitle",
                    "KeyType": "HASH"
                }
            ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 10,
                "WriteCapacityUnits": 10
            },
            "Projection": {
                "ProjectionType": "ALL"
            }
        }
    }
]
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
```

```
        ],
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "CREATING",
                "Backfilling": false,
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ]
```

```
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 3: So aktivieren Sie DynamoDB Streams für eine Tabelle

Der folgende Befehl aktiviert DynamoDB Streams für die `MusicCollection`-Tabelle.

```
aws dynamodb update-table \
    --table-name MusicCollection \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_IMAGE
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
```

```
        "TableStatus": "UPDATING",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Year",
                        "Genre"
                    ]
                },
                "IndexSizeBytes": 139,
                "ItemCount": 2,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ],
```

```
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "ACTIVE",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ],
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_IMAGE"
        },
        "LatestStreamLabel": "2020-07-28T21:53:39.112",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

Beispiel 4: So aktivieren Sie die serverseitige Verschlüsselung

Im folgenden Beispiel wird die serverseitige Verschlüsselung für die `MusicCollection`-Tabelle aktiviert.

```
aws dynamodb update-table \
```

```
    --table-name MusicCollection \
    --sse-specification Enabled=true,SSEType=KMS
```

Ausgabe:

```
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "AlbumTitle",
                "AttributeType": "S"
            },
            {
                "AttributeName": "Artist",
                "AttributeType": "S"
            },
            {
                "AttributeName": "SongTitle",
                "AttributeType": "S"
            }
        ],
        "TableName": "MusicCollection",
        "KeySchema": [
            {
                "AttributeName": "Artist",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "SongTitle",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": "2020-05-26T15:59:49.473000-07:00",
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": "2020-07-28T12:59:17.537000-07:00",
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 15,
            "WriteCapacityUnits": 10
        },
        "TableSizeBytes": 182,
        "ItemCount": 2,
```

```
        "TableArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection",
        "TableId": "abcd0123-01ab-23cd-0123-abcdef123456",
        "BillingModeSummary": {
            "BillingMode": "PROVISIONED",
            "LastUpdateToPayPerRequestDateTime":
 "2020-07-28T13:14:48.366000-07:00"
        },
        "LocalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitleIndex",
                "KeySchema": [
                    {
                        "AttributeName": "Artist",
                        "KeyType": "HASH"
                    },
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "RANGE"
                    }
                ],
                "Projection": {
                    "ProjectionType": "INCLUDE",
                    "NonKeyAttributes": [
                        "Year",
                        "Genre"
                    ]
                },
                "IndexSizeBytes": 139,
                "ItemCount": 2,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitleIndex"
            }
        ],
        "GlobalSecondaryIndexes": [
            {
                "IndexName": "AlbumTitle-index",
                "KeySchema": [
                    {
                        "AttributeName": "AlbumTitle",
                        "KeyType": "HASH"
                    }
                ],
                "Projection": {
```

```
                    "ProjectionType": "ALL"
                },
                "IndexStatus": "ACTIVE",
                "ProvisionedThroughput": {
                    "NumberOfDecreasesToday": 0,
                    "ReadCapacityUnits": 10,
                    "WriteCapacityUnits": 10
                },
                "IndexSizeBytes": 0,
                "ItemCount": 0,
                "IndexArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/index/AlbumTitle-index"
            }
        ],
        "StreamSpecification": {
            "StreamEnabled": true,
            "StreamViewType": "NEW_IMAGE"
        },
        "LatestStreamLabel": "2020-07-28T21:53:39.112",
        "LatestStreamArn": "arn:aws:dynamodb:us-west-2:123456789012:table/
MusicCollection/stream/2020-07-28T21:53:39.112",
        "SSEDescription": {
            "Status": "UPDATING"
        }
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren einer Tabelle im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie UpdateTablein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert den bereitgestellten Durchsatz für die angegebene Tabelle.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Einzelheiten zur API finden Sie unter UpdateTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert den bereitgestellten Durchsatz für die angegebene Tabelle.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- Einzelheiten zur API finden Sie unter UpdateTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateTimeToLive`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateTimeToLive` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten Sie mit Streams und Time-to-Live

CLI

AWS CLI

So aktualisieren Sie die Time-to-Live-Einstellungen für eine Tabelle

Im folgenden Beispiel für `update-time-to-live` wird Time to Live für die angegebene Tabelle aktiviert.

```
aws dynamodb update-time-to-live \
    --table-name MusicCollection \
    --time-to-live-specification Enabled=true,AttributeName=ttl
```

Ausgabe:

```
{
    "TimeToLiveSpecification": {
        "Enabled": true,
        "AttributeName": "ttl"
    }
}
```

Weitere Informationen finden Sie unter Time to Live im Entwicklerhandbuch zu Amazon DynamoDB.

- Einzelheiten zur API finden Sie UpdateTimeToLivein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Aktivieren Sie mit AWS SDK for Java 2.x TTL für eine vorhandene DynamoDB-Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final
 String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            throw e;
        }
    }
```

Deaktivieren Sie mit AWS SDK for Java 2.x TTL für eine vorhandene DynamoDB-Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region)
 {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateTimeToLivein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Aktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const enableTTL = async (tableName, ttlAttribute, region = 'us-east-1') =>
 {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: true,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL enabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to enable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error enabling TTL: ${e}`);
        throw e;
```

```
    }
};

// Example usage (commented out for testing)
// enableTTL('ExampleTable', 'exampleTtlAttribute');
```

Deaktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const disableTTL = async (tableName, ttlAttribute, region = 'us-east-1')
 => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: false,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL disabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to disable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error disabling TTL: ${e}`);
        throw e;
    }
};
```

```
// Example usage (commented out for testing)
// disableTTL('ExampleTable', 'exampleTtlAttribute');
```

- Einzelheiten zur API finden Sie UpdateTimeToLivein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Aktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import boto3


def enable_ttl(table_name, ttl_attribute_name):
    """
    Enables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table
    :param ttl_attribute_name: The name of the TTL attribute being provided to
 the table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": True, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been enabled successfully.")
        else:
            print(
                f"Failed to enable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
```

```
        return response
    except Exception as ex:
        print("Couldn't enable TTL in table %s. Here's why: %s" % (table_name,
 ex))
        raise


# your values
enable_ttl("your-table-name", "expireAt")
```

Deaktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import boto3


def disable_ttl(table_name, ttl_attribute_name):
    """
    Disables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table being modified
    :param ttl_attribute_name: The name of the TTL attribute being provided to
 the table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": False, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been disabled successfully.")
        else:
            print(
                f"Failed to disable TTL, status code
 {response['ResponseMetadata']['HTTPStatusCode']}"
```

```
            )
    except Exception as ex:
        print("Couldn't disable TTL in table %s. Here's why: %s" % (table_name,
 ex))
        raise


# your values
disable_ttl("your-table-name", "expireAt")
```

- Einzelheiten zur API finden Sie UpdateTimeToLivein AWS SDK for Python (Boto3) API Reference.

Szenarien für die Verwendung von DynamoDB AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in DynamoDB mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von DynamoDB oder in Kombination mit anderen AWS-Services erledigen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Beschleunigen Sie DynamoDB-Lesevorgänge mit DAX mithilfe eines AWS SDK
- Arbeiten Sie mit erweiterten DynamoDB Global Secondary Index-Szenarien unter Verwendung von AWS Command Line Interface v2
- Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
- Vergleichen Sie mehrere Werte mit einem einzelnen Attribut in DynamoDB mit einem AWS SDK
- Bedingtes Aktualisieren eines DynamoDB-Elements mit einer TTL unter Verwendung eines AWS SDK
- Stellen Sie mithilfe eines eine Connect zu einer lokalen DynamoDB-Instanz her AWS SDK
- Zählen Sie Ausdrucksoperatoren in DynamoDB mit einem AWS SDK
- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Erstellen einer Messenger-Anwendung mit Step Functions

- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine DynamoDB-Tabelle mit einem globalen sekundären Index mithilfe der AWS SDK
- Erstellen Sie eine DynamoDB-Tabelle mit einer Warmdurchsatzeinstellung AWS SDK
- Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten
- Erstellen einer Websocket-Chat-Anwendung mit API Gateway
- Erstellen Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK
- Erstellen und verwalten Sie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency mithilfe eines AWS SDK
- Erstellen und verwalten Sie globale DynamoDB-Tabellen, die MREC demonstrieren, mit einem AWS SDK
- Löschen Sie DynamoDB-Daten mithilfe von PartiQL DELETE-Anweisungen mit einem AWS SDK
- Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erste Schritte mit NoSQL-Datenbanken
- Fügen Sie DynamoDB-Daten mithilfe von PartiQL INSERT-Anweisungen mit einem AWS SDK
- Aufrufen einer Lambda-Funktion von einem Browser aus
- Globale Sekundärindizes von DynamoDB verwalten mit AWS Command Line Interface v2
- Verwalten Sie ressourcenbasierte DynamoDB-Richtlinien mit AWS Command Line Interface v2
- Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK
- Führen Sie erweiterte DynamoDB-Abfrageoperationen mit einem AWS SDK
- Führen Sie Listenoperationen in DynamoDB mit einem AWS SDK
- Führen Sie Kartenoperationen in DynamoDB mit einem AWS SDK
- Führen Sie Mengenoperationen in DynamoDB mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mithilfe von Batches von PartiQL-Anweisungen und AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mit PartiQL ab und AWS SDK
- Abfragen einer DynamoDB-Tabelle mithilfe eines globalen sekundären Indexes mit einem AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mithilfe einer Begins_with-Bedingung mit einem ab AWS SDK
- Fragen Sie eine DynamoDB-Tabelle unter Verwendung eines Datumsbereichs im Sortierschlüssel mit einem ab AWS SDK

- Abfragen einer DynamoDB-Tabelle mit einem komplexen Filterausdruck mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit einem dynamischen Filterausdruck mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit einem Filterausdruck und einem Limit mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle mit verschachtelten Attributen unter Verwendung eines AWS SDK
- Abfragen einer DynamoDB-Tabelle mit Paginierung mithilfe eines AWS SDK
- Fragen Sie eine DynamoDB-Tabelle mit stark konsistenten Lesevorgängen ab, indem Sie eine AWS SDK
- Abfragen von DynamoDB-Daten mithilfe von PartiQL SELECT-Anweisungen mit einem AWS SDK
- Abfragen einer DynamoDB-Tabelle nach TTL-Elementen mithilfe eines AWS SDK
- Abfragen von DynamoDB-Tabellen mithilfe von Datums- und Uhrzeitmustern mit einem AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK
- Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2
- Verstehen Sie die Reihenfolge der Aktualisierungsausdrücke in DynamoDB mit einem AWS SDK
- Aktualisieren Sie eine DynamoDB-Tabelleneinstellung mit warmem Durchsatz mithilfe eines AWS SDK
- Aktualisieren Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK
- Aktualisieren Sie DynamoDB-Daten mithilfe von PartiQL UPDATE-Anweisungen mit einem AWS SDK
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Verwenden Sie ein Dokumentmodell für DynamoDB mit einem AWS SDK
- Verwenden Sie ein Objektpersistenzmodell auf hoher Ebene für DynamoDB mit einem AWS SDK
- Verwenden Sie atomare Zähleroperationen in DynamoDB mit einem AWS SDK
- Verwenden Sie bedingte Operationen in DynamoDB mit einem AWS SDK
- Verwenden Sie Ausdrucksattributnamen in DynamoDB mit einem AWS SDK
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion
- Arbeiten Sie mit lokalen sekundären DynamoDB-Indizes unter Verwendung von AWS Command Line Interface v2
- Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2

- Arbeiten Sie mit globalen DynamoDB-Tabellen und Multi-Region-Replikation mit Eventual Consistency (MREC) unter Verwendung von AWS Command Line Interface v2
- Arbeiten Sie mit DynamoDB-Ressourcen-Tagging mit AWS Command Line Interface v2
- Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

Beschleunigen Sie DynamoDB-Lesevorgänge mit DAX mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen und Schreiben von Daten in eine Tabelle mit sowohl den DAX- als auch den SDK-Clients.
- Abrufen, Abfragen und Scannen der Tabelle mit beiden Clients und Vergleichen ihrer Leistung.

Weitere Informationen finden Sie unter Entwickeln mit dem DynamoDB-Accelerator-Client.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Tabelle mit dem DAX- oder Boto3-Client.

```
import boto3


def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")
```

```
    table_name = "TryDaxTable"
    params = {
        "TableName": table_name,
        "KeySchema": [
            {"AttributeName": "partition_key", "KeyType": "HASH"},
            {"AttributeName": "sort_key", "KeyType": "RANGE"},
        ],
        "AttributeDefinitions": [
            {"AttributeName": "partition_key", "AttributeType": "N"},
            {"AttributeName": "sort_key", "AttributeType": "N"},
        ],
        "BillingMode": "PAY_PER_REQUEST",
    }
    table = dyn_resource.create_table(**params)
    print(f"Creating {table_name}...")
    table.wait_until_exists()
    return table


if __name__ == "__main__":
    dax_table = create_dax_table()
    print(f"Created table.")
```

Schreiben Sie Testdaten in die Tabelle.

```
import boto3


def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate
 the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
```

```
    some_data = "X" * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(
                Item={
                    "partition_key": partition_key,
                    "sort_key": sort_key,
                    "some_data": some_data,
                }
            )
            print(f"Put item ({partition_key}, {sort_key}) succeeded.")


if __name__ == "__main__":
    write_key_count = 10
    write_item_size = 1000
    print(
        f"Writing {write_key_count*write_key_count} items to the table. "
        f"Each item is {write_item_size} characters."
    )
    write_data_to_dax_table(write_key_count, write_item_size)
```

Rufen Sie Elemente für eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client ab und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import sys
import time
import amazondax
import boto3


def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param key_count: The number of items to get from the table in each
 iteration.
    :param iterations: The number of iterations to run.
```

```
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        for partition_key in range(1, key_count + 1):
            for sort_key in range(1, key_count + 1):
                table.get_item(
                    Key={"partition_key": partition_key, "sort_key": sort_key}
                )
                print(".", end="")
                sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the DAX client."
        )
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
 as dax:
            test_start, test_end = get_item_test(
```

```
                test_key_count, test_iterations, dyn_resource=dax
            )
    else:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the Boto3 client."
        )
        test_start, test_end = get_item_test(test_key_count, test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/ test_iterations}."
    )
```

Fragen Sie die Tabelle im Hinblick auf eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client ab und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import time
import sys
import amazondax
import boto3
from boto3.dynamodb.conditions import Key


def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
    Queries the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param partition_key: The partition key value to use in the query. The query
                          returns items that have partition keys equal to this
 value.
    :param sort_keys: The range of sort key values for the query. The query
 returns
                      items that have sort key values between these two values.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")
```

```
    table = dyn_resource.Table("TryDaxTable")
    key_condition_expression = Key("partition_key").eq(partition_key) & Key(
        "sort_key"
    ).between(*sort_keys)

    start = time.perf_counter()
    for _ in range(iterations):
        table.query(KeyConditionExpression=key_condition_expression)
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX
 client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
 as dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations,
 dyn_resource=dax
            )
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3
 client.")
```

```
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations
        )

    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Scannen Sie die Tabelle auf eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import time
import sys
import amazondax
import boto3


def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end
```

```
if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX
 client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url)
 as dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = scan_test(test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Löschen Sie die Tabelle.

```
import boto3


def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
```

```
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    table.delete()

    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()


if __name__ == "__main__":
    delete_dax_table()
    print("Table deleted!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTable
  - DeleteTable
  - GetItem
  - PutItem
  - Query
  - Scan

Arbeiten Sie mit erweiterten DynamoDB Global Secondary Index-Szenarien unter Verwendung von AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie Sie mit erweiterten Konfigurationen für globale sekundäre Indizes arbeiten.

- Erstellen Sie eine Tabelle mit mehreren GSIs.
- Erstellen Sie eine Tabelle mit On-Demand-Kapazität und einem GSI.
- Fügen Sie Elemente in eine Tabelle mit mehreren GSIs ein.
- Fragen Sie mehrere GSIs mit unterschiedlichen Bedingungen ab.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit mehreren GSIs.

```
# Create a table with multiple GSIs
aws dynamodb create-table \
    --table-name MusicLibrary \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
        AttributeName=Year,AttributeType=N \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumIndex\",
                \"KeySchema\": [{\"AttributeName\":\"AlbumTitle\",\"KeyType\":
\"HASH\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            },
            {
                \"IndexName\": \"GenreYearIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Year\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",
\"NonKeyAttributes\":[\"Artist\",\"SongTitle\"]}
            }
        ]"
```

Erstellen Sie eine Tabelle mit On-Demand-Kapazität und einem GSI.

```
# Create a table with on-demand capacity and GSI
aws dynamodb create-table \
```

```
    --table-name MusicOnDemand \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GenreIndex\",
                \"KeySchema\": [{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH
\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            }
        ]"
```

Fügen Sie Elemente in eine Tabelle mit mehreren GSIs ein.

```
# Add items to MusicLibrary table
aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "The Beatles"},
        "SongTitle": {"S": "Hey Jude"},
        "AlbumTitle": {"S": "Past Masters"},
        "Genre": {"S": "Rock"},
        "Year": {"N": "1968"}
    }'

aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "Miles Davis"},
        "SongTitle": {"S": "So What"},
        "AlbumTitle": {"S": "Kind of Blue"},
        "Genre": {"S": "Jazz"},
        "Year": {"N": "1959"}
    }'
```

Fragen Sie Elemente aus einer Tabelle mit mehreren GSIs ab.

```
# Query the AlbumIndex GSI
echo "Querying AlbumIndex GSI:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Kind of Blue"}}'

# Query the GenreYearIndex GSI with a range condition
echo "Querying GenreYearIndex GSI with range condition:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name GenreYearIndex \
    --key-condition-expression "Genre = :genre AND #yr > :year" \
    --expression-attribute-names '{"#yr": "Year"}' \
    --expression-attribute-values '{":genre":{"S":"Rock"},":year":{"N":"1965"}}'
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - PutItem
  - Query

Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Code-Beispiele zeigen, wie man eine Anwendung erstellt, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

Java

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Vergleichen Sie mehrere Werte mit einem einzelnen Attribut in DynamoDB mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Sie mehrere Werte mit einem einzigen Attribut in DynamoDB vergleichen.

- Verwenden des IN-Operators, um mehrere Werte mit einem einzigen Attribut zu vergleichen
- Vergleichen des IN-Operators mit mehreren OR-Bedingungen
- Verständnis der Vorteilen von IN hinsichtlich Leistung und Ausdruckskomplexität

Java

SDK für Java 2.x

Vergleichen Sie mehrere Werte mit einem einzelnen Attribut in DynamoDB mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

    /**
     * Queries a table using the IN operator to compare multiple values with a
 single attribute.
     *
     * <p>This method demonstrates how to use the IN operator in a filter
 expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
```

```
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values
with a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
```

```
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
```

```
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the
IN operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
```

```
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number
of values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
```

```
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or
boolean)
     * @param count The number of values to generate
```

```
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value =
AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: "
+ valueType);
            }

            values.add(value);
        }

        return values;
    }
```

Beispiel für die Verwendung des Vergleichs mehrerer Werte mit. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
```

```
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
```

```
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);

            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator
limit for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
condition expressions");
```

```
            System.out.println("7. The IN operator supports up to 100 comparison
 values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - Query
  - Scan

JavaScript

SDK für JavaScript (v3)

Vergleichen Sie mehrere Werte mit einem einzigen Attribut mithilfe von AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  ScanCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Query or scan a DynamoDB table to find items where an attribute matches any
 value from a list.
 *
 * This function demonstrates the use of the IN operator to compare a single
 attribute
 * against multiple possible values, which is more efficient than using multiple
 OR conditions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against
 the values list
```

```
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key
 attribute for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the
 matching items
 */
async function compareMultipleValues(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the filter expression using the IN operator
  const filterExpression = `${attributeName} IN (${valuesList.map((_, index) =>
 `:val${index}`).join(', ')})`;

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
```

```
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous
query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }
```

```
      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}

/**
 * Alternative implementation using multiple OR conditions instead of the IN
 operator.
 *
 * This function is provided for comparison to show why using the IN operator is
 preferable.
 * With many values, this approach becomes verbose and less efficient.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against
 the values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key
 attribute for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the
 matching items
 */
async function compareWithOrConditions(
  config,
  tableName,
  attributeName,
```

```
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // If no values provided, return empty result
  if (!valuesList || valuesList.length === 0) {
    return {
      Items: [],
      Count: 0
    };
  }

  // Create the filter expression using multiple OR conditions
  const filterConditions = valuesList.map((_, index) => `${attributeName} = :val
${index}`);
  const filterExpression = filterConditions.join(' OR ');

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };
```

```
      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous
query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));
```

```
      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}
```

Beispiel für die Verwendung des Vergleichs mehrerer Werte mit AWS SDK für JavaScript.

```
/**
 * Example of how to use the compareMultipleValues function.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const attributeName = "Category";
  const valuesList = ["Electronics", "Computers", "Accessories"];

  console.log(`Searching for products in any of these categories:
 ${valuesList.join(', ')}`);

  try {
    // Using the IN operator (recommended approach)
    console.log("\nApproach 1: Using the IN operator");
    const response = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList
    );
```

```
    console.log(`Found ${response.Count} products in the specified categories`);

    // Using multiple OR conditions (alternative approach)
    console.log("\nApproach 2: Using multiple OR conditions");
    const response2 = await compareWithOrConditions(
      config,
      tableName,
      attributeName,
      valuesList
    );

    console.log(`Found ${response2.Count} products in the specified categories`);

    // Example with a query operation
    console.log("\nQuerying a specific manufacturer's products in multiple
 categories");
    const partitionKeyName = "Manufacturer";
    const partitionKeyValue = "Acme";

    const response3 = await compareMultipleValues(
      config,
      tableName,
      attributeName,
      valuesList,
      partitionKeyName,
      partitionKeyValue
    );

    console.log(`Found ${response3.Count} Acme products in the specified
 categories`);

    // Explain the benefits of using the IN operator
    console.log("\nBenefits of using the IN operator:");
    console.log("1. More concise expression compared to multiple OR conditions");
    console.log("2. Better readability and maintainability");
    console.log("3. Potentially better performance with large value lists");
    console.log("4. Simpler code that's less prone to errors");
    console.log("5. Easier to modify when adding or removing values");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - Query
  - Scan

Python

SDK für Python (Boto3)

Vergleichen Sie mehrere Werte mit einem einzigen Attribut mithilfe von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key
from typing import Any, Dict, List, Optional


def compare_multiple_values(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Query or scan a DynamoDB table to find items where an attribute matches any
 value from a list.

    This function demonstrates the use of the IN operator to compare a single
 attribute
    against multiple possible values, which is more efficient than using multiple
 OR conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key
 attribute for query operations.
```

```
        partition_key_value (Optional[str]): The value of the partition key to
query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create the filter expression using the is_in method
    filter_expression = Attr(attribute_name).is_in(values_list)

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition,
FilterExpression=filter_expression
        )
    else:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}
```

```
def compare_with_or_conditions(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Alternative implementation using multiple OR conditions instead of the IN
 operator.

    This function is provided for comparison to show why using the IN operator is
 preferable.
    With many values, this approach becomes verbose and less efficient.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key
 attribute for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
 query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create a filter expression with multiple OR conditions
    filter_expression = None
    for value in values_list:
        condition = Attr(attribute_name).eq(value)
        if filter_expression is None:
            filter_expression = condition
        else:
            filter_expression = filter_expression | condition

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value and filter_expression:
        key_condition = Key(partition_key_name).eq(partition_key_value)
```

```
        response = table.query(
            KeyConditionExpression=key_condition,
 FilterExpression=filter_expression
        )
    elif filter_expression:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)
    else:
        # Return empty response if no values provided
        return {"Items": [], "Count": 0}

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}
```

Beispiel für die Verwendung des Vergleichs mehrerer Werte mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the compare_multiple_values function."""
    # Example parameters
    table_name = "Products"
    attribute_name = "Category"
    values_list = ["Electronics", "Computers", "Accessories"]

    print(f"Searching for products in any of these categories: {values_list}")
```

```
    # Using the IN operator (recommended approach)
    print("\nApproach 1: Using the IN operator")
    response = compare_multiple_values(
        table_name=table_name, attribute_name=attribute_name,
values_list=values_list
    )

    print(f"Found {response['Count']} products in the specified categories")

    # Using multiple OR conditions (alternative approach)
    print("\nApproach 2: Using multiple OR conditions")
    response2 = compare_with_or_conditions(
        table_name=table_name, attribute_name=attribute_name,
values_list=values_list
    )

    print(f"Found {response2['Count']} products in the specified categories")

    # Example with a query operation
    print("\nQuerying a specific manufacturer's products in multiple categories")
    partition_key_name = "Manufacturer"
    partition_key_value = "Acme"

    response3 = compare_multiple_values(
        table_name=table_name,
        attribute_name=attribute_name,
        values_list=values_list,
        partition_key_name=partition_key_name,
        partition_key_value=partition_key_value,
    )

    print(f"Found {response3['Count']} Acme products in the specified
categories")

    # Explain the benefits of using the IN operator
    print("\nBenefits of using the IN operator:")
    print("1. More concise expression compared to multiple OR conditions")
    print("2. Better readability and maintainability")
    print("3. Potentially better performance with large value lists")
    print("4. Simpler code that's less prone to errors")
    print("5. Easier to modify when adding or removing values")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Query
  - Scan

Bedingtes Aktualisieren eines DynamoDB-Elements mit einer TTL unter Verwendung eines AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie die TTL eines Elements bedingungsabhängig aktualisiert wird.

Java

SDK für Java 2.x

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
```

```
 */
public class UpdateTTLConditional {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
 or partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table
 is located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR +
 "=:c, " + EXPIRE_AT_ATTR + "=:e";
    private static final String CONDITION_EXPRESSION = EXPIRE_AT_ATTR + " > :c";
    private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with
 TTL successful.";
    private static final String CONDITION_FAILED_MESSAGE = "Condition check
 failed. The item may have already expired.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon
 DynamoDB table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs an UpdateTTLConditional with a default DynamoDB client.
     */
    public UpdateTTLConditional() {
        this.dynamoDbClient = null;
    }

    /**
     * Constructs an UpdateTTLConditional with the specified DynamoDB client.
     *
```

```
     * @param dynamoDbClient The DynamoDB client to use
     */
    public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Main method to demonstrate conditionally updating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new UpdateTTLConditional().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this
code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] :
null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // Get current time in epoch second format
```

```
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c",
 AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e",
 AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItemConditional = async (tableName, partitionKey, sortKey,
 region = 'us-east-1', newAttribute = 'default-value') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            artist: partitionKey,
            album: sortKey
        }),
        UpdateExpression: "SET newAttribute = :newAttribute",
        ConditionExpression: "expireAt > :expiration",
        ExpressionAttributeValues: marshall({
            ':newAttribute': newAttribute,
            ':expiration': currentTime
        }),
        ReturnValues: "ALL_NEW"
    };

    try {
        const response = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(response.Attributes);
        console.log("Item updated successfully: ", responseData);
        return responseData;
```

```
    } catch (error) {
        if (error.name === "ConditionalCheckFailedException") {
            console.log("Condition check failed: Item's 'expireAt' is expired.");
        } else {
            console.error("Error updating item: ", error);
        }
        throw error;
    }
};

// Example usage (commented out for testing)
// updateItemConditional('your-table-name', 'your-partition-key-value', 'your-
sort-key-value');
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
from datetime import datetime, timedelta

import boto3
from botocore.exceptions import ClientError


def update_dynamodb_item_ttl(table_name, region, primary_key, sort_key,
 ttl_attribute):
    """
    Updates an existing record in a DynamoDB table with a new or updated TTL
 attribute.

    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :param ttl_attribute: name of the TTL attribute in the target DynamoDB table
    :return:
    """
```

```
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Generate updated TTL in epoch second format
        updated_expiration_time = int((datetime.now() +
 timedelta(days=90)).timestamp())

        # Define the update expression for adding/updating a new attribute
        update_expression = "SET newAttribute = :val1"

        # Define the condition expression for checking if 'expireAt' is not
 expired
        condition_expression = "expireAt > :val2"

        # Define the expression attribute values
        expression_attribute_values = {":val1": ttl_attribute, ":val2":
 updated_expiration_time}

        response = table.update_item(
            Key={"primaryKey": primary_key, "sortKey": sort_key},
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeValues=expression_attribute_values,
        )

        print("Item updated successfully.")
        return response["ResponseMetadata"]["HTTPStatusCode"]  # Ideally a 200 OK
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            print("Condition check failed: Item's 'expireAt' is expired.")
        else:
            print(f"Error updating item: {e}")
    except Exception as e:
        print(f"Error updating item: {e}")


# replace with your values
update_dynamodb_item_ttl(
    "your-table-name",
    "us-east-1",
    "your-partition-key-value",
    "your-sort-key-value",
    "your-ttl-attribute-value",
```

```
)
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Stellen Sie mithilfe eines eine Connect zu einer lokalen DynamoDB-Instanz her AWS SDK

Das folgende Codebeispiel zeigt, wie eine Endpunkt-URL überschrieben wird, um eine Verbindung zu einer lokalen Entwicklungsbereitstellung von DynamoDB und einem AWS SDK herzustellen.

Weitere Informationen finden Sie unter Lokale Version von DynamoDB.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Lists your tables from a local DynamoDB instance by setting the SDK Config's
/// endpoint_url and test_credentials.
#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();

    let config = aws_config::defaults(aws_config::BehaviorVersion::latest())
        .test_credentials()
        // DynamoDB run locally uses port 8000 by default.
        .endpoint_url("http://localhost:8000")
        .load()
        .await;
    let dynamodb_local_config =
 aws_sdk_dynamodb::config::Builder::from(&config).build();

    let client = aws_sdk_dynamodb::Client::from_conf(dynamodb_local_config);
```

```
    let list_resp = client.list_tables().send().await;
    match list_resp {
        Ok(resp) => {
            println!("Found {} tables", resp.table_names().len());
            for name in resp.table_names() {
                println!("  {}", name);
            }
        }
        Err(err) => eprintln!("Failed to list local dynamodb tables: {err:?}"),
    }
}
```

Zählen Sie Ausdrucksoperatoren in DynamoDB mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Ausdrucksoperatoren in DynamoDB gezählt werden.

- Machen Sie sich mit der Obergrenze von 300 Operatoren in DynamoDB vertraut.
- Zählen Sie Operatoren in komplexen Ausdrücken.
- Optimieren Sie Ausdrücke, um innerhalb der Grenzwerte zu bleiben.

Java

SDK für Java 2.x

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
```

```
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param conditionsCount Number of conditions to include
     * @param useAnd Whether to use AND (true) or OR (false) between conditions
     * @return Map containing the filter expression, attribute values, and
operator count
     */
    public static Map<String, Object> createComplexFilterExpression(int
conditionsCount, boolean useAnd) {
        // Initialize the expression parts and attribute values
        StringBuilder filterExpression = new StringBuilder();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of conditions
        for (int i = 0; i < conditionsCount; i++) {
            // Add the operator between conditions (except for the first one)
            if (i > 0) {
                filterExpression.append(useAnd ? " AND " : " OR ");
            }

            // Alternate between different comparison operators for variety
            String valueKey = ":val" + i;

            switch (i % 5) {
                case 0:
                    filterExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" +
i).build());
                    break;
                case 1:
                    filterExpression.append("attribute").append(i).append(" >
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                    expressionAttributeValues.put(
```

```
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i *
10)).build());
                    break;
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ?
conditionsCount - 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
```

```
    * Creates a complex update expression with a specified number of operations.
    *
    * <p>This method demonstrates how to generate a complex update expression
with
    * a specific number of operators to test the 300 operator limit.
    *
    * @param operationsCount Number of operations to include
    * @return Map containing the update expression, attribute values, and
operator count
    */
   public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
       // Initialize the expression parts and attribute values
       StringBuilder updateExpression = new StringBuilder("SET ");
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

       // Generate the specified number of SET operations
       for (int i = 0; i < operationsCount; i++) {
           // Add comma between operations (except for the first one)
           if (i > 0) {
               updateExpression.append(", ");
           }

           // Alternate between different types of SET operations
           String valueKey = ":val" + i;

           switch (i % 3) {
               case 0:
                   // Simple assignment (1 operator: =)
                   updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey, AttributeValue.builder().s("value" +
i).build());
                   break;
               case 1:
                   // Addition (2 operators: = and +)
                   updateExpression
                       .append("attribute")
                       .append(i)
                       .append(" = attribute")
                       .append(i)
                       .append(" + ")
                       .append(valueKey);
```

```
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: =
and if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i *
10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
```

```
    * Test the operator limit by attempting an operation with a complex
expression.
    *
    * <p>This method demonstrates what happens when an expression approaches or
    * exceeds the 300 operator limit.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @param operatorCount Target number of operators to include
    * @return Map containing the result of the operation attempt
    */
   public static Map<String, Object> testOperatorLimit(
       DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, int operatorCount) {

       // Create a complex update expression with the specified operator count
       Map<String, Object> expressionData =
           createComplexUpdateExpression((int) Math.ceil(operatorCount /
1.5)); // Adjust to get close to target count

       String updateExpression = (String)
expressionData.get("updateExpression");
       @SuppressWarnings("unchecked")
       Map<String, AttributeValue> expressionAttributeValues =
           (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
       int actualCount = (int) expressionData.get("operatorCount");

       System.out.println("Generated update expression with approximately " +
actualCount + " operators");

       // Define the update parameters
       UpdateItemRequest request = UpdateItemRequest.builder()
           .tableName(tableName)
           .key(key)
           .updateExpression(updateExpression)
           .expressionAttributeValues(expressionAttributeValues)
           .returnValues("UPDATED_NEW")
           .build();

       try {
           // Attempt the update operation
           UpdateItemResponse response = dynamoDbClient.updateItem(request);
```

```
            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
```

```
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into
" + batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
```

```
                .build();

            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String
expression) {
```

```
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists,
 attribute_type, begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\
\-]\\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);
```

```
        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

Beispiel für die Verwendung des Ausdrucksoperators Counting With AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
```

```
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating DynamoDB expression operator counting
and the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
```

```
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " +
complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching
the operator limit");
            Map<String, Object> approachingLimit =
testOperatorLimit(dynamoDbClient, tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit =
testOperatorLimit(dynamoDbClient, tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression
into multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>)
breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any
expression is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as
1 operator");
```

```
            System.out.println("4. Each function call (attribute_exists,
 contains, etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
 operator");
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself
 and the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the
 number of values");
            System.out.println("8. Parentheses for grouping and attribute paths
 don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
 reports '301 operators'");
            System.out.println("10. For complex operations, break them into
 multiple smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Create a complex filter expression with a specified number of conditions.
 *
 * This function demonstrates how to generate a complex expression with
 * a specific number of operators to test the 300 operator limit.
 *
```

```
 * @param {number} conditionsCount - Number of conditions to include
 * @param {boolean} useAnd - Whether to use AND (true) or OR (false) between
 conditions
 * @returns {Object} - Object containing the filter expression and attribute
 values
 */
function createComplexFilterExpression(conditionsCount, useAnd = true) {
  // Initialize the expression parts and attribute values
  const conditions = [];
  const expressionAttributeValues = {};

  // Generate the specified number of conditions
  for (let i = 0; i < conditionsCount; i++) {
    // Alternate between different comparison operators for variety
    let condition;
    const valueKey = `:val${i}`;

    switch (i % 5) {
      case 0:
        condition = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        condition = `attribute${i} > ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        condition = `attribute${i} < ${valueKey}`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
      case 3:
        condition = `contains(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = `substring${i}`;
        break;
      case 4:
        condition = `attribute_exists(attribute${i})`;
        break;
    }

    conditions.push(condition);
  }

  // Join the conditions with AND or OR
  const operator = useAnd ? " AND " : " OR ";
```

```
  const filterExpression = conditions.join(operator);

  // Calculate the operator count
  // Each condition has 1 operator (=, >, <, contains, attribute_exists)
  // Each AND or OR between conditions is 1 operator
  const operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
 - 1 : 0);

  return {
    filterExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Create a complex update expression with a specified number of operations.
 *
 * This function demonstrates how to generate a complex update expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} operationsCount - Number of operations to include
 * @returns {Object} - Object containing the update expression and attribute
 values
 */
function createComplexUpdateExpression(operationsCount) {
  // Initialize the expression parts and attribute values
  const setOperations = [];
  const expressionAttributeValues = {};

  // Generate the specified number of SET operations
  for (let i = 0; i < operationsCount; i++) {
    // Alternate between different types of SET operations
    let operation;
    const valueKey = `:val${i}`;

    switch (i % 3) {
      case 0:
        // Simple assignment (1 operator: =)
        operation = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        // Addition (2 operators: = and +)
```

```
        operation = `attribute${i} = attribute${i} + ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        // Conditional assignment with if_not_exists (2 operators: = and
 if_not_exists)
        operation = `attribute${i} = if_not_exists(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
    }

    setOperations.push(operation);
  }

  // Create the update expression
  const updateExpression = `SET ${setOperations.join(", ")}`;

  // Calculate the operator count
  // Each operation has 1-2 operators as noted above
  let operatorCount = 0;
  for (let i = 0; i < operationsCount; i++) {
    operatorCount += (i % 3 === 0) ? 1 : 2;
  }

  return {
    updateExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Test the operator limit by attempting an operation with a complex expression.
 *
 * This function demonstrates what happens when an expression approaches or
 * exceeds the 300 operator limit.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} operatorCount - Target number of operators to include
 * @returns {Promise<Object>} - Result of the operation attempt
 */
async function testOperatorLimit(
```

```
  config,
  tableName,
  key,
  operatorCount
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create a complex update expression with the specified operator count
  const { updateExpression, expressionAttributeValues, operatorCount:
 actualCount } =
    createComplexUpdateExpression(Math.ceil(operatorCount / 1.5)); // Adjust to
 get close to target count

  console.log(`Generated update expression with approximately ${actualCount}
 operators`);

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Attempt the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      message: `Operation succeeded with ${actualCount} operators`,
      data: response
    };
  } catch (error) {
    // Check if the error is due to exceeding the operator limit
    if (error.name === "ValidationException" &&
        error.message.includes("too many operators")) {
      return {
        success: false,
        message: `Operation failed: ${error.message}`,
        operatorCount: actualCount
      };
```

```
    }

    // Return other errors
    return {
      success: false,
      message: `Operation failed: ${error.message}`,
      error
    };
  }
}

/**
 * Break down a complex expression into multiple simpler operations.
 *
 * This function demonstrates how to handle expressions that would exceed
 * the 300 operator limit by breaking them into multiple operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} totalOperations - Total number of operations to perform
 * @returns {Promise<Object>} - Result of the operations
 */
async function breakDownComplexExpression(
  config,
  tableName,
  key,
  totalOperations
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Calculate how many operations we can safely include in each batch
  // Using 150 as a conservative limit (well below 300)
  const operationsPerBatch = 100;
  const batchCount = Math.ceil(totalOperations / operationsPerBatch);

  console.log(`Breaking down ${totalOperations} operations into ${batchCount}
 batches`);

  const results = [];

  // Process each batch
```

```
  for (let batch = 0; batch < batchCount; batch++) {
    // Calculate the operations for this batch
    const batchStart = batch * operationsPerBatch;
    const batchEnd = Math.min(batchStart + operationsPerBatch, totalOperations);
    const batchSize = batchEnd - batchStart;

    console.log(`Processing batch ${batch + 1}/${batchCount} with ${batchSize}
 operations`);

    // Create an update expression for this batch
    const { updateExpression, expressionAttributeValues, operatorCount } =
      createComplexUpdateExpression(batchSize);

    // Define the update parameters
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: updateExpression,
      ExpressionAttributeValues: expressionAttributeValues,
      ReturnValues: "UPDATED_NEW"
    };

    try {
      // Perform the update operation for this batch
      const response = await docClient.send(new UpdateCommand(params));

      results.push({
        batch: batch + 1,
        success: true,
        operatorCount,
        attributes: response.Attributes
      });
    } catch (error) {
      results.push({
        batch: batch + 1,
        success: false,
        operatorCount,
        error: error.message
      });

      // Stop processing if an error occurs
      break;
    }
  }
```

```
  return {
    totalBatches: batchCount,
    results
  };
}

/**
 * Count operators in a DynamoDB expression based on the rules in the
 documentation.
 *
 * This function demonstrates how operators are counted according to the
 * DynamoDB documentation.
 *
 * @param {string} expression - The DynamoDB expression to analyze
 * @returns {Object} - Breakdown of operator counts
 */
function countOperatorsInExpression(expression) {
  // Initialize counters for different operator types
  const counts = {
    comparisonOperators: 0,
    logicalOperators: 0,
    functions: 0,
    arithmeticOperators: 0,
    specialOperators: 0,
    total: 0
  };

  // Count comparison operators (=, <>, <, <=, >, >=)
  const comparisonRegex = /[^<>]=[^=]|<>|<=|>=|[^<]>[^=]|[^>]<[^=]/g;
  const comparisonMatches = expression.match(comparisonRegex) || [];
  counts.comparisonOperators = comparisonMatches.length;

  // Count logical operators (AND, OR, NOT)
  const andMatches = expression.match(/\bAND\b/g) || [];
  const orMatches = expression.match(/\bOR\b/g) || [];
  const notMatches = expression.match(/\bNOT\b/g) || [];
  counts.logicalOperators = andMatches.length + orMatches.length +
 notMatches.length;

  // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
  const functionRegex = /\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\(/g;
```

```
  const functionMatches = expression.match(functionRegex) || [];
  counts.functions = functionMatches.length;

  // Count arithmetic operators (+ and -)
  const arithmeticMatches = expression.match(/[a-zA-Z0-9_)\]]\s*[\+\-]\s*[a-zA-
Z0-9_(:]/g) || [];
  counts.arithmeticOperators = arithmeticMatches.length;

  // Count special operators (BETWEEN, IN)
  const betweenMatches = expression.match(/\bBETWEEN\b/g) || [];
  const inMatches = expression.match(/\bIN\b/g) || [];
  counts.specialOperators = betweenMatches.length + inMatches.length;

  // Add extra operators for BETWEEN (each BETWEEN includes an AND)
  counts.logicalOperators += betweenMatches.length;

  // Calculate total
  counts.total = counts.comparisonOperators +
                 counts.logicalOperators +
                 counts.functions +
                 counts.arithmeticOperators +
                 counts.specialOperators;

  return counts;
}
```

Beispiel für die Verwendung des Ausdrucksoperators Counting with AWS SDK für JavaScript.

```
/**
 * Example of how to work with expression operator counting.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating DynamoDB expression operator counting and the 300
 operator limit");

  try {
    // Example 1: Analyze a simple expression
```

```
   console.log("\nExample 1: Analyzing a simple expression");
   const simpleExpression = "Price = :price AND Rating > :rating AND Category IN
(:cat1, :cat2, :cat3)";
   const simpleCount = countOperatorsInExpression(simpleExpression);

   console.log(`Expression: ${simpleExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${simpleCount.comparisonOperators}`);
   console.log(`- Logical operators: ${simpleCount.logicalOperators}`);
   console.log(`- Functions: ${simpleCount.functions}`);
   console.log(`- Arithmetic operators: ${simpleCount.arithmeticOperators}`);
   console.log(`- Special operators: ${simpleCount.specialOperators}`);
   console.log(`- Total operators: ${simpleCount.total}`);

   // Example 2: Analyze a complex expression
   console.log("\nExample 2: Analyzing a complex expression");
   const complexExpression =
     "(attribute_exists(Category) AND Size BETWEEN :min AND :max) OR " +
     "(Price > :price AND contains(Description, :keyword) AND " +
     "(Rating >= :minRating OR Reviews > :minReviews))";
   const complexCount = countOperatorsInExpression(complexExpression);

   console.log(`Expression: ${complexExpression}`);
   console.log("Operator count breakdown:");
   console.log(`- Comparison operators: ${complexCount.comparisonOperators}`);
   console.log(`- Logical operators: ${complexCount.logicalOperators}`);
   console.log(`- Functions: ${complexCount.functions}`);
   console.log(`- Arithmetic operators: ${complexCount.arithmeticOperators}`);
   console.log(`- Special operators: ${complexCount.specialOperators}`);
   console.log(`- Total operators: ${complexCount.total}`);

   // Example 3: Test approaching the operator limit
   console.log("\nExample 3: Testing an expression approaching the operator
limit");
   const approachingLimit = await testOperatorLimit(config, tableName, key,
290);
   console.log(approachingLimit.message);

   // Example 4: Test exceeding the operator limit
   console.log("\nExample 4: Testing an expression exceeding the operator
limit");
   const exceedingLimit = await testOperatorLimit(config, tableName, key, 310);
   console.log(exceedingLimit.message);
```

```
    // Example 5: Breaking down a complex expression
    console.log("\nExample 5: Breaking down a complex expression into multiple
 operations");
    const breakdownResult = await breakDownComplexExpression(config, tableName,
 key, 500);
    console.log(`Processed ${breakdownResult.results.length} of
 ${breakdownResult.totalBatches} batches`);

    // Explain the operator counting rules
    console.log("\nKey points about DynamoDB expression operator counting:");
    console.log("1. The maximum number of operators in any expression is 300");
    console.log("2. Each comparison operator (=, <>, <, <=, >, >=) counts as 1
 operator");
    console.log("3. Each logical operator (AND, OR, NOT) counts as 1 operator");
    console.log("4. Each function call (attribute_exists, contains, etc.) counts
 as 1 operator");
    console.log("5. Each arithmetic operator (+ or -) counts as 1 operator");
    console.log("6. BETWEEN counts as 2 operators (BETWEEN itself and the AND
 within it)");
    console.log("7. IN counts as 1 operator regardless of the number of values");
    console.log("8. Parentheses for grouping and attribute paths don't count as
 operators");
    console.log("9. When you exceed the limit, the error always reports '301
 operators'");
    console.log("10. For complex operations, break them into multiple smaller
 operations");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Einzelheiten zur API finden Sie UpdateItemunter AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
```

```
from typing import Any, Dict, List, Optional, Tuple



def create_complex_filter_expression(
    attribute_name: str, values: List[Any], use_or: bool = True
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a complex filter expression with multiple conditions.

    This function demonstrates how to build a complex filter expression
    and count the number of operators used.

    Args:
        attribute_name (str): The name of the attribute to filter on.
        values (List[Any]): List of values to compare against.
        use_or (bool, optional): Whether to use OR between conditions. Defaults
 to True.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if not values:
        return "", {}, {}, 0

    # Initialize expression components
    filter_expression = ""
    expression_attribute_values = {}
    expression_attribute_names = {"#attr": attribute_name}
    operator_count = 0

    # Build the filter expression
    for i, value in enumerate(values):
        value_placeholder = f":val{i}"
        expression_attribute_values[value_placeholder] = value

        if i > 0:
            # Add OR or AND operator between conditions
            filter_expression += " OR " if use_or else " AND "
            operator_count += 1  # Count the OR/AND operator
```

```
        # Add the condition
        filter_expression += f"#attr = {value_placeholder}"
        operator_count += 1  # Count the = operator

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def create_nested_filter_expression(
    depth: int, conditions_per_level: int
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a deeply nested filter expression with multiple conditions.

    This function demonstrates how to build a complex nested filter expression
    and count the number of operators used.

    Args:
        depth (int): The depth of nesting.
        conditions_per_level (int): Number of conditions at each level.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if depth <= 0 or conditions_per_level <= 0:
        return "", {}, {}, 0

    # Initialize expression components
    expression_attribute_values = {}
    expression_attribute_names = {}
    operator_count = 0

    def build_nested_expression(current_depth: int, prefix: str) -> str:
        nonlocal operator_count
```

```
        if current_depth <= 0:
            return ""

        # Build conditions at this level
        conditions = []
        for i in range(conditions_per_level):
            attr_name = f"attr{prefix}_{i}"
            attr_placeholder = f"#attr{prefix}_{i}"
            val_placeholder = f":val{prefix}_{i}"

            expression_attribute_names[attr_placeholder] = attr_name
            expression_attribute_values[val_placeholder] = i

            conditions.append(f"{attr_placeholder} = {val_placeholder}")
            operator_count += 1  # Count the = operator

        # Join conditions with AND
        level_expression = " AND ".join(conditions)
        operator_count += max(0, len(conditions) - 1)  # Count the AND operators

        # If not at the deepest level, add nested expressions
        if current_depth > 1:
            nested_expr = build_nested_expression(current_depth - 1,
 f"{prefix}_{current_depth}")
            if nested_expr:
                level_expression = f"({level_expression}) OR ({nested_expr})"
                operator_count += 1  # Count the OR operator

        return level_expression

    # Build the expression starting from the top level
    filter_expression = build_nested_expression(depth, "1")

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def count_operators_in_update_expression(update_expression: str) -> int:
    """
    Count the number of operators in an update expression.
```

```
    This function demonstrates how to count operators in an update expression
    based on DynamoDB's rules.

    Args:
        update_expression (str): The update expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count SET operations
    if "SET" in update_expression:
        set_section = (
            update_expression.split("SET")[1].split("REMOVE")[0].split("ADD")
[0].split("DELETE")[0]
        )

        # Count assignment operators (=)
        operator_count += set_section.count("=")

        # Count arithmetic operators (+, -)
        operator_count += set_section.count("+")
        operator_count += set_section.count("-")

        # Count list_append function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("list_append")

        # Count if_not_exists function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("if_not_exists")

    # Count REMOVE operations (no additional operators)

    # Count ADD operations (each ADD counts as 1 operator)
    if "ADD" in update_expression:
        add_section = (
            update_expression.split("ADD")[1].split("DELETE")[0].split("SET")
[0].split("REMOVE")[0]
        )
        operator_count += add_section.count(",") + 1

    # Count DELETE operations (each DELETE counts as 1 operator)
    if "DELETE" in update_expression:
```

```
        delete_section = (
            update_expression.split("DELETE")[1].split("SET")[0].split("ADD")
[0].split("REMOVE")[0]
        )
        operator_count += delete_section.count(",") + 1

    return operator_count


def count_operators_in_condition_expression(condition_expression: str) -> int:
    """
    Count the number of operators in a condition expression.

    This function demonstrates how to count operators in a condition expression
    based on DynamoDB's rules.

    Args:
        condition_expression (str): The condition expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count comparison operators
    comparison_operators = ["=", "<>", "<", "<=", ">", ">="]
    for op in comparison_operators:
        operator_count += condition_expression.count(op)

    # Count logical operators
    operator_count += condition_expression.upper().count(" AND ")
    operator_count += condition_expression.upper().count(" OR ")
    operator_count += condition_expression.upper().count("NOT ")

    # Count BETWEEN operator (counts as 2: BETWEEN + AND)
    between_count = condition_expression.upper().count(" BETWEEN ")
    operator_count += between_count * 2

    # Count IN operator (counts as 1 regardless of number of values)
    operator_count += condition_expression.upper().count(" IN ")

    # Count functions (each counts as 1 operator)
    functions = [
        "attribute_exists",
```

```
        "attribute_not_exists",
        "attribute_type",
        "begins_with",
        "contains",
        "size",
    ]
    for func in functions:
        operator_count += condition_expression.lower().count(func)

    return operator_count


# Note: This function is for demonstration purposes only and should be called
 from example_usage()
# It's not meant to be used directly as a test function
def _test_expression_limit(
    table_name: str, key: Dict[str, Any], operator_count: int, attribute_name:
 str = "TestAttribute"
) -> Tuple[bool, Optional[str]]:
    """
    Test if an expression with a specific number of operators exceeds the limit.

    This function demonstrates how to test the 300 operator limit by creating
    an expression with a specified number of operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        operator_count (int): The number of operators to include in the
 expression.
        attribute_name (str, optional): The name of the attribute to update.
 Defaults to "TestAttribute".

    Returns:
        Tuple[bool, Optional[str]]: A tuple containing:
            - A boolean indicating if the operation succeeded
            - The error message if it failed, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create an update expression with the specified number of operators
    update_expression = f"SET #{attribute_name} = :val0"
```

```
    expression_attribute_names = {f"#{attribute_name}": attribute_name}
    expression_attribute_values = {":val0": 0}

    # Add additional SET operations to reach the desired operator count
    # Each assignment adds 1 operator
    for i in range(1, operator_count):
        attr_name = f"{attribute_name}{i}"
        attr_placeholder = f"#attr{i}"
        val_placeholder = f":val{i}"

        update_expression += f", {attr_placeholder} = {val_placeholder}"
        expression_attribute_names[attr_placeholder] = attr_name
        expression_attribute_values[val_placeholder] = i

    try:
        # Attempt the update operation
        table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues=expression_attribute_values,
        )
        return True, None
    except ClientError as e:
        error_message = e.response["Error"]["Message"]

        if "expression contains too many operators" in error_message.lower():
            return False, error_message
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung des Ausdrucksoperators Counting with AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the expression operator counting functions."""

    print("Example 1: Creating a complex filter expression with multiple
 conditions")
```

```
    attribute_name = "Status"
    values = ["Active", "Pending", "Processing", "Shipped", "Delivered"]

    filter_expr, expr_attr_vals, expr_attr_names, op_count =
create_complex_filter_expression(
        attribute_name=attribute_name, values=values, use_or=True
    )

    print(f"Filter Expression: {filter_expr}")
    print(f"Expression Attribute Values: {expr_attr_vals}")
    print(f"Expression Attribute Names: {expr_attr_names}")
    print(f"Operator Count: {op_count}")

    print("\nExample 2: Creating a nested filter expression")
    nested_expr, nested_vals, nested_names, nested_count =
create_nested_filter_expression(
        depth=3, conditions_per_level=2
    )

    print(f"Nested Filter Expression: {nested_expr}")
    print(f"Operator Count: {nested_count}")

    print("\nExample 3: Counting operators in an update expression")
    update_expression = "SET #name = :name, #age = :age + :increment,
#address.#city = :city, #status = if_not_exists(#status, :default_status) REMOVE
#old_field ADD #counter :value DELETE #set_attr :set_val"
    update_op_count = count_operators_in_update_expression(update_expression)

    print(f"Update Expression: {update_expression}")
    print(f"Operator Count: {update_op_count}")

    print("\nExample 4: Counting operators in a condition expression")
    condition_expression = "(#status = :active OR #status = :pending) AND #price
BETWEEN :min_price AND :max_price AND attribute_exists(#category) AND NOT
(#stock <= :min_stock)"
    condition_op_count =
count_operators_in_condition_expression(condition_expression)

    print(f"Condition Expression: {condition_expression}")
    print(f"Operator Count: {condition_op_count}")

    print("\nExample 5: Testing the 300 operator limit")
```

```
    # This is just for demonstration - in a real application, you would use your
actual table
    # Note: This function is renamed to _test_expression_limit to avoid pytest
trying to run it
    print("In a real application, you would test with _test_expression_limit
function")
    print("Expression with 250 operators would be under the limit")
    print("Expression with 350 operators would exceed the 300 operator limit")

    print("\nOperator Counting Rules in DynamoDB:")
    print("1. Comparison Operators (=, <>, <, <=, >, >=): 1 operator each")
    print("2. Logical Operators (AND, OR, NOT): 1 operator each")
    print("3. BETWEEN: 2 operators (BETWEEN + AND)")
    print("4. IN: 1 operator (regardless of number of values)")
    print("5. Functions (attribute_exists, begins_with, etc.): 1 operator each")
    print("6. Arithmetic Operators (+, -): 1 operator each")
    print("7. SET assignments (=): 1 operator each")
    print("8. ADD and DELETE operations: 1 operator each")

    print("\nStrategies for Working Within the 300 Operator Limit:")
    print("1. Break operations into multiple requests")
    print("2. Use DynamoDB Transactions for complex operations")
    print("3. Optimize data model to reduce query complexity")
    print("4. Use application-side filtering for less critical filters")
    print("5. Consider using IN operator instead of multiple OR conditions")
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die anhand fiktiver Daten ein System zur Nachverfolgung täglicher Fälle COVID-19 in den Vereinigte Staaten simuliert.

Python

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon AWS Lambda

DynamoDB verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle von Infektionen COVID-19 in Vereinigte Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Erstellen einer Messenger-Anwendung mit Step Functions

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

Python

SDK für Python (Boto3)

Zeigt, wie AWS SDK für Python (Boto3) mit AWS Step Functions dem with eine Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. Die Zustandsmaschine ist mit einer AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht werden kann.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen Sie eine DynamoDB-Tabelle mit einem globalen sekundären Index mithilfe der AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine Tabelle mit einem globalen sekundären Index erstellen.

Java

SDK für Java 2.x

Erstellen Sie eine DynamoDB-Tabelle mit Global Secondary Index mithilfe von. AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(ISSUE_ID_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(TITLE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(CREATE_DATE_ATTR)
```

```
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(DUE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());

            // Key schema for table
            final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.HASH)
                .build()); // Partition key
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.RANGE)
                .build()); // Sort key

            // Initial provisioned throughput settings for the indexes
            final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
                .readCapacityUnits(1L)
                .writeCapacityUnits(1L)
                .build();

            // CreateDateIndex
            final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(CREATE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
```

```
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex =
GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();
```

```
            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex,
dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie unter CreateTable AWS SDK for Java 2.xAPI-Referenz.

Erstellen Sie eine DynamoDB-Tabelle mit einer Warmdurchsatzeinstellung AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit aktiviertem Warmdurchsatz erstellt wird.

Java

SDK für Java 2.x

Erstellen Sie mithilfe von AWS SDK for Java 2.x eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long
 readUnitsPerSecond, final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }

    /**
```

```
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType
 scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
 attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
 throughput settings.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to create
     * @param partitionKey The partition key attribute name
     * @param sortKey The sort key attribute name
     * @param miscellaneousKeyAttribute Additional key attribute name for GSI
     * @param nonKeyAttribute Non-key attribute to include in GSI projection
     * @param tableReadCapacityUnits Read capacity units for the table
     * @param tableWriteCapacityUnits Write capacity units for the table
     * @param tableWarmReadUnitsPerSecond Warm read units per second for the
 table
```

```
    * @param tableWarmWriteUnitsPerSecond Warm write units per second for the
table
    * @param globalSecondaryIndexName The name of the GSI to create
    * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the
GSI
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per
second for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
       final Long tableWarmWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadCapacityUnits,
       final Long globalSecondaryIndexWriteCapacityUnits,
       final Long globalSecondaryIndexWarmReadUnitsPerSecond,
       final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

       // Define the table attributes
       final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
       final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
       final AttributeDefinition miscellaneousKeyAttributeDefinition =
           buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
       final AttributeDefinition[] attributeDefinitions = {
           partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
       };

       // Define the table key schema
```

```
        final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
        final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
        final KeySchemaElement[] keySchema = {partitionKeyElement,
sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();
```

```
        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
 tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Erstellen Sie eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung mithilfe von AWS SDK für JavaScript.

```
import { DynamoDBClient, CreateTableCommand } from "@aws-sdk/client-dynamodb";

export async function createDynamoDBTableWithWarmThroughput(
  tableName,
  partitionKey,
  sortKey,
  miscKeyAttr,
  nonKeyAttr,
  tableProvisionedReadUnits,
  tableProvisionedWriteUnits,
  tableWarmReads,
  tableWarmWrites,
  indexName,
  indexProvisionedReadUnits,
```

```
  indexProvisionedWriteUnits,
  indexWarmReads,
  indexWarmWrites,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });
    const command = new CreateTableCommand({
      TableName: tableName,
      AttributeDefinitions: [
          { AttributeName: partitionKey, AttributeType: "S" },
          { AttributeName: sortKey, AttributeType: "S" },
          { AttributeName: miscKeyAttr, AttributeType: "N" },
      ],
      KeySchema: [
          { AttributeName: partitionKey, KeyType: "HASH" },
          { AttributeName: sortKey, KeyType: "RANGE" },
      ],
      ProvisionedThroughput: {
          ReadCapacityUnits: tableProvisionedReadUnits,
          WriteCapacityUnits: tableProvisionedWriteUnits,
      },
      WarmThroughput: {
          ReadUnitsPerSecond: tableWarmReads,
          WriteUnitsPerSecond: tableWarmWrites,
      },
      GlobalSecondaryIndexes: [
          {
            IndexName: indexName,
            KeySchema: [
                { AttributeName: sortKey, KeyType: "HASH" },
                { AttributeName: miscKeyAttr, KeyType: "RANGE" },
            ],
            Projection: {
                ProjectionType: "INCLUDE",
                NonKeyAttributes: [nonKeyAttr],
            },
            ProvisionedThroughput: {
                ReadCapacityUnits: indexProvisionedReadUnits,
                WriteCapacityUnits: indexProvisionedWriteUnits,
            },
            WarmThroughput: {
                ReadUnitsPerSecond: indexWarmReads,
                WriteUnitsPerSecond: indexWarmWrites,
```

```
            },
          },
      ],
    });
    const response = await ddbClient.send(command);
    console.log(response);
    return response;
  } catch (error) {
    console.error(`Error creating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
createDynamoDBTableWithWarmThroughput(
  'example-table',
  'pk',
  'sk',
  'gsiKey',
  'data',
  10, 10, 5, 5,
  'example-index',
  5, 5, 2, 2
);
*/
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Erstellen Sie eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung mithilfe von AWS SDK für Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def create_dynamodb_table_warm_throughput(
```

```
    table_name,
    partition_key,
    sort_key,
    misc_key_attr,
    non_key_attr,
    table_provisioned_read_units,
    table_provisioned_write_units,
    table_warm_reads,
    table_warm_writes,
    gsi_name,
    gsi_provisioned_read_units,
    gsi_provisioned_write_units,
    gsi_warm_reads,
    gsi_warm_writes,
    region_name="us-east-1",
):
    """
    Creates a DynamoDB table with a warm throughput setting configured.

    :param table_name: The name of the table to be created.
    :param partition_key: The partition key for the table being created.
    :param sort_key: The sort key for the table being created.
    :param misc_key_attr: A miscellaneous key attribute for the table being
 created.
    :param non_key_attr: A non-key attribute for the table being created.
    :param table_provisioned_read_units: The newly created table's provisioned
 read capacity units.
    :param table_provisioned_write_units: The newly created table's provisioned
 write capacity units.
    :param table_warm_reads: The read units per second setting for the table's
 warm throughput.
    :param table_warm_writes: The write units per second setting for the table's
 warm throughput.
    :param gsi_name: The name of the Global Secondary Index (GSI) to be created
 on the table.
    :param gsi_provisioned_read_units: The configured Global Secondary Index
 (GSI) provisioned read capacity units.
    :param gsi_provisioned_write_units: The configured Global Secondary Index
 (GSI) provisioned write capacity units.
    :param gsi_warm_reads: The read units per second setting for the Global
 Secondary Index (GSI)'s warm throughput.
    :param gsi_warm_writes: The write units per second setting for the Global
 Secondary Index (GSI)'s warm throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
```

```
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Define the table attributes
        attribute_definitions = [
            {"AttributeName": partition_key, "AttributeType": "S"},
            {"AttributeName": sort_key, "AttributeType": "S"},
            {"AttributeName": misc_key_attr, "AttributeType": "N"},
        ]

        # Define the table key schema
        key_schema = [
            {"AttributeName": partition_key, "KeyType": "HASH"},
            {"AttributeName": sort_key, "KeyType": "RANGE"},
        ]

        # Define the provisioned throughput for the table
        provisioned_throughput = {
            "ReadCapacityUnits": table_provisioned_read_units,
            "WriteCapacityUnits": table_provisioned_write_units,
        }

        # Define the global secondary index
        gsi_key_schema = [
            {"AttributeName": sort_key, "KeyType": "HASH"},
            {"AttributeName": misc_key_attr, "KeyType": "RANGE"},
        ]
        gsi_projection = {"ProjectionType": "INCLUDE", "NonKeyAttributes":
[non_key_attr]}
        gsi_provisioned_throughput = {
            "ReadCapacityUnits": gsi_provisioned_read_units,
            "WriteCapacityUnits": gsi_provisioned_write_units,
        }
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_warm_reads,
            "WriteUnitsPerSecond": gsi_warm_writes,
        }
        global_secondary_indexes = [
            {
                "IndexName": gsi_name,
                "KeySchema": gsi_key_schema,
                "Projection": gsi_projection,
                "ProvisionedThroughput": gsi_provisioned_throughput,
```

```
                "WarmThroughput": gsi_warm_throughput,
            }
        ]

        # Define the warm throughput for the table
        warm_throughput = {
            "ReadUnitsPerSecond": table_warm_reads,
            "WriteUnitsPerSecond": table_warm_writes,
        }

        # Create the DynamoDB client and create the table
        response = ddb.create_table(
            TableName=table_name,
            AttributeDefinitions=attribute_definitions,
            KeySchema=key_schema,
            ProvisionedThroughput=provisioned_throughput,
            GlobalSecondaryIndexes=global_secondary_indexes,
            WarmThroughput=warm_throughput,
        )

        print(response)
        return response
    except ClientError as e:
        print(f"Error creating table: {e}")
        raise e
```

- Einzelheiten zur API finden Sie CreateTablein AWS SDK for Python (Boto3) API Reference.

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie man die Amazon-DynamoDB-.NET-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in Amazon DynamoDB nachverfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer DynamoDB-Tabelle gespeichert sind.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel im AWS Code Examples Repository unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen einer Websocket-Chat-Anwendung mit API Gateway

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon API Gateway V2 eine Websocket-API erstellen, die in Amazon DynamoDB integriert AWS Lambda werden kann.

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.
- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Erstellen Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie ein Element mit TTL erstellen.

Java

SDK für Java 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB
 items.
 */
public class CreateTTL {

    private static final String USAGE =
        """
```

```
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table
is located in. (Default: us-east-1)
            """;
   private static final int DAYS_TO_EXPIRE = 90;
   private static final int SECONDS_PER_DAY = 24 * 60 * 60;
   private static final String PRIMARY_KEY_ATTR = "primaryKey";
   private static final String SORT_KEY_ATTR = "sortKey";
   private static final String CREATION_DATE_ATTR = "creationDate";
   private static final String EXPIRE_AT_ATTR = "expireAt";
   private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
successful.";
   private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon
DynamoDB table \"%s\" can't be found.";

   private final DynamoDbClient dynamoDbClient;

   /**
    * Constructs a CreateTTL instance with the specified DynamoDB client.
    *
    * @param dynamoDbClient The DynamoDB client to use
    */
   public CreateTTL(final DynamoDbClient dynamoDbClient) {
       this.dynamoDbClient = dynamoDbClient;
   }

   /**
    * Constructs a CreateTTL with a default DynamoDB client.
    */
   public CreateTTL() {
       this.dynamoDbClient = null;
   }

   /**
    * Main method to demonstrate creating an item with TTL.
    *
    * @param args Command line arguments
```

```
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this
code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] :
null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }
```

```
    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR,
AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            throw e;
        }
    }
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

```
import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

export function createDynamoDBItem(table_name, region, partition_key, sort_key) {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    // Get the current time in epoch second format
    const current_time = Math.floor(new Date().getTime() / 1000);

    // Calculate the expireAt time (90 days from now) in epoch second format
    const expire_at = Math.floor((new Date().getTime() + 90 * 24 * 60 * 60 *
 1000) / 1000);

    // Create DynamoDB item
    const item = {
        'partitionKey': {'S': partition_key},
        'sortKey': {'S': sort_key},
        'createdAt': {'N': current_time.toString()},
        'expireAt': {'N': expire_at.toString()}
    };

    const putItemCommand = new PutItemCommand({
        TableName: table_name,
        Item: item,
        ProvisionedThroughput: {
            ReadCapacityUnits: 1,
            WriteCapacityUnits: 1,
        },
    });
```

```
    client.send(putItemCommand, function(err, data) {
        if (err) {
            console.log("Exception encountered when creating item %s, here's what
 happened: ", data, err);
            throw err;
        } else {
            console.log("Item created successfully: %s.", data);
            return data;
        }
    });
}

// Example usage (commented out for testing)
// createDynamoDBItem('your-table-name', 'us-east-1', 'your-partition-key-value',
 'your-sort-key-value');
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def create_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Creates a DynamoDB item with an attached expiry attribute.

    :param table_name: Table name for the boto3 resource to target when creating
 an item
    :param region: string representing the AWS region. Example: `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)
```

```
        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expiration time (90 days from now) in epoch second format
        expiration_time = int((datetime.now() + timedelta(days=90)).timestamp())

        item = {
            "primaryKey": primary_key,
            "sortKey": sort_key,
            "creationDate": current_time,
            "expireAt": expiration_time,
        }
        response = table.put_item(Item=item)

        print("Item created successfully.")
        return response
    except Exception as e:
        print(f"Error creating item: {e}")
        raise e


# Use your own values
create_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- Einzelheiten zur API finden Sie PutItemin AWS SDK for Python (Boto3) API Reference.

Erstellen und verwalten Sie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency (MRSC) erstellt und verwaltet werden.

- Erstellen Sie eine Tabelle mit Multi-Region starker Konsistenz.
- Überprüfen Sie die MRSC-Konfiguration und den Replikatstatus.
- Testen Sie die starke Konsistenz in allen Regionen mit sofortigen Lesevorgängen.
- Führen Sie bedingte Schreibvorgänge mit MRSC-Garantien durch.

- Bereinigen Sie die globalen MRSC-Tabellenressourcen.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Multi-Region starker Konsistenz.

```
# Step 1: Create a new table in us-east-2 (primary region for MRSC)
# Note: Table must be empty when enabling MRSC
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --region us-east-2

# Wait for table to become active
aws dynamodb wait table-exists --table-name MusicTable --region us-east-2

# Step 2: Add replica and witness with Multi-Region Strong Consistency
# MRSC requires exactly three replicas in supported regions
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Create": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Create": {"RegionName": "us-west-2"}}]' \
    --multi-region-consistency STRONG \
    --region us-east-2
```

Überprüfen Sie die MRSC-Konfiguration und den Replikatstatus.

```
# Verify the global table configuration and MRSC setting
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
```

```
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistenc
{Region:RegionName,Status:ReplicaStatus}}'
```

Testen Sie die starke Konsistenz mit sofortigen Lesevorgängen in allen Regionen.

```
# Write an item to the primary region
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"},"Album":
 {"S":"The Beatles 1967-1970"},"Year": {"N":"1968"}}' \
    --region us-east-2

# Read the item from replica region to verify strong consistency (cannot read or
 write to witness)
# No wait time needed - MRSC provides immediate consistency
echo "Reading from us-east-1 (immediate consistency):"
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --consistent-read \
    --region us-east-1
```

Führen Sie bedingte Schreibvorgänge mit MRSC-Garantien durch.

```
# Perform a conditional update from a different region
# This demonstrates that conditions work consistently across all regions
aws dynamodb update-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --update-expression "SET #rating = :rating" \
    --condition-expression "attribute_exists(Artist)" \
    --expression-attribute-names '{"#rating": "Rating"}' \
    --expression-attribute-values '{":rating": {"N":"5"}}' \
    --region us-east-1
```

Bereinigen Sie die globalen MRSC-Tabellenressourcen.

```
# Remove replica tables (must be done before deleting the primary table)
```

```
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Delete": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Delete": {"RegionName": "us-west-2"}}]' \
    --region us-east-2

# Wait for replicas to be deleted
echo "Waiting for replicas to be deleted..."
sleep 30

# Delete the primary table
aws dynamodb delete-table \
    --table-name MusicTable \
    --region us-east-2
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Java

SDK für Java 2.x

Erstellen Sie eine regionale Tabelle, die für die MRSC-Konvertierung bereit ist, mit AWS SDK for Java 2.x.

```
    public static CreateTableResponse createRegionalTable(final DynamoDbClient
 dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
```

```
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Creating regional table: " + tableName + " (must be
empty for MRSC)");

            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .tableName(tableName)
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("Artist")
                        .attributeType(ScalarAttributeType.S)
                        .build(),
                    AttributeDefinition.builder()
                        .attributeName("SongTitle")
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
```

```
                .cause(e)
                .build();
        }
    }
```

Konvertiert eine regionale Tabelle in eine MRSC-Tabelle mit Replikaten und Zeugen unter Verwendung von. AWS SDK for Java 2.x

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());

            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate =
ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
```

```
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Beschreiben Sie eine globale MRSC-Tabellenkonfiguration mit. AWS SDK for Java 2.x

```
    public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
```

```
        }

        try {
            LOGGER.info("Describing MRSC global table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response =
dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());

            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: "
+ replica.replicaStatus()));
            }

            if (response.table().globalTableWitnesses() != null
                && !response.table().globalTableWitnesses().isEmpty()) {
                LOGGER.info("Number of witnesses: "
                    + response.table().globalTableWitnesses().size());
                response.table()
                    .globalTableWitnesses()
                    .forEach(witness -> LOGGER.info(
                        "Witness region: " + witness.regionName() + ", Status: "
+ witness.witnessStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
```

```
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Fügen Sie mit AWS SDK for Java 2.x Testelemente hinzu, um die starke Konsistenz von MRSC zu überprüfen.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
```

```
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Lesen Sie Elemente mit konsistenten Lesevorgängen aus MRSC-Replikaten unter Verwendung von. AWS SDK for Java 2.x

```
    public static GetItemResponse getItemWithConsistentRead(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
```

```
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Reading item from MRSC global table with consistent
read: " + tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            GetItemRequest getItemRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .consistentRead(true)
                .build();

            GetItemResponse response = dynamoDbClient.getItem(getItemRequest);

            if (response.hasItem()) {
                LOGGER.info("Item found with strong consistency - no wait time
needed");
            } else {
                LOGGER.info("Item not found");
            }

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Führen Sie bedingte Aktualisierungen mit MRSC-Garantien durch, indem Sie. AWS SDK for Java 2.x

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());
```

```
            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName +
" - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Warten Sie, bis MRSC-Replikate und Zeugen die Nutzung aktiviert haben. AWS SDK for Java 2.x

```
    public static void waitForMRSCReplicasActive(
        final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
        throws InterruptedException {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (maxWaitTimeSeconds <= 0) {
            throw new IllegalArgumentException("Max wait time must be positive");
        }

        try {
            LOGGER.info("Waiting for MRSC replicas and witnesses to become
active: " + tableName);

            final long startTime = System.currentTimeMillis();
            final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
            int backoffSeconds = 5; // Start with 5 second intervals
            final int maxBackoffSeconds = 30; // Cap at 30 seconds

            while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
                DescribeTableResponse response =
describeMRSCTable(dynamoDbClient, tableName);

                boolean allActive = true;
                StringBuilder statusReport = new StringBuilder();

                if (response.table().multiRegionConsistency() == null
                    || !MultiRegionConsistency.STRONG
                        .toString()

.equals(response.table().multiRegionConsistency().toString())) {
                    allActive = false;
                    statusReport
                        .append("MultiRegionConsistency: ")
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
                if (response.table().globalTableWitnesses() == null
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }
```

```
                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString()))
{
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString()))
{
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
                                .append(" ");
                        }
                    }
                }

                // Check witness status
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
                        if (!"ACTIVE".equals(witness.witnessStatus().toString()))
{
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }

                if (allActive) {
```

```
                    LOGGER.info("All MRSC replicas and witnesses are now active:
" + tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active.
Status: " + statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active
after " + maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Bereinigen Sie MRSC-Replikate und Zeugen mithilfe von. AWS SDK for Java 2.x

```
    public static UpdateTableResponse cleanupMRSCReplicas(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
```

```
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

            // Remove replica using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate =
ReplicationGroupUpdate.builder()
                .delete(DeleteReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Remove witness
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - "
+ e.getMessage());
            throw DynamoDbException.builder()
```

```
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Vollständige Demonstration des MRSC-Workflows mithilfe von. AWS SDK for Java 2.x

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);
```

```
            // Use the existing GlobalTableOperations method for basic table
waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "]
to become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "]
to become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
            LOGGER.info("Step 4: Testing strong consistency with data
operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");

            // Immediately read from replica region (no wait needed with MRSC)
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The
Beatles", "Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
            }
```

```
            // Test conditional update from replica region
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles",
"Hey Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null)
{
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
                cleanupVerification = describeMRSCTable(primaryClient,
tableName);
            }

            // Delete the primary table
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency
(MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle
eventual consistency)");
```

```
            LOGGER.info("- Support for conditional writes and transactions across
regions");
            LOGGER.info("- Consistent read operations from any region without
waiting");

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Erstellen und verwalten Sie globale DynamoDB-Tabellen, die MREC demonstrieren, mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie globale DynamoDB-Tabellen mit Replikaten in mehreren Regionen erstellt und verwaltet werden.

- Erstellen Sie eine Tabelle mit einem globalen sekundären Index und DynamoDB Streams.
- Fügen Sie Replikate in verschiedenen Regionen hinzu, um eine globale Tabelle zu erstellen.
- Entfernen Sie Replikate aus einer globalen Tabelle.
- Fügen Sie Testelemente hinzu, um die regionsübergreifende Replikation zu überprüfen.
- Beschreiben Sie die globale Tabellenkonfiguration und den Replikatstatus.

Java

SDK für Java 2.x

Erstellen Sie eine Tabelle mit Global Secondary Index und DynamoDB Streams mithilfe von. AWS SDK for Java 2.x

```
   public static CreateTableResponse createTableWithGSI(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (indexName == null || indexName.trim().isEmpty()) {
           throw new IllegalArgumentException("Index name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating table: " + tableName + " with GSI: " +
indexName);

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
                   KeySchemaElement.builder()
                       .attributeName("Artist")
                       .keyType(KeyType.HASH)
                       .build(),
                   KeySchemaElement.builder()
```

```
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Warten Sie, bis eine Tabelle aktiv wird mit. AWS SDK for Java 2.x

```
    public static void waitForTableActive(final DynamoDbClient dynamoDbClient,
final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
```

```
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Waiting for table to become active: " + tableName);

            try (DynamoDbWaiter waiter =
                DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
                DescribeTableRequest request =
                    DescribeTableRequest.builder().tableName(tableName).build();

                waiter.waitUntilTableExists(request);
                LOGGER.info("Table is now active: " + tableName);
            }

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to wait for table to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Fügen Sie ein Replikat hinzu, um eine globale Tabelle zu erstellen oder zu erweitern mit AWS SDK for Java 2.x.

```
    public static UpdateTableResponse addReplica(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final String indexName,
        final Long readCapacity) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
```

```
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }

        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
```

```
            throw e;
        }
    }
```

Entfernen Sie ein Replikat aus einer globalen Tabelle mit. AWS SDK for Java 2.x

```
    public static UpdateTableResponse removeReplica(
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + "
for table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .delete(builder ->
builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());
```

```
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

Fügen Sie Testelemente hinzu, um die Replikation mithilfe von AWS SDK for Java 2.x zu überprüfen.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to table: " + tableName);

            Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
            item.put(
                "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(artist)
```

```
                    .build());
            item.put(
                "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(songTitle)
                    .build());

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " -
" + e.getMessage());
            throw e;
        }
    }
```

Beschreiben Sie die globale Tabellenkonfiguration und die Verwendung von AWS SDK for Java 2.x Replikaten.

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();
```

```
            DescribeTableResponse response =
dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: "
+ replica.replicaStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " +
e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Vollständiges Beispiel für globale Tabellenoperationen unter Verwendung von AWS SDK for Java 2.x.

```
    public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

        String tableName = "Music";
        String indexName = "SongTitleIndex";
        Long readCapacity = 15L;

        // Create DynamoDB client for the source region
        try (DynamoDbClient dynamoDbClient =
            DynamoDbClient.builder().region(sourceRegion).build()) {
```

```
            try {
                // Step 1: Create the initial table with GSI and streams
                LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
                createTableWithGSI(dynamoDbClient, tableName, indexName);

                // Step 2: Wait for table to become active
                LOGGER.info("Step 2: Waiting for table to become active");
                waitForTableActive(dynamoDbClient, tableName);

                // Step 3: Add replica in destination region
                LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
                addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);

                // Step 4: Wait a moment for replica creation to start
                Thread.sleep(5000);

                // Step 5: Describe table to view replica information
                LOGGER.info("Step 5: Describing table to view replicas");
                describeTable(dynamoDbClient, tableName);

                // Step 6: Add a test item to verify replication
                LOGGER.info("Step 6: Adding test item to verify replication");
                putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

                LOGGER.info("Global table setup completed successfully!");
                LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

                // Step 7: Remove replica and clean up table
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
```

```
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTable
  - DescribeTable
  - PutItem
  - UpdateTable

Löschen Sie DynamoDB-Daten mithilfe von PartiQL DELETE-Anweisungen mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-DELETE-Anweisungen löschen.

JavaScript

SDK für JavaScript (v3)

Löschen Sie Elemente aus einer DynamoDB-Tabelle mithilfe von PartiQL DELETE-Anweisungen mit. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to delete items from a DynamoDB table using
 PartiQL.
 * It shows different ways to delete documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
```

```
/**
 * Delete a single item by its partition key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByCompositeKey = async (
  tableName: string,
```

```
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item with a condition to ensure the delete only happens if a
 condition is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${conditionAttribute} = ?`,
    Parameters: [partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item with condition:", err);
    throw err;
  }
};

/**
 * Batch delete multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param keys - Array of objects containing key information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchDeleteItems = async (
  tableName: string,
  keys: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    sortKeyName?: string;
    sortKeyValue?: string | number;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each delete
  const statements = keys.map((key) => {
    if (key.sortKeyName && key.sortKeyValue !== undefined) {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?
 AND ${key.sortKeyName} = ?`,
        Parameters: [key.partitionKeyValue, key.sortKeyValue],
      };
```

```
    } else {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?
`,
        Parameters: [key.partitionKeyValue],
      };
    }
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch deleted successfully");
    return data;
  } catch (err) {
    console.error("Error batch deleting items:", err);
    throw err;
  }
};

/**
 * Delete multiple items that match a filter condition.
 * Note: This performs a scan operation which can be expensive on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemsByFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };
```

```
  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items deleted by filter successfully");
    return data;
  } catch (err) {
    console.error("Error deleting items by filter:", err);
    throw err;
  }
};

/**
 * Example usage showing how to delete items with different index types
 */
export const deleteExamples = async () => {
  // Delete an item by partition key (simple primary key)
  await deleteItemByPartitionKey("UsersTable", "userId", "user123");

  // Delete an item by composite key (partition key + sort key)
  await deleteItemByCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789"
  );

  // Delete with a condition
  await deleteItemWithCondition(
    "UsersTable",
    "userId",
    "user789",
    "userStatus",
    "inactive"
  );

  // Batch delete multiple items
  await batchDeleteItems("UsersTable", [
    { partitionKeyName: "userId", partitionKeyValue: "user234" },
    { partitionKeyName: "userId", partitionKeyValue: "user345" },
  ]);

  // Batch delete items with composite keys
  await batchDeleteItems("OrdersTable", [
```

```
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order567",
      sortKeyName: "productId",
      sortKeyValue: "prod123",
    },
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order678",
      sortKeyName: "productId",
      sortKeyValue: "prod456",
    },
  ]);

  // Delete items by filter (use with caution)
  await deleteItemsByFilter("UsersTable", "userStatus", "deleted");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erste Schritte mit NoSQL-Datenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Tabelle in DynamoDB
- Daten in einer DynamoDB-Tabelle aktualisieren
- Daten in einer DynamoDB-Tabelle abfragen
- Löschen Sie Ihre DynamoDB-Tabelle

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# DynamoDB Getting Started Tutorial Script
# This script demonstrates basic operations with Amazon DynamoDB:
# - Creating a table
# - Writing data to the table
# - Reading data from the table
# - Updating data in the table
# - Querying data in the table
# - Deleting the table (cleanup)
```

```
set -uo pipefail

# Set up logging with secure permissions
LOG_DIR="${XDG_STATE_HOME:-.}/dynamodb-tutorial-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/dynamodb-tutorial-$(date +%Y%m%d-%H%M%S).log"
chmod 700 "$LOG_DIR"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting DynamoDB Getting Started Tutorial at $(date)"
echo "Logging to $LOG_FILE"

# Validate AWS CLI is configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH"
    exit 1
fi

# Check AWS credentials are available
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd_name=$2

    if echo "$output" | grep -qi "error\|failed"; then
        echo "ERROR detected in $cmd_name command:" >&2
        echo "$output" >&2
        return 1
    fi
    return 0
}

# Function to wait for table to be in ACTIVE state
wait_for_table_active() {
    local table_name=$1
    local max_attempts=60
    local attempt=0
```

```
    local status=""

    echo "Waiting for table $table_name to become ACTIVE..."

    while [[ "$status" != "ACTIVE" && $attempt -lt $max_attempts ]]; do
        sleep 5
        status=$(aws dynamodb describe-table --table-name "$table_name" --query
 "Table.TableStatus" --output text 2>/dev/null || echo "UNKNOWN")
        echo "Current status: $status"
        ((attempt++))
    done

    if [[ "$status" != "ACTIVE" ]]; then
        echo "ERROR: Table $table_name did not become ACTIVE within timeout
 period" >&2
        return 1
    fi

    echo "Table $table_name is now ACTIVE"
    return 0
}

# Track created resources for cleanup
declare -a RESOURCES=()

# Cleanup function
cleanup() {
    local exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
        echo "Script encountered an error (exit code: $exit_code)" >&2
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP"
    echo "============================================"
    echo "Resources to clean up:"
    for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
        echo "- $resource"
    done
    echo ""

    if [[ ${#RESOURCES[@]} -gt 0 ]]; then
```

```
        echo "Proceeding with cleanup of all created resources..."

        for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
            if [[ "$resource" == Table:* ]]; then
                local table_name="${resource#Table:}"
                echo "Deleting table: $table_name"
                if aws dynamodb delete-table --table-name "$table_name" 2>/dev/
null; then
                    echo "Waiting for table deletion to complete..."
                    aws dynamodb wait table-not-exists --table-name "$table_name"
 2>/dev/null || true
                else
                    echo "Warning: Failed to delete table $table_name" >&2
                fi
            fi
        done

        echo "Cleanup completed."
    fi

    return $exit_code
}

trap cleanup EXIT

# Validate table name
validate_table_name() {
    local name=$1
    if [[ ! $name =~ ^[a-zA-Z0-9._-]+$ ]] || [[ ${#name} -gt 255 ]]; then
        echo "ERROR: Invalid table name: $name" >&2
        return 1
    fi
    return 0
}

# Step 1: Create a table in DynamoDB
echo "Step 1: Creating Music table in DynamoDB..."

TABLE_NAME="Music"
validate_table_name "$TABLE_NAME"

CREATE_TABLE_OUTPUT=$(aws dynamodb create-table \
    --table-name "$TABLE_NAME" \
    --attribute-definitions \
```

```
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH
 AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --table-class STANDARD \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-dynamodb-gs
 2>&1) || {
    echo "ERROR: Failed to create table" >&2
    exit 1
}

check_error "$CREATE_TABLE_OUTPUT" "create-table"
echo "$CREATE_TABLE_OUTPUT"

# Add table to resources list
RESOURCES+=("Table:$TABLE_NAME")

# Wait for table to be active
wait_for_table_active "$TABLE_NAME"

# Enable point-in-time recovery (best practice)
echo "Enabling point-in-time recovery for the $TABLE_NAME table..."

PITR_OUTPUT=$(aws dynamodb update-continuous-backups \
    --table-name "$TABLE_NAME" \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true 2>&1)
 || {
    echo "ERROR: Failed to enable PITR" >&2
    exit 1
}

check_error "$PITR_OUTPUT" "update-continuous-backups"
echo "$PITR_OUTPUT"

# Step 2: Write data to the DynamoDB table
echo "Step 2: Writing data to the $TABLE_NAME table..."

# Use a temporary file for item data
ITEMS_TEMP=$(mktemp)
trap "rm -f '$ITEMS_TEMP'" EXIT

cat > "$ITEMS_TEMP" << 'EOF'
```

```
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"},
 "AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Howdy"}, "AlbumTitle":
 {"S": "Somewhat Famous"}, "Awards": {"N": "2"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle":
 {"S": "Songs About Life"}, "Awards": {"N": "10"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "PartiQL Rocks"}, "AlbumTitle":
 {"S": "Another Album Title"}, "Awards": {"N": "8"}}
EOF

declare -i item_num=0
while IFS= read -r item_data; do
    ((item_num++))
    ITEM_OUTPUT=$(aws dynamodb put-item \
        --table-name "$TABLE_NAME" \
        --item "$item_data" 2>&1) || {
        echo "ERROR: Failed to put item $item_num" >&2
        exit 1
    }
    check_error "$ITEM_OUTPUT" "put-item (item $item_num)"
    echo "Item $item_num added successfully"
done < "$ITEMS_TEMP"

# Step 3: Read data from the DynamoDB table
echo "Step 3: Reading data from the $TABLE_NAME table..."

# Get a specific item
GET_ITEM_OUTPUT=$(aws dynamodb get-item --consistent-read \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' 2>&1)
 || {
    echo "ERROR: Failed to get item" >&2
    exit 1
}

check_error "$GET_ITEM_OUTPUT" "get-item"
echo "Retrieved item:"
echo "$GET_ITEM_OUTPUT"

# Step 4: Update data in the DynamoDB table
echo "Step 4: Updating data in the $TABLE_NAME table..."

# Update an item
UPDATE_ITEM_OUTPUT=$(aws dynamodb update-item \
```

```
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' \
    --update-expression "SET AlbumTitle = :newval" \
    --expression-attribute-values '{":newval": {"S": "Updated Album Title"}}' \
    --return-values ALL_NEW 2>&1) || {
    echo "ERROR: Failed to update item" >&2
    exit 1
}

check_error "$UPDATE_ITEM_OUTPUT" "update-item"
echo "Updated item:"
echo "$UPDATE_ITEM_OUTPUT"

# Step 5: Query data in the DynamoDB table
echo "Step 5: Querying data in the $TABLE_NAME table..."

# Query items by Artist
QUERY_OUTPUT=$(aws dynamodb query \
    --table-name "$TABLE_NAME" \
    --key-condition-expression "Artist = :name" \
    --expression-attribute-values '{":name": {"S": "Acme Band"}}' 2>&1) || {
    echo "ERROR: Failed to query table" >&2
    exit 1
}

check_error "$QUERY_OUTPUT" "query"
echo "Query results:"
echo "$QUERY_OUTPUT"

echo "DynamoDB Getting Started Tutorial completed successfully at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- UpdateContinuousBackups
- UpdateItem
- Wait

Fügen Sie DynamoDB-Daten mithilfe von PartiQL INSERT-Anweisungen mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-INSERT-Anweisungen einfügen.

JavaScript

SDK für JavaScript (v3)

Fügen Sie Elemente mithilfe von PartiQL INSERT-Anweisungen mit in eine DynamoDB-Tabelle ein. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to insert items into a DynamoDB table using
 PartiQL.
 * It shows different ways to insert documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Insert a single item into a DynamoDB table using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItem = async (tableName: string, item: Record<string, any>) =>
 {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Convert the item to a string representation for PartiQL
  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item:", err);
    throw err;
  }
};

/**
 * Insert multiple items into a DynamoDB table using PartiQL batch operation.
 * This is more efficient than inserting items one by one.
 *
 * @param tableName - The name of the DynamoDB table
 * @param items - Array of items to insert
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchInsertItems = async (tableName: string, items: Record<string,
 any>[]) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each item
  const statements = items.map((item) => {
    const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
    return {
      Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
    };
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
```

```
    console.log("Items inserted successfully");
    return data;
  } catch (err) {
    console.error("Error batch inserting items:", err);
    throw err;
  }
};

/**
 * Insert an item with a condition to prevent overwriting existing items.
 * This is useful for ensuring you don't accidentally overwrite data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @param partitionKeyName - The name of the partition key attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItemWithCondition = async (
  tableName: string,
  item: Record<string, any>,
  partitionKeyName: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
  const partitionKeyValue = JSON.stringify(item[partitionKeyName]);

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString} WHERE
 attribute_not_exists(${partitionKeyName})`,
    Parameters: [{ S: partitionKeyValue }],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item with condition:", err);
    throw err;
  }
};
```

```
/**
 * Example usage showing how to insert items with different index types
 */
export const insertExamples = async () => {
  // Example table with a simple primary key (just partition key)
  const simpleKeyItem = {
    userId: "user123",
    name: "John Doe",
    email: "john@example.com",
  };
  await insertItem("UsersTable", simpleKeyItem);

  // Example table with composite key (partition key + sort key)
  const compositeKeyItem = {
    orderId: "order456",
    productId: "prod789",
    quantity: 2,
    price: 29.99,
  };
  await insertItem("OrdersTable", compositeKeyItem);

  // Example with Global Secondary Index (GSI)
  // The GSI might be on the email attribute
  const gsiItem = {
    userId: "user789",
    email: "jane@example.com",
    name: "Jane Smith",
    userType: "premium",  // This could be part of a GSI
  };
  await insertItem("UsersTable", gsiItem);

  // Example with Local Secondary Index (LSI)
  // LSI uses the same partition key but different sort key
  const lsiItem = {
    orderId: "order567",  // Partition key
    productId: "prod123", // Sort key for the table
    orderDate: "2023-11-15", // Potential sort key for an LSI
    quantity: 1,
    price: 19.99,
  };
  await insertItem("OrdersTable", lsiItem);

  // Batch insert example with multiple items
  const batchItems = [
```

```
    {
      userId: "user234",
      name: "Alice Johnson",
      email: "alice@example.com",
    },
    {
      userId: "user345",
      name: "Bob Williams",
      email: "bob@example.com",
    },
  ];
  await batchInsertItems("UsersTable", batchItems);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

JavaScript

SDK für JavaScript (v2)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion verwendet, um eine Amazon DynamoDB-Tabelle mit Benutzerauswahlen zu aktualisieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

SDK für JavaScript (v3)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion verwendet, um eine Amazon DynamoDB-Tabelle mit Benutzerauswahlen zu aktualisieren. Diese App verwendet v3. AWS SDK für JavaScript

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

Globale Sekundärindizes von DynamoDB verwalten mit AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie den gesamten Lebenszyklus globaler sekundärer Indizes verwalten.

- Erstellen Sie eine Tabelle mit einem globalen sekundären Index.
- Fügen Sie einer neuen GSI zu einer vorhandenen Tabelle.
- Aktualisieren (erhöhen) Sie den GSI-Warmdurchsatz.
- Fragen Sie Daten mithilfe von GSIs ab.
- Löschen Sie einen GSI.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einem globalen sekundären Index.

```
# Create a table with a GSI
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
    --key-schema \
```

```
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "IndexName=AlbumIndex,\
        KeySchema=[{AttributeName=AlbumTitle,KeyType=HASH}],\
        Projection={ProjectionType=ALL}"
```

Fügen Sie einer vorhandenen Tabelle einen neuen (On-Demand-)GSI hinzu.

```
# Add a new GSI to an existing table
aws dynamodb update-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Genre,AttributeType=S \
    --global-secondary-index-updates \
        "[{\"Create\":{\"IndexName\":\"GenreIndex\",\
        \"KeySchema\":[{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],\
        \"Projection\":{\"ProjectionType\":\"ALL\"}}}]"
```

Aktualisieren (erhöhen) Sie den GSI-Warmdurchsatz.

```
# Increase the warm throughput of a GSI (default values are 12k reads, 4k writes)
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Update\":{\"IndexName\":\"AlbumIndex\",\
        \"WarmThroughput\":{\"ReadUnitsPerSecond\":15000,\"WriteUnitsPerSecond
\":6000}}}]"
```

Fragen Sie Daten mithilfe von GSIs ab.

```
# Query the AlbumIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Let It Be"}}'
```

```
# Query the GenreIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name GenreIndex \
    --key-condition-expression "Genre = :genre" \
    --expression-attribute-values '{":genre":{"S":"Jazz"}}'
```

Löschen Sie einen GSI.

```
# Delete a GSI from a table
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Delete\":{\"IndexName\":\"GenreIndex\"}}]"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - Query
  - UpdateTable

Verwalten Sie ressourcenbasierte DynamoDB-Richtlinien mit AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie den gesamten Lebenszyklus ressourcenbasierter Richtlinien für DynamoDB-Tabellen verwalten.

- Erstellen einer Tabelle mit einer Ressourcenrichtlinie
- Rufen Sie eine Ressourcenrichtlinie ab.
- Aktualisieren Sie eine Ressourcenrichtlinie.
- Löschen Sie eine Ressourcenrichtlinie.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einer Ressourcenrichtlinie.

```
# Step 1: Create a DynamoDB table
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST

# Step 2: Create a resource-based policy document
cat > policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/DynamoDBReadOnly"
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 3: Attach the resource-based policy to the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
 \
    --policy file://policy.json
```

Rufen Sie eine Ressourcenrichtlinie ab.

```
# Get the resource-based policy attached to a table
aws dynamodb get-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

Aktualisieren Sie eine Ressourcenrichtlinie.

```
# Step 1: Create an updated policy document
cat > updated-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/DynamoDBReadOnly",
          "arn:aws:iam::123456789012:role/DynamoDBAnalytics"
        ]
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 2: Update the resource-based policy on the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
 \
    --policy file://updated-policy.json
```

Löschen Sie eine Ressourcenrichtlinie.

```
# Delete the resource-based policy from a table
aws dynamodb delete-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - DeleteResourcePolicy
  - GetResourcePolicy
  - PutResourcePolicy

Überwachen Sie die Leistung von Amazon DynamoDB mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie die Verwendung von DynamoDB durch eine Anwendung zur Leistungsüberwachung konfiguriert wird.

Java

SDK für Java 2.x

Im folgenden Codebeispiel wird gezeigt, wie eine Java-Anwendung zur Leistungsüberwachung von DynamoDB konfiguriert wird. Die Anwendung sendet Metrikdaten, CloudWatch an die Sie die Leistung überwachen können.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch
- DynamoDB

Führen Sie erweiterte DynamoDB-Abfrageoperationen mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Sie erweiterte Abfrageoperationen in DynamoDB ausführen.

- Abfragen von Tabellen mithilfe verschiedener Filter- und Bedingungstechniken
- Implementieren der Paginierung für große Ergebnismengen
- Verwenden globaler sekundärer Indizes für alternative Zugriffsmuster
- Anwenden von Konsistenzkontrollen auf Grundlage der Anwendungsanforderungen

Java

SDK für Java 2.x

Abfrage mit stark konsistenten Lesevorgängen unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
```

```
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

Abfrage unter Verwendung eines globalen sekundären Indexes mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
```

```
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
```

```
            throw new DynamoDbQueryException("Table or index not found: " +
 tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI",
 e);
        }
    }
```

Abfrage mit Paginierung unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
```

```
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;
```

```
            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total
 of " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " +
 e.getMessage());
            throw e;
        }
    }
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
```

```
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
```

```
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
```

```
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName,
partitionKeyName, partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " +
e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
```

```
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " +
filterAttrName + " = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
```

```
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items
 with multiple filters:");
            multiFilterResponse.items().forEach(item ->
 System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

Abfrage mit einem Filterausdruck und Grenzwert unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
```

```
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
 filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();
```

```
        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
 {0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
 filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
 e.getMessage());
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Abfrage mit stark konsistenten Lesevorgängen unter Verwendung von AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
```

```
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

Abfrage unter Verwendung eines globalen sekundären Indexes mit AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
```

```
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}

/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
```

```
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

Abfrage mit Paginierung unter Verwendung von AWS SDK für JavaScript.

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }

      // Execute the query
      const command = new QueryCommand(input);
```

```
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in
 ${pageCount} pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to
 retrieve
```

```
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
```

```
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
```

```
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
```

```
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Abfrage mit stark konsistenten Lesevorgängen unter Verwendung von AWS SDK für Python (Boto3).

```
import time

import boto3
from boto3.dynamodb.conditions import Key


def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent
 reads. Defaults to True.
```

```
    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

Abfrage unter Verwendung eines globalen sekundären Indexes mit AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
```

```
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

Abfrage mit Paginierung unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key
```

```
def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If
 None, retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
```

```
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large
 result sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
```

```
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
```

```
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
 max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating
```

```
    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
```

```
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions
where
            keys are attribute names and values are dictionaries with 'operator'
and 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
{'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
```

```
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Abfrage mit einem Filterausdruck und Grenzwert unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
```

```
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of
 results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults
 to 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] =
 Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value":
 filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Führen Sie Listenoperationen in DynamoDB mit einem AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Listenoperationen für DynamoDB-Tabellen ausgeführt werden.

- Hinzufügen von Elementen zu einem Listenattribut
- Entfernen von Elementen aus einem Listenattribut
- Aktualisieren von bestimmten Elementen in einer Liste nach Index
- Verwenden von Funktionen zum Anfügen und zum Auflisten von Indizes

Java

SDK für Java 2.x

Demonstrieren Sie Listenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
        AttributeValue listValue =
AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {

        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append
function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
     *
     * <p>This method demonstrates how to update a specific element in a list
```

```
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
```

```
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
```

```
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating list operations in DynamoDB");

        try {
            // Example 1: Append items to a list
            System.out.println("\nExample 1: Appending items to a list");
            List<AttributeValue> tagsToAppend = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Gadget").build());

            UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);
```

```
            System.out.println("Updated list attribute: " +
appendResponse.attributes());

            // Example 2: Prepend items to a list
            System.out.println("\nExample 2: Prepending items to a list");
            List<AttributeValue> tagsToPrepend = List.of(
                AttributeValue.builder().s("Featured").build(),
                AttributeValue.builder().s("New").build());

            UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

            System.out.println("Updated list attribute: " +
prependResponse.attributes());

            // Example 3: Update a specific element in a list
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");
```

```
            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " +
 currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
 newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
 existingList)");
            System.out.println("5. Use index notation (list[0]) to access or
 update specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
 list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie Listenoperationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
```

```
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Append elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the end of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to append to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function appendToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} =
 list_append(if_not_exists(${listName}, :empty_list), :values)`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Prepend elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the beginning of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to prepend to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function prependToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  // Note: To prepend, we put the new values first in the list_append function
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = list_append(:values,
 if_not_exists(${listName}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Update a specific element in a list by index.
 *
 * This function demonstrates how to update a specific element in a list
 * using the index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to update
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateListElement(
  config,
  tableName,
  key,
  listName,
  index,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${index}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Remove an element from a list by index.
 *
 * This function demonstrates how to remove a specific element from a list
 * using the REMOVE action with index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to remove
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeListElement(
  config,
  tableName,
  key,
  listName,
  index
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using REMOVE with index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `REMOVE ${listName}[${index}]`,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Concatenate two lists.
 *
```

```
 * This function demonstrates how to concatenate two lists using the list_append
 function.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName1 - The name of the first list attribute
 * @param {string} listName2 - The name of the second list attribute
 * @param {string} resultListName - The name of the attribute to store the
 concatenated list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function concatenateLists(
  config,
  tableName,
  key,
  listName1,
  listName2,
  resultListName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${resultListName} =
 list_append(if_not_exists(${listName1}, :empty_list),
 if_not_exists(${listName2}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": []
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
```

```
 * Create a nested list structure.
 *
 * This function demonstrates how to create and work with nested lists.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} nestedLists - An array of arrays to create a nested list
 structure
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createNestedList(
  config,
  tableName,
  key,
  listName,
  nestedLists
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters to create a nested list
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = :nested_lists`,
    ExpressionAttributeValues: {
      ":nested_lists": nestedLists
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an element in a nested list.
 *
 * This function demonstrates how to update an element in a nested list
```

```
 * using multiple index notations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} outerIndex - The index in the outer list
 * @param {number} innerIndex - The index in the inner list
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedListElement(
  config,
  tableName,
  key,
  listName,
  outerIndex,
  innerIndex,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using multiple index notations
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${outerIndex}][${innerIndex}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
```

```
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to work with lists in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "UserProfiles";
  const key = { UserId: "U12345" };

  console.log("Demonstrating list operations in DynamoDB");
```

```
  try {
    // Example 1: Append elements to a list
    console.log("\nExample 1: Appending elements to a list");
    const response1 = await appendToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["laptop", "headphones", "monitor"]
    );

    console.log("Appended to list:", response1.Attributes);

    // Example 2: Prepend elements to a list
    console.log("\nExample 2: Prepending elements to a list");
    const response2 = await prependToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["keyboard", "mouse"]
    );

    console.log("Prepended to list:", response2.Attributes);

    // Get the current state of the item
    let currentItem = await getItem(config, tableName, key);
    console.log("\nCurrent state of RecentSearches:",
 currentItem?.RecentSearches);

    // Example 3: Update a specific element in a list
    console.log("\nExample 3: Updating a specific element in a list");
    const response3 = await updateListElement(
      config,
      tableName,
      key,
      "RecentSearches",
      0, // Update the first element
      "mechanical keyboard" // New value
    );

    console.log("Updated list element:", response3.Attributes);

    // Example 4: Remove an element from a list
```

```
    console.log("\nExample 4: Removing an element from a list");
    const response4 = await removeListElement(
      config,
      tableName,
      key,
      "RecentSearches",
      2 // Remove the third element
    );

    console.log("List after removing element:", response4.Attributes);

    // Example 5: Create and concatenate lists
    console.log("\nExample 5: Creating and concatenating lists");

    // First, create two separate lists
    await updateWithMultipleActions(
      config,
      tableName,
      key,
      "SET WishList = :wishlist, SavedItems = :saveditems",
      null,
      {
        ":wishlist": ["gaming laptop", "wireless earbuds"],
        ":saveditems": ["smartphone", "tablet"]
      }
    );

    // Then, concatenate them
    const response5 = await concatenateLists(
      config,
      tableName,
      key,
      "WishList",
      "SavedItems",
      "AllItems"
    );

    console.log("Concatenated lists:", response5.Attributes);

    // Example 6: Create a nested list structure
    console.log("\nExample 6: Creating a nested list structure");
    const response6 = await createNestedList(
      config,
      tableName,
```

```
      key,
      "Categories",
      [
        ["Electronics", "Computers", "Accessories"],
        ["Books", "Magazines", "E-books"],
        ["Clothing", "Shoes", "Watches"]
      ]
    );

    console.log("Created nested list:", response6.Attributes);

    // Example 7: Update an element in a nested list
    console.log("\nExample 7: Updating an element in a nested list");
    const response7 = await updateNestedListElement(
      config,
      tableName,
      key,
      "Categories",
      0, // First inner list
      1, // Second element in that list
      "Laptops" // New value
    );

    console.log("Updated nested list element:", response7.Attributes);

    // Get the final state of the item
    currentItem = await getItem(config, tableName, key);
    console.log("\nFinal state of the item:", JSON.stringify(currentItem, null,
  2));

    // Explain list operations
    console.log("\nKey points about list operations in DynamoDB:");
    console.log("1. Use list_append to add elements to a list");
    console.log("2. To append elements, use list_append(existingList,
  newElements)");
    console.log("3. To prepend elements, use list_append(newElements,
  existingList)");
    console.log("4. Use if_not_exists to handle cases where the list might not
  exist yet");
    console.log("5. Use index notation (list[0]) to access or update specific
  elements");
    console.log("6. Use REMOVE with index notation to remove elements from a
  list");
    console.log("7. Lists can contain elements of different types");
```

```
    console.log("8. Lists can be nested (lists of lists)");
    console.log("9. Use multiple index notations (list[0][1]) to access nested
 list elements");

  } catch (error) {
    console.error("Error:", error);
  }
}

/**
 * Helper function for the examples.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} updateExpression - The update expression
 * @param {Object} expressionAttributeNames - Expression attribute name
 placeholders
 * @param {Object} expressionAttributeValues - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
```

```
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie Listenoperationen mit AWS SDK für Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, List, Optional, Union


def create_list_attribute(
    table_name: str, key: Dict[str, Any], list_name: str, list_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a new list attribute or replace an existing one.

    This function demonstrates how to create a new list attribute or replace
    an existing list with new values.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        list_values (List[Any]): The values to set in the list.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation to create or replace the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = :list_values",
        ExpressionAttributeValues={":list_values": list_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_append:
 List[Any]
) -> Dict[str, Any]:
    """
    Append values to the end of a list attribute.

    This function demonstrates how to use the list_append function to add
 elements
    to the end of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Use list_append to add values to the end of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append({list_name}, :values)",
        ExpressionAttributeValues={":values": values_to_append},
        ReturnValues="UPDATED_NEW",
    )

    return response


def prepend_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_prepend:
 List[Any]
) -> Dict[str, Any]:
    """
    Prepend values to the beginning of a list attribute.

    This function demonstrates how to use the list_append function to add
 elements
    to the beginning of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_prepend (List[Any]): The values to prepend to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append with reversed order to add values to the beginning of the
 list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append(:values, {list_name})",
        ExpressionAttributeValues={":values": values_to_prepend},
        ReturnValues="UPDATED_NEW",
    )
```

```
    return response


def update_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int, new_value:
 Any
) -> Dict[str, Any]:
    """
    Update a specific element in a list attribute.

    This function demonstrates how to update a specific element in a list
 attribute
    using the index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to update.
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the index notation to update a specific element
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name}[{index}] = :value",
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int
) -> Dict[str, Any]:
```

```
    """
    Remove a specific element from a list attribute.

    This function demonstrates how to remove a specific element from a list
 attribute
    using the REMOVE action with index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to remove.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the REMOVE action with index notation to remove a specific element
    response = table.update_item(
        Key=key, UpdateExpression=f"REMOVE {list_name}[{index}]",
 ReturnValues="UPDATED_NEW"
    )

    return response


def update_nested_list_element(
    table_name: str, key: Dict[str, Any], path: str, new_value: Any
) -> Dict[str, Any]:
    """
    Update an element in a nested list structure.

    This function demonstrates how to update an element in a nested list
 structure
    using expression attribute names for the path components.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested element (e.g., "parent[0].child[1]").
```

```
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
attribute values.
    """
    # Define a type for path parts
    path_part = Dict[str, Union[str, int]]
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Parse the path to extract attribute names and indices
    path_parts: List[path_part] = []
    current_part = ""
    in_bracket = False

    for char in path:
        if char == "[":
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
            in_bracket = True
        elif char == "]":
            if current_part:
                # Fix for mypy: Use a properly typed dictionary with Union type
                path_parts.append({"type": "index", "value": int(current_part)})
                current_part = ""
            in_bracket = False
        elif char == "." and not in_bracket:
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
        else:
            current_part += char

    if current_part:
        path_parts.append({"type": "attribute", "value": current_part})

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
```

```
    path_expression = ""
    for i, part in enumerate(path_parts):
        if part["type"] == "attribute":
            name_placeholder = f"#attr{i}"
            expression_attribute_names[name_placeholder] = part["value"]

            if path_expression:
                path_expression += "."
            path_expression += name_placeholder
        elif part["type"] == "index":
            path_expression += f"[{part['value']}]"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def create_list_if_not_exists(
    table_name: str, key: Dict[str, Any], list_name: str, default_values:
 List[Any]
) -> Dict[str, Any]:
    """
    Create a list attribute if it doesn't exist.

    This function demonstrates how to use if_not_exists to create a list
 attribute
    with default values if it doesn't already exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        default_values (List[Any]): The default values for the list.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to create the list if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} =
 if_not_exists({list_name}, :default)",
        ExpressionAttributeValues={":default": default_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list_safely(
    table_name: str,
    key: Dict[str, Any],
    list_name: str,
    values_to_append: List[Any],
    default_values: Optional[List[Any]] = None,
) -> Dict[str, Any]:
    """
    Append values to a list, creating it if it doesn't exist.

    This function demonstrates how to safely append values to a list attribute,
    creating the list with default values if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.
        default_values (Optional[List[Any]]): The default values if the list
 doesn't exist.
            If not provided, values_to_append will be used as the default.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # If default_values is not provided, use values_to_append
    if default_values is None:
        default_values = values_to_append

    # Use if_not_exists with list_append to safely append to the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} =
 list_append(if_not_exists({list_name}, :default), :values)",
        ExpressionAttributeValues={
            ":default": default_values if default_values else [],
            ":values": values_to_append,
        },
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use list operations in DynamoDB."""
    # Example parameters
    table_name = "UserData"
    key = {"UserId": "user123"}

    print("Example 1: Creating a list attribute")
    try:
        response = create_list_attribute(
            table_name=table_name,
            key=key,
            list_name="Interests",
            list_values=["Reading", "Hiking", "Photography"],
```

```
        )
        print(
            f"List attribute created successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list attribute: {e}")

    print("\nExample 2: Appending values to a list")
    try:
        response = append_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_append=["Cooking", "Gardening"],
        )
        print(
            f"Values appended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error appending to list: {e}")

    print("\nExample 3: Prepending values to a list")
    try:
        response = prepend_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_prepend=["Travel", "Music"],
        )
        print(
            f"Values prepended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error prepending to list: {e}")

    print("\nExample 4: Updating a specific list element")
    try:
        response = update_list_element(
            table_name=table_name,
            key=key,
            list_name="Interests",
```

```
            index=2,
            new_value="Mountain Hiking",
        )
        print(
            f"List element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating list element: {e}")

    print("\nExample 5: Removing a list element")
    try:
        response = remove_list_element(
            table_name=table_name, key=key, list_name="Interests", index=0
        )
        print(
            f"List element removed successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error removing list element: {e}")

    print("\nExample 6: Working with nested lists")
    try:
        # First, create an item with a nested structure
        dynamodb = boto3.resource("dynamodb")
        table = dynamodb.Table(table_name)

        table.update_item(
            Key={"UserId": "user456"},
            UpdateExpression="SET #skills = :skills",
            ExpressionAttributeNames={"#skills": "Skills"},
            ExpressionAttributeValues={
                ":skills": [
                    {"Category": "Programming", "Languages": ["Python", "Java",
"JavaScript"]},
                    {"Category": "Database", "Systems": ["DynamoDB", "MongoDB",
"PostgreSQL"]},
                ]
            },
        )

        # Now update a nested element
        response = update_nested_list_element(
```

```
            table_name=table_name,
            key={"UserId": "user456"},
            path="Skills[0].Languages[1]",
            new_value="TypeScript",
        )
        print(
            f"Nested list element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error working with nested lists: {e}")

    print("\nExample 7: Creating a list if it doesn't exist")
    try:
        response = create_list_if_not_exists(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Preferences",
            default_values=["Default1", "Default2", "Default3"],
        )
        print(
            f"List created with default values:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list with default values: {e}")

    print("\nExample 8: Safely appending to a list")
    try:
        response = append_to_list_safely(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Notifications",
            values_to_append=["New message received"],
            default_values=[],
        )
        print(f"Safely appended to list: {json.dumps(response.get('Attributes',
{}), default=str)}")
    except Exception as e:
        print(f"Error safely appending to list: {e}")

    print("\nKey Points About Working with Lists in DynamoDB:")
    print("1. Lists are ordered collections of elements that can be of different
types")
```

```
    print("2. Use the SET operation with direct assignment to create or replace a
list")
    print("3. Use list_append() to add elements to a list without replacing the
entire list")
    print("4. To append to the end: list_append(list_name, :values)")
    print("5. To prepend to the beginning: list_append(:values, list_name)")
    print("6. Use index notation list_name[index] to access or update specific
elements")
    print("7. Use the REMOVE action with index notation to remove specific
elements")
    print("8. Lists can contain nested structures like maps and other lists")
    print("9. Use if_not_exists() to create a list with default values if it
doesn't exist")
    print("10. List indices are zero-based (the first element is at index 0)")
    print("11. Attempting to access an index beyond the list bounds will result
in an error")
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Führen Sie Kartenoperationen in DynamoDB mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Map-Operationen in DynamoDB ausgeführt werden.

- Fügen Sie verschachtelte Attribute in Map-Strukturen hinzu und aktualisieren Sie sie.
- Entfernen bestimmter Felder aus Maps
- Arbeiten mit tief verschachtelten Map-Attributen

Java

SDK für Java 2.x

Demonstrieren Sie Kartenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // A single UpdateExpression can't reference both a map and a path inside
        // that same map (for example "#mapName" and "#mapName.#mapKey"):
 DynamoDB
        // rejects overlapping document paths with a ValidationException.
 Instead,
        // perform the update in two steps.

        // Step 1: create the map with an empty value if it doesn't already
 exist.
        dynamoDbClient.updateItem(UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("SET #mapName =
if_not_exists(#mapName, :emptyMap)")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(
                ":emptyMap", AttributeValue.builder().m(new
HashMap<>()).build()))
            .build());

        // Step 2: set the key within the now-guaranteed-to-exist map.
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {
```

```
        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
```

```
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
```

```
            if (response.item() != null && response.item().containsKey(mapName))
{
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Kartenoperationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");

        try {
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color",
AttributeValue.builder().s("Red").build());
            productDetails.put("Weight",
AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key,
"Details", productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());
```

```
            // Example 2: Update a specific attribute in a map
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());

            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested
map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap =
getMapAttribute(dynamoDbClient, tableName, key, "Details");

            if (currentMap != null) {
```

```
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
            System.out.println("3. You can update individual attributes without
overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes
in a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie Kartenoperationen mit AWS SDK für JavaScript.

```
/**
 * Example of updating map attributes in DynamoDB.
```

```
 *
 * This module demonstrates how to update map attributes that may not exist,
 * how to update nested attributes, and how to handle various map update
 scenarios.
 */

const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update a map attribute safely, handling the case where the map might not
 exist.
 *
 * This function demonstrates using the if_not_exists function to safely update
 * a map attribute that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeSafe(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
```

```
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return response;
  } catch (error) {
    // If the error is because the map doesn't exist, create it
    if (error.name === "ValidationException" &&
        error.message.includes("The document path provided in the update
 expression is invalid")) {

      // Create the map with the specified key-value pair
      const createParams = {
        TableName: tableName,
        Key: key,
        UpdateExpression: `SET ${mapName} = :map`,
        ExpressionAttributeValues: {
          ":map": { [mapKey]: value }
        },
        ReturnValues: "UPDATED_NEW"
      };

      return await docClient.send(new UpdateCommand(createParams));
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Update a map attribute using the if_not_exists function.
 *
 * This function demonstrates a more elegant approach using if_not_exists
 * to handle the case where the map doesn't exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeWithIfNotExists(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // A single UpdateExpression can't reference both a map and a path inside that
  // same map (for example `${mapName}` and `${mapName}.${mapKey}`): DynamoDB
  // rejects overlapping document paths with a ValidationException. Instead,
  // perform the update in two steps.

  // Step 1: create the map with an empty value if it doesn't already exist.
  await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`,
    ExpressionAttributeValues: {
      ":emptyMap": {}
    }
  }));

  // Step 2: set the key within the now-guaranteed-to-exist map.
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };
```

```
  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Add a value to a deeply nested map, creating parent maps if they don't exist.
 *
 * This function demonstrates how to update a deeply nested attribute,
 * creating any parent maps that don't exist along the way.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} path - The path to the nested attribute as an array of keys
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToNestedMap(
  config,
  tableName,
  key,
  path,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET";
  const expressionAttributeValues = {};

  // For each level in the path, create a map if it doesn't exist
  for (let i = 0; i < path.length; i++) {
    const currentPath = path.slice(0, i + 1).join(".");
    const parentPath = i > 0 ? path.slice(0, i).join(".") : null;

    if (parentPath) {
      updateExpression += ` ${parentPath} =
 if_not_exists(${parentPath}, :emptyMap${i}),`;
      expressionAttributeValues[`:emptyMap${i}`] = {};
    }
```

```
  }

  // Set the final value
  const fullPath = path.join(".");
  updateExpression += ` ${fullPath} = :value`;
  expressionAttributeValues[":value"] = value;

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update multiple fields in a map attribute in a single operation.
 *
 * This function demonstrates how to update multiple fields in a map
 * in a single DynamoDB operation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {Object} updates - Object containing key-value pairs to update
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMultipleMapFields(
  config,
  tableName,
  key,
  mapName,
  updates
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
```

```
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`;
  const expressionAttributeValues = {
    ":emptyMap": {}
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([field, value], index) => {
    updateExpression += `, ${mapName}.${field} = :val${index}`;
    expressionAttributeValues[`:val${index}`] = value;
  });

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
```

```
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}

/**
 * Example of how to use the map attribute update functions.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating different approaches to update map attributes in
 DynamoDB");

  try {
    // Example 1: Update a map attribute that might not exist (two-step approach)
    console.log("\nExample 1: Updating a map attribute that might not exist (two-
step approach)");
    const response1 = await updateMapAttributeSafe(
      config,
      tableName,
      key,
      "Preferences",
      "Theme",
      "Dark"
    );

    console.log("Updated preferences:", response1.Attributes);
```

```
    // Example 2: Update a map attribute using if_not_exists (elegant approach)
    console.log("\nExample 2: Updating a map attribute using if_not_exists
 (elegant approach)");
    const response2 = await updateMapAttributeWithIfNotExists(
      config,
      tableName,
      key,
      "Settings",
      "NotificationsEnabled",
      true
    );

    console.log("Updated settings:", response2.Attributes);

    // Example 3: Update a deeply nested attribute
    console.log("\nExample 3: Updating a deeply nested attribute");
    const response3 = await addToNestedMap(
      config,
      tableName,
      key,
      ["Profile", "Address", "City"],
      "Seattle"
    );

    console.log("Updated nested attribute:", response3.Attributes);

    // Example 4: Update multiple fields in a map
    console.log("\nExample 4: Updating multiple fields in a map");
    const response4 = await updateMultipleMapFields(
      config,
      tableName,
      key,
      "ContactInfo",
      {
        Email: "user@example.com",
        Phone: "555-123-4567",
        PreferredContact: "Email"
      }
    );

    console.log("Updated multiple fields:", response4.Attributes);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
```

```
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Explain the benefits of different approaches
    console.log("\nKey points about updating map attributes:");
    console.log("1. Use if_not_exists to handle maps that might not exist");
    console.log("2. Multiple updates can be combined in a single operation");
    console.log("3. Deeply nested attributes require creating parent maps");
    console.log("4. DynamoDB expressions are atomic - the entire update succeeds
 or fails");
    console.log("5. Using a single operation is more efficient than multiple
 separate updates");

  } catch (error) {
    console.error("Error:", error);
  }
}

// Export the functions
module.exports = {
  updateMapAttributeSafe,
  updateMapAttributeWithIfNotExists,
  addToNestedMap,
  updateMultipleMapFields,
  getItem,
  exampleUsage
};

// Run the example if this file is executed directly
if (require.main === module) {
  exampleUsage();
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie Kartenoperationen mit AWS SDK für Python (Boto3).

```
"""
Example of updating map attributes in DynamoDB.

This module demonstrates how to update map attributes in DynamoDB, including
handling cases where the map attribute might not exist yet.
"""


import boto3
from typing import Any, Dict, Optional



def update_map_attribute_safe(
    table_name: str, key: Dict[str, Any], map_name: str, map_key: str, value: Any
) -> Dict[str, Any]:
    """
    Update a specific key in a map attribute, creating the map if it doesn't
 exist.

    This function demonstrates how to safely update a key within a map attribute,
    even if the map doesn't exist yet in the item.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with attribute_not_exists to safely update the map
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map.#key = :value",
        ExpressionAttributeNames={"#map": map_name, "#key": map_key},
        ExpressionAttributeValues={":value": value},
```

```
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_nested_map(
    table_name: str, key: Dict[str, Any], path: str, value: Any
) -> Dict[str, Any]:
    """
    Add or update a value in a deeply nested map structure.

    This function demonstrates how to update a value at a specific path in a
    nested map structure, creating any intermediate maps as needed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested attribute (e.g.,
 "user.preferences.theme").
        value (Any): The value to set at the specified path.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
```

```
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_map_with_if_not_exists(
    table_name: str,
    key: Dict[str, Any],
    map_name: str,
    map_key: str,
    value: Any,
    default_map: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update a key in a map, creating the map with default values if it doesn't
 exist.

    This function demonstrates how to use if_not_exists to initialize a map with
    default values if it doesn't exist yet, and then update a specific key.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.
        default_map (Optional[Dict[str, Any]]): Default map values if the map
 doesn't exist.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Set default map if not provided
    if default_map is None:
        default_map = {}

    # Create a map with the new key-value pair
    updated_map = default_map.copy()
    updated_map[map_key] = value

    # Use if_not_exists to initialize the map if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map = if_not_exists(#map, :default_map)",
        ExpressionAttributeNames={"#map": map_name},
        ExpressionAttributeValues={":default_map": updated_map},
        ReturnValues="UPDATED_NEW",
    )

    return response


def merge_into_map(
    table_name: str, key: Dict[str, Any], map_name: str, values_to_merge:
 Dict[str, Any]
) -> Dict[str, Any]:
    """
    Merge multiple key-value pairs into a map attribute.

    This function demonstrates how to update multiple keys in a map attribute
    in a single operation, without overwriting the entire map.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        values_to_merge (Dict[str, Any]): Key-value pairs to merge into the map.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression for each key-value pair
    update_expression = "SET "
    expression_attribute_names = {"#map": map_name}
    expression_attribute_values = {}

    # Add each key-value pair to the update expression
    for i, (k, v) in enumerate(values_to_merge.items()):
        key_placeholder = f"#key{i}"
        value_placeholder = f":value{i}"

        expression_attribute_names[key_placeholder] = k
        expression_attribute_values[value_placeholder] = v

        if i > 0:
            update_expression += ", "
        update_expression += f"#map.{key_placeholder} = {value_placeholder}"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response




def example_usage():
    """Example of how to use the map attribute update functions."""
    # Example parameters
    table_name = "UserProfiles"
    key = {"UserId": "user123"}
```

```
    print("Example 1: Updating a specific key in a map attribute")
    try:
        response = update_map_attribute_safe(
            table_name=table_name, key=key, map_name="Preferences",
map_key="Theme", value="Dark"
        )
        print(f"Map attribute updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating map attribute: {e}")

    print("\nExample 2: Adding a value to a deeply nested map")
    try:
        response = add_to_nested_map(
            table_name=table_name, key=key, path="Settings.Notifications.Email",
value=True
        )
        print(f"Nested map updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating nested map: {e}")

    print("\nExample 3: Initializing a map with default values if it doesn't
exist")
    try:
        default_map = {"Language": "English", "Currency": "USD", "TimeZone":
"UTC"}

        response = update_map_with_if_not_exists(
            table_name=table_name,
            key={"UserId": "newuser456"},
            map_name="Preferences",
            map_key="Theme",
            value="Light",
            default_map=default_map,
        )
        print(f"Map initialized with defaults: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error initializing map: {e}")

    print("\nExample 4: Merging multiple values into a map")
    try:
        values_to_merge = {
            "NotificationsEnabled": True,
```

```
            "EmailFrequency": "Daily",
            "PushNotifications": False,
        }

        response = merge_into_map(
            table_name=table_name,
            key=key,
            map_name="NotificationSettings",
            values_to_merge=values_to_merge,
        )
        print(f"Multiple values merged into map: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error merging values into map: {e}")

    print("\nBest practices for working with map attributes in DynamoDB:")
    print("1. Use dot notation to access and update nested attributes")
    print("2. Use ExpressionAttributeNames to handle reserved words and special
 characters")
    print("3. Use if_not_exists() to handle cases where attributes might not
 exist")
    print("4. Update specific map keys rather than overwriting the entire map")
    print("5. Use a single update operation to modify multiple map keys for
 better performance")
    print("6. Consider your data model carefully to minimize the need for deeply
 nested attributes")



if __name__ == "__main__":
    example_usage()
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Führen Sie Mengenoperationen in DynamoDB mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Set-Operationen in DynamoDB ausgeführt werden.

- Hinzufügen von Elementen zu einem Set-Attribut
- Entfernen von Elementen aus einem Set-Attribut

- Verwenden der Operationen ADD und DELETE mit Sätzen

Java

SDK für Java 2.x

Demonstrieren Sie festgelegte Operationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {
```

```
        // Create a string set value from the values to add
        AttributeValue setValue =
AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }
```

```
        // Create a number set value from the values to add
        AttributeValue setValue =
AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove
values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {
```

```
        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers
for a number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
```

```
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);

            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the
set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
```

```
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not
exist in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
```

```
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers
for a number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Mengenoperationen mit AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating set operations in DynamoDB");

       try {
           // Example 1: Create a string set with multiple values
           System.out.println("\nExample 1: Creating a string set with multiple
values");
           Set<String> tags = new HashSet<>();
           tags.add("Electronics");
           tags.add("Gadget");
           tags.add("Smartphone");

           UpdateItemResponse createResponse = createSetWithValues(
               dynamoDbClient, tableName, key, "Tags", tags, false // Not a
number set
               );

           System.out.println("Created set attribute: " +
createResponse.attributes());

           // Example 2: Add values to a string set
           System.out.println("\nExample 2: Adding values to a string set");
           Set<String> additionalTags = new HashSet<>();
           additionalTags.add("Mobile");
           additionalTags.add("Wireless");

           UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);

           System.out.println("Updated set attribute: " +
addResponse.attributes());

           // Example 3: Create a number set with multiple values
           System.out.println("\nExample 3: Creating a number set with multiple
values");
           Set<Number> ratings = new HashSet<>();
           ratings.add(4);
```

```
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );

            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false //
Not a number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());

            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a
set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));
```

```
            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number
sets (NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
            System.out.println("8. You can create a set with multiple values in a
single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
```

```
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie die Bedienung von Sets mithilfe von AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Add elements to a set attribute.
 *
 * This function demonstrates using the ADD operation to add elements to a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to add to the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove elements from a set attribute.
 *
 * This function demonstrates using the DELETE operation to remove elements from
 a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to remove from the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeFromSet(
```

```
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using DELETE
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `DELETE ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a new set attribute with initial values.
 *
```

```
 * This function demonstrates using the SET operation to create a new set
 attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The initial values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using SET
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${setName} = :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
```

```
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Replace an entire set attribute with a new set of values.
 *
 * This function demonstrates using the SET operation to replace an entire set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The new values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function replaceSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // This is the same as createSet, but included for clarity of intent
  return await createSet(config, tableName, key, setName, values, setType);
}

/**
 * Remove the last element from a set and handle the empty set case.
 *
 * This function demonstrates what happens when you delete the last element of a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @returns {Promise<Object>} - The result of the operation
```

```
 */
async function removeLastElementFromSet(
  config,
  tableName,
  key,
  setName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // First, get the current item to check the set
  const currentItem = await getItem(config, tableName, key);

  // Check if the set exists and has elements
  if (!currentItem || !currentItem[setName] || currentItem[setName].size === 0) {
    return {
      success: false,
      message: "Set doesn't exist or is already empty",
      item: currentItem
    };
  }

  // Get the set values
  const setValues = Array.from(currentItem[setName]);

  // If there's only one element left, remove the attribute entirely
  if (setValues.length === 1) {
    // Define the update parameters to remove the attribute
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: `REMOVE ${setName}`,
      ReturnValues: "UPDATED_NEW"
    };

    // Perform the update operation
    await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Last element removed, attribute has been deleted",
      removedValue: setValues[0]
    };
```

```
  } else {
    // Otherwise, remove just the last element
    // Create a set with just the last element
    const lastElement = setValues[setValues.length - 1];
    const setType = typeof lastElement === 'number' ? 'number' : 'string';

    // Remove the last element
    const response = await removeFromSet(
      config,
      tableName,
      key,
      setName,
      [lastElement],
      setType
    );

    return {
      success: true,
      message: "Last element removed, set still contains elements",
      removedValue: lastElement,
      remainingSet: response.Attributes[setName]
    };
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Mengenoperationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to work with sets in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating set operations in DynamoDB");

  try {
    // Example 1: Create a string set
    console.log("\nExample 1: Creating a string set");
    const response1 = await createSet(
      config,
      tableName,
      key,
      "Interests",
      ["Reading", "Hiking", "Cooking"],
      "string"
    );

    console.log("Created set:", response1.Attributes);

    // Example 2: Add elements to a set
    console.log("\nExample 2: Adding elements to a set");
```

```
    const response2 = await addToSet(
      config,
      tableName,
      key,
      "Interests",
      ["Photography", "Travel"],
      "string"
    );

    console.log("Updated set after adding elements:", response2.Attributes);

    // Example 3: Remove elements from a set
    console.log("\nExample 3: Removing elements from a set");
    const response3 = await removeFromSet(
      config,
      tableName,
      key,
      "Interests",
      ["Cooking"],
      "string"
    );

    console.log("Updated set after removing elements:", response3.Attributes);

    // Example 4: Create a number set
    console.log("\nExample 4: Creating a number set");
    const response4 = await createSet(
      config,
      tableName,
      key,
      "FavoriteNumbers",
      [7, 42, 99],
      "number"
    );

    console.log("Created number set:", response4.Attributes);

    // Example 5: Replace an entire set
    console.log("\nExample 5: Replacing an entire set");
    const response5 = await replaceSet(
      config,
      tableName,
      key,
      "Interests",
```

```
      ["Gaming", "Movies", "Music"],
      "string"
    );

    console.log("Replaced set:", response5.Attributes);

    // Example 6: Remove the last element from a set
    console.log("\nExample 6: Removing the last element from a set");

    // First, create a set with just one element
    await createSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags",
      ["LastTag"],
      "string"
    );

    // Then, remove the last element
    const response6 = await removeLastElementFromSet(
      config,
      tableName,
      { UserId: "U67890" },
      "Tags"
    );

    console.log(response6.message);
    console.log("Removed value:", response6.removedValue);

    // Get the final state of the items
    console.log("\nFinal state of the items:");
    const item1 = await getItem(config, tableName, key);
    console.log("User U12345:", JSON.stringify(item1, null, 2));

    const item2 = await getItem(config, tableName, { UserId: "U67890" });
    console.log("User U67890:", JSON.stringify(item2, null, 2));

    // Explain set operations
    console.log("\nKey points about set operations in DynamoDB:");
    console.log("1. Use ADD to add elements to a set (duplicates are
automatically removed)");
    console.log("2. Use DELETE to remove elements from a set");
    console.log("3. Use SET to create a new set or replace an existing one");
```

```
    console.log("4. DynamoDB supports three types of sets: string sets, number
 sets, and binary sets");
    console.log("5. When you delete the last element from a set, the attribute
 remains as an empty set");
    console.log("6. To remove an empty set, use the REMOVE operation");
    console.log("7. Sets automatically maintain unique values (no duplicates)");
    console.log("8. You cannot mix data types within a set");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie festgelegte Operationen mit AWS SDK für Python (Boto3).

```
import boto3
from typing import Any, Dict, List


def create_set_attribute(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    set_values: List[Any],
    set_type: str = "string",
) -> Dict[str, Any]:
    """
    Create a new set attribute or add elements to an existing set.

    This function demonstrates how to use the ADD operation to create a new set
    or add elements to an existing set.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
```

```
        set_values (List[Any]): The values to add to the set.
        set_type (str, optional): The type of set to create: "string", "number",
 or "binary".
            Defaults to "string".

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a DynamoDB set based on the specified type
    if set_type == "string":
        dynamo_set = set(str(value) for value in set_values)
    elif set_type == "number":
        # We need to use actual float values for the DynamoDB API
        # but mypy expects strings in sets, so we need to use type: ignore
        dynamo_set = set(float(value) for value in set_values)  # type: ignore
    else:  # binary set is not directly supported in high-level API, handled
 differently
        raise ValueError("Binary sets are not supported in this example")

    # Use the ADD operation to create or update the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :set_values",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":set_values": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_add: List[Any]
) -> Dict[str, Any]:
    """
    Add elements to an existing set attribute.

    This function demonstrates how to use the ADD operation to add elements to an
 existing set.
```

```
    If the set doesn't exist, it will be created.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_add (List[Any]): The values to add to the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_add)

    # Use the ADD operation to add values to the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :values_to_add",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_add": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_from_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_remove:
 List[Any]
) -> Dict[str, Any]:
    """
    Remove elements from a set attribute.

    This function demonstrates how to use the DELETE operation to remove elements
 from a set.
    If the last element is removed, the attribute will be deleted entirely.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_remove (List[Any]): The values to remove from the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_remove)

    # Use the DELETE operation to remove values from the set
    response = table.update_item(
        Key=key,
        UpdateExpression="DELETE #set_attr :values_to_remove",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_remove": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def check_if_set_exists(table_name: str, key: Dict[str, Any], set_name: str) ->
 bool:
    """
    Check if a set attribute exists in an item.

    This function demonstrates how to check if a set attribute exists after
    potentially removing all elements from it.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to check.
        set_name (str): The name of the set attribute.

    Returns:
        bool: True if the set attribute exists, False otherwise.
    """
    # Initialize the DynamoDB resource
```

```
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Get the item
    response = table.get_item(
        Key=key, ProjectionExpression="#set_attr",
 ExpressionAttributeNames={"#set_attr": set_name}
    )

    # Check if the item exists and has the set attribute
    return "Item" in response and set_name in response["Item"]


def demonstrate_last_element_removal(
    table_name: str, key: Dict[str, Any], set_name: str
) -> Dict[str, Any]:
    """
    Demonstrate what happens when you remove the last element from a set.

    This function creates a set with a single element, then removes that element,
    showing that the attribute is completely removed when the last element is
 deleted.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Step 1: Create a set with a single element
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=["last_element"],
        set_type="string",
    )

    # Step 2: Check that the set exists
    exists_before = check_if_set_exists(table_name, key, set_name)

    # Step 3: Remove the last element
```

```
    delete_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
 values_to_remove=["last_element"]
    )

    # Step 4: Check if the set still exists
    exists_after = check_if_set_exists(table_name, key, set_name)

    # Return the results
    return {
        "create_response": create_response,
        "exists_before": exists_before,
        "delete_response": delete_response,
        "exists_after": exists_after,
    }


def work_with_number_set(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    initial_values: List[float],
    values_to_add: List[float],
    values_to_remove: List[float],
) -> Dict[str, Any]:
    """
    Demonstrate working with a number set in DynamoDB.

    This function shows how to create and manipulate a set of numbers.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        initial_values (List[float]): The initial values for the set.
        values_to_add (List[float]): Values to add to the set.
        values_to_remove (List[float]): Values to remove from the set.

    Returns:
        Dict[str, Any]: A dictionary containing the responses from each
 operation.
    """
    # Step 1: Create the number set
    create_response = create_set_attribute(
```

```
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=initial_values,
        set_type="number",
    )

    # Step 2: Add more numbers to the set
    add_response = add_to_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_add=values_to_add
    )

    # Step 3: Remove some numbers from the set
    remove_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_remove=values_to_remove
    )

    # Step 4: Get the final state
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    get_response = table.get_item(
        Key=key,
        ProjectionExpression=f"#{set_name}",
        ExpressionAttributeNames={f"#{set_name}": set_name},
    )

    # Return all responses
    return {
        "create_response": create_response,
        "add_response": add_response,
        "remove_response": remove_response,
        "final_state": get_response.get("Item", {}),
    }
```

Beispiel für die Verwendung von Mengenoperationen mit AWS SDK für Python (Boto3).

```
def example_usage():
```

```
    """Example of how to use the set operations functions."""
    # Example parameters
    table_name = "UserPreferences"
    key = {"UserId": "user123"}

    print("Example 1: Creating a string set attribute")
    try:
        response = create_set_attribute(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            set_values=["AWS", "DynamoDB", "NoSQL"],
            set_type="string",
        )
        print(f"Set attribute created successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error creating set attribute: {e}")

    print("\nExample 2: Adding elements to an existing set")
    try:
        response = add_to_set(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            values_to_add=["Database", "Serverless"],
        )
        print(f"Elements added to set successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error adding to set: {e}")

    print("\nExample 3: Removing elements from a set")
    try:
        response = remove_from_set(
            table_name=table_name, key=key, set_name="FavoriteTags",
values_to_remove=["NoSQL"]
        )
        print(f"Elements removed from set successfully:
{response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error removing from set: {e}")
```

```
    print("\nExample 4: Demonstrating what happens when you remove the last
element from a set")
    try:
        results = demonstrate_last_element_removal(
            table_name=table_name, key={"UserId": "tempUser"},
set_name="SingleElementSet"
        )

        print(f"Set exists before removal: {results['exists_before']}")
        print(f"Set exists after removal: {results['exists_after']}")

        if not results["exists_after"]:
            print("The set attribute was completely removed when the last element
was deleted.")
        else:
            print("The set attribute still exists after removing the last
element.")
    except Exception as e:
        print(f"Error in last element removal demonstration: {e}")

    print("\nExample 5: Working with a number set")
    try:
        results = work_with_number_set(
            table_name=table_name,
            key={"UserId": "user456"},
            set_name="LuckyNumbers",
            initial_values=[7, 13, 42],
            values_to_add=[99, 100],
            values_to_remove=[13],
        )

        print(f"Initial number set: {results['create_response'].get('Attributes',
{})}")
        print(f"After adding numbers: {results['add_response'].get('Attributes',
{})}")
        print(f"After removing numbers:
{results['remove_response'].get('Attributes', {})}")
        print(f"Final state: {results['final_state']}")
    except Exception as e:
        print(f"Error working with number set: {e}")

    print("\nKey Points About DynamoDB Sets:")
    print("1. Sets can only contain elements of the same type (string, number, or
binary)")
```

```
    print("2. Sets automatically eliminate duplicate values")
    print("3. The ADD operation creates a set if it doesn't exist")
    print("4. The DELETE operation removes specified elements from a set")
    print("5. When the last element is removed from a set, the entire attribute
 is deleted")
    print("6. Empty sets are not allowed in DynamoDB")
    print("7. Sets are unordered collections")
    print("8. The ADD operation is atomic for sets")
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Abfragen einer DynamoDB-Tabelle mithilfe von Batches von PartiQL-Anweisungen und AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
```

```
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
const string movieFileName = @"..\..\..\..\..\..\..\..\resources\sample_files
\movies.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
{
    Console.WriteLine($"Successfully created table: {tableName}.");
}

WaitForEnter();

// Add movie information to the table from moviedata.json. See the
// instructions at the top of this file to download the JSON file.
Console.WriteLine($"Inserting movies into the new table. Please wait...");
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);
if (success)
{
    Console.WriteLine("Movies successfully added to the table.");
}
else
{
    Console.WriteLine("Movies could not be added to the table.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var title1 = "Star Wars";
var year1 = 1977;
var title2 = "Wizard of Oz";
var year2 = 1939;

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
```

```
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1,
 year2);
if (success)
{
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");
}
else
{
    Console.WriteLine("Select statement failed.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var producer1 = "LucasFilm";
var producer2 = "MGM";

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1,
 year1, producer2, title2, year2);
if (success)
{
    Console.WriteLine($"Successfully updated {title1} and {title2}.");
}
else
{
    Console.WriteLine("Update failed.");
}

WaitForEnter();

// Delete multiple movies by using the BatchExecute statement.
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2,
 year2);

if (success)
{
    Console.WriteLine($"Deleted {title1} and {title2}");
}
else
{
    Console.WriteLine($"could not delete {title1} or {title2}");
```

```
}

WaitForEnter();

// DNow that the PartiQL Batch scenario is complete, delete the movie table.
success = await DynamoDBMethods.DeleteTableAsync(tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
void DisplayInstructions()
{
    Console.Clear();
    Console.WriteLine();
    Console.Write(new string(' ', 24));
    Console.WriteLine("DynamoDB PartiQL Basics Example");
    Console.WriteLine(SepBar);
    Console.WriteLine("This demo application shows the basics of using Amazon
 DynamoDB with the AWS SDK for");
    Console.WriteLine(".NET version 3.7 and .NET 6.");
    Console.WriteLine(SepBar);
    Console.WriteLine("Creates a table by using the CreateTable method.");
    Console.WriteLine("Gets multiple movies by using a PartiQL SELECT
 statement.");
    Console.WriteLine("Updates multiple movies by using the ExecuteBatch
 method.");
    Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE
 statement.");
    Console.WriteLine("Cleans up the resources created for the demo by deleting
 the table.");
    Console.WriteLine(SepBar);

    WaitForEnter();
}
```

```
/// <summary>
/// Simple method to wait for the <Enter> key to be pressed.
/// </summary>
void WaitForEnter()
{
    Console.WriteLine("\nPress <Enter> to continue.");
    Console.Write(SepBar);
    _ = Console.ReadLine();
}


        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
```

```
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {

Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }

        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
```

```
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });
```

```
                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
```

```
                return null!;
            }
        }

        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</
param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</
param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title
 = ? AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
```

```
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }

        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {
```

```
            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND
year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
  \sa partiqlBatchExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
```

```
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/
n) "));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

 attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute
 = Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);
```

```
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
              << std::endl;

    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
```

```
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }
```

```
   std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
               titles.size());

       std::stringstream sqlStream;
       sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                 << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                 << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


       std::string sql(sqlStream.str());

       for (size_t i = 0; i < statements.size(); ++i) {
           statements[i].SetStatement(sql);

           Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);
       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to update movie information: "
                     << outcome.GetError().GetMessage() << std::endl;
           return false;
       }
   }

   std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
             << std::endl;
```

```
   // 5. Get the updated data for multiple movies using "Select" statements.
(BatchExecuteStatement)
   {
       Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
               titles.size());
       std::stringstream sqlStream;
       sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       std::string sql(sqlStream.str());

       for (size_t i = 0; i < statements.size(); ++i) {
           statements[i].SetStatement(sql);
           Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
           attributes.push_back(
                   Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
           statements[i].SetParameters(attributes);
       }

       Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

       request.SetStatements(statements);

       Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
               request);
       if (outcome.IsSuccess()) {
           const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

           const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

           for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
               const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

               printMovieInfo(item);
           }
       }
```

```
        else {
            std::cerr << "Failed to retrieve the movies information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
              << std::endl;

    // 6. Delete multiple movies using "Delete" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
    }

    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
```

```
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
 std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
```

```
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und Stapel von PartiQL-Abfragen ausführt.

```
import (
```

```
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
// to run batches of PartiQL statements to query a table that stores data about
 movies.
//
//   - Use batches of PartiQL statements to add, get, update, and delete data for
//     individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
```

```
exists, err := tableBasics.TableExists(ctx)
if err != nil {
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{{
 Title: "House PartiQL",
 Year:  currentYear - 5,
 Info: map[string]interface{}{
  "plot":   "Wacky high jinks result from querying a mysterious database.",
  "rating": 8.5}}, {
 Title: "House PartiQL 2",
 Year:  currentYear - 3,
 Info: map[string]interface{}{
  "plot":   "Moderate high jinks result from querying another mysterious
database.",
  "rating": 6.5}}, {
 Title: "House PartiQL 3",
 Year:  currentYear - 1,
 Info: map[string]interface{}{
  "plot":   "Tepid high jinks result from querying yet another mysterious
database.",
  "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
err = runner.AddMovieBatch(ctx, customMovies)
if err == nil {
 log.Printf("Added %v movies to the table.\n", len(customMovies))
}
```

```
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(ctx, customMovies)
if err == nil {
 for _, movie := range movies {
  log.Println(movie)
 }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
log.Println("Updating a batch of movies with new ratings.")
err = runner.UpdateMovieBatch(ctx, customMovies, newRatings)
if err == nil {
 log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting projected data from the table to verify our update.")
log.Println("Using a page size of 2 to demonstrate paging.")
projections, err := runner.GetAllMovies(ctx, 2)
if err == nil {
 log.Println("All movies:")
 for _, projection := range projections {
  log.Println(projection)
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Deleting a batch of movies.")
err = runner.DeleteMovieBatch(ctx, customMovies)
if err == nil {
 log.Printf("Deleted %v movies.\n", len(customMovies))
}

err = tableBasics.DeleteTable(ctx)
if err == nil {
 log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
```

```
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
```

```
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die PartiQL-Anweisungen ausführen.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
 to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
```

```
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year, movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
 err)
 }
 return err
}



// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
```

```
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
  }
 }
 return outMovies, err
}



// GetAllMovies runs a PartiQL SELECT statement to get all movies from the
 DynamoDB table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
```

```
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}



// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the
 rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}



// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("Creating an Amazon DynamoDB table named " + tableName
            + " with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Adding multiple records into the " + tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        System.out.println("Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
```

```
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND
year = ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
```

```
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
 scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
```

```
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse = dbWaiter
                .waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
 BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();
```

```
            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);
```

```
            BatchStatementRequest statementRequestMovie3 =
 BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);

            BatchExecuteStatementRequest batchRequest =
 BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
 ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
 response.toString());
            System.out.println("Added new movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
```

```
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();
```

```
        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
```

```
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
```

```
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
{
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,

List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
```

```
    }
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie Batch-PartiQL-Anweisungen aus.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "Cities";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
```

```
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({}, { confirmAll });
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
```

```
      {
        AttributeName: "name",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "name", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert items.
   */

  log("Inserting cities into the table.");
  const addItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statements: [
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["Alachua", 10712],
      },
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
```

```
        Parameters: ["High Springs", 6415],
      },
    ],
  });
  await docClient.send(addItemsStatementCommand);
  log("Cities inserted.");

  /**
   * Select items.
   */

  log("Selecting cities from the table.");
  const selectItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statements: [
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  const selectItemResponse = await docClient.send(selectItemsStatementCommand);
  log(
    `Got cities: ${selectItemResponse.Responses.map(
      (r) => `${r.Item.name} (${r.Item.population})`,
    ).join(", ")}`,
  );

  /**
   * Update items.
   */

  log("Modifying the populations.");
  const updateItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statements: [
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
```

```
        Parameters: [10, "Alachua"],
      },
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [5, "High Springs"],
      },
    ],
  });
  await docClient.send(updateItemStatementCommand);
  log("Updated cities.");

  /**
   * Delete the items.
   */

  log("Deleting the cities.");
  const deleteItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statements: [
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Cities deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named
 id and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }
```

```
    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
 'title' : ?, 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
```

```
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
```

```
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
```

```
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
```

```
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
```

```
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo
 using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
```

```
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
```

```
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']
['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
 lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
 born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
 of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
```

```
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array
 $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }
```

```
        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array
$parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Stapel von PartiQL-Anweisungen ausführen kann.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
 the
        resource transforms input and output from plain old Python objects
 (POPOs) to
```

```
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated
with
                           each statement. This list must be in the same order as
the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the
table "
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why:
%s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen in Stapeln ausführt.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    movie_data = [
        {
            "title": f"House PartiQL",
            "year": datetime.now().year - 5,
            "info": {
                "plot": "Wacky high jinks result from querying a mysterious
 database.",
                "rating": Decimal("8.5"),
            },
        },
        {
            "title": f"House PartiQL 2",
            "year": datetime.now().year - 3,
            "info": {
                "plot": "Moderate high jinks result from querying another
 mysterious database.",
                "rating": Decimal("6.5"),
            },
        },
        {
            "title": f"House PartiQL 3",
            "year": datetime.now().year - 1,
            "info": {
                "plot": "Tepid high jinks result from querying yet another
 mysterious database.",
                "rating": Decimal("2.5"),
            },
        },
    ]

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
```

```
        f'INSERT INTO "{table_name}" ' f"VALUE {{'title': ?, 'year': ?,
'info': ?}}"
    ] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting data for a batch of movies.")
    statements = [f'SELECT * FROM "{table_name}" WHERE title=? AND year=?'] *
len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    output = wrapper.run_partiql(statements, params)
    for item in output["Responses"]:
        print(f"\n{item['Item']['title']}, {item['Item']['year']}")
        pprint(item["Item"])
    print("-" * 88)

    ratings = [Decimal("7.7"), Decimal("5.5"), Decimal("1.3")]
    print(f"Updating a batch of movies with new ratings.")
    statements = [
        f'UPDATE "{table_name}" SET info.rating=? ' f"WHERE title=? AND year=?"
    ] * len(movie_data)
    params = [
        [rating, movie["title"], movie["year"]]
        for rating, movie in zip(ratings, movie_data)
    ]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting projected data from the table to verify our update.")
    output = wrapper.dyn_resource.meta.client.execute_statement(
        Statement=f'SELECT title, info.rating FROM "{table_name}"'
    )
    pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting a batch of movies from the table.")
    statements = [f'DELETE FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
```

```
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem eine Tabelle erstellt wird und PartiQL-Stapelabfragen ausgeführt werden.

```
  table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
```

```
  sdk = DynamoDBPartiQLBatch.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Populate DynamoDB table with movie data.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a batch of items from the movies table.')
  puts "Let's select some popular movies for side-by-side comparison."
  response = sdk.batch_execute_select([['Mean Girls', 2004], ['Goodfellas',
 1977], ['The Prancing of the Lambs', 2005]])
  puts("Items selected: #{response['responses'].length}\n")
  print "\nDone!\n".green

  new_step(4, 'Delete a batch of items from the movies table.')
  sdk.batch_execute_write([['Mean Girls', 2004], ['Goodfellas', 1977], ['The
 Prancing of the Lambs', 2005]])
  print "\nDone!\n".green

  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für Ruby API-Referenz.

Fragen Sie eine DynamoDB-Tabelle mit PartiQL ab und AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
        private static readonly AmazonDynamoDBClient Client = new
 AmazonDynamoDBClient();


        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table where the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
 movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;
```

```
            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE
{{'title': ?, 'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset +=
25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await
Client.BatchExecuteStatementAsync(new BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully
added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
```

```
                }
            }
            catch (AmazonDynamoDBException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        return success;
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public static List<Movie> ImportMovies(string movieFileName)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        if (allMovies is not null)
        {
            // Return the first 250 entries.
            return allMovies.GetRange(0, 250);
        }
        else
        {
            return null!;
        }
    }



    /// <summary>
    /// Uses a PartiQL SELECT statement to retrieve a single movie from the
    /// movie database.
```

```
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
```

```
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }


        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
```

```
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
```

```
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Displays the list of movies returned from a database query.
        /// </summary>
        /// <param name="items">The list of movie information to display.</param>
        private static void DisplayMovies(List<Dictionary<string,
 AttributeValue>> items)
        {
            if (items.Count > 0)
            {
                Console.WriteLine($"Found {items.Count} movies.");
                items.ForEach(item =>
 Console.WriteLine($"{item["year"].N}\t{item["title"].S}"));
            }
            else
            {
                Console.WriteLine($"Didn't find a movie that matched the supplied
 criteria.");
            }
        }


    }
}



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
```

```
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
```

```
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
```

```
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {
```

```
            AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using single PartiQL
 statements.
/*!
  \sa partiqlExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
 it? ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
```

```
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    //  3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
```

```
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
```

```
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

       request.SetStatement(sqlStream.str());

       Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

       request.SetParameters(attributes);

       Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
               request);

       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to update a movie: "
                     << outcome.GetError().GetMessage();
           return false;
       }
   }

   std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

   //  5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
   {
       Aws::DynamoDB::Model::ExecuteStatementRequest request;
       std::stringstream sqlStream;
       sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       request.SetStatement(sqlStream.str());

       Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
       request.SetParameters(attributes);

       Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
               request);
       if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were
retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    std::cout << "Deleting the movie" << std::endl;

    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
```

```
            return false;
        }
    }

    std::cout << "Movie successfully deleted." << std::endl;
    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);
```

```
        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." <<
 std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
```

```
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
```

```
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
import (
```

```
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual
 movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(ctx context.Context, sdkConfig aws.Config,
 tableName string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }

 exists, err := tableBasics.TableExists(ctx)
```

```
if err != nil {
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
 Title: "24 Hour PartiQL People",
 Year:  currentYear,
 Info: map[string]interface{}{
  "plot":   "A group of data developers discover a new query language they can't
stop using.",
  "rating": 9.9,
 },
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
movie, err := runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

newRating := 6.6
```

```
 log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
 newRating)
 err = runner.UpdateMovie(ctx, customMovie, newRating)
 if err == nil {
  log.Printf("Updated %v with a new rating.\n", customMovie.Title)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Getting data again to verify the update.")
 movie, err = runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
 if err == nil {
  log.Println(movie)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Deleting movie '%v'.\n", customMovie.Title)
 err = runner.DeleteMovie(ctx, customMovie)
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite
 primary key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die PartiQL-Anweisungen ausführen.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB
 table by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie
 that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
```

```
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}



// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the
 DynamoDB table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a
 key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("Updating a record.");
        updateTableItem(ddb);

        System.out.println("Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
```

```
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();
```

```
            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf(year))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s(title)
```

```
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s(info)
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " + title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and
title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
```

```
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
 {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse
 executeStatementRequest(DynamoDbClient ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie einzelne PartiQL-Anweisungen aus.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "SingleOriginCoffees";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
```

```
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({});
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "varietal",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
```

```
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "varietal", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert an item.
   */

  log("Inserting a coffee into the table.");
  const addItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statement: `INSERT INTO ${tableName} value {'varietal':?, 'profile':?}`,
    Parameters: ["arabica", ["chocolate", "floral"]],
  });
  await client.send(addItemStatementCommand);
  log("Coffee inserted.");

  /**
   * Select an item.
   */

  log("Selecting the coffee from the table.");
  const selectItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statement: `SELECT * FROM ${tableName} WHERE varietal=?`,
```

```
    Parameters: ["arabica"],
  });
  const selectItemResponse = await docClient.send(selectItemStatementCommand);
  log(`Got coffee: ${JSON.stringify(selectItemResponse.Items[0])}`);

  /**
   * Update the item.
   */

  log("Add a flavor profile to the coffee.");
  const updateItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statement: `UPDATE ${tableName} SET profile=list_append(profile, ?) WHERE
 varietal=?`,
    Parameters: [["fruity"], "arabica"],
  });
  await client.send(updateItemStatementCommand);
  log("Updated coffee");

  /**
   * Delete the item.
   */

  log("Deleting the coffee.");
  const deleteItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statement: `DELETE FROM ${tableName} WHERE varietal=?`,
    Parameters: ["arabica"],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Coffee deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key
 named id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
```

```
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}
```

```
suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}
```

```
suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
```

```
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
```

```
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo
 using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);
```

```
        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']
['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1
 || $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
```

```
        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
{$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something
lighter, in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were
born:\n";
        $oops = "Oops! There were no movies released in that year (that we know
of).\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
```

```
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
```

```
            'Parameters' => $parameters,
        ]);
    }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die PartiQL-Anweisungen ausführen kann.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
```

```
    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because
the
        resource transforms input and output from plain old Python objects
(POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't
 stop using."
    rating = Decimal("9.9")

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"{table_name}\" VALUE {{'title': ?, 'year': ?,
 'info': ?}}",
        [title, year, {"plot": plot, "rating": rating}],
    )
    print("Success!")
    print("-" * 88)

    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    rating = Decimal("2.4")
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f'UPDATE "{table_name}" SET info.rating=? WHERE title=? AND year=?',
        [rating, title, year],
    )
    print("Success!")
    print("-" * 88)
```

```
    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f'DELETE FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- Einzelheiten zur API finden Sie ExecuteStatementin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLSingle.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, 'Populate DynamoDB table with movie data.')
download_file = 'moviedata.json'
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, 'Select a single item from the movies table.')
response = sdk.select_item_by_title('Star Wars')
puts("Items selected for title 'Star Wars': #{response.items.length}\n")
print response.items.first.to_s.yellow
print "\n\nDone!\n".green

new_step(4, 'Update a single item from the movies table.')
puts "Let's correct the rating on The Big Lebowski to 10.0."
sdk.update_rating_by_title('The Big Lebowski', 1998, 10.0)
print "\nDone!\n".green
```

```
  new_step(5, 'Delete a single item from the movies table.')
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title('The Silence of the Lambs', 1991)
  print "\nDone!\n".green

  new_step(6, 'Insert a new item into the movies table.')
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item('The Prancing of the Lambs', 2005, 'A movie about happy
 livestock.', 5.0)
  print "\nDone!\n".green

  new_step(7, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<(), SdkError<CreateTableError>> {
    let ad = AttributeDefinition::builder()
        .attribute_name(key)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .expect("creating AttributeDefinition");

    let ks = KeySchemaElement::builder()
```

```
        .attribute_name(key)
        .key_type(KeyType::Hash)
        .build()
        .expect("creating KeySchemaElement");

    match client
        .create_table()
        .table_name(table)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn add_item(client: &Client, item: Item) -> Result<(),
 SdkError<ExecuteStatementError>> {
    match client
        .execute_statement()
        .statement(format!(
            r#"INSERT INTO "{}" VALUE {{
                "{}": ?,
                "acount_type": ?,
                "age": ?,
                "first_name": ?,
                "last_name": ?
        }} "#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![
            AttributeValue::S(item.utype),
            AttributeValue::S(item.age),
            AttributeValue::S(item.first_name),
            AttributeValue::S(item.last_name),
        ]))
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
```

```
    }
}

async fn query_item(client: &Client, item: Item) -> bool {
    match client
        .execute_statement()
        .statement(format!(
            r#"SELECT * FROM "{}" WHERE "{}" = ?"#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![AttributeValue::S(item.value)]))
        .send()
        .await
    {
        Ok(resp) => {
            if !resp.items().is_empty() {
                println!("Found a matching entry in the table:");
                println!("{:?}", resp.items.unwrap_or_default().pop());
                true
            } else {
                println!("Did not find a match.");
                false
            }
        }
        Err(e) => {
            println!("Got an error querying table:");
            println!("{}", e);
            process::exit(1);
        }
    }
}

async fn remove_item(client: &Client, table: &str, key: &str, value: String) ->
 Result<(), Error> {
    client
        .execute_statement()
        .statement(format!(r#"DELETE FROM "{table}" WHERE "{key}" = ?"#))
        .set_parameters(Some(vec![AttributeValue::S(value)]))
        .send()
        .await?;

    println!("Deleted item.");

    Ok(())
```

```
}

async fn remove_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der API-Referenz zum AWS SDK für Rust.

Abfragen einer DynamoDB-Tabelle mithilfe eines globalen sekundären Indexes mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit einem globalen sekundären Index abgefragt wird.

- Abfragen einer DynamoDB-Tabelle mit ihrem Primärschlüssel
- Abfragen eines globalen sekundären Index (GSI) für alternative Zugriffsmuster
- Vergleichen von Tabellenabfragen und GSI-Abfragen

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit ihrem Primärschlüssel und einem Global Secondary Index (GSI) mit ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
```

```
    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
```

```
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " +
response.count() + " items");
            return response;
```

```
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI",
e);
        }
    }
```

Vergleichen Sie die direkte Abfrage einer Tabelle mit der Abfrage einer GSI mit. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition
key attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        try (DynamoDbClient ddb =
DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " +
basePartitionKeyName + " = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName +
" = " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in
GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the
base table (projection)");
```

```
            System.out.println("3. GSIs consume read capacity units from the
 GSI's capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data
 (cannot use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mithilfe des Primärschlüssels mit ab. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
```

```
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}
```

Fragen Sie einen DynamoDB Global Secondary Index (GSI) mit ab. AWS SDK für JavaScript

```
/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
```

```
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit ihrem Primärschlüssel und einem Global Secondary Index (GSI) mit ab. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Fragen Sie eine DynamoDB-Tabelle mithilfe einer Begins_with-Bedingung mit einem ab AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit einer „begins_with“-Bedingung abgefragt wird.

- Verwenden der Funktion „begins_with“ in einem Schlüsselbedingungsausdruck
- Filtern der Elemente auf Grundlage eines Präfixmusters für den Sortierschlüssel

Java

SDK für Java 2.x

Fragen Sie mit AWS SDK for Java 2.x eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung für den Sortierschlüssel ab.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix",
sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
```

```
    }
```

Demonstrieren Sie mit AWS SDK for Java 2.x die Verwendung von „begins_with“ mit unterschiedlichen Präfixlängen.

```
    public static void main(String[] args) {
        try {
            CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
            LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
                config.getPartitionKeyName(),
                config.getPartitionKeyValue(),
                config.getSortKeyName(),
                config.getSortKeyPrefix()
            });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new BeginsWithQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withSortKeyName(config.getSortKeyName())
                .withSortKeyPrefix(config.getSortKeyPrefix())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> LOGGER.info(item.toString()));

            // Demonstrate with a different prefix
            if (!config.getSortKeyPrefix().isEmpty()) {
                String shorterPrefix = config.getSortKeyPrefix()
                    .substring(0, Math.max(1,
config.getSortKeyPrefix().length() / 2));
                LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

                final QueryResponse response2 = new BeginsWithQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
```

```
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withSortKeyName(config.getSortKeyName())
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
 response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie mit AWS SDK für JavaScript eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung im Sortierschlüssel ab.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items where the sort key begins with a specific
 prefix
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
```

```
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key
 * @param {string} prefix - The prefix to match at the beginning of the sort key
 * @returns {Promise<Object>} - The query response
 */
async function queryWithBeginsWith(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  prefix
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND begins_with(#sk, :prefix)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":prefix": { S: prefix }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with begins_with: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie mit AWS SDK für Python (Boto3) eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung im Sortierschlüssel ab.

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_begins_with(
    table_name, partition_key_name, partition_key_value, sort_key_name, prefix
):
    """
    Query a DynamoDB table with a begins_with condition on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute.
        prefix (str): The prefix to match at the beginning of the sort key.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query with a begins_with condition on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) & Key(
        sort_key_name
    ).begins_with(prefix)
    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Fragen Sie eine DynamoDB-Tabelle unter Verwendung eines Datumsbereichs im Sortierschlüssel mit einem ab AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mithilfe eines Datumsbereichs im Sortierschlüssel abgefragt wird.

- Abfragen von Elementen innerhalb eines bestimmten Datumsbereichs
- Verwenden von Vergleichsoperatoren für Sortierschlüssel im Datumsformat

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {
```

```
        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start
date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
```

```
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Zeigt, wie eine DynamoDB-Tabelle mit Datumsbereichsfilterung abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<dateKeyName> <startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);
```

```
            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });

            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date
range: {0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
```

```
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date
range:", response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with
time components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort
 key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
```

```
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
```

```
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
```

```
    table_name, partition_key_name, partition_key_value, sort_key_name, year,
 month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)
```

```
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer DynamoDB-Tabelle mit einem komplexen Filterausdruck mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit einem komplexen Filterausdruck abgefragt wird.

- Anwenden von komplexen Filterausdrücken auf Abfrageergebnisse
- Kombinieren von mehreren Bedingungen über logische Operatoren
- Filtern von Elementen auf der Grundlage von Nicht-Schlüsselattributen.

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
```

```
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck ab mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
```

```
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck ab mit. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
```

```
            query_params["ExpressionAttributeValues"] =
 expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
```

```
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] =
expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer DynamoDB-Tabelle mit einem dynamischen Filterausdruck mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit einem dynamischen Filterausdruck abgefragt wird.

- Erstellen von Filterausdrücken dynamisch zur Laufzeit
- Konstruieren von Filterbedingungen auf der Grundlage von Benutzereingaben oder Anwendungsstatus
- Hinzufügen oder Entfernen von Filterkriterien unter bestimmten Bedingungen

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {
```

```
        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName,
 partitionKeyName, partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " +
 e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

Demonstriert, wie dynamische Filterausdrücke mit verwendet werden. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
 <filterAttrName> <filterAttrValue> [region]
```

```
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " +
filterAttrName + " = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
```

```
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items
with multiple filters:");
            multiFilterResponse.items().forEach(item ->
System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
```

```
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab mit. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
```

```
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions
 where
            keys are attribute names and values are dictionaries with 'operator'
 and 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
```

```
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Demonstriert, wie dynamische Filterausdrücke mit verwendet werden. AWS SDK für Python (Boto3)

```
def example_usage():
    """Example of how to use the query_with_dynamic_filter function."""
    # Example parameters
    table_name = "Products"
    partition_key_name = "Category"
    partition_key_value = "Electronics"

    # Define dynamic filter conditions based on user input or runtime conditions
```

```
    user_min_rating = 4  # This could come from user input
    user_status_filter = "active"  # This could come from user input

    filter_conditions = {}

    # Only add conditions that are actually specified
    if user_min_rating is not None:
        filter_conditions["rating"] = {"operator": ">=", "value":
user_min_rating}

    if user_status_filter:
        filter_conditions["status"] = {"operator": "=", "value":
user_status_filter}

    print(
        f"Querying products in category '{partition_key_value}' with filter
conditions: {filter_conditions}"
    )

    # Execute the query with dynamic filter
    response = query_with_dynamic_filter(
        table_name, partition_key_name, partition_key_value, filter_conditions
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Product: {item}")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer DynamoDB-Tabelle mit einem Filterausdruck und einem Limit mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit einem Filterausdruck und Limit abgefragt wird.

- Anwenden von Filterausdrücken auf Abfrageergebnisse mit einem Limit für die ausgewerteten Elemente
- Verstehen, wie sich Limits auf gefilterte Abfrageergebnisse auswirken
- Steuern der maximalen Anzahl der in einer Abfrage verarbeiteten Elemente

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem Filterausdruck ab und verwenden Sie das Limit.
AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
```

```
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem Filterausdruck ab und verwenden Sie das Limit.
AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of
 results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults
 to 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] =
 Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value":
 filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

Demonstriert, wie Filterausdrücke mit Grenzwerten in verwendet werden. AWS SDK für Python (Boto3)

```
def example_usage():
    """Example of how to use the query_with_filter_and_limit function."""
    # Example parameters
    table_name = "ProductReviews"
    partition_key_name = "ProductId"
    partition_key_value = "P123456"
    filter_attribute = "Rating"
    filter_value = 3  # Filter for ratings > 3
    limit = 5

    print(f"Querying reviews for product '{partition_key_value}' with rating >
 {filter_value}")
    print(f"Limiting to {limit} evaluated items")

    # Execute the query with filter and limit
    response = query_with_filter_and_limit(
```

```
        table_name, partition_key_name, partition_key_value, filter_attribute,
filter_value, limit
    )

    # Process the results
    items = response.get("Items", [])
    print(f"\nReturned {len(items)} items that passed the filter")

    for item in items:
        print(f"Review: {item}")

    # Explain the difference between Limit and actual results
    explain_limit_vs_results(response)

    # Check if there are more results
    if "LastEvaluatedKey" in response:
        print("\nThere are more results available. Use the LastEvaluatedKey for
pagination.")
    else:
        print("\nAll matching results have been retrieved.")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer DynamoDB-Tabelle mit verschachtelten Attributen unter Verwendung eines AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit verschachtelten Attributen abgefragt wird.

- Zugreifen und Filtern nach verschachtelten Attributen in DynamoDB-Elementen
- Verwenden von Dokumentpfadausdrücken, um auf verschachtelte Elemente zu verweisen

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit verschachtelten Attributen ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
```

```
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful.
Found " + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
```

```
    }
```

Demonstriert, wie eine DynamoDB-Tabelle mit verschachtelten Attributen abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<nestedPath> <nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
```

```
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM())
{
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath +
"." + nestedAttr + ": "
                                + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to
access nested attributes");
            System.out.println("2. Use expression attribute names for each
component of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
```

```
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit verschachtelten Attributen ab mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table filtering on a nested attribute
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} productId - The product ID to query by (partition key)
 * @param {string} category - The category to filter by (nested attribute)
 * @returns {Promise<Object>} - The query response
 */
async function queryWithNestedAttribute(
  config,
  tableName,
  productId,
  category
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
```

```
      TableName: tableName,
      KeyConditionExpression: "product_id = :productId",
      FilterExpression: "details.category = :category",
      ExpressionAttributeValues: {
        ":productId": { S: productId },
        ":category": { S: category }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with nested attribute: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit verschachtelten Attributen ab mit. AWS SDK für Python (Boto3)

```
from typing import Any, Dict, List

import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_path: str,
    comparison_operator: str,
    comparison_value: Any,
```

```
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_path (str): The path to the nested attribute (e.g.,
 'specs.weight').
        comparison_operator (str): The comparison operator to use ('=', '!=',
 '<', '<=', '>', '>=').
        comparison_value (any): The value to compare against.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the filter expression based on the nested attribute path and
 comparison operator
    filter_expression = None
    if comparison_operator == "=":
        filter_expression = Attr(nested_path).eq(comparison_value)
    elif comparison_operator == "!=":
        filter_expression = Attr(nested_path).ne(comparison_value)
    elif comparison_operator == "<":
        filter_expression = Attr(nested_path).lt(comparison_value)
    elif comparison_operator == "<=":
        filter_expression = Attr(nested_path).lte(comparison_value)
    elif comparison_operator == ">":
        filter_expression = Attr(nested_path).gt(comparison_value)
    elif comparison_operator == ">=":
        filter_expression = Attr(nested_path).gte(comparison_value)
    elif comparison_operator == "contains":
        filter_expression = Attr(nested_path).contains(comparison_value)
    elif comparison_operator == "begins_with":
        filter_expression = Attr(nested_path).begins_with(comparison_value)
```

```
    # Execute the query with the filter expression
    response = table.query(KeyConditionExpression=key_condition,
 FilterExpression=filter_expression)

    return response


def query_with_multiple_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_conditions: List[Dict[str, Any]],
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by multiple nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_conditions (list): A list of dictionaries, each containing:
            - path (str): The path to the nested attribute
            - operator (str): The comparison operator
            - value (any): The value to compare against

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the combined filter expression for all nested attributes
    combined_filter = None

    for condition in nested_conditions:
        if not isinstance(condition, dict):
            continue
        path = condition.get("path", "")
        operator = condition.get("operator", "")
        value = condition.get("value")
```

```
        if not path or not operator:
            continue

        # Build the individual filter expression
        current_filter = None
        if operator == "=":
            current_filter = Attr(path).eq(value)
        elif operator == "!=":
            current_filter = Attr(path).ne(value)
        elif operator == "<":
            current_filter = Attr(path).lt(value)
        elif operator == "<=":
            current_filter = Attr(path).lte(value)
        elif operator == ">":
            current_filter = Attr(path).gt(value)
        elif operator == ">=":
            current_filter = Attr(path).gte(value)
        elif operator == "contains":
            current_filter = Attr(path).contains(value)
        elif operator == "begins_with":
            current_filter = Attr(path).begins_with(value)

        # Combine with the existing filter using AND
        if current_filter:
            if combined_filter is None:
                combined_filter = current_filter
            else:
                combined_filter = combined_filter & current_filter

    # Execute the query with the combined filter expression
    response = table.query(KeyConditionExpression=key_condition,
FilterExpression=combined_filter)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer DynamoDB-Tabelle mit Paginierung mithilfe eines AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit Paginierung abgefragt wird.

- Implementieren der Paginierung für DynamoDB-Abfrageergebnisse
- Verwenden Sie das LastEvaluatedKey , um nachfolgende Seiten abzurufen.
- Steuern der Anzahl der Elemente pro Seite mit dem Parameter „Limit“

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit Paginierung ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
```

```
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }
```

```
                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total
of " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " +
e.getMessage());
            throw e;
        }
    }
```

Demonstriert, wie eine DynamoDB-Tabelle mit Paginierung abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    pageSize (optional) - The maximum number of items to return
per page. (Default: 10)
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;
```

```
        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

        System.out.println("Querying items with pagination (page size: " +
pageSize + ")");

        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " +
allItems.size() + " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() -
sampleSize) + " more items");
                }
            }
```

```
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit Paginierung ab mit. AWS SDK für JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
```

```
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }

      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);
```

```
      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in
 ${pageCount} pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to
 retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
```

```
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit Paginierung ab mit. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If
 None, retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
```

```
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large
 result sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page.
 Defaults to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)
```

```
        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Fragen Sie eine DynamoDB-Tabelle mit stark konsistenten Lesevorgängen ab, indem Sie eine AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie eine Tabelle mit strikt konsistenten Lesevorgängen abgefragt wird.

- Konfigurieren des Konsistenzniveaus für DynamoDB-Abfragen
- Verwenden der strikt konsistenten Lesevorgänge, um die aktuellen Daten zu erhalten
- Erkennen der Kompromisse zwischen letztendlicher Konsistenz und starker Konsistenz

Java

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit konfigurierbarer Lesekonsistenz ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
```

```
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit konfigurierbarer Lesekonsistenz ab mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
```

```
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit der Option für stark konsistente Lesevorgänge mit ab. AWS SDK für Python (Boto3)

```
import time

import boto3
from boto3.dynamodb.conditions import Key
```

```
def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent
 reads. Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen von DynamoDB-Daten mithilfe von PartiQL SELECT-Anweisungen mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-SELECT-Anweisungen abfragen.

JavaScript

SDK für JavaScript (v3)

Fragen Sie Elemente aus einer DynamoDB-Tabelle mithilfe von PartiQL SELECT-Anweisungen mit ab. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to query items from a DynamoDB table using
 PartiQL.
 * It shows different ways to select data with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Select all items from a DynamoDB table using PartiQL.
 * Note: This should be used with caution on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @returns The response from the ExecuteStatementCommand
 */
export const selectAllItems = async (tableName: string) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}"`,
  };
```

```
  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select an item by its primary key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select an item by its composite key (partition key + sort key) using PartiQL.
```

```
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select items using a filter condition with PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsWithFilter = async (
  tableName: string,
```

```
  filterAttribute: string,
  filterValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a begins_with function for prefix matching.
 * This is useful for querying hierarchical data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for prefix
 * @param prefix - The prefix to match
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByPrefix = async (
  tableName: string,
  attributeName: string,
  prefix: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE
 begins_with(${attributeName}, ?)`,
    Parameters: [prefix],
  };
```

```
  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a between condition for range queries.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for range
 * @param startValue - The start value of the range
 * @param endValue - The end value of the range
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByRange = async (
  tableName: string,
  attributeName: string,
  startValue: number | string,
  endValue: number | string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${attributeName} BETWEEN ?
 AND ?`,
    Parameters: [startValue, endValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};
```

```
/**
 * Example usage showing how to select items with different index types
 */
export const selectExamples = async () => {
  // Select all items from a table (use with caution on large tables)
  await selectAllItems("UsersTable");

  // Select by partition key (simple primary key)
  await selectItemByPartitionKey("UsersTable", "userId", "user123");

  // Select by composite key (partition key + sort key)
  await selectItemByCompositeKey("OrdersTable", "orderId", "order456",
 "productId", "prod789");

  // Select with a filter condition (can use any attribute)
  await selectItemsWithFilter("UsersTable", "userType", "premium");

  // Select items with a prefix (useful for hierarchical data)
  await selectItemsByPrefix("ProductsTable", "category", "electronics");

  // Select items within a range (useful for numeric or date ranges)
  await selectItemsByRange("OrdersTable", "orderDate", "2023-01-01",
 "2023-12-31");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Abfragen einer DynamoDB-Tabelle nach TTL-Elementen mithilfe eines AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie TTL-Elemente abgefragt werden.

Java

SDK für Java 2.x

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK für JavaScript

```
import { DynamoDBClient, QueryCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const queryFiltered = async (tableName, primaryKey, region = 'us-east-1')
 => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pk",
        FilterExpression: "#ea > :ea",
        ExpressionAttributeNames: {
            "#pk": "primaryKey",
            "#ea": "expireAt"
        },
        ExpressionAttributeValues: marshall({
            ":pk": primaryKey,
            ":ea": currentTime
        })
    };

    try {
        const { Items } = await client.send(new QueryCommand(params));
        Items.forEach(item => {
            console.log(unmarshall(item))
        });
        return Items;
    } catch (err) {
        console.error(`Error querying items: ${err}`);
        throw err;
    }
}
```

```
// Example usage (commented out for testing)
// queryFiltered('your-table-name', 'your-partition-key-value');
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK für Python (Boto3)

```
from datetime import datetime

import boto3


def query_dynamodb_items(table_name, partition_key):
    """

    :param table_name: Name of the DynamoDB table
    :param partition_key:
    :return:
    """
    try:
        # Initialize a DynamoDB resource
        dynamodb = boto3.resource("dynamodb", region_name="us-east-1")

        # Specify your table
        table = dynamodb.Table(table_name)

        # Get the current time in epoch format
        current_time = int(datetime.now().timestamp())

        # Perform the query operation with a filter expression to exclude expired
 items
        # response = table.query(
        #
 KeyConditionExpression=boto3.dynamodb.conditions.Key('partitionKey').eq(partition_key),
```

```
        #
 FilterExpression=boto3.dynamodb.conditions.Attr('expireAt').gt(current_time)
        # )
        response = table.query(

 KeyConditionExpression=dynamodb.conditions.Key("partitionKey").eq(partition_key),

 FilterExpression=dynamodb.conditions.Attr("expireAt").gt(current_time),
        )

        # Print the items that are not expired
        for item in response["Items"]:
            print(item)

    except Exception as e:
        print(f"Error querying items: {e}")


# Call the function with your values
query_dynamodb_items("Music", "your-partition-key-value")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen von DynamoDB-Tabellen mithilfe von Datums- und Uhrzeitmustern mit einem AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie Datums- und Uhrzeitmuster zum Abfragen von Tabellen verwendet werden.

- date/time Werte in DynamoDB speichern und abfragen.
- Implementieren von Datumsbereichsabfragen mithilfe von Sortierschlüsseln
- Formatieren von Datumszeichenfolgen für effektive Abfragen

Java

SDK für Java 2.x

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start
 date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Abfrage mit Datums- und Uhrzeitvariablen mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
```

```
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " +
 e.getMessage());
            throw e;
        }
    }
```

Führen Sie mit AWS SDK for Java 2.x Abfragen innerhalb von Datumsbereichen in Unix-Epochenzeitstempeln aus.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;
```

```
   public QueryResponse queryWithDateTimeEpoch(
       final String tableName,
       final String partitionKeyName,
       final String partitionKeyValue,
       final String dateKeyName,
       final long startEpoch,
       final long endEpoch) {

       CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
       CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
       CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
       CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

       // Create expression attribute names for the column names
       final Map<String, String> expressionAttributeNames = new HashMap<>();
       expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
       expressionAttributeNames.put("#dateKey", dateKeyName);

       // Create expression attribute values for the column values
       final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
       expressionAttributeValues.put(
           EXPRESSION_ATTRIBUTE_VALUE_PK,
           AttributeValue.builder().s(partitionKeyValue).build());
       expressionAttributeValues.put(
           ":startDate",
AttributeValue.builder().n(String.valueOf(startEpoch)).build());
       expressionAttributeValues.put(
           ":endDate",
AttributeValue.builder().n(String.valueOf(endEpoch)).build());

       // Create the query request
       final QueryRequest queryRequest = QueryRequest.builder()
           .tableName(tableName)
           .keyConditionExpression(KEY_CONDITION_EXPRESSION)
           .expressionAttributeNames(expressionAttributeNames)
           .expressionAttributeValues(expressionAttributeValues)
           .build();

       try {
           final QueryResponse response = dynamoDbClient.query(queryRequest);
```

```
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
 e.getMessage());
            throw e;
        }
    }
```

Abfrage innerhalb von Datumsbereichen mithilfe von LocalDateTime Objekten mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
```

```
            throw new IllegalArgumentException("Start and end LocalDateTime must
 not be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct
 format
        final String startDate =
 startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
 endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
 dateKeyName, startDate, endDate);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort
 key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
```

```
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

Abfrage mit Datums- und Uhrzeitvariablen mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} dateKeyName - The name of the date attribute to filter on
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRange(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  dateKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: `#pk = :pkValue AND #dateAttr BETWEEN :startDate
 AND :endDate`,
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#dateAttr": dateKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
```

```
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Python

SDK für Python (Boto3)

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year,
 month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

Abfrage mit Datums- und Uhrzeitvariablen mit. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_datetime(
    table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
):
    """
    Query a DynamoDB table with a date range filter on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date/
time values).
        start_date (datetime): The start date/time for the query range.
        end_date (datetime): The end date/time for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date/time values as ISO 8601 strings
    # DynamoDB works well with ISO format for date/time values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
```

```
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response



def example_usage():
    """Example of how to use the query_with_datetime function."""
    # Example parameters
    table_name = "Events"
    partition_key_name = "EventType"
    partition_key_value = "UserLogin"
    sort_key_name = "Timestamp"

    # Create date/time variables for the query
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)  # Query events from the last 7
 days

    print(f"Querying events from {start_date.isoformat()} to
 {end_date.isoformat()}")

    # Execute the query
    response = query_with_datetime(
        table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Event: {item}")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Rust

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie Attribute-Based Access Control (ABAC) für DynamoDB implementiert wird.

- Erstellen Sie eine IAM-Richtlinie für ABAC.
- Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.
- Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine IAM-Richtlinie für ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
```

```
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreatePolicy
  - CreateTable
  - ListTables

Verstehen Sie die Reihenfolge der Aktualisierungsausdrücke in DynamoDB mit einem AWS SDK

In den folgenden Codebeispielen wird die Reihenfolge der Aktualisierungsausdrücke erklärt.

- Erfahren, wie DynamoDB Aktualisierungsausdrücke verarbeitet
- Verstehen der Reihenfolge der Operationen in Aktualisierungsausdrücken
- Vermeiden von unerwarteten Ergebnissen durch Verständnis der Ausdrucksauswertung

Java

SDK für Java 2.x

Demonstrieren Sie die Reihenfolge der Aktualisierungsausdrücke mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
 AttributeValue> key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
```

```
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD
 OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
 dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter
 = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
 dynamoDbClient.updateItem(addFirstRequest);
```

```
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter
= Counter")
                .expressionAttributeValues(
                    Map.of(":increment",
AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }
```

```
        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
```

```
                .message("Failed to update item with SET first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
```

```
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
```

```
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
```

```
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating update expression order in DynamoDB");

       try {
           // Example 1: Demonstrate update order effects
           System.out.println("\nExample 1: Demonstrating update order
effects");
           Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

           if ((boolean) orderResults.get("success")) {
               System.out.println("SET first, then ADD:");
               System.out.println("  " + orderResults.get("setFirstResponse"));

               System.out.println("ADD first, then SET:");
               System.out.println("  " + orderResults.get("addFirstResponse"));

               System.out.println("SET with multiple attributes:");
               System.out.println("  " + orderResults.get("multiSetResponse"));

               System.out.println("SET with self-reference:");
               System.out.println("  " +
orderResults.get("selfReferenceResponse"));
           } else {
               System.out.println("Error: " + orderResults.get("error"));
           }

           // Example 2: Update with SET first, then REMOVE
```

```
            System.out.println("\nExample 2: Update with SET first, then
REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then
SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
```

```
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
 DynamoDB:");
            System.out.println("1. Update expressions are processed in this
 order: SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed
 from left to right");
            System.out.println("3. SET operations use the item state before any
 updates in the expression");
            System.out.println("4. When an attribute is referenced multiple
 times, the first operation wins");
            System.out.println("5. To reference a new value, split the update
 into multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
 change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
 separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie die Reihenfolge der Aktualisierungsausdrücke mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");
```

```
/**
 * Update an item with multiple actions in a single update expression.
 *
 * This function demonstrates how to use multiple actions in a single update
 expression
 * and how DynamoDB processes these actions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} updateExpression - The update expression with multiple actions
 * @param {Object} [expressionAttributeNames] - Expression attribute name
 placeholders
 * @param {Object} [expressionAttributeValues] - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
```

```
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}

/**
 * Demonstrate that variables hold copies of existing values before
 modifications.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that uses the values of attributes before they are modified in the same
 expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateValueCopying(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = { ...key, a: 1, b: 2, c: 3 };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
```

```
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with an expression that uses values before they are
 modified
  // This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to
 the value of 'b'
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE a SET b = a, c = b",
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Demonstrate the order in which different action types are processed.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the
 order
 * in which they are processed.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateActionOrder(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    set_attr: new Set(["A", "B", "C"]),
    to_remove: "This will be removed",
    to_modify: "Original value"
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with multiple action types
  // The actions will be processed in this order: REMOVE, SET, ADD, DELETE
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE to_remove SET to_modify = :new_value ADD
 counter :increment DELETE set_attr :elements",
    ExpressionAttributeValues: {
      ":new_value": "Updated value",
```

```
      ":increment": 5,
      ":elements": new Set(["B"])
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Update multiple attributes with a single SET action.
 *
 * This function demonstrates how to update multiple attributes in a single SET
 action,
 * which is more efficient than using multiple separate update operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {Object} attributes - The attributes to update and their new values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleSetActions(
  config,
  tableName,
  key,
  attributes
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Build the update expression and expression attribute values
  let updateExpression = "SET ";
  const expressionAttributeValues = {};

  // Add each attribute to the update expression
  Object.entries(attributes).forEach(([attrName, attrValue], index) => {
    const valuePlaceholder = `:val${index}`;

    if (index > 0) {
      updateExpression += ", ";
    }
    updateExpression += `${attrName} = ${valuePlaceholder}`;

    expressionAttributeValues[valuePlaceholder] = attrValue;
  });

  // Execute the update
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Update an attribute with a value from another attribute or a default value.
 *
 * This function demonstrates how to use if_not_exists to conditionally copy a
 value
 * from one attribute to another, or use a default value if the source doesn't
 exist.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} sourceAttribute - The attribute to copy the value from
 * @param {string} targetAttribute - The attribute to update
 * @param {any} defaultValue - The default value to use if the source attribute
 doesn't exist
```

```
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithConditionalValueCopying(
  config,
  tableName,
  key,
  sourceAttribute,
  targetAttribute,
  defaultValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Use if_not_exists to conditionally copy the value
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${targetAttribute} =
 if_not_exists(${sourceAttribute}, :default)`,
    ExpressionAttributeValues: {
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Demonstrate complex update expressions with multiple operations on the same
 attribute.
 *
 * This function shows how DynamoDB processes multiple operations on the same
 attribute
 * in a single update expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateMultipleOperationsOnSameAttribute(
```

```
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    list_attr: [1, 2, 3],
    map_attr: {
      nested1: "value1",
      nested2: "value2"
    }
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with multiple operations on the same attributes
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `
      SET counter = counter + :inc1,
          counter = counter + :inc2,
          map_attr.nested1 = :new_val1,
          map_attr.nested3 = :new_val3,
          list_attr[0] = list_attr[1],
          list_attr[1] = list_attr[2]
    `,
```

```
    ExpressionAttributeValues: {
      ":inc1": 5,
      ":inc2": 3,
      ":new_val1": "updated_value1",
      ":new_val3": "new_value3"
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK für JavaScript.

```
/**
 * Example of how to use update expression order of operations in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "OrderProcessing";

  console.log("Demonstrating update expression order of operations in DynamoDB");

  try {
    // Example 1: Demonstrating value copying in update expressions
    console.log("\nExample 1: Demonstrating value copying in update
 expressions");
    const results1 = await demonstrateValueCopying(
```

```
      config,
      tableName,
      { OrderId: "order123" }
    );

    console.log("Initial state:", JSON.stringify(results1.initialState, null,
2));
    console.log("Update response:", JSON.stringify(results1.updateResponse, null,
2));
    console.log("Final state:", JSON.stringify(results1.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The initial state had a=1, b=2, c=3");
    console.log("2. The update expression 'REMOVE a SET b = a, c = b' did the
following:");
    console.log("   - Copied the value of 'a' (which was 1) to be used for 'b'");
    console.log("   - Copied the value of 'b' (which was 2) to be used for 'c'");
    console.log("   - Removed the attribute 'a'");
    console.log("3. The final state has b=1, c=2, and 'a' is removed");
    console.log("4. This demonstrates that DynamoDB uses the values of attributes
as they were BEFORE any modifications");

    // Example 2: Demonstrating the order of different action types
    console.log("\nExample 2: Demonstrating the order of different action
types");
    const results2 = await demonstrateActionOrder(
      config,
      tableName,
      { OrderId: "order456" }
    );

    console.log("Initial state:", JSON.stringify(results2.initialState, null,
2));
    console.log("Update response:", JSON.stringify(results2.updateResponse, null,
2));
    console.log("Final state:", JSON.stringify(results2.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The update expression contained multiple action types:
REMOVE, SET, ADD, DELETE");
    console.log("2. DynamoDB processes these actions in this order: REMOVE, SET,
ADD, DELETE");
    console.log("3. First, 'to_remove' was removed");
    console.log("4. Then, 'to_modify' was set to a new value");
```

```
   console.log("5. Next, 'counter' was incremented by 5");
   console.log("6. Finally, 'B' was removed from the set attribute");

   // Example 3: Updating multiple attributes in a single SET action
   console.log("\nExample 3: Updating multiple attributes in a single SET
action");
   const response3 = await updateWithMultipleSetActions(
     config,
     tableName,
     { OrderId: "order789" },
     {
       Status: "Shipped",
       ShippingDate: "2025-05-28",
       TrackingNumber: "1Z999AA10123456784"
     }
   );

   console.log("Multiple attributes updated successfully:",
JSON.stringify(response3.Attributes, null, 2));

   // Example 4: Conditional value copying with if_not_exists
   console.log("\nExample 4: Conditional value copying with if_not_exists");
   const response4 = await updateWithConditionalValueCopying(
     config,
     tableName,
     { OrderId: "order101" },
     "PreferredShippingMethod",
     "ShippingMethod",
     "Standard"
   );

   console.log("Conditional value copying result:",
JSON.stringify(response4.Attributes, null, 2));

   // Example 5: Multiple operations on the same attribute
   console.log("\nExample 5: Multiple operations on the same attribute");
   const results5 = await demonstrateMultipleOperationsOnSameAttribute(
     config,
     tableName,
     { OrderId: "order202" }
   );

   console.log("Initial state:", JSON.stringify(results5.initialState, null,
2));
```

```
    console.log("Update response:", JSON.stringify(results5.updateResponse, null,
 2));
    console.log("Final state:", JSON.stringify(results5.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The counter was incremented twice (first by 5, then by 3) for
 a total of +8");
    console.log("2. The map attribute had one value updated and a new nested
 attribute added");
    console.log("3. The list attribute had values shifted (value at index 1 moved
 to index 0, value at index 2 moved to index 1)");
    console.log("4. All operations within the SET action are processed from left
 to right");

    // Key points about update expression order of operations
    console.log("\nKey Points About Update Expression Order of Operations:");
    console.log("1. Variables in expressions hold copies of attribute values as
 they existed BEFORE any modifications");
    console.log("2. Multiple actions in an update expression are processed in
 this order: REMOVE, SET, ADD, DELETE");
    console.log("3. Within each action type, operations are processed from left
 to right");
    console.log("4. You can reference the same attribute multiple times in an
 expression");
    console.log("5. You can use if_not_exists() to conditionally set values based
 on attribute existence");
    console.log("6. Using a single update expression with multiple actions is
 more efficient than multiple separate updates");
    console.log("7. The update expression is atomic - either all actions succeed
 or none do");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Einzelheiten zur API finden Sie UpdateItemunter AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie die Reihenfolge der Aktualisierungsausdrücke mit AWS SDK für Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, Optional


def update_with_multiple_actions(
    table_name: str,
    key: Dict[str, Any],
    update_expression: str,
    expression_attribute_names: Optional[Dict[str, str]] = None,
    expression_attribute_values: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update an item with multiple actions in a single update expression.

    This function demonstrates how to use multiple actions in a single update
 expression
    and how DynamoDB processes these actions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        update_expression (str): The update expression with multiple actions.
        expression_attribute_names (Optional[Dict[str, str]]): Expression
 attribute name placeholders.
        expression_attribute_values (Optional[Dict[str, Any]]): Expression
 attribute value placeholders.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Prepare the update parameters
```

```
    update_params = {
        "Key": key,
        "UpdateExpression": update_expression,
        "ReturnValues": "UPDATED_NEW",
    }

    # Add expression attribute names if provided
    if expression_attribute_names:
        update_params["ExpressionAttributeNames"] = expression_attribute_names

    # Add expression attribute values if provided
    if expression_attribute_values:
        update_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the update
    response = table.update_item(**update_params)

    return response


def demonstrate_value_copying(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate that variables hold copies of existing values before
 modifications.

    This function creates an item with initial values, then updates it with an
 expression
    that uses the values of attributes before they are modified in the same
 expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
```

```
    initial_item.update({"a": 1, "b": 2, "c": 3})

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with an expression that uses values before they are
 modified
    # This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to
 the value of 'b'
    update_response = table.update_item(
        Key=key, UpdateExpression="REMOVE a SET b = a, c = b",
 ReturnValues="UPDATED_NEW"
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def demonstrate_action_order(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate the order in which different action types are processed.

    This function creates an item with initial values, then updates it with an
 expression
    that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the
 order
    in which they are processed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.
```

```
    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update(
        {
            "counter": 10,
            "set_attr": set(["A", "B", "C"]),
            "to_remove": "This will be removed",
            "to_modify": "Original value",
        }
    )

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with multiple action types
    # The actions will be processed in this order: REMOVE, SET, ADD, DELETE
    update_response = table.update_item(
        Key=key,
        UpdateExpression="REMOVE to_remove SET to_modify = :new_value ADD
counter :increment DELETE set_attr :elements",
        ExpressionAttributeValues={
            ":new_value": "Updated value",
            ":increment": 5,
            ":elements": set(["B"]),
        },
        ReturnValues="UPDATED_NEW",
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
```

```
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def update_with_multiple_set_actions(
    table_name: str, key: Dict[str, Any], attributes: Dict[str, Any]
) -> Dict[str, Any]:
    """
    Update multiple attributes with a single SET action.

    This function demonstrates how to update multiple attributes in a single SET
 action,
    which is more efficient than using multiple separate update operations.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attributes (Dict[str, Any]): The attributes to update and their new
 values.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression and expression attribute values
    update_expression = "SET "
    expression_attribute_values = {}

    # Add each attribute to the update expression
    for i, (attr_name, attr_value) in enumerate(attributes.items()):
        value_placeholder = f":val{i}"

        if i > 0:
            update_expression += ", "
        update_expression += f"{attr_name} = {value_placeholder}"

        expression_attribute_values[value_placeholder] = attr_value
```

```
    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_with_conditional_value_copying(
    table_name: str,
    key: Dict[str, Any],
    source_attribute: str,
    target_attribute: str,
    default_value: Any,
) -> Dict[str, Any]:
    """
    Update an attribute with a value from another attribute or a default value.

    This function demonstrates how to use if_not_exists to conditionally copy a
 value
    from one attribute to another, or use a default value if the source doesn't
 exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        source_attribute (str): The attribute to copy the value from.
        target_attribute (str): The attribute to update.
        default_value (Any): The default value to use if the source attribute
 doesn't exist.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to conditionally copy the value
    response = table.update_item(
```

```
        Key=key,
        UpdateExpression=f"SET {target_attribute} =
 if_not_exists({source_attribute}, :default)",
        ExpressionAttributeValues={":default": default_value},
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use update expression order of operations in
 DynamoDB."""
    # Example parameters
    table_name = "OrderProcessing"
    key = {"OrderId": "order123"}

    print("Example 1: Demonstrating value copying in update expressions")
    try:
        results = demonstrate_value_copying(table_name=table_name, key=key)

        print(f"Initial state: {json.dumps(results['initial_state'],
 default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
 default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The initial state had a=1, b=2, c=3")
        print("2. The update expression 'REMOVE a SET b = a, c = b' did the
 following:")
        print("   - Copied the value of 'a' (which was 1) to be used for 'b'")
        print("   - Copied the value of 'b' (which was 2) to be used for 'c'")
        print("   - Removed the attribute 'a'")
        print("3. The final state has b=1, c=2, and 'a' is removed")
        print(
            "4. This demonstrates that DynamoDB uses the values of attributes as
 they were BEFORE any modifications"
        )
```

```
    except Exception as e:
        print(f"Error demonstrating value copying: {e}")

    print("\nExample 2: Demonstrating the order of different action types")
    try:
        results = demonstrate_action_order(table_name=table_name, key={"OrderId":
"order456"})

        print(f"Initial state: {json.dumps(results['initial_state'],
default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The update expression contained multiple action types: REMOVE,
SET, ADD, DELETE")
        print("2. DynamoDB processes these actions in this order: REMOVE, SET,
ADD, DELETE")
        print("3. First, 'to_remove' was removed")
        print("4. Then, 'to_modify' was set to a new value")
        print("5. Next, 'counter' was incremented by 5")
        print("6. Finally, 'B' was removed from the set attribute")
    except Exception as e:
        print(f"Error demonstrating action order: {e}")

    print("\nExample 3: Updating multiple attributes in a single SET action")
    try:
        response = update_with_multiple_set_actions(
            table_name=table_name,
            key={"OrderId": "order789"},
            attributes={
                "Status": "Shipped",
                "ShippingDate": "2025-05-14",
                "TrackingNumber": "1Z999AA10123456784",
            },
        )

        print(
            f"Multiple attributes updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating multiple attributes: {e}")
```

```
    print("\nExample 4: Conditional value copying with if_not_exists")
    try:
        response = update_with_conditional_value_copying(
            table_name=table_name,
            key={"OrderId": "order101"},
            source_attribute="PreferredShippingMethod",
            target_attribute="ShippingMethod",
            default_value="Standard",
        )

        print(
            f"Conditional value copying result:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error with conditional value copying: {e}")

    print("\nKey Points About Update Expression Order of Operations:")
    print(
        "1. Variables in expressions hold copies of attribute values as they
existed BEFORE any modifications"
    )
    print(
        "2. Multiple actions in an update expression are processed in this order:
REMOVE, SET, ADD, DELETE"
    )
    print("3. Within each action type, operations are processed from left to
right")
    print("4. You can reference the same attribute multiple times in an
expression")
    print("5. You can use if_not_exists() to conditionally set values based on
attribute existence")
    print(
        "6. Using a single update expression with multiple actions is more
efficient than multiple separate updates"
    )
    print("7. The update expression is atomic - either all actions succeed or
none do")
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie eine DynamoDB-Tabelleneinstellung mit warmem Durchsatz mithilfe eines AWS SDK

In den folgenden Codebeispielen wird gezeigt, wie die Warmdurchsatzeinstellung einer Tabelle aktualisiert wird.

Java

SDK für Java 2.x

Aktualisieren Sie mit AWS SDK for Java 2.x die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long
 readUnitsPerSecond, final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table
 and a global secondary index.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to update
     * @param tableReadUnitsPerSecond Read units per second for the table
     * @param tableWriteUnitsPerSecond Write units per second for the table
     * @param globalSecondaryIndexName The name of the global secondary index to
 update
     * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for
 the GSI
     * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for
 the GSI
```

```
     */
    public static void updateDynamoDBTable(
        final DynamoDbClient ddb,
        final String tableName,
        final Long tableReadUnitsPerSecond,
        final Long tableWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadUnitsPerSecond,
        final Long globalSecondaryIndexWriteUnitsPerSecond) {

        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableReadUnitsPerSecond,
tableWriteUnitsPerSecond);
        final WarmThroughput gsiWarmThroughput =
            buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

        final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
            .update(UpdateGlobalSecondaryIndexAction.builder()
                .indexName(globalSecondaryIndexName)
                .warmThroughput(gsiWarmThroughput)
                .build())
            .build();

        final UpdateTableRequest request = UpdateTableRequest.builder()
            .tableName(tableName)
            .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
            .warmThroughput(tableWarmThroughput)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Aktualisieren Sie mit AWS SDK für JavaScript die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTableCommand } from "@aws-sdk/client-dynamodb";

export async function updateDynamoDBTableWarmThroughput(
  tableName,
  tableReadUnits,
  tableWriteUnits,
  gsiName,
  gsiReadUnits,
  gsiWriteUnits,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });

    // Construct the update table request
    const updateTableRequest = {
      TableName: tableName,
      GlobalSecondaryIndexUpdates: [
        {
            Update: {
                IndexName: gsiName,
                WarmThroughput: {
                    ReadUnitsPerSecond: gsiReadUnits,
                    WriteUnitsPerSecond: gsiWriteUnits,
                },
            },
        },
      ],
      WarmThroughput: {
          ReadUnitsPerSecond: tableReadUnits,
          WriteUnitsPerSecond: tableWriteUnits,
      },
    };

    const command = new UpdateTableCommand(updateTableRequest);
    const response = await ddbClient.send(command);
```

```
    console.log(`Table updated successfully! Response:
 ${JSON.stringify(response)}`);
    return response;
  } catch (error) {
    console.error(`Error updating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
updateDynamoDBTableWarmThroughput(
  'example-table',
  5, 5,
  'example-index',
  2, 2
);
*/
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Aktualisieren Sie mit AWS SDK für Python (Boto3) die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
from boto3 import client
from botocore.exceptions import ClientError


def update_dynamodb_table_warm_throughput(
    table_name,
    table_read_units,
    table_write_units,
    gsi_name,
    gsi_read_units,
    gsi_write_units,
    region_name="us-east-1",
):
```

```
    """
    Updates the warm throughput of a DynamoDB table and a global secondary index.

    :param table_name: The name of the table to update.
    :param table_read_units: The new read units per second for the table's warm
throughput.
    :param table_write_units: The new write units per second for the table's warm
throughput.
    :param gsi_name: The name of the global secondary index to update.
    :param gsi_read_units: The new read units per second for the GSI's warm
throughput.
    :param gsi_write_units: The new write units per second for the GSI's warm
throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    :return: The response from the update_table operation
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Update the table's warm throughput
        table_warm_throughput = {
            "ReadUnitsPerSecond": table_read_units,
            "WriteUnitsPerSecond": table_write_units,
        }

        # Update the global secondary index's warm throughput
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_read_units,
            "WriteUnitsPerSecond": gsi_write_units,
        }

        # Construct the global secondary index update
        global_secondary_index_update = [
            {"Update": {"IndexName": gsi_name, "WarmThroughput":
gsi_warm_throughput}}
        ]

        # Construct the update table request
        update_table_request = {
            "TableName": table_name,
            "GlobalSecondaryIndexUpdates": global_secondary_index_update,
            "WarmThroughput": table_warm_throughput,
        }
```

```
        # Update the table
        response = ddb.update_table(**update_table_request)
        print("Table updated successfully!")
        return response  # Make sure to return the response
    except ClientError as e:
        print(f"Error updating table: {e}")
        raise e
```

- Einzelheiten zur API finden Sie UpdateTablein AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie ein DynamoDB-Element mit einer TTL mithilfe eines AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie die TTL eines Elements aktualisieren.

Java

SDK für Java 2.x

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
```

```
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":c",
AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e",
AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response =
dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItem = async (tableName, partitionKey, sortKey, region = 'us-
east-1') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);
    const expireAt = Math.floor((Date.now() + 90 * 24 * 60 * 60 * 1000) / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            partitionKey: partitionKey,
            sortKey: sortKey
        }),
        UpdateExpression: "SET updatedAt = :c, expireAt = :e",
        ExpressionAttributeValues: marshall({
            ":c": currentTime,
            ":e": expireAt
        }),
    };

    try {
        const data = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(data.Attributes);
        console.log("Item updated successfully: %s", responseData);
        return responseData;
    } catch (err) {
        console.error("Error updating item:", err);
        throw err;
    }
}

// Example usage (commented out for testing)
// updateItem('your-table-name', 'your-partition-key-value', 'your-sort-key-
value');
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def update_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Update an existing DynamoDB item with a TTL.
    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        # Create the DynamoDB resource.
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expireAt time (90 days from now) in epoch second format
        expire_at = int((datetime.now() + timedelta(days=90)).timestamp())

        table.update_item(
            Key={"partitionKey": primary_key, "sortKey": sort_key},
            UpdateExpression="set updatedAt=:c, expireAt=:e",
            ExpressionAttributeValues={":c": current_time, ":e": expire_at},
        )

        print("Item updated successfully.")
    except Exception as e:
        print(f"Error updating item: {e}")
```

```
# Replace with your own values
update_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie DynamoDB-Daten mithilfe von PartiQL UPDATE-Anweisungen mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-UPDATE-Anweisungen aktualisieren.

JavaScript

SDK für JavaScript (v3)

Aktualisieren Sie Elemente in einer DynamoDB-Tabelle mithilfe von PartiQL UPDATE-Anweisungen mit. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to update items in a DynamoDB table using
 PartiQL.
 * It shows different ways to update documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Update a single attribute of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
```

```
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateSingleAttribute = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update multiple attributes of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeUpdates - Object containing attribute names and their new
 values
 * @returns The response from the ExecuteStatementCommand
 */
export const updateMultipleAttributes = async (
  tableName: string,
  partitionKeyName: string,
```

```
  partitionKeyValue: string | number,
  attributeUpdates: Record<string, any>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create SET clause for each attribute
  const setClause = Object.keys(attributeUpdates)
    .map((attr, index) => `${attr} = ?`)
    .join(", ");

  // Create parameters array with attribute values followed by the partition key
 value
  const parameters = [...Object.values(attributeUpdates), partitionKeyValue];

  const params = {
    Statement: `UPDATE "${tableName}" SET ${setClause} WHERE ${partitionKeyName}
 = ?`,
    Parameters: parameters,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item identified by a composite key (partition key + sort key) using
 PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
```

```
export const updateItemWithCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${sortKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update an item with a condition to ensure the update only happens if a
 condition is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
```

```
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${conditionAttribute} = ?`,
    Parameters: [attributeValue, partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated with condition successfully");
    return data;
  } catch (err) {
    console.error("Error updating item with condition:", err);
    throw err;
  }
};

/**
 * Batch update multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param updates - Array of objects containing key and update information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchUpdateItems = async (
  tableName: string,
  updates: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    attributeName: string;
    attributeValue: any;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Create statements for each update
  const statements = updates.map((update) => {
    return {
      Statement: `UPDATE "${tableName}" SET ${update.attributeName} = ? WHERE
 ${update.partitionKeyName} = ?`,
      Parameters: [update.attributeValue, update.partitionKeyValue],
    };
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch updated successfully");
    return data;
  } catch (err) {
    console.error("Error batch updating items:", err);
    throw err;
  }
};

/**
 * Example usage showing how to update items with different index types
 */
export const updateExamples = async () => {
  // Update a single attribute using a simple primary key
  await updateSingleAttribute("UsersTable", "userId", "user123", "email",
 "newemail@example.com");

  // Update multiple attributes at once
  await updateMultipleAttributes("UsersTable", "userId", "user123", {
    email: "newemail@example.com",
    name: "John Smith",
    lastLogin: new Date().toISOString(),
  });

  // Update an item with a composite key (partition key + sort key)
  await updateItemWithCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
```

```
    "prod789",
    "quantity",
    5
  );

  // Update with a condition
  await updateItemWithCondition(
    "UsersTable",
    "userId",
    "user123",
    "userStatus",
    "active",
    "userType",
    "premium"
  );

  // Batch update multiple items
  await batchUpdateItems("UsersTable", [
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user123",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user456",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
  ]);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie Sie eine AWS Step Functions Zustandsmaschine erstellen, die nacheinander AWS Lambda Funktionen aufruft.

Java

SDK für Java 2.x

Zeigt, wie Sie einen AWS serverlosen Workflow mithilfe von AWS Step Functions und erstellen. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Verwenden Sie ein Dokumentmodell für DynamoDB mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Create, Read, Update, Delete (CRUD) und Batch-Operationen mithilfe eines Dokumentmodells für DynamoDB und eines SDK ausgeführt werden. AWS

Weitere Informationen finden Sie unter Dokumentmodell.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie CRUD-Operationen unter Verwendung eines Dokumentmodells durch.

```
/// <summary>
/// Performs CRUD operations on an Amazon DynamoDB table.
/// </summary>
public class MidlevelItemCRUD
{
    public static async Task Main()
    {
        var tableName = "ProductCatalog";
        var sampleBookId = 555;

        var client = new AmazonDynamoDBClient();
        var productCatalog = LoadTable(client, tableName);

        await CreateBookItem(productCatalog, sampleBookId);
        RetrieveBook(productCatalog, sampleBookId);

        // Couple of sample updates.
        UpdateMultipleAttributes(productCatalog, sampleBookId);
        UpdateBookPriceConditionally(productCatalog, sampleBookId);

        // Delete.
        await DeleteBook(productCatalog, sampleBookId);
    }

    /// <summary>
    /// Loads the contents of a DynamoDB table.
    /// </summary>
    /// <param name="client">An initialized DynamoDB client object.</param>
    /// <param name="tableName">The name of the table to load.</param>
    /// <returns>A DynamoDB table object.</returns>
    public static Table LoadTable(IAmazonDynamoDB client, string tableName)
    {
        Table productCatalog = Table.LoadTable(client, tableName);
        return productCatalog;
    }

    /// <summary>
    /// Creates an example book item and adds it to the DynamoDB table
    /// ProductCatalog.
    /// </summary>
    /// <param name="productCatalog">A DynamoDB table object.</param>
```

```
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task CreateBookItem(Table productCatalog, int
sampleBookId)
        {
            Console.WriteLine("\n*** Executing CreateBookItem() ***");
            var book = new Document
            {
                ["Id"] = sampleBookId,
                ["Title"] = "Book " + sampleBookId,
                ["Price"] = 19.99,
                ["ISBN"] = "111-1111111111",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["PageCount"] = 500,
                ["Dimensions"] = "8.5x11x.5",
                ["InPublication"] = new DynamoDBBool(true),
                ["InStock"] = new DynamoDBBool(false),
                ["QuantityOnHand"] = 0,
            };

            // Adds the book to the ProductCatalog table.
            await productCatalog.PutItemAsync(book);
        }

        /// <summary>
        /// Retrieves an item, a book, from the DynamoDB ProductCatalog table.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void RetrieveBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing RetrieveBook() ***");

            // Optional configuration.
            var config = new GetItemOperationConfig
            {
                AttributesToGet = new List<string> { "Id", "ISBN", "Title",
"Authors", "Price" },
                ConsistentRead = true,
            };
```

```
            Document document = await productCatalog.GetItemAsync(sampleBookId,
config);
            Console.WriteLine("RetrieveBook: Printing book retrieved...");
            PrintDocument(document);
        }

        /// <summary>
        /// Updates multiple attributes for a book and writes the changes to the
        /// DynamoDB table ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateMultipleAttributes(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\nUpdating multiple attributes....");
            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,

                // List of attribute updates.
                // The following replaces the existing authors list.
                ["Authors"] = new List<string> { "Author x", "Author y" },
                ["newAttribute"] = "New Value",
                ["ISBN"] = null, // Remove it.
            };

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                // Gets updated item in response.
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateMultipleAttributes: Printing item after
updates ...");
            PrintDocument(updatedBook);
```

```
        }

        /// <summary>
        /// Updates a book item if it meets the specified criteria.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateBookPriceConditionally(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing UpdateBookPriceConditionally()
***");

            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,
                ["Price"] = 29.99,
            };

            // For conditional price update, creating a condition expression.
            var expr = new Expression
            {
                ExpressionStatement = "Price = :val",
            };
            expr.ExpressionAttributeValues[":val"] = 19.00;

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                ConditionalExpression = expr,
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateBookPriceConditionally: Printing item whose
price was conditionally updated");
            PrintDocument(updatedBook);
        }
```

```
        /// <summary>
        /// Deletes the book with the supplied Id value from the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task DeleteBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing DeleteBook() ***");

            // Optional configuration.
            var config = new DeleteItemOperationConfig
            {
                // Returns the deleted item.
                ReturnValues = ReturnValues.AllOldAttributes,
            };
            Document document = await
productCatalog.DeleteItemAsync(sampleBookId, config);
            Console.WriteLine("DeleteBook: Printing deleted just deleted...");

            PrintDocument(document);
        }

        /// <summary>
        /// Prints the information for the supplied DynamoDB document.
        /// </summary>
        /// <param name="updatedDocument">A DynamoDB document object.</param>
        public static void PrintDocument(Document updatedDocument)
        {
            if (updatedDocument is null)
            {
                return;
            }

            foreach (var attribute in updatedDocument.GetAttributeNames())
            {
                string stringValue = null;
                var value = updatedDocument[attribute];

                if (value is null)
                {
```

```
                    continue;
                }

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                        select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}", attribute,
stringValue);
            }
        }
    }
```

Führen Sie Batch-Schreibvorgänge unter Verwendung eines Dokumentmodells durch.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to perform batch
    /// operations.
    /// </summary>
    public class MidLevelBatchWriteItem
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            await SingleTableBatchWrite(client);
            await MultiTableBatchWrite(client);
        }

        /// <summary>
        /// Perform a batch operation on a single DynamoDB table.
        /// </summary>
```

```
        /// <param name="client">An initialized DynamoDB object.</param>
        public static async Task SingleTableBatchWrite(IAmazonDynamoDB client)
        {
            Table productCatalog = Table.LoadTable(client, "ProductCatalog");
            var batchWrite = productCatalog.CreateBatchWrite();

            var book1 = new Document
            {
                ["Id"] = 902,
                ["Title"] = "My book1 in batch write using .NET helper classes",
                ["ISBN"] = "902-11-11-1111",
                ["Price"] = 10,
                ["ProductCategory"] = "Book",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["Dimensions"] = "8.5x11x.5",
                ["InStock"] = new DynamoDBBool(true),
                ["QuantityOnHand"] = new DynamoDBNull(), // Quantity is unknown
at this time.
            };

            batchWrite.AddDocumentToPut(book1);

            // Specify delete item using overload that takes PK.
            batchWrite.AddKeyToDelete(12345);
            Console.WriteLine("Performing batch write in
SingleTableBatchWrite()");
            await batchWrite.ExecuteAsync();
        }

        /// <summary>
        /// Perform a batch operation involving multiple DynamoDB tables.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        public static async Task MultiTableBatchWrite(IAmazonDynamoDB client)
        {
            // Specify item to add in the Forum table.
            Table forum = Table.LoadTable(client, "Forum");
            var forumBatchWrite = forum.CreateBatchWrite();

            var forum1 = new Document
            {
                ["Name"] = "Test BatchWrite Forum",
                ["Threads"] = 0,
```

```
            };
            forumBatchWrite.AddDocumentToPut(forum1);

            // Specify item to add in the Thread table.
            Table thread = Table.LoadTable(client, "Thread");
            var threadBatchWrite = thread.CreateBatchWrite();

            var thread1 = new Document
            {
                ["ForumName"] = "S3 forum",
                ["Subject"] = "My sample question",
                ["Message"] = "Message text",
                ["KeywordTags"] = new List<string> { "S3", "Bucket" },
            };
            threadBatchWrite.AddDocumentToPut(thread1);

            // Specify item to delete from the Thread table.
            threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");

            // Create multi-table batch.
            var superBatch = new MultiTableDocumentBatchWrite();
            superBatch.AddBatch(forumBatchWrite);
            superBatch.AddBatch(threadBatchWrite);
            Console.WriteLine("Performing batch write in
MultiTableBatchWrite()");

            // Execute the batch.
            await superBatch.ExecuteAsync();
        }
    }
```

Scannen Sie eine Tabelle unter Verwendung eines Dokumentmodells.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to scan a DynamoDB
    /// table for values.
    /// </summary>
    public class MidLevelScanOnly
    {
        public static async Task Main()
```

```
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            Table productCatalogTable = Table.LoadTable(client,
"ProductCatalog");

            await FindProductsWithNegativePrice(productCatalogTable);
            await FindProductsWithNegativePriceWithConfig(productCatalogTable);
        }

        /// <summary>
        /// Retrieves any products that have a negative price in a DynamoDB
table.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePrice(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced <
0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            Search search = productCatalogTable.Scan(scanFilter);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePrice:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Finds any items in the ProductCatalog table using a DynamoDB
        /// configuration object.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
```

```
        public static async Task FindProductsWithNegativePriceWithConfig(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced <
0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            var config = new ScanOperationConfig()
            {
                Filter = scanFilter,
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string> { "Title", "Id" },
            };

            Search search = productCatalogTable.Scan(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePriceWithConfig:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the details of the passed DynamoDB document object on the
        /// console.
        /// </summary>
        /// <param name="document">A DynamoDB document object.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
                if (value is Primitive)
```

```
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                        select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Verwenden Sie ein Dokumentmodell, um eine Tabelle abzufragen und zu scannen.

```
    /// <summary>
    /// Shows how to perform mid-level query procedures on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class MidLevelQueryAndScan
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            // Query examples.
            Table replyTable = Table.LoadTable(client, "Reply");
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 2";

            await FindRepliesInLast15Days(replyTable);
            await FindRepliesInLast15DaysWithConfig(replyTable, forumName,
threadSubject);
            await FindRepliesPostedWithinTimePeriod(replyTable, forumName,
threadSubject);

            // Get Example.
```

```
            Table productCatalogTable = Table.LoadTable(client,
"ProductCatalog");
            int productId = 101;

            await GetProduct(productCatalogTable, productId);
        }

        /// <summary>
        /// Retrieves information about a product from the DynamoDB table
        /// ProductCatalog based on the product ID and displays the information
        /// on the console.
        /// </summary>
        /// <param name="tableName">The name of the table from which to retrieve
        /// product information.</param>
        /// <param name="productId">The ID of the product to retrieve.</param>
        public static async Task GetProduct(Table tableName, int productId)
        {
            Console.WriteLine("*** Executing GetProduct() ***");
            Document productDocument = await tableName.GetItemAsync(productId);
            if (productDocument != null)
            {
                PrintDocument(productDocument);
            }
            else
            {
                Console.WriteLine("Error: product " + productId + " does not
exist");
            }
        }

        /// <summary>
        /// Retrieves replies from the passed DynamoDB table object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        public static async Task FindRepliesInLast15Days(
          Table table)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, "Id");
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            // Use Query overloads that take the minimum required query
parameters.
```

```
            Search search = table.Query(filter);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15Days:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Retrieve replies made during a specific time period.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The subject of the thread, which we are
        /// searching for replies.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          Table table,
          string forumName,
          string threadSubject)
        {
            DateTime startDate = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0,
0));
            DateTime endDate = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0,
0));

            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName +
"#" + threadSubject);
            filter.AddCondition("ReplyDateTime", QueryOperator.Between,
startDate, endDate);

            var config = new QueryOperationConfig()
            {
                Limit = 2, // 2 items/page.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
```

```
        {
          "Message",
          "ReplyDateTime",
          "PostedBy",
        },
                ConsistentRead = true,
                Filter = filter,
            };

            Search search = table.Query(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: printing
 replies posted within dates: {0} and {1} ............", startDate, endDate);

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Perform a query for replies made in the last 15 days using a DynamoDB
        /// QueryOperationConfig object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
 in.</param>
        /// <param name="threadName">The bane of the thread that we are searching
        /// for replies.</param>
        public static async Task FindRepliesInLast15DaysWithConfig(
          Table table,
          string forumName,
          string threadName)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName +
 "#" + threadName);
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
 twoWeeksAgoDate);
```

```
            var config = new QueryOperationConfig()
            {
                Filter = filter,

                // Optional parameters.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
                {
                  "Message",
                  "ReplyDateTime",
                  "PostedBy",
                },
                ConsistentRead = true,
            };

            Search search = table.Query(config);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15DaysWithConfig:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the contents of the passed DynamoDB document on the console.
        /// </summary>
        /// <param name="document">A DynamoDB document to display.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
```

```
                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Verwenden Sie ein Objektpersistenzmodell auf hoher Ebene für DynamoDB mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Create, Read, Update, Delete (CRUD) und Batch-Operationen mit einem Objektpersistenzmodell für DynamoDB und einem SDK ausgeführt werden. AWS

Weitere Informationen finden Sie unter Object-Persistence-Modell.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie CRUD-Operationen unter Verwendung eines übergeordneten Object-Persistence-Modells durch.

```
    /// <summary>
    /// Shows how to perform high-level CRUD operations on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class HighLevelItemCrud
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await PerformCRUDOperations(context);
        }

        public static async Task PerformCRUDOperations(IDynamoDBContext context)
        {
            int bookId = 1001; // Some unique value.
            Book myBook = new Book
            {
                Id = bookId,
                Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
                Isbn = "111-1111111001",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
            };

            // Save the book to the ProductCatalog table.
            await context.SaveAsync(myBook);

            // Retrieve the book from the ProductCatalog table.
            Book bookRetrieved = await context.LoadAsync<Book>(bookId);

            // Update some properties.
            bookRetrieved.Isbn = "222-2222221001";

            // Update existing authors list with the following values.
            bookRetrieved.BookAuthors = new List<string> { " Author 1", "Author
x" };
            await context.SaveAsync(bookRetrieved);

            // Retrieve the updated book. This time, add the optional
            // ConsistentRead parameter using DynamoDBContextConfig object.
            await context.LoadAsync<Book>(bookId, new DynamoDBContextConfig
            {
                ConsistentRead = true,
            });
```

```
            // Delete the book.
            await context.DeleteAsync<Book>(bookId);

            // Try to retrieve deleted book. It should return null.
            Book deletedBook = await context.LoadAsync<Book>(bookId, new
 DynamoDBContextConfig
            {
                ConsistentRead = true,
            });

            if (deletedBook == null)
            {
                Console.WriteLine("Book is deleted");
            }
        }
    }
```

Führen Sie Batch-Schreibvorgänge unter Verwendung eines übergeordneten Object-Persistence-Modells durch.

```
    /// <summary>
    /// Performs high-level batch write operations to an Amazon DynamoDB table.
    /// This example was written using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class HighLevelBatchWriteItem
    {
        public static async Task SingleTableBatchWrite(IDynamoDBContext context)
        {
            Book book1 = new Book
            {
                Id = 902,
                InPublication = true,
                Isbn = "902-11-11-1111",
                PageCount = "100",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book3 in batch write",
            };
```

```
            Book book2 = new Book
            {
                Id = 903,
                InPublication = true,
                Isbn = "903-11-11-1111",
                PageCount = "200",
                Price = 10,
                ProductCategory = "Book",
                Title = "My book4 in batch write",
            };

            var bookBatch = context.CreateBatchWrite<Book>();
            bookBatch.AddPutItems(new List<Book> { book1, book2 });

            Console.WriteLine("Adding two books to ProductCatalog table.");
            await bookBatch.ExecuteAsync();
        }

        public static async Task MultiTableBatchWrite(IDynamoDBContext context)
        {
            // New Forum item.
            Forum newForum = new Forum
            {
                Name = "Test BatchWrite Forum",
                Threads = 0,
            };
            var forumBatch = context.CreateBatchWrite<Forum>();
            forumBatch.AddPutItem(newForum);

            // New Thread item.
            Thread newThread = new Thread
            {
                ForumName = "S3 forum",
                Subject = "My sample question",
                KeywordTags = new List<string> { "S3", "Bucket" },
                Message = "Message text",
            };

            DynamoDBOperationConfig config = new DynamoDBOperationConfig();
            config.SkipVersionCheck = true;
            var threadBatch = context.CreateBatchWrite<Thread>(config);
            threadBatch.AddPutItem(newThread);
```

```
            threadBatch.AddDeleteKey("some partition key value", "some sort key
value");

            var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);

            Console.WriteLine("Performing batch write in
MultiTableBatchWrite().");
            await superBatch.ExecuteAsync();
        }

        public static async Task Main()
        {
            AmazonDynamoDBClient client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);

            await SingleTableBatchWrite(context);
            await MultiTableBatchWrite(context);
        }
    }
```

Weisen Sie einer Tabelle unter Verwendung eines übergeordneten Object-Persistence-Modells beliebige Daten zu.

```
    /// <summary>
    /// Shows how to map arbitrary data to an Amazon DynamoDB table.
    /// </summary>
    public class HighLevelMappingArbitraryData
    {
        /// <summary>
        /// Creates a book, adds it to the DynamoDB ProductCatalog table,
retrieves
        /// the new book from the table, updates the dimensions and writes the
        /// changed item back to the table.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to write and
        /// read data from the table.</param>
        public static async Task AddRetrieveUpdateBook(IDynamoDBContext context)
        {
            // Create a book.
            DimensionType myBookDimensions = new DimensionType()
```

```
            {
                Length = 8M,
                Height = 11M,
                Thickness = 0.5M,
            };

            Book myBook = new Book
            {
                Id = 501,
                Title = "AWS SDK for .NET Object Persistence Model Handling
 Arbitrary Data",
                Isbn = "999-9999999999",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
                Dimensions = myBookDimensions,
            };

            // Add the book to the DynamoDB table ProductCatalog.
            await context.SaveAsync(myBook);

            // Retrieve the book.
            Book bookRetrieved = await context.LoadAsync<Book>(501);

            // Update the book dimensions property.
            bookRetrieved.Dimensions.Height += 1;
            bookRetrieved.Dimensions.Length += 1;
            bookRetrieved.Dimensions.Thickness += 0.2M;

            // Write the changed item to the table.
            await context.SaveAsync(bookRetrieved);
        }

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await AddRetrieveUpdateBook(context);
        }
    }
```

Verwenden Sie ein übergeordnetes Object-Persistence-Modell, um eine Tabelle abzufragen und zu scannen.

```
    /// <summary>
    /// Shows how to perform high-level query and scan operations to Amazon
    /// DynamoDB tables.
    /// </summary>
    public class HighLevelQueryAndScan
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            DynamoDBContext context = new DynamoDBContext(client);

            // Get an item.
            await GetBook(context, 101);

            // Sample forum and thread to test queries.
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 1";

            // Sample queries.
            await FindRepliesInLast15Days(context, forumName, threadSubject);
            await FindRepliesPostedWithinTimePeriod(context, forumName,
threadSubject);

            // Scan table.
            await FindProductsPricedLessThanZero(context);
        }

        public static async Task GetBook(IDynamoDBContext context, int productId)
        {
            Book bookItem = await context.LoadAsync<Book>(productId);

            Console.WriteLine("\nGetBook: Printing result.....");
            Console.WriteLine($"Title: {bookItem.Title} \n ISBN:{bookItem.Isbn}
\n No. of pages: {bookItem.PageCount}");
        }

        /// <summary>
        /// Queries a DynamoDB table to find replies posted within the last 15
days.
        /// </summary>
```

```
        /// <param name="context">The DynamoDB context used to perform the
query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The thread object containing the query
parameters.</param>
        public static async Task FindRepliesInLast15Days(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string replyId = $"{forumName} #{threadSubject}";
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);

            List<object> times = new List<object>();
            times.Add(twoWeeksAgoDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("PostedBy", ScanOperator.GreaterThan,
times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(replyId,
cfg);
            IEnumerable<Reply> latestReplies = await
response.GetRemainingAsync();

            Console.WriteLine("\nReplies in last 15 days:");

            foreach (Reply r in latestReplies)
            {

Console.WriteLine($"{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries for replies posted within a specific time period.
        /// </summary>
```

```
        /// <param name="context">The DynamoDB context used to perform the
query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">Information about the subject that we're
        /// interested in.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string forumId = forumName + "#" + threadSubject;
            Console.WriteLine("\nReplies posted within time period:");

            DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(30);
            DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(1);

            List<object> times = new List<object>();
            times.Add(startDate);
            times.Add(endDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("LastPostedBy", ScanOperator.Between,
times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(forumId,
cfg);
            IEnumerable<Reply> repliesInAPeriod = await
response.GetRemainingAsync();

            foreach (Reply r in repliesInAPeriod)
            {

Console.WriteLine("{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
```

```
        /// Queries the DynamoDB ProductCatalog table for products costing less
        /// than zero.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to perform the
        /// query.</param>
        public static async Task FindProductsPricedLessThanZero(IDynamoDBContext
context)
        {
            int price = 0;

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc1 = new ScanCondition("Price", ScanOperator.LessThan, price);
            var sc2 = new ScanCondition("ProductCategory", ScanOperator.Equal,
"Book");
            scs.Add(sc1);
            scs.Add(sc2);

            AsyncSearch<Book> response = context.ScanAsync<Book>(scs);

            IEnumerable<Book> itemsWithWrongPrice = await
response.GetRemainingAsync();

            Console.WriteLine("\nFindProductsPricedLessThanZero: Printing
result.....");

            foreach (Book r in itemsWithWrongPrice)
            {
                Console.WriteLine($"{r.Id}\t{r.Title}\t{r.Price}\t{r.Isbn}");
            }
        }
    }
```

Verwenden Sie atomare Zähleroperationen in DynamoDB mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie unteilbare Zähleroperationen in DynamoDB verwendet werden.

- Inkrementieren von unteilbaren Zählern mithilfe von ADD- und SET-Operationen
- Sicheres Erhöhen von Zählern, die möglicherweise nicht existieren

- Implementieren einer optimistischen Sperre für Zähleroperationen

Java

SDK für Java 2.x

Demonstrieren Sie atomare Gegenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
```

```
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an
expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
```

```
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName =
if_not_exists(#counterName, :zero) + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
```

```
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if
the counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",

AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();
```

```
        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse =
incrementCounterWithAdd(dynamoDbClient, tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
            UpdateItemResponse setResponse =
incrementCounterWithSet(dynamoDbClient, tableName, key, "SetCounter", 1);
```

```
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;

            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);
```

```
        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName))
{
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in
DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse =
incrementCounterWithAdd(dynamoDbClient, tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());

            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse =
incrementCounterWithSet(dynamoDbClient, tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse =
incrementCounterSafely(dynamoDbClient, tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());
```

```
            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " +
comparison.get("addDuration") + " ns");
                System.out.println("SET duration: " +
comparison.get("setDuration") + " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " +
comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
```

```
            System.out.println("4. Use if_not_exists to handle the case where the
 counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
 from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be
 isolated from other writes");
            System.out.println("7. ADD can only be used with number and set data
 types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie atomare Gegenoperationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Increment a counter using the ADD operation.
 *
 * This function demonstrates using the ADD operation for atomic increments.
 * The ADD operation is atomic and is the recommended way to increment counters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
```

```
 */
async function incrementCounterWithAdd(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${counterName} :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter using the SET operation with an expression.
 *
 * This function demonstrates using the SET operation with an expression for
 increments.
 * While this approach works, it's less idiomatic for simple increments than
 using ADD.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
```

```
async function incrementCounterWithSet(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with an expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter safely, handling the case where the counter might not
 exist.
 *
 * This function demonstrates using the if_not_exists function with SET to safely
 * increment a counter that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} defaultValue - The default value if the counter doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterSafely(
```

```
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  defaultValue = 0
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} =
 if_not_exists(${counterName}, :default) + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter with optimistic locking to prevent race conditions.
 *
 * This function demonstrates using a condition expression to implement
 optimistic
 * locking, which prevents race conditions when multiple processes try to update
 * the same counter.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} expectedValue - The expected current value of the counter
```

```
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithLocking(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  expectedValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ConditionExpression: `${counterName} = :expected`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":expected": expectedValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      data: response
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        error: "Optimistic locking failed: the counter value has changed"
      };
    }
    // Re-throw other errors
    throw error;
```

```
  }
}

/**
 * Get the current value of a counter.
 *
 * Helper function to retrieve the current value of a counter attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @param {string} counterName - The name of the counter attribute
 * @returns {Promise<number|null>} - The current counter value or null if not
 found
 */
async function getCounterValue(
  config,
  tableName,
  key,
  counterName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the counter value if it exists, otherwise null
  return response.Item && counterName in response.Item
    ? response.Item[counterName]
    : null;
}
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use the atomic counter operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };
  const counterName = "ViewCount";
  const incrementValue = 1;

  console.log("Demonstrating different approaches to increment counters in
 DynamoDB");

  try {
    // Example 1: Using ADD operation (recommended for simple increments)
    console.log("\nExample 1: Incrementing counter with ADD operation");
    const response1 = await incrementCounterWithAdd(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response1.Attributes[counterName]}`);

    // Example 2: Using SET operation with an expression
    console.log("\nExample 2: Incrementing counter with SET operation");
    const response2 = await incrementCounterWithSet(
      config,
      tableName,
      key,
      counterName,
      incrementValue
    );

    console.log(`Counter incremented to: ${response2.Attributes[counterName]}`);

    // Example 3: Safely incrementing a counter that might not exist
    console.log("\nExample 3: Safely incrementing counter that might not exist");
    const newKey = { ProductId: "P67890" };
    const response3 = await incrementCounterSafely(
```

```
      config,
      tableName,
      newKey,
      counterName,
      incrementValue,
      0
    );

    console.log(`Counter initialized and incremented to:
${response3.Attributes[counterName]}`);

    // Example 4: Incrementing with optimistic locking
    console.log("\nExample 4: Incrementing with optimistic locking");

    // First, get the current counter value
    const currentValue = await getCounterValue(config, tableName, key,
counterName);
    console.log(`Current counter value: ${currentValue}`);

    // Then, try to increment with optimistic locking
    const response4 = await incrementCounterWithLocking(
      config,
      tableName,
      key,
      counterName,
      incrementValue,
      currentValue
    );

    if (response4.success) {
      console.log(`Counter successfully incremented to:
${response4.data.Attributes[counterName]}`);
    } else {
      console.log(response4.error);
    }

    // Explain the differences between ADD and SET
    console.log("\nKey differences between ADD and SET for counter operations:");
    console.log("1. ADD is more concise and idiomatic for simple increments");
    console.log("2. SET with expressions is more flexible for complex
operations");
    console.log("3. Both operations are atomic and safe for concurrent updates");
    console.log("4. SET with if_not_exists is required when the attribute might
not exist");
```

```
    console.log("5. Optimistic locking can be added to either approach for
 additional safety");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Demonstrieren Sie atomare Zähleroperationen mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Union


def increment_counter_with_add(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int
 = 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the ADD operation.

    This function demonstrates the atomic ADD operation, which is ideal for
    incrementing counters without the risk of race conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the ADD operation to atomically increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #counter :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_with_set(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int
 = 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the SET operation with an expression.

    This function demonstrates using SET with an expression to increment a
 counter.
    While this works, it's generally recommended to use ADD for simple
 increments.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation with an expression to increment the counter
```

```
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = #counter + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_safely(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    increment_value: int = 1,
    initial_value: int = 0,
) -> Dict[str, Any]:
    """
    Increment a counter attribute safely, handling the case where it might not
 exist.

    This function demonstrates a best practice for incrementing counters by using
    the if_not_exists function to handle the case where the counter doesn't exist
 yet.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to
 1.
        initial_value (int, optional): The initial value if the counter doesn't
 exist. Defaults to 0.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with if_not_exists to safely increment the counter
```

```
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = if_not_exists(#counter, :initial)
 + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value, ":initial":
 initial_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def atomic_conditional_increment(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    condition_attribute: str,
    condition_value: Any,
    increment_value: int = 1,
) -> Union[Dict[str, Any], None]:
    """
    Atomically increment a counter only if a condition is met.

    This function demonstrates combining atomic counter operations with
    conditional expressions for more complex update scenarios.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        increment_value (int, optional): The value to increment by. Defaults to
 1.

    Returns:
        Optional[Dict[str, Any]]: The response from DynamoDB if successful, None
 if condition failed.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)
```

```
    try:
        # Use ADD with a condition expression
        response = table.update_item(
            Key=key,
            UpdateExpression="ADD #counter :increment",
            ConditionExpression="#condition = :value",
            ExpressionAttributeNames={"#counter": counter_name, "#condition":
 condition_attribute},
            ExpressionAttributeValues={":increment": increment_value, ":value":
 condition_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the atomic counter operations functions."""
    # Example parameters
    table_name = "GameScores"
    key = {"UserId": "user123", "GameId": "game456"}
    counter_name = "Score"

    print("Example 1: Incrementing a counter with ADD operation")
    try:
        response = increment_counter_with_add(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=10
        )
        print(
            f"Counter incremented successfully. New value:
 {response.get('Attributes', {}).get(counter_name)}"
        )
```

```
    except Exception as e:
        print(f"Error incrementing counter with ADD: {e}")

    print("\nExample 2: Incrementing a counter with SET operation")
    try:
        response = increment_counter_with_set(
            table_name=table_name, key=key, counter_name=counter_name,
increment_value=5
        )
        print(
            f"Counter incremented successfully. New value:
{response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with SET: {e}")

    print("\nExample 3: Safely incrementing a counter that might not exist")
    try:
        new_key = {"UserId": "newuser789", "GameId": "game456"}
        response = increment_counter_safely(
            table_name=table_name,
            key=new_key,
            counter_name=counter_name,
            increment_value=15,
            initial_value=100,
        )
        print(
            f"Counter safely incremented. New value: {response.get('Attributes',
{}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error safely incrementing counter: {e}")

    print("\nExample 4: Conditional counter increment")
    try:
        # Fix for mypy: Handle the case where response might be None
        result = atomic_conditional_increment(
            table_name=table_name,
            key=key,
            counter_name="Achievements",
            condition_attribute="Level",
            condition_value=5,
            increment_value=1,
        )
```

```
        if result is not None:
            print(
                f"Conditional increment succeeded. New value:
{result.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
        if response:
            print(
                f"Conditional increment succeeded. New value:
{response.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
    except Exception as e:
        print(f"Error with conditional increment: {e}")

    print("\nComparison of ADD vs SET for counter operations:")
    print("1. ADD is specifically designed for atomic numeric increments and set
operations")
    print("2. SET with an expression can be used for more complex calculations")
    print("3. Both operations are atomic, preventing race conditions")
    print("4. ADD is more concise for simple increments")
    print("5. SET with if_not_exists() is recommended when the attribute might
not exist")
    print("6. For counters, ADD is generally preferred for clarity and
simplicity")
```

- Einzelheiten zur API finden Sie UpdateItemin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie bedingte Operationen in DynamoDB mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie bedingte Operationen in DynamoDB verwendet werden.

- Implementieren von bedingten Schreibvorgängen, um das Überschreiben von Daten zu verhindern
- Verwenden von Bedingungsausdrücken, um Geschäftsregeln durchzusetzen
- Ordnungsgemäßer Umgang mit Fehlern bei bedingten Prüfungen

Java

SDK für Java 2.x

Demonstrieren Sie bedingte Operationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
```

```
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }

    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr",
conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());
```

```
        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a
version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();
```

```
            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr
 = :data")
                    .expressionAttributeNames(Map.of("#verAttr",
 versionAttribute, "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
```

```
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
"#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
     * <p>This method demonstrates how to use multiple conditions in a condition
expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditions Map of attribute names to values for conditions
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdateWithMultipleConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        Map<String, AttributeValue> conditions,
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Build the condition expression and attribute names/values
            StringBuilder conditionExpression = new StringBuilder();
            Map<String, String> expressionAttributeNames = new HashMap<>();
            Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();

            // Add update attribute
            expressionAttributeNames.put("#updateAttr", updateAttribute);
            expressionAttributeValues.put(":updateVal", updateValue);

            // Add conditions
            int i = 0;
            for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
                String attrName = condition.getKey();
                AttributeValue attrValue = condition.getValue();
```

```
                String nameKey = "#cond" + i;
                String valueKey = ":val" + i;

                expressionAttributeNames.put(nameKey, attrName);
                expressionAttributeValues.put(valueKey, attrValue);

                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" =
").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update
was not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
```

```
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

Beispiel für die Verwendung von bedingten Operationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
 updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
```

```
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " +
lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price",
AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());
```

```
            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }

            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the
condition fails");
            System.out.println("4. Conditions can be used with update, delete,
and put operations");
            System.out.println("5. Multiple conditions can be combined with AND
and OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write
capacity whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteItem
  - PutItem
  - UpdateItem

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie bedingte Operationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  DeleteCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Perform a conditional update operation.
 *
 * This function demonstrates how to update an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
 * @param {string} updateAttribute - The attribute to update
 * @param {any} updateValue - The new value to set
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalUpdate(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue,
  updateAttribute,
  updateValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
```

```
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateAttribute} = :value`,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":value": updateValue,
      ":condition": conditionValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Condition was met and update was performed",
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, update was not performed",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Perform a conditional delete operation.
 *
 * This function demonstrates how to delete an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to delete
 * @param {string} conditionAttribute - The attribute to check in the condition
```

```
 * @param {any} conditionValue - The value to compare against
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalDelete(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the delete parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":condition": conditionValue
    },
    ReturnValues: "ALL_OLD"
  };

  try {
    // Perform the delete operation
    const response = await docClient.send(new DeleteCommand(params));

    return {
      success: true,
      message: "Condition was met and item was deleted",
      deletedItem: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, item was not deleted",
        error: "ConditionalCheckFailedException"
      };
    }
```

```
    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement optimistic locking with a version number.
 *
 * This function demonstrates how to use a version number for optimistic locking
 * to prevent race conditions when multiple processes update the same item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {Object} updates - The attributes to update
 * @param {number} expectedVersion - The expected current version number
 * @returns {Promise<Object>} - Result of the operation
 */
async function updateWithOptimisticLocking(
  config,
  tableName,
  key,
  updates,
  expectedVersion
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression
  const updateExpressions = [];
  const expressionAttributeValues = {
    ":expectedVersion": expectedVersion,
    ":newVersion": expectedVersion + 1
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([attribute, value], index) => {
    updateExpressions.push(`${attribute} = :val${index}`);
    expressionAttributeValues[`:val${index}`] = value;
  });

  // Add the version update
  updateExpressions.push("version = :newVersion");
```

```
  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateExpressions.join(", ")}`,
    ConditionExpression: "version = :expectedVersion",
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Update succeeded with optimistic locking",
      newVersion: expectedVersion + 1,
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Optimistic locking failed: the item was modified by another
 process",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement a conditional write that creates an item only if it doesn't exist.
 *
 * This function demonstrates how to use attribute_not_exists to create an item
 * only if it doesn't already exist (similar to an "INSERT IF NOT EXISTS"
 operation).
 *
```

```
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} item - The item to create
 * @returns {Promise<Object>} - Result of the operation
 */
async function createIfNotExists(
  config,
  tableName,
  item
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Extract the primary key attributes
  const keyAttributes = Object.keys(item).filter(attr =>
    attr === "id" || attr === "ID" || attr === "Id" ||
    attr.endsWith("Id") || attr.endsWith("ID") ||
    attr.endsWith("Key")
  );

  if (keyAttributes.length === 0) {
    throw new Error("Could not determine primary key attributes");
  }

  // Create a condition expression that checks if the item doesn't exist
  const conditionExpression = `attribute_not_exists(${keyAttributes[0]})`;

  // Define the put parameters with a condition expression
  const params = {
    TableName: tableName,
    Item: item,
    ConditionExpression: conditionExpression
  };

  try {
    // Perform the put operation
    await docClient.send(new PutCommand(params));

    return {
      success: true,
      message: "Item was created because it didn't exist",
      item
    };
```

```
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Item already exists, creation was skipped",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
```

```
  return response.Item || null;
}
```

Beispiel für die Verwendung von bedingten Operationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use conditional operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating conditional operations in DynamoDB");

  try {
    // Example 1: Conditional update based on attribute value
    console.log("\nExample 1: Conditional update based on attribute value");
    const updateResult = await conditionalUpdate(
      config,
      tableName,
      key,
      "Category",
      "Electronics",
      "Price",
      299.99
    );

    console.log(`Result: ${updateResult.message}`);
    if (updateResult.success) {
      console.log("Updated attributes:", updateResult.updatedAttributes);
    }

    // Example 2: Conditional delete based on attribute value
    console.log("\nExample 2: Conditional delete based on attribute value");
    const deleteResult = await conditionalDelete(
      config,
      tableName,
      key,
      "InStock",
      false
```

```
    );

    console.log(`Result: ${deleteResult.message}`);
    if (deleteResult.success) {
      console.log("Deleted item:", deleteResult.deletedItem);
    }

    // Example 3: Optimistic locking with version number
    console.log("\nExample 3: Optimistic locking with version number");

    // First, get the current item to check its version
    const currentItem = await getItem(config, tableName, { ProductId:
 "P67890" });
    const currentVersion = currentItem ? (currentItem.version || 0) : 0;

    console.log(`Current version: ${currentVersion}`);

    // Then, update with optimistic locking
    const lockingResult = await updateWithOptimisticLocking(
      config,
      tableName,
      { ProductId: "P67890" },
      {
        Name: "Updated Product Name",
        Description: "This is an updated description"
      },
      currentVersion
    );

    console.log(`Result: ${lockingResult.message}`);
    if (lockingResult.success) {
      console.log(`New version: ${lockingResult.newVersion}`);
      console.log("Updated attributes:", lockingResult.updatedAttributes);
    }

    // Example 4: Create item only if it doesn't exist
    console.log("\nExample 4: Create item only if it doesn't exist");
    const createResult = await createIfNotExists(
      config,
      tableName,
      {
        ProductId: "P99999",
        Name: "New Product",
        Category: "Accessories",
```

```
        Price: 19.99,
        InStock: true
      }
    );

    console.log(`Result: ${createResult.message}`);
    if (createResult.success) {
      console.log("Created item:", createResult.item);
    }

    // Explain conditional operations
    console.log("\nKey points about conditional operations:");
    console.log("1. Conditional operations only succeed if the condition is
 met");
    console.log("2. ConditionalCheckFailedException indicates the condition
 wasn't met");
    console.log("3. Optimistic locking prevents race conditions in concurrent
 updates");
    console.log("4. attribute_exists and attribute_not_exists are useful for
 checking if attributes are present");
    console.log("5. Conditional operations are atomic - they either succeed
 completely or fail completely");
    console.log("6. You can use any valid comparison operators and functions in
 condition expressions");
    console.log("7. Conditional operations don't consume write capacity if the
 condition fails");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - DeleteItem
  - PutItem
  - UpdateItem

Python

SDK für Python (Boto3)

Demonstrieren Sie bedingte Operationen mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Optional, Tuple, Union


def conditional_update(
    table_name: str,
    key: Dict[str, Any],
    condition_attribute: str,
    condition_value: Any,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item only if a condition is met.

    This function demonstrates how to perform a conditional update operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional update
```

```
        response = table.update_item(
            Key=key,
            UpdateExpression="SET #update_attr = :update_val",
            ConditionExpression="#cond_attr = :cond_val",
            ExpressionAttributeNames={
                "#update_attr": update_attribute,
                "#cond_attr": condition_attribute,
            },
            ExpressionAttributeValues={":update_val": update_value, ":cond_val":
 condition_value},
            ReturnValues="UPDATED_NEW",
        )
        # Update succeeded, condition was met
        return True, response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False, None
        else:
            # Other error occurred
            raise


def conditional_delete(
    table_name: str, key: Dict[str, Any], condition_attribute: str,
 condition_value: Any
) -> bool:
    """
    Delete an item only if a condition is met.

    This function demonstrates how to perform a conditional delete operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to delete.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.

    Returns:
        bool: True if the delete succeeded (condition was met), False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional delete
        table.delete_item(
            Key=key,
            ConditionExpression="#attr = :val",
            ExpressionAttributeNames={"#attr": condition_attribute},
            ExpressionAttributeValues={":val": condition_value},
        )
        # Delete succeeded, condition was met
        return True
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False
        else:
            # Other error occurred
            raise


def optimistic_locking_update(
    table_name: str,
    key: Dict[str, Any],
    version_attribute: str,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item using optimistic locking with a version attribute.

    This function demonstrates how to implement optimistic locking using
    a version attribute that is incremented with each update.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        version_attribute (str): The name of the version attribute.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
```

```
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # First, get the current version
    try:
        response = table.get_item(
            Key=key,
            ProjectionExpression=f"#{version_attribute}",
            ExpressionAttributeNames={f"#{version_attribute}":
version_attribute},
        )

        item = response.get("Item", {})
        current_version = item.get(version_attribute, 0)

        # Now, try to update with a condition on the version
        try:
            update_response = table.update_item(
                Key=key,
                UpdateExpression=f"SET #{update_attribute} = :update_val,
#{version_attribute} = :new_version",
                ConditionExpression=f"#{version_attribute} = :current_version",
                ExpressionAttributeNames={
                    f"#{update_attribute}": update_attribute,
                    f"#{version_attribute}": version_attribute,
                },
                ExpressionAttributeValues={
                    ":update_val": update_value,
                    ":current_version": current_version,
                    ":new_version": current_version + 1,
                },
                ReturnValues="UPDATED_NEW",
            )
            # Update succeeded
            return True, update_response
        except ClientError as e:
            if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
                # Version has changed, optimistic locking failed
                return False, None
            else:
                # Other error occurred
```

```
                raise
    except ClientError:
        # Error getting the item
        raise


def conditional_check_and_update(
    table_name: str,
    key: Dict[str, Any],
    check_attribute: str,
    check_value: Any,
    update_attribute: str,
    update_value: Any,
    create_if_not_exists: bool = False,
) -> Union[Dict[str, Any], None]:
    """
    Check if an attribute has a specific value and update another attribute if it
 does.

    This function demonstrates a more complex conditional update that can also
    create the item if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        check_attribute (str): The attribute to check in the condition.
        check_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.
        create_if_not_exists (bool, optional): Whether to create the item if it
 doesn't exist.

    Returns:
        Union[Dict[str, Any], None]: The response from DynamoDB if successful,
 None otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        if create_if_not_exists:
            # Use attribute_not_exists to create the item if it doesn't exist
```

```
            condition_expression = "attribute_not_exists(#pk) OR #check_attr
= :check_val"
            update_expression = "SET #update_attr = :update_val, #check_attr =
if_not_exists(#check_attr, :check_val)"

            # Get the partition key name from the key dictionary
            pk_name = next(iter(key))

            expression_attribute_names = {
                "#pk": pk_name,
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }
        else:
            # Only update if the check attribute has the expected value
            condition_expression = "#check_attr = :check_val"
            update_expression = "SET #update_attr = :update_val"

            expression_attribute_names = {
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }

        # Perform the conditional update
        response = table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues={":check_val": check_value, ":update_val":
update_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung von bedingten Operationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the conditional operations functions."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Conditional Update")
    try:
        # Update the price only if the current stock is greater than 10
        success, response = conditional_update(
            table_name=table_name,
            key=key,
            condition_attribute="Stock",
            condition_value=10,
            update_attribute="Price",
            update_value=99.99,
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes",
 {})
            print(f"Update succeeded! New values: {attributes}")
        else:
            print("Update failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional update: {e}")

    print("\nExample 2: Conditional Delete")
    try:
        # Delete the product only if it's discontinued
        success = conditional_delete(
            table_name=table_name,
            key=key,
            condition_attribute="Status",
            condition_value="Discontinued",
        )

        if success:
```

```
            print("Delete succeeded! The item was deleted.")
        else:
            print("Delete failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional delete: {e}")

    print("\nExample 3: Optimistic Locking")
    try:
        # Update with optimistic locking using a version attribute
        success, response = optimistic_locking_update(
            table_name=table_name,
            key=key,
            version_attribute="Version",
            update_attribute="Description",
            update_value="Updated product description",
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes",
{})
            print(f"Optimistic locking update succeeded! New values:
{attributes}")
        else:
            print("Optimistic locking update failed because the version has
changed.")
    except Exception as e:
        print(f"Error during optimistic locking update: {e}")

    print("\nExample 4: Conditional Check and Update")
    try:
        # Update the featured status if the product is in stock
        response = conditional_check_and_update(
            table_name=table_name,
            key=key,
            check_attribute="InStock",
            check_value=True,
            update_attribute="Featured",
            update_value=True,
            create_if_not_exists=True,
        )

        if response:
            print(
```

```
                f"Conditional check and update succeeded! New values:
{response.get('Attributes', {})}"
            )
        else:
            print("Conditional check and update failed because the condition was
not met.")
    except Exception as e:
        print(f"Error during conditional check and update: {e}")

    print("\nUnderstanding Conditional Operations in DynamoDB:")
    print("1. Conditional operations help maintain data integrity")
    print("2. They prevent race conditions in concurrent environments")
    print("3. Failed conditions result in ConditionalCheckFailedException")
    print("4. No DynamoDB capacity is consumed when conditions fail")
    print("5. Optimistic locking is a common pattern using version attributes")
    print("6. Conditions can be combined with logical operators (AND, OR, NOT)")
    print("7. Conditions can use comparison operators (=, <>, <, <=, >, >=)")
    print(
        "8. attribute_exists() and attribute_not_exists() are useful for checking
attribute presence"
    )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteItem
  - PutItem
  - UpdateItem

Verwenden Sie Ausdrucksattributnamen in DynamoDB mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Ausdrucksattributnamen in DynamoDB verwendet werden.

- Arbeiten mit reservierten Wörtern in DynamoDB-Ausdrücken
- Verwenden von Platzhaltern für Ausdrucksattributnamen
- Behandeln von Sonderzeichen in Attributnamen

Java

SDK für Java 2.x

Demonstrieren Sie die Namen der Ausdrucksattribute mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to
 update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
```

```
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to
update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to
update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
```

```
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Queries items using an attribute that is a reserved word.
     *
     * <p>This method demonstrates how to use expression attribute names in a
query
     * when the attribute is a reserved word.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key
     * @param reservedWordAttribute The reserved word attribute to filter on
     * @param value The value to compare against
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse queryWithReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the query parameters using expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attr", reservedWordAttribute);

        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);
        expressionAttributeValues.put(":value", value);

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attr = :value")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to
update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }
```

```
        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String,
AttributeValue> filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;
```

```
            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating expression attribute names in
 DynamoDB");

        try {
            // Example 1: Update an attribute that is a reserved word
            System.out.println("\nExample 1: Updating an attribute that is a
 reserved word");
            UpdateItemResponse response1 = updateReservedWordAttribute(
                dynamoDbClient,
```

```
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special
character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the
path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());
```

```
            System.out.println("Updated nested attribute: " +
 response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute
 name placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
            filters.put(
                "Product-Type",
 AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 =
 scanWithMultipleAttributeNames(dynamoDbClient, tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
 reserved words");
            System.out.println("2. Use expression attribute names for attributes
 with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
 nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute
 names in a single expression");
            System.out.println("6. Expression attribute names are case-
sensitive");
            System.out.println("7. Expression attribute names are only used in
 expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
```

```
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - Query
  - UpdateItem

JavaScript

SDK für JavaScript (v3)

Demonstrieren Sie die Namen der Ausdrucksattribute mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  QueryCommand,
  ScanCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an attribute that is a reserved word in DynamoDB.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that are reserved words in DynamoDB.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} reservedWordAttribute - The reserved word attribute to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateReservedWordAttribute(
  config,
  tableName,
  key,
```

```
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an attribute that contains special characters.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that contain special characters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} specialCharAttribute - The attribute with special characters
 to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateSpecialCharacterAttribute(
  config,
  tableName,
```

```
  key,
  specialCharAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": specialCharAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Query items using an attribute that is a reserved word.
 *
 * This function demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key attribute
 * @param {any} partitionKeyValue - The value of the partition key
 * @param {string} reservedWordAttribute - The reserved word attribute to filter
 on
 * @param {any} value - The value to compare against
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function queryWithReservedWordAttribute(
```

```
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the query parameters using expression attribute names
  const params = {
    TableName: tableName,
    KeyConditionExpression: "#pkName = :pkValue",
    FilterExpression: "#attr = :value",
    ExpressionAttributeNames: {
      "#pkName": partitionKeyName,
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":pkValue": partitionKeyValue,
      ":value": value
    }
  };

  // Perform the query operation
  const response = await docClient.send(new QueryCommand(params));

  return response;
}

/**
 * Update a nested attribute with a path that contains reserved words.
 *
 * This function demonstrates how to use expression attribute names to update
 * nested attributes where the path contains reserved words.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} attributePath - The path to the nested attribute as an array
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
```

```
 */
async function updateNestedReservedWordAttribute(
  config,
  tableName,
  key,
  attributePath,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names for each part of the path
  const expressionAttributeNames = {};
  for (let i = 0; i < attributePath.length; i++) {
    expressionAttributeNames[`#attr${i}`] = attributePath[i];
  }

  // Build the attribute path using the expression attribute names
  const attributePathExpression = attributePath
    .map((_, i) => `#attr${i}`)
    .join(".");

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${attributePathExpression} = :value`,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Scan a table with multiple attribute name placeholders.
 *
```

```
 * This function demonstrates how to use multiple expression attribute names
 * in a complex filter expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} filters - Object mapping attribute names to filter values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function scanWithMultipleAttributeNames(
  config,
  tableName,
  filters
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names and values
  const expressionAttributeNames = {};
  const expressionAttributeValues = {};
  const filterConditions = [];

  // Build the filter expression
  Object.entries(filters).forEach(([attrName, value], index) => {
    const nameKey = `#attr${index}`;
    const valueKey = `:val${index}`;

    expressionAttributeNames[nameKey] = attrName;
    expressionAttributeValues[valueKey] = value;
    filterConditions.push(`${nameKey} = ${valueKey}`);
  });

  // Join the filter conditions with AND
  const filterExpression = filterConditions.join(" AND ");

  // Define the scan parameters
  const params = {
    TableName: tableName,
    FilterExpression: filterExpression,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: expressionAttributeValues
  };

  // Perform the scan operation
```

```
  const response = await docClient.send(new ScanCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use expression attribute names.
 */
```

```
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating expression attribute names in DynamoDB");

  try {
    // Example 1: Update an attribute that is a reserved word
    console.log("\nExample 1: Updating an attribute that is a reserved word");
    const response1 = await updateReservedWordAttribute(
      config,
      tableName,
      key,
      "Size", // "SIZE" is a reserved word in DynamoDB
      "Large"
    );

    console.log("Updated attribute:", response1.Attributes);

    // Example 2: Update an attribute with special characters
    console.log("\nExample 2: Updating an attribute with special characters");
    const response2 = await updateSpecialCharacterAttribute(
      config,
      tableName,
      key,
      "Product-Type", // Contains a hyphen, which is a special character
      "Electronics"
    );

    console.log("Updated attribute:", response2.Attributes);

    // Example 3: Query with a reserved word attribute
    console.log("\nExample 3: Querying with a reserved word attribute");
    const response3 = await queryWithReservedWordAttribute(
      config,
      tableName,
      "Category",
      "Electronics",
      "Count", // "COUNT" is a reserved word in DynamoDB
      10
    );
```

```
   console.log(`Found ${response3.Items.length} items`);

   // Example 4: Update a nested attribute with reserved words in the path
   console.log("\nExample 4: Updating a nested attribute with reserved words in
the path");
   const response4 = await updateNestedReservedWordAttribute(
     config,
     tableName,
     key,
     ["Dimensions", "Size", "Height"], // "SIZE" is a reserved word
     30
   );

   console.log("Updated nested attribute:", response4.Attributes);

   // Example 5: Scan with multiple attribute name placeholders
   console.log("\nExample 5: Scanning with multiple attribute name
placeholders");
   const response5 = await scanWithMultipleAttributeNames(
     config,
     tableName,
     {
       "Size": "Large",
       "Count": 10,
       "Product-Type": "Electronics"
     }
   );

   console.log(`Found ${response5.Items.length} items`);

   // Get the final state of the item
   console.log("\nFinal state of the item:");
   const item = await getItem(config, tableName, key);
   console.log(JSON.stringify(item, null, 2));

   // Show some common reserved words
   console.log("\nSome common DynamoDB reserved words:");
   const commonReservedWords = [
     "ABORT", "ABSOLUTE", "ACTION", "ADD", "ALL", "ALTER", "AND", "ANY", "AS",
     "ASC", "BETWEEN", "BY", "CASE", "CAST", "COLUMN", "CONNECT", "COUNT",
     "CREATE", "CURRENT", "DATE", "DELETE", "DESC", "DROP", "ELSE", "EXISTS",
     "FOR", "FROM", "GRANT", "GROUP", "HAVING", "IN", "INDEX", "INSERT", "INTO",
     "IS", "JOIN", "KEY", "LEVEL", "LIKE", "LIMIT", "LOCAL", "MAX", "MIN",
"NAME",
```

```
      "NOT", "NULL", "OF", "ON", "OR", "ORDER", "OUTER", "REPLACE", "RETURN",
      "SELECT", "SET", "SIZE", "TABLE", "THEN", "TO", "UPDATE", "USER", "VALUES",
      "VIEW", "WHERE"
    ];
    console.log(commonReservedWords.join(", "));

    // Explain expression attribute names
    console.log("\nKey points about expression attribute names:");
    console.log("1. Use expression attribute names (#name) for reserved words");
    console.log("2. Use expression attribute names for attributes with special
 characters");
    console.log("3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
    console.log("4. Expression attribute names are required for nested attributes
 with reserved words");
    console.log("5. You can use multiple expression attribute names in a single
 expression");
    console.log("6. Expression attribute names are case-sensitive");
    console.log("7. Expression attribute names are only used in expressions, not
 in the actual data");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - Query
  - UpdateItem

Python

SDK für Python (Boto3)

Demonstrieren Sie Ausdrucksattributnamen mithilfe von AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List
```

```
def use_reserved_word_attribute(
    table_name: str, key: Dict[str, Any], reserved_word: str, value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name is a DynamoDB reserved word.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names that are DynamoDB reserved words.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        reserved_word (str): The reserved word to use as an attribute name.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle the reserved word
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #reserved_attr = :value",
        ExpressionAttributeNames={"#reserved_attr": reserved_word},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def use_special_character_attribute(
    table_name: str, key: Dict[str, Any], attribute_with_special_chars: str,
 value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name contains special characters.

    This function demonstrates how to use expression attribute names to work with
```

```
    attributes that have names containing special characters like spaces, dots,
 or hyphens.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attribute_with_special_chars (str): The attribute name with special
 characters.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle special characters
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #special_attr = :value",
        ExpressionAttributeNames={"#special_attr": attribute_with_special_chars},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def query_with_attribute_names(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    filter_attribute_name: str,
    filter_value: Any,
) -> Dict[str, Any]:
    """
    Query a table using expression attribute names for both key and filter
 attributes.

    This function demonstrates how to use expression attribute names in a query
 operation
    for both the key condition expression and filter expression.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute_name (str): The name of the attribute to filter on.
        filter_value (Any): The value to compare against in the filter.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names for both key condition and filter
    response = table.query(
        KeyConditionExpression="#pk = :pk_val",
        FilterExpression="#filter_attr = :filter_val",
        ExpressionAttributeNames={"#pk": partition_key_name, "#filter_attr":
 filter_attribute_name},
        ExpressionAttributeValues={":pk_val": partition_key_value, ":filter_val":
 filter_value},
    )

    return response


def update_nested_attribute_with_dots(
    table_name: str, key: Dict[str, Any], path_with_dots: str, value: Any
) -> Dict[str, Any]:
    """
    Update a nested attribute using a path with dot notation.

    This function demonstrates how to use expression attribute names to work with
    nested attributes specified using dot notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path_with_dots (str): The path to the nested attribute using dot notation
 (e.g., "a.b.c").
        value (Any): The value to set for the nested attribute.
```

```
    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated
 attribute values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path_with_dots.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part

        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def demonstrate_attribute_name_requirements(table_name: str, key: Dict[str, Any])
 -> Dict[str, Any]:
    """
```

```
    Demonstrate the requirements and allowed characters for attribute names.

    This function shows examples of valid and invalid attribute names and how to
    handle them using expression attribute names.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Examples of attribute names with different characteristics
    examples = {
        "valid_standard": "NormalAttribute",  # Standard attribute name (no
placeholder needed)
        "valid_with_underscore": "Normal_Attribute",  # Underscore is allowed
        "valid_with_number": "Attribute123",  # Numbers are allowed
        "reserved_word": "Timestamp",  # Reserved word (requires placeholder)
        "starts_with_number": "123Attribute",  # Starts with number (valid but
may need placeholder in some contexts)
        "with_space": "Attribute Name",  # Contains space (requires placeholder)
        "with_dot": "Attribute.Name",  # Contains dot (requires placeholder)
        "with_hyphen": "Attribute-Name",  # Contains hyphen (requires
placeholder)
        "with_special_chars": "Attribute#$%",  # Contains special characters
(requires placeholder)
    }

    results = {}

    # Try to update each attribute type
    for example_type, attr_name in examples.items():
        try:
            # For attributes that don't need placeholders, try direct reference
            if example_type in ["valid_standard", "valid_with_underscore",
"valid_with_number"]:
                try:
                    # Try without expression attribute names first
                    response = table.update_item(
```

```
                        Key=key,
                        UpdateExpression=f"SET {attr_name} = :value",
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": False,
                        "response": response,
                    }
                except ClientError:
                    # If direct reference fails, try with placeholder
                    response = table.update_item(
                        Key=key,
                        UpdateExpression="SET #attr = :value",
                        ExpressionAttributeNames={"#attr": attr_name},
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": True,
                        "response": response,
                    }
            else:
                # For attributes that definitely need placeholders
                response = table.update_item(
                    Key=key,
                    UpdateExpression="SET #attr = :value",
                    ExpressionAttributeNames={"#attr": attr_name},
                    ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                    ReturnValues="UPDATED_NEW",
                )
                results[example_type] = {
                    "attribute_name": attr_name,
                    "success": True,
                    "needed_placeholder": True,
                    "response": response,
                }
```

```
        except ClientError as e:
            results[example_type] = {"attribute_name": attr_name, "success":
 False, "error": str(e)}

    return results
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use expression attribute names in DynamoDB."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Using a reserved word as an attribute name")
    try:
        response = use_reserved_word_attribute(
            table_name=table_name, key=key, reserved_word="Timestamp",
 value="2025-05-14T12:00:00Z"
        )
        print(f"Reserved word attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating reserved word attribute: {e}")

    print("\nExample 2: Using an attribute name with special characters")
    try:
        response = use_special_character_attribute(
            table_name=table_name,
            key=key,
            attribute_with_special_chars="Product Info",
            value="Special product information",
        )
        print(f"Special character attribute updated successfully:
 {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating special character attribute: {e}")

    print("\nExample 3: Querying with expression attribute names")
    try:
```

```
        response = query_with_attribute_names(
            table_name=table_name,
            partition_key_name="Category",
            partition_key_value="Electronics",
            filter_attribute_name="Price",
            filter_value=500,
        )
        print(
            f"Query with expression attribute names returned
{len(response.get('Items', []))} items"
        )
    except Exception as e:
        print(f"Error querying with expression attribute names: {e}")

    print("\nExample 4: Updating a nested attribute with dot notation")
    try:
        response = update_nested_attribute_with_dots(
            table_name=table_name,
            key=key,
            path_with_dots="Product.Details.Specifications",
            value={"Weight": "2.5 kg", "Dimensions": "30x20x10 cm"},
        )
        print(f"Nested attribute updated successfully:
{response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating nested attribute: {e}")

    print("\nExample 5: Demonstrating attribute name requirements")
    try:
        results = demonstrate_attribute_name_requirements(table_name=table_name,
key=key)

        print("Attribute Name Requirements Results:")
        for example_type, result in results.items():
            if result.get("success", False):
                needed_placeholder = result.get("needed_placeholder", True)
                print(
                    f"  - {example_type}: '{result['attribute_name']}' -
{'Requires' if needed_placeholder else 'Does not require'} placeholder"
                )
            else:
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - Failed:
{result.get('error', 'Unknown error')}"
```

```
                )
    except Exception as e:
        print(f"Error demonstrating attribute name requirements: {e}")

    print("\nCommon DynamoDB Reserved Words (sample):")
    reserved_words = get_common_reserved_words()
    print(", ".join(reserved_words[:20]) + "... (and many more)")

    print("\nWhen to Use Expression Attribute Names:")
    print("1. When the attribute name is a DynamoDB reserved word")
    print("2. When the attribute name contains special characters (spaces, dots,
hyphens)")
    print("3. When the attribute name begins with a number")
    print("4. When working with nested attributes using dot notation")
    print("5. When you need to reference the same attribute multiple times in an
expression")

    print("\nExpression Attribute Name Requirements:")
    print("1. Must begin with a pound sign (#)")
    print("2. After the pound sign, must contain at least one character")
    print("3. Can contain alphanumeric characters and underscore (_)")
    print("4. Are case-sensitive")
    print("5. Must be unique within a single expression")

    print("\nAttribute Name Requirements in DynamoDB:")
    print("1. Can begin with a-z, A-Z, or 0-9")
    print("2. Can contain a-z, A-Z, 0-9, underscore (_), dash (-), and dot (.)")
    print("3. Are case-sensitive")
    print("4. No length restrictions, but practical limits apply")
    print("5. Cannot be a DynamoDB reserved word if used directly in
expressions")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Query
  - UpdateItem

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

Arbeiten Sie mit lokalen sekundären DynamoDB-Indizes unter Verwendung von AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie Tabellen mit lokalen sekundären Indizes erstellen und abfragen.

- Erstellen einer Tabelle mit einem lokalen sekundären Index (LSI)
- Erstellen einer Tabelle mit mehreren LSIs mit unterschiedlichen Projektionstypen
- Abfragen von Daten mithilfe von LSIs

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einem lokalen sekundären Index.

```
# Create a table with a Local Secondary Index
aws dynamodb create-table \
    --table-name CustomerOrders \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=OrderID,AttributeType=S \
        AttributeName=OrderDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=OrderID,KeyType=RANGE \
    --local-secondary-indexes \
        "IndexName=OrderDateIndex,\
        KeySchema=[{AttributeName=CustomerID,KeyType=HASH},
{AttributeName=OrderDate,KeyType=RANGE}],\
        Projection={ProjectionType=ALL}" \
    --billing-mode PAY_PER_REQUEST
```

Erstellen Sie eine Tabelle mit mehreren LSIs.

```
# Create a table with multiple Local Secondary Indexes
aws dynamodb create-table \
    --table-name CustomerDetails \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=Name,AttributeType=S \
        AttributeName=Email,AttributeType=S \
        AttributeName=RegistrationDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=Name,KeyType=RANGE \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"EmailIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Email\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",
\"NonKeyAttributes\":[\"Address\",\"Phone\"]}
            },
            {
                \"IndexName\": \"RegistrationIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"RegistrationDate\",\"KeyType\":\"RANGE
\"}
                ],
                \"Projection\": {\"ProjectionType\":\"KEYS_ONLY\"}
            }
        ]" \
    --billing-mode PAY_PER_REQUEST
```

Fragen Sie Daten mithilfe von LSIs ab.

```
# Query the OrderDateIndex LSI
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId AND OrderDate BETWEEN :date1
 AND :date2" \
```

```
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":date1": {"S": "2023-01-01"},
        ":date2": {"S": "2023-02-01"}
    }'

# Query with a filter expression
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId" \
    --filter-expression "Amount > :amount" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":amount": {"N": "150"}
    }'
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - Query

Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie DynamoDB Streams und Time-to-Live -Funktionen verwaltet werden.

- Erstellen Sie eine Tabelle mit aktiviertem Streams.
- Beschreiben Sie Streams.
- Erstellen Sie eine Lambda-Funktion für die Verarbeitung von Streams.
- Aktivieren Sie TTL für eine Tabelle.
- Fügen Sie Elemente mit TTL-Attributen hinzu.
- Beschreiben Sie TTL-Einstellungen.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit aktiviertem Streams.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Beschreiben Sie Streams.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Erstellen Sie eine Lambda-Funktion für Streams.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Aktivieren Sie TTL für eine Tabelle.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
```

```
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Fügen Sie Elemente mit TTL-Attributen hinzu.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Beschreiben Sie TTL-Einstellungen.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

Arbeiten Sie mit globalen DynamoDB-Tabellen und Multi-Region-Replikation mit Eventual Consistency (MREC) unter Verwendung von AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie globale DynamoDB-Tabellen mit letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC) verwaltet werden.

- Erstellen Sie eine Tabelle mit multiregionaler Replikation (MREC).
- Fügen Sie Elemente in eine Replikattabelle ein und rufen Sie sie ab.
- Entfernen Sie die Replikate nacheinander.
- Führen Sie eine Bereinigung durch Löschen der Tabelle aus.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit multiregionaler Replikation.

```
# Step 1: Create a new table (MusicTable) in US East (Ohio), with DynamoDB
 Streams enabled (NEW_AND_OLD_IMAGES)
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
```

```
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
    --region us-east-2

# Step 2: Create an identical MusicTable table in US East (N. Virginia)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2

# Step 3: Create a table in Europe (Ireland)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2
```

Beschreiben Sie die multiregionale Tabelle.

```
# Step 4: View the list of replicas created using describe-table
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
```

```
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistenc
{Region:RegionName,Status:ReplicaStatus}}'
```

Fügen Sie Elemente in eine Replikattabelle ein.

```
# Step 5: To verify that replication is working, add a new item to the Music
 table in US East (Ohio)
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-2
```

Rufen Sie Elemente aus Replikattabellen ab.

```
# Step 6: Wait for a few seconds, and then check to see whether the item has
 been
# successfully replicated to US East (N. Virginia) and Europe (Ireland)
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-1

aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region eu-west-1
```

Entfernen Sie Replikate.

```
# Step 7: Delete the replica table in Europe (Ireland) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "eu-west-1"
      }
```

```
    }
  ]
}' \
--region us-east-2

# Delete the replica table in US East (N. Virginia) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2
```

Führen Sie eine Bereinigung durch Löschen der Tabelle aus.

```
# Clean up: Delete the primary table
aws dynamodb delete-table --table-name MusicTable --region us-east-2

echo "Global table demonstration complete."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateTable

Arbeiten Sie mit DynamoDB-Ressourcen-Tagging mit AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie Tags für DynamoDB-Ressourcen verwalten.

- Erstellen Sie eine Tabelle mit Tags.
- Listen Sie Tags für eine Ressource auf.
- Fügen Sie Tags zu einer Ressource hinzu.
- Entfernen Sie Tags aus einer Ressource.
- Filtern Sie Tabellen nach Tags.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Tags.

```
# Create a table with tags
aws dynamodb create-table \
    --table-name TaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Production \
        Key=Project,Value=Analytics \
        Key=Owner,Value=DataTeam
```

Listen Sie Tags für eine Ressource auf.

```
# Get the table ARN
TABLE_ARN=$(aws dynamodb describe-table \
    --table-name TaggedTable \
    --query "Table.TableArn" \
    --output text)

# List tags for the table
```

```
aws dynamodb list-tags-of-resource \
    --resource-arn $TABLE_ARN
```

Fügen Sie Tags zu einer Ressource hinzu.

```
# Add tags to an existing table
aws dynamodb tag-resource \
    --resource-arn $TABLE_ARN \
    --tags \
        Key=CostCenter,Value=12345 \
        Key=BackupSchedule,Value=Daily
```

Entfernen Sie Tags aus einer Ressource.

```
# Remove tags from a table
aws dynamodb untag-resource \
    --resource-arn $TABLE_ARN \
    --tag-keys Owner BackupSchedule
```

Filtern Sie Tabellen nach Tags.

```
# Create another table with different tags
aws dynamodb create-table \
    --table-name AnotherTaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Development \
        Key=Project,Value=Testing

# Wait for table to become active
aws dynamodb wait table-exists --table-name AnotherTaggedTable

# List all tables
echo "All tables:"
aws dynamodb list-tables
```

```
# Get ARNs for all tables
echo -e "\nFiltering tables by Environment=Production tag:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# Find tables with specific tag
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTable
  - ListTagsOfResource
  - TagResource
  - UntagResource

Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie Verschlüsselungsoptionen für DynamoDB-Tabellen verwalten.

- Erstellen Sie eine Tabelle mit Standardverschlüsselung.
- Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.
- Aktualisieren Sie die Einstellungen für die Tabellenverschlüsselung.
- Beschreiben Sie die Tabellenverschlüsselung.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Standardverschlüsselung.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Aktualisieren Sie die Tabellenverschlüsselung.

```
# Update a table to use a different KMS key
```

```
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Beschreiben Sie die Tabellenverschlüsselung.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Serverless-Beispiele für DynamoDB

Die folgenden Codebeispiele zeigen, wie DynamoDB mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext
 context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
```

```
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string,
 error) {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
```

```
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new
 GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
```

```
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " +
 GSON.toJson(record.getDynamodb()));
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines DynamoDB-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Konsumieren eines DynamoDB-Ereignisses mit Lambda unter Verwendung. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
```

```
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
```

```
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
```

```
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }
```

```
    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet

die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
```

```
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent)
 (*BatchResult, error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
```

```
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier:
 curRecordSequenceNumber }] };
    }
  }
```

```
  return { batchItemFailures: [] };
};
```

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
```

```
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
```

```
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
```

```
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS Community-Beiträge für DynamoDB

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden AWS. Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Beispiele

- Erstellen und Testen einer Serverless-Anwendung

Erstellen und Testen einer Serverless-Anwendung

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

.NET

SDK für .NET

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des .NET SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Go

SDK für Go V2

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Java

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Rust

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Codebeispiele für Amazon EBS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic Block Store mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Amazon-EBS-Benutzerhandbuch – Weitere Informationen zu Amazon EBS.
- API-Referenz für Amazon EBS – Details zu allen verfügbaren Amazon-EBS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon EBS AWS SDKs
  - Aktionen für Amazon EBS mit AWS SDKs
    - Verwenden Sie CompleteSnapshotmit einem AWS SDK
    - Verwenden Sie PutSnapshotBlockmit einem AWS SDK
    - Verwenden Sie StartSnapshotmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon EBS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic Block Store mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon EBS mit AWS SDKs
  - Verwenden Sie CompleteSnapshotmit einem AWS SDK
  - Verwenden Sie PutSnapshotBlockmit einem AWS SDK
  - Verwenden Sie StartSnapshotmit einem AWS SDK

Aktionen für Amazon EBS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon EBS-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Elastic Block Store.

Beispiele

- Verwenden Sie CompleteSnapshotmit einem AWS SDK
- Verwenden Sie PutSnapshotBlockmit einem AWS SDK
- Verwenden Sie StartSnapshotmit einem AWS SDK

Verwenden Sie **`CompleteSnapshot`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CompleteSnapshot`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;

    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been
 transferred to Amazon S3.");
```

```
    println!("Use the get-snapshot-state code example to get the state of the
 snapshot.");

    Ok(())
}
```

- Einzelheiten zur API finden Sie CompleteSnapshotin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`PutSnapshotBlock`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PutSnapshotBlock`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_block(
    client: &Client,
    id: &str,
    idx: usize,
    block: Vec<u8>,
    checksum: &str,
) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;
```

```
    Ok(())
}
```

- Einzelheiten zur API finden Sie PutSnapshotBlockin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`StartSnapshot`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`StartSnapshot`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;

    Ok(snapshot.snapshot_id.unwrap())
}
```

- Einzelheiten zur API finden Sie StartSnapshotin der API-Referenz zum AWS SDK für Rust.

Codebeispiele für Amazon EC2 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic Compute Cloud mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-EC2-Benutzerhandbuch – Weitere Informationen zu Amazon EC2.
- API-Referenz für Amazon EC2 – Details zu allen verfügbaren Amazon-EC2-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon EC2 AWS SDKs
  - Hello Amazon EC2
  - Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK
  - Aktionen für Amazon EC2 mit AWS SDKs
    - Verwendung von AcceptVpcPeeringConnection mit einer CLI
    - Verwenden Sie mit einem AllocateAddress AWS SDK oder CLI
    - Verwendung von AllocateHosts mit einer CLI
    - Verwendung von AssignPrivateIpAddresses mit einer CLI
    - Verwenden Sie mit einem AssociateAddress AWS SDK oder CLI
    - Verwendung von AssociateDhcpOptions mit einer CLI
    - Verwendung von AssociateRouteTable mit einer CLI

- Verwendung von AttachInternetGateway mit einer CLI
- Verwendung von AttachNetworkInterface mit einer CLI
- Verwendung von AttachVolume mit einer CLI
- Verwendung von AttachVpnGateway mit einer CLI
- Verwendung von AuthorizeSecurityGroupEgress mit einer CLI
- Verwenden Sie mit einem AuthorizeSecurityGroupIngress AWS SDK oder CLI
- Verwendung von CancelCapacityReservation mit einer CLI
- Verwendung von CancelImportTask mit einer CLI
- Verwendung von CancelSpotFleetRequests mit einer CLI
- Verwendung von CancelSpotInstanceRequests mit einer CLI
- Verwendung von ConfirmProductInstance mit einer CLI
- Verwendung von CopyImage mit einer CLI
- Verwendung von CopySnapshot mit einer CLI
- Verwendung von CreateCapacityReservation mit einer CLI
- Verwendung von CreateCustomerGateway mit einer CLI
- Verwendung von CreateDhcpOptions mit einer CLI
- Verwendung von CreateFlowLogs mit einer CLI
- Verwendung von CreateImage mit einer CLI
- Verwendung von CreateInstanceExportTask mit einer CLI
- Verwendung von CreateInternetGateway mit einer CLI
- Verwenden Sie mit einem CreateKeyPair AWS SDK oder CLI
- Verwenden Sie CreateLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von CreateNetworkAcl mit einer CLI
- Verwendung von CreateNetworkAclEntry mit einer CLI
- Verwendung von CreateNetworkInterface mit einer CLI
- Verwendung von CreatePlacementGroup mit einer CLI
- Verwendung von CreateRoute mit einer CLI
- Verwenden Sie mit einem CreateRouteTable AWS SDK oder CLI
- Verwenden Sie CreateSecurityGroupmit einem AWS SDK oder CLI

- Verwendung von CreateSnapshot mit einer CLI

- Verwendung von CreateSpotDatafeedSubscription mit einer CLI
- Verwenden Sie mit einem CreateSubnet AWS SDK oder CLI
- Verwenden Sie CreateTagsmit einem AWS SDK oder CLI
- Verwendung von CreateVolume mit einer CLI
- Verwenden Sie mit einem CreateVpc AWS SDK oder CLI
- Verwenden Sie CreateVpcEndpointmit einem AWS SDK oder CLI
- Verwendung von CreateVpnConnection mit einer CLI
- Verwendung von CreateVpnConnectionRoute mit einer CLI
- Verwendung von CreateVpnGateway mit einer CLI
- Verwendung von DeleteCustomerGateway mit einer CLI
- Verwendung von DeleteDhcpOptions mit einer CLI
- Verwendung von DeleteFlowLogs mit einer CLI
- Verwendung von DeleteInternetGateway mit einer CLI
- Verwenden Sie mit einem DeleteKeyPair AWS SDK oder CLI
- Verwenden Sie DeleteLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von DeleteNetworkAcl mit einer CLI
- Verwendung von DeleteNetworkAclEntry mit einer CLI
- Verwendung von DeleteNetworkInterface mit einer CLI
- Verwendung von DeletePlacementGroup mit einer CLI
- Verwendung von DeleteRoute mit einer CLI
- Verwendung von DeleteRouteTable mit einer CLI
- Verwenden Sie mit einem DeleteSecurityGroup AWS SDK oder CLI
- Verwenden Sie DeleteSnapshotmit einem AWS SDK oder CLI
- Verwendung von DeleteSpotDatafeedSubscription mit einer CLI
- Verwendung von DeleteSubnet mit einer CLI
- Verwendung von DeleteTags mit einer CLI
- Verwendung von DeleteVolume mit einer CLI
- Verwenden Sie mit einem DeleteVpc AWS SDK oder CLI
- Verwenden Sie DeleteVpcEndpointsmit einem AWS SDK oder CLI

- Verwendung von DeleteVpnConnection mit einer CLI

- Verwendung von DeleteVpnConnectionRoute mit einer CLI
- Verwendung von DeleteVpnGateway mit einer CLI
- Verwendung von DeregisterImage mit einer CLI
- Verwendung von DescribeAccountAttributes mit einer CLI
- Verwenden Sie mit einem DescribeAddresses AWS SDK oder CLI
- Verwenden Sie DescribeAvailabilityZonesmit einem AWS SDK oder CLI
- Verwendung von DescribeBundleTasks mit einer CLI
- Verwendung von DescribeCapacityReservations mit einer CLI
- Verwendung von DescribeCustomerGateways mit einer CLI
- Verwendung von DescribeDhcpOptions mit einer CLI
- Verwendung von DescribeFlowLogs mit einer CLI
- Verwendung von DescribeHostReservationOfferings mit einer CLI
- Verwendung von DescribeHosts mit einer CLI
- Verwenden Sie mit einem DescribeIamInstanceProfileAssociations AWS SDK oder CLI
- Verwendung von DescribeIdFormat mit einer CLI
- Verwendung von DescribeIdentityIdFormat mit einer CLI
- Verwendung von DescribeImageAttribute mit einer CLI
- Verwenden Sie mit einem DescribeImages AWS SDK oder CLI
- Verwendung von DescribeImportImageTasks mit einer CLI
- Verwendung von DescribeImportSnapshotTasks mit einer CLI
- Verwendung von DescribeInstanceAttribute mit einer CLI
- Verwenden Sie mit einem DescribeInstanceStatus AWS SDK oder CLI
- Verwenden Sie DescribeInstanceTypesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeInstancesmit einem AWS SDK oder CLI
- Verwendung von DescribeInternetGateways mit einer CLI
- Verwenden Sie mit einem DescribeKeyPairs AWS SDK oder CLI
- Verwendung von DescribeNetworkAcls mit einer CLI
- Verwendung von DescribeNetworkInterfaceAttribute mit einer CLI
- Verwendung von DescribeNetworkInterfaces mit einer CLI

- Verwendung von DescribePlacementGroups mit einer CLI

- Verwendung von DescribePrefixLists mit einer CLI
- Verwenden Sie mit einem DescribeRegions AWS SDK oder CLI
- Verwenden Sie DescribeRouteTablesmit einem AWS SDK oder CLI
- Verwendung von DescribeScheduledInstanceAvailability mit einer CLI
- Verwendung von DescribeScheduledInstances mit einer CLI
- Verwenden Sie mit einem DescribeSecurityGroups AWS SDK oder CLI
- Verwendung von DescribeSnapshotAttribute mit einer CLI
- Verwenden Sie mit einem DescribeSnapshots AWS SDK oder CLI
- Verwendung von DescribeSpotDatafeedSubscription mit einer CLI
- Verwendung von DescribeSpotFleetInstances mit einer CLI
- Verwendung von DescribeSpotFleetRequestHistory mit einer CLI
- Verwendung von DescribeSpotFleetRequests mit einer CLI
- Verwendung von DescribeSpotInstanceRequests mit einer CLI
- Verwendung von DescribeSpotPriceHistory mit einer CLI
- Verwenden Sie mit einem DescribeSubnets AWS SDK oder CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeVolumeAttribute mit einer CLI
- Verwendung von DescribeVolumeStatus mit einer CLI
- Verwendung von DescribeVolumes mit einer CLI
- Verwendung von DescribeVpcAttribute mit einer CLI
- Verwendung von DescribeVpcClassicLink mit einer CLI
- Verwendung von DescribeVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von DescribeVpcEndpointServices mit einer CLI
- Verwendung von DescribeVpcEndpoints mit einer CLI
- Verwenden Sie mit einem DescribeVpcs AWS SDK oder CLI
- Verwendung von DescribeVpnConnections mit einer CLI
- Verwendung von DescribeVpnGateways mit einer CLI
- Verwendung von DetachInternetGateway mit einer CLI
- Verwendung von DetachNetworkInterface mit einer CLI

- Verwendung von DetachVolume mit einer CLI

- Verwendung von DetachVpnGateway mit einer CLI
- Verwendung von DisableVgwRoutePropagation mit einer CLI
- Verwendung von DisableVpcClassicLink mit einer CLI
- Verwendung von DisableVpcClassicLinkDnsSupport mit einer CLI
- Verwenden Sie mit einem DisassociateAddress AWS SDK oder CLI
- Verwendung von DisassociateRouteTable mit einer CLI
- Verwendung von EnableVgwRoutePropagation mit einer CLI
- Verwendung von EnableVolumeIo mit einer CLI
- Verwendung von EnableVpcClassicLink mit einer CLI
- Verwendung von EnableVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von GetConsoleOutput mit einer CLI
- Verwendung von GetHostReservationPurchasePreview mit einer CLI
- Verwenden Sie mit einem GetPasswordData AWS SDK oder CLI
- Verwendung von ImportImage mit einer CLI
- Verwendung von ImportKeyPair mit einer CLI
- Verwendung von ImportSnapshot mit einer CLI
- Verwendung von ModifyCapacityReservation mit einer CLI
- Verwendung von ModifyHosts mit einer CLI
- Verwendung von ModifyIdFormat mit einer CLI
- Verwendung von ModifyImageAttribute mit einer CLI
- Verwendung von ModifyInstanceAttribute mit einer CLI
- Verwendung von ModifyInstanceCreditSpecification mit einer CLI
- Verwendung von ModifyNetworkInterfaceAttribute mit einer CLI
- Verwendung von ModifyReservedInstances mit einer CLI
- Verwendung von ModifySnapshotAttribute mit einer CLI
- Verwendung von ModifySpotFleetRequest mit einer CLI
- Verwendung von ModifySubnetAttribute mit einer CLI
- Verwendung von ModifyVolumeAttribute mit einer CLI
- Verwendung von ModifyVpcAttribute mit einer CLI

- Verwenden Sie mit einem MonitorInstances AWS SDK oder CLI

- Verwendung von MoveAddressToVpc mit einer CLI
- Verwendung von PurchaseHostReservation mit einer CLI
- Verwendung von PurchaseScheduledInstances mit einer CLI
- Verwenden Sie mit einem RebootInstances AWS SDK oder CLI
- Verwendung von RegisterImage mit einer CLI
- Verwendung von RejectVpcPeeringConnection mit einer CLI
- Verwenden Sie mit einem ReleaseAddress AWS SDK oder CLI
- Verwendung von ReleaseHosts mit einer CLI
- Verwenden Sie mit einem ReplaceIamInstanceProfileAssociation AWS SDK oder CLI
- Verwendung von ReplaceNetworkAclAssociation mit einer CLI
- Verwendung von ReplaceNetworkAclEntry mit einer CLI
- Verwendung von ReplaceRoute mit einer CLI
- Verwendung von ReplaceRouteTableAssociation mit einer CLI
- Verwendung von ReportInstanceStatus mit einer CLI
- Verwendung von RequestSpotFleet mit einer CLI
- Verwendung von RequestSpotInstances mit einer CLI
- Verwendung von ResetImageAttribute mit einer CLI
- Verwendung von ResetInstanceAttribute mit einer CLI
- Verwendung von ResetNetworkInterfaceAttribute mit einer CLI
- Verwendung von ResetSnapshotAttribute mit einer CLI
- Verwendung von RevokeSecurityGroupEgress mit einer CLI
- Verwendung von RevokeSecurityGroupIngress mit einer CLI
- Verwenden Sie mit einem RunInstances AWS SDK oder CLI
- Verwendung von RunScheduledInstances mit einer CLI
- Verwenden Sie mit einem StartInstances AWS SDK oder CLI
- Verwenden Sie StopInstancesmit einem AWS SDK oder CLI
- Verwenden Sie TerminateInstancesmit einem AWS SDK oder CLI
- Verwendung von UnassignPrivateIpAddresses mit einer CLI
- Verwenden Sie mit einem UnmonitorInstances AWS SDK oder CLI

- Verwendung von UpdateSecurityGroupRuleDescriptionsIngress mit einer CLI

- Szenarien für die Verwendung von Amazon EC2 AWS SDKs
  - Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Erste Schritte mit virtuellen privaten Netzwerken
  - Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
  - Einen Container-Service für Instanzen virtueller Maschinen erstellen
  - Eine relationale Datenbankinstanz erstellen
  - Ein Service Networking Mesh erstellen und verwalten
  - Blockspeicher-Volumes erstellen und verwalten
  - Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI
  - Beginnen Sie mit dem Kauf von Softwaremarktplätzen über die CLI
  - Erste Schritte mit Big-Data-Verarbeitungsclustern
  - Erste Schritte mit Dokumentendatenbanken
  - Erste Schritte mit Graphdatenbanken
  - Erste Schritte mit In-Memory-Caching
  - Erste Schritte mit dem Lastenausgleich
  - Erste Schritte mit verwaltetem Streaming
  - Erste Schritte mit Network Transit Gateways
  - Erste Schritte mit Videotransport-Streaming
  - Erste Schritte mit virtuellen Maschinen
  - Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
  - Erste Schritte mit der IP-Adressverwaltung über die CLI
  - Erstellen Sie mithilfe der CLI ein virtuelles privates Netzwerk mit privaten Subnetzen und Gateways zur Netzwerkadressübersetzung
  - Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung
  - Ich arbeite mit Netzwerk-Peering-Verbindungen

Grundlegende Beispiele für die Verwendung von Amazon EC2 AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic Compute Cloud mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon EC2
- Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK
- Aktionen für Amazon EC2 mit AWS SDKs
  - Verwendung von AcceptVpcPeeringConnection mit einer CLI
  - Verwenden Sie mit einem AllocateAddress AWS SDK oder CLI
  - Verwendung von AllocateHosts mit einer CLI
  - Verwendung von AssignPrivateIpAddresses mit einer CLI
  - Verwenden Sie mit einem AssociateAddress AWS SDK oder CLI
  - Verwendung von AssociateDhcpOptions mit einer CLI
  - Verwendung von AssociateRouteTable mit einer CLI
  - Verwendung von AttachInternetGateway mit einer CLI
  - Verwendung von AttachNetworkInterface mit einer CLI
  - Verwendung von AttachVolume mit einer CLI
  - Verwendung von AttachVpnGateway mit einer CLI
  - Verwendung von AuthorizeSecurityGroupEgress mit einer CLI
  - Verwenden Sie mit einem AuthorizeSecurityGroupIngress AWS SDK oder CLI
  - Verwendung von CancelCapacityReservation mit einer CLI
  - Verwendung von CancelImportTask mit einer CLI
  - Verwendung von CancelSpotFleetRequests mit einer CLI
  - Verwendung von CancelSpotInstanceRequests mit einer CLI
  - Verwendung von ConfirmProductInstance mit einer CLI
  - Verwendung von CopyImage mit einer CLI
  - Verwendung von CopySnapshot mit einer CLI
  - Verwendung von CreateCapacityReservation mit einer CLI
  - Verwendung von CreateCustomerGateway mit einer CLI
  - Verwendung von CreateDhcpOptions mit einer CLI
  - Verwendung von CreateFlowLogs mit einer CLI
  - Verwendung von CreateImage mit einer CLI

  - Verwendung von CreateInstanceExportTask mit einer CLI

- Verwendung von CreateInternetGateway mit einer CLI
- Verwenden Sie mit einem CreateKeyPair AWS SDK oder CLI
- Verwenden Sie CreateLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von CreateNetworkAcl mit einer CLI
- Verwendung von CreateNetworkAclEntry mit einer CLI
- Verwendung von CreateNetworkInterface mit einer CLI
- Verwendung von CreatePlacementGroup mit einer CLI
- Verwendung von CreateRoute mit einer CLI
- Verwenden Sie mit einem CreateRouteTable AWS SDK oder CLI
- Verwenden Sie CreateSecurityGroupmit einem AWS SDK oder CLI
- Verwendung von CreateSnapshot mit einer CLI
- Verwendung von CreateSpotDatafeedSubscription mit einer CLI
- Verwenden Sie mit einem CreateSubnet AWS SDK oder CLI
- Verwenden Sie CreateTagsmit einem AWS SDK oder CLI
- Verwendung von CreateVolume mit einer CLI
- Verwenden Sie mit einem CreateVpc AWS SDK oder CLI
- Verwenden Sie CreateVpcEndpointmit einem AWS SDK oder CLI
- Verwendung von CreateVpnConnection mit einer CLI
- Verwendung von CreateVpnConnectionRoute mit einer CLI
- Verwendung von CreateVpnGateway mit einer CLI
- Verwendung von DeleteCustomerGateway mit einer CLI
- Verwendung von DeleteDhcpOptions mit einer CLI
- Verwendung von DeleteFlowLogs mit einer CLI
- Verwendung von DeleteInternetGateway mit einer CLI
- Verwenden Sie mit einem DeleteKeyPair AWS SDK oder CLI
- Verwenden Sie DeleteLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von DeleteNetworkAcl mit einer CLI
- Verwendung von DeleteNetworkAclEntry mit einer CLI
- Verwendung von DeleteNetworkInterface mit einer CLI

- Verwendung von DeletePlacementGroup mit einer CLI

- Verwendung von DeleteRoute mit einer CLI
- Verwendung von DeleteRouteTable mit einer CLI
- Verwenden Sie mit einem DeleteSecurityGroup AWS SDK oder CLI
- Verwenden Sie DeleteSnapshotmit einem AWS SDK oder CLI
- Verwendung von DeleteSpotDatafeedSubscription mit einer CLI
- Verwendung von DeleteSubnet mit einer CLI
- Verwendung von DeleteTags mit einer CLI
- Verwendung von DeleteVolume mit einer CLI
- Verwenden Sie mit einem DeleteVpc AWS SDK oder CLI
- Verwenden Sie DeleteVpcEndpointsmit einem AWS SDK oder CLI
- Verwendung von DeleteVpnConnection mit einer CLI
- Verwendung von DeleteVpnConnectionRoute mit einer CLI
- Verwendung von DeleteVpnGateway mit einer CLI
- Verwendung von DeregisterImage mit einer CLI
- Verwendung von DescribeAccountAttributes mit einer CLI
- Verwenden Sie mit einem DescribeAddresses AWS SDK oder CLI
- Verwenden Sie DescribeAvailabilityZonesmit einem AWS SDK oder CLI
- Verwendung von DescribeBundleTasks mit einer CLI
- Verwendung von DescribeCapacityReservations mit einer CLI
- Verwendung von DescribeCustomerGateways mit einer CLI
- Verwendung von DescribeDhcpOptions mit einer CLI
- Verwendung von DescribeFlowLogs mit einer CLI
- Verwendung von DescribeHostReservationOfferings mit einer CLI
- Verwendung von DescribeHosts mit einer CLI
- Verwenden Sie mit einem DescribeIamInstanceProfileAssociations AWS SDK oder CLI
- Verwendung von DescribeIdFormat mit einer CLI
- Verwendung von DescribeIdentityIdFormat mit einer CLI
- Verwendung von DescribeImageAttribute mit einer CLI
- Verwenden Sie mit einem DescribeImages AWS SDK oder CLI

- Verwendung von DescribeImportImageTasks mit einer CLI

- Verwendung von DescribeImportSnapshotTasks mit einer CLI
- Verwendung von DescribeInstanceAttribute mit einer CLI
- Verwenden Sie mit einem DescribeInstanceStatus AWS SDK oder CLI
- Verwenden Sie DescribeInstanceTypesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeInstancesmit einem AWS SDK oder CLI
- Verwendung von DescribeInternetGateways mit einer CLI
- Verwenden Sie mit einem DescribeKeyPairs AWS SDK oder CLI
- Verwendung von DescribeNetworkAcls mit einer CLI
- Verwendung von DescribeNetworkInterfaceAttribute mit einer CLI
- Verwendung von DescribeNetworkInterfaces mit einer CLI
- Verwendung von DescribePlacementGroups mit einer CLI
- Verwendung von DescribePrefixLists mit einer CLI
- Verwenden Sie mit einem DescribeRegions AWS SDK oder CLI
- Verwenden Sie DescribeRouteTablesmit einem AWS SDK oder CLI
- Verwendung von DescribeScheduledInstanceAvailability mit einer CLI
- Verwendung von DescribeScheduledInstances mit einer CLI
- Verwenden Sie mit einem DescribeSecurityGroups AWS SDK oder CLI
- Verwendung von DescribeSnapshotAttribute mit einer CLI
- Verwenden Sie mit einem DescribeSnapshots AWS SDK oder CLI
- Verwendung von DescribeSpotDatafeedSubscription mit einer CLI
- Verwendung von DescribeSpotFleetInstances mit einer CLI
- Verwendung von DescribeSpotFleetRequestHistory mit einer CLI
- Verwendung von DescribeSpotFleetRequests mit einer CLI
- Verwendung von DescribeSpotInstanceRequests mit einer CLI
- Verwendung von DescribeSpotPriceHistory mit einer CLI
- Verwenden Sie mit einem DescribeSubnets AWS SDK oder CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeVolumeAttribute mit einer CLI
- Verwendung von DescribeVolumeStatus mit einer CLI

- Verwendung von DescribeVolumes mit einer CLI

- Verwendung von DescribeVpcAttribute mit einer CLI
- Verwendung von DescribeVpcClassicLink mit einer CLI
- Verwendung von DescribeVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von DescribeVpcEndpointServices mit einer CLI
- Verwendung von DescribeVpcEndpoints mit einer CLI
- Verwenden Sie mit einem DescribeVpcs AWS SDK oder CLI
- Verwendung von DescribeVpnConnections mit einer CLI
- Verwendung von DescribeVpnGateways mit einer CLI
- Verwendung von DetachInternetGateway mit einer CLI
- Verwendung von DetachNetworkInterface mit einer CLI
- Verwendung von DetachVolume mit einer CLI
- Verwendung von DetachVpnGateway mit einer CLI
- Verwendung von DisableVgwRoutePropagation mit einer CLI
- Verwendung von DisableVpcClassicLink mit einer CLI
- Verwendung von DisableVpcClassicLinkDnsSupport mit einer CLI
- Verwenden Sie mit einem DisassociateAddress AWS SDK oder CLI
- Verwendung von DisassociateRouteTable mit einer CLI
- Verwendung von EnableVgwRoutePropagation mit einer CLI
- Verwendung von EnableVolumeIo mit einer CLI
- Verwendung von EnableVpcClassicLink mit einer CLI
- Verwendung von EnableVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von GetConsoleOutput mit einer CLI
- Verwendung von GetHostReservationPurchasePreview mit einer CLI
- Verwenden Sie mit einem GetPasswordData AWS SDK oder CLI
- Verwendung von ImportImage mit einer CLI
- Verwendung von ImportKeyPair mit einer CLI
- Verwendung von ImportSnapshot mit einer CLI
- Verwendung von ModifyCapacityReservation mit einer CLI
- Verwendung von ModifyHosts mit einer CLI

- Verwendung von ModifyIdFormat mit einer CLI

- Verwendung von ModifyImageAttribute mit einer CLI
- Verwendung von ModifyInstanceAttribute mit einer CLI
- Verwendung von ModifyInstanceCreditSpecification mit einer CLI
- Verwendung von ModifyNetworkInterfaceAttribute mit einer CLI
- Verwendung von ModifyReservedInstances mit einer CLI
- Verwendung von ModifySnapshotAttribute mit einer CLI
- Verwendung von ModifySpotFleetRequest mit einer CLI
- Verwendung von ModifySubnetAttribute mit einer CLI
- Verwendung von ModifyVolumeAttribute mit einer CLI
- Verwendung von ModifyVpcAttribute mit einer CLI
- Verwenden Sie mit einem MonitorInstances AWS SDK oder CLI
- Verwendung von MoveAddressToVpc mit einer CLI
- Verwendung von PurchaseHostReservation mit einer CLI
- Verwendung von PurchaseScheduledInstances mit einer CLI
- Verwenden Sie mit einem RebootInstances AWS SDK oder CLI
- Verwendung von RegisterImage mit einer CLI
- Verwendung von RejectVpcPeeringConnection mit einer CLI
- Verwenden Sie mit einem ReleaseAddress AWS SDK oder CLI
- Verwendung von ReleaseHosts mit einer CLI
- Verwenden Sie mit einem ReplaceIamInstanceProfileAssociation AWS SDK oder CLI
- Verwendung von ReplaceNetworkAclAssociation mit einer CLI
- Verwendung von ReplaceNetworkAclEntry mit einer CLI
- Verwendung von ReplaceRoute mit einer CLI
- Verwendung von ReplaceRouteTableAssociation mit einer CLI
- Verwendung von ReportInstanceStatus mit einer CLI
- Verwendung von RequestSpotFleet mit einer CLI
- Verwendung von RequestSpotInstances mit einer CLI
- Verwendung von ResetImageAttribute mit einer CLI
- Verwendung von ResetInstanceAttribute mit einer CLI

- Verwendung von ResetNetworkInterfaceAttribute mit einer CLI

- Verwendung von ResetSnapshotAttribute mit einer CLI
- Verwendung von RevokeSecurityGroupEgress mit einer CLI
- Verwendung von RevokeSecurityGroupIngress mit einer CLI
- Verwenden Sie mit einem RunInstances AWS SDK oder CLI
- Verwendung von RunScheduledInstances mit einer CLI
- Verwenden Sie mit einem StartInstances AWS SDK oder CLI
- Verwenden Sie StopInstancesmit einem AWS SDK oder CLI
- Verwenden Sie TerminateInstancesmit einem AWS SDK oder CLI
- Verwendung von UnassignPrivateIpAddresses mit einer CLI
- Verwenden Sie mit einem UnmonitorInstances AWS SDK oder CLI
- Verwendung von UpdateSecurityGroupRuleDescriptionsIngress mit einer CLI

Hello Amazon EC2

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon EC2 beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace EC2Actions;

public class HelloEc2
{
    /// <summary>
    /// HelloEc2 lists the existing security groups for the default users.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
```

```
    {
        // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
EC2).
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                .AddTransient<EC2Wrapper>()
            )
            .Build();

        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();

        try
        {
            // Retrieve information for up to 10 Amazon EC2 security groups.
            var request = new DescribeSecurityGroupsRequest { MaxResults = 10 };
            var securityGroups = new List<SecurityGroup>();

            var paginatorForSecurityGroups =
                ec2Client.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            // Now print the security groups returned by the call to
            // DescribeSecurityGroupsAsync.
            Console.WriteLine("Welcome to the EC2 Hello Service example. " +
                              "\nLet's list your Security Groups:");
            securityGroups.ForEach(group =>
            {
                Console.WriteLine(
                    $"Security group: {group.GroupName} ID: {group.GroupId}");
            });
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"An Amazon EC2 service error occurred while
listing security groups. {ex.Message}");
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing security groups.
 {ex.Message}");
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
project("hello_ec2")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
 *  A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 Cloud (Amazon EC2) client and describes
 *  the Amazon EC2 instances.
 *
 *  main function
 *
```

```
 *  Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)argv;

    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                              std::setw(48) << "Name" <<
                              std::setw(20) << "ID" <<
                              std::setw(25) << "Ami" <<
                              std::setw(15) << "Type" <<
                              std::setw(15) << "State" <<
                              std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }

                const std::vector<Aws::EC2::Model::Reservation> &reservations =
                        outcome.GetResult().GetReservations();

                for (const auto &reservation: reservations) {
                    const std::vector<Aws::EC2::Model::Instance> &instances =
                            reservation.GetInstances();
                    for (const auto &instance: instances) {
```

```
                        Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                        instance.GetState().GetName());

                        Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                        instance.GetInstanceType());

                        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                        instance.GetMonitoring().GetState());
                        Aws::String name = "Unknown";

                        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                     [](const
Aws::EC2::Model::Tag &tag) {
                                                         return tag.GetKey() ==
"Name";
                                                     });
                        if (nameIter != tags.cend()) {
                            name = nameIter->GetValue();
                        }
                        std::cout <<
                                  std::setw(48) << name <<
                                  std::setw(20) << instance.GetInstanceId() <<
                                  std::setw(25) << instance.GetImageId() <<
                                  std::setw(15) << typeString <<
                                  std::setw(15) << instanceStateString <<
                                  std::setw(15) << monitorString << std::endl;
                    }
                }

                if (!outcome.GetResult().GetNextToken().empty()) {
                    request.SetNextToken(outcome.GetResult().GetNextToken());
                } else {
                    done = true;
                }
            } else {
                std::cerr << "Failed to describe EC2 instances:" <<
```

```
                            outcome.GetError().GetMessage() << std::endl;
                    result = 1;
                    break;
                }
            }
        }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsunter AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
```

```
            .groupNames(groupName)
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
  const client = new EC2Client();
  try {
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({}),
    );

    const securityGroupList = SecurityGroups.slice(0, 9)
      .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
      .join("\n");

    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your
 account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_client: A Boto3 EC2 client. This client provides low-level
                       access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
```

```
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(PaginationConfig={'MaxItems': 10})
 # List only 10 security groups.
        logging.basicConfig(level=logging.INFO) # Enable logging.
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
        raise


if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
```

```
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
    instances = []

    paginator.each_page do |page|
      page.reservations.each do |reservation|
        reservation.instances.each do |instance|
          instances << instance
        end
      end
    end

    instances
  end

  # Prints details of the given EC2 instances.
  #
  # @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
 print.
  def print_instances(instances)
    instances.each do |instance|
      @logger.info("Instance ID: #{instance.instance_id}")
      @logger.info("Instance Type: #{instance.instance_type}")
      @logger.info("Public IP: #{instance.public_ip_address}")
```

```
      @logger.info("Public DNS Name: #{instance.public_dns_name}")
      @logger.info("\n")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
 Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
```

```
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "hello-ec2",
    // Let Xcode know the minimum Apple platforms supported.
```

```
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "hello-ec2",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Die Datei `entry.swift`

```
// An example that shows how to use the AWS SDK for Swift to perform a simple
// operation using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation

import AWSEC2
```

```
struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "hello-ec2",
        abstract: """
        Demonstrates a simple operation using Amazon EC2.
        """,
        discussion: """
        An example showing how to make a call to Amazon EC2 using the AWS SDK for
 Swift.
        """
    )

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []
```

```
        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let groupNames = await getSecurityGroupNames(ec2Client: ec2Client)

        print("Found \(groupNames.count) security group(s):")

        for group in groupNames {
            print("    \(group)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
```

```
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Swift.

Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    public static ILogger<EC2Basics> _logger = null!;
```

```
    public static EC2Wrapper _ec2Wrapper = null!;
    public static SsmWrapper _ssmWrapper = null!;
    public static UiMethods _uiMethods = null!;

    public static string associationId = null!;
    public static string allocationId = null!;
    public static string instanceId = null!;
    public static string keyPairName = null!;
    public static string groupName = null!;
    public static string tempFileName = null!;
    public static string secGroupId = null!;
    public static bool isInteractive = true;

    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management (Amazon SSM) Service.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        SetUpServices(host);

        var uniqueName = Guid.NewGuid().ToString();
        keyPairName = "mvp-example-key-pair" + uniqueName;
        groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        try
        {
            // Start the scenario.
            _uiMethods.DisplayOverview();
```

```
            _uiMethods.PressEnter(isInteractive);

            // Create the key pair.
            _uiMethods.DisplayTitle("Create RSA key pair");
            Console.Write("Let's create an RSA key pair that you can be use to
");
            Console.WriteLine("securely connect to your EC2 instance.");
            var keyPair = await _ec2Wrapper.CreateKeyPair(keyPairName);

            // Save key pair information to a temporary file.
            tempFileName = _ec2Wrapper.SaveKeyPair(keyPair);

            Console.WriteLine(
                $"Created the key pair: {keyPair.KeyName} and saved it to:
{tempFileName}");
            string? answer = "";
            if (isInteractive)
            {
                do
                {
                    Console.Write("Would you like to list your existing key
pairs? ");
                    answer = Console.ReadLine();
                } while (answer!.ToLower() != "y" && answer.ToLower() != "n");
            }

            if (!isInteractive || answer == "y")
            {
                // List existing key pairs.
                _uiMethods.DisplayTitle("Existing key pairs");

                // Passing an empty string to the DescribeKeyPairs method will
return
                // a list of all existing key pairs.
                var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
                keyPairs.ForEach(kp =>
                {
                    Console.WriteLine(
                        $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
                });
            }

            _uiMethods.PressEnter(isInteractive);
```

```
            // Create the security group.
            Console.WriteLine(
                "Let's create a security group to manage access to your
instance.");
            secGroupId = await _ec2Wrapper.CreateSecurityGroup(groupName,
groupDescription);
            Console.WriteLine(
                "Let's add rules to allow all HTTP and HTTPS inbound traffic and
to allow SSH only from your current IP address.");

            _uiMethods.DisplayTitle("Security group information");
            var secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);

            Console.WriteLine($"Created security group {groupName} in your
default VPC.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll authorize the security group we just created so that
it can");
            Console.WriteLine("access the EC2 instances you create.");
            await _ec2Wrapper.AuthorizeSecurityGroupIngress(groupName);

            secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);
            Console.WriteLine($"Now let's look at the permissions again.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            // Get list of available Amazon Linux 2 Amazon Machine Images (AMIs).
            var parameters =
                await _ssmWrapper.GetParametersByPath(
                    "/aws/service/ami-amazon-linux-latest");

            List<string> imageIds = parameters.Select(param =>
param.Value).ToList();
```

```
            var images = await _ec2Wrapper.DescribeImages(imageIds);

            var i = 1;
            images.ForEach(image =>
            {
                Console.WriteLine($"\t{i++}\t{image.Description}");
            });

            int choice = 1;
            bool validNumber = false;
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an image: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedImage = images[choice - 1];

            // Display available instance types.
            _uiMethods.DisplayTitle("Instance Types");
            var instanceTypes =
                await
_ec2Wrapper.DescribeInstanceTypes(selectedImage.Architecture);

            i = 1;
            instanceTypes.ForEach(instanceType =>
            {
                Console.WriteLine($"\t{i++}\t{instanceType.InstanceType}");
            });
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an instance type: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedInstanceType = instanceTypes[choice - 1].InstanceType;
```

```
            // Create an EC2 instance.
            _uiMethods.DisplayTitle("Creating an EC2 Instance");
            instanceId = await _ec2Wrapper.RunInstances(selectedImage.ImageId,
                selectedInstanceType, keyPairName, secGroupId);

            _uiMethods.PressEnter(isInteractive);

            var instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine(
                "\nYou can use SSH to connect to your instance. For example:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll stop the instance and then start it again to see
what's changed.");

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nNotice the change in the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we will stop the instance again. Then we will create and
associate an");
            Console.WriteLine("Elastic IP address to use with our instance.");
```

```
            await _ec2Wrapper.StopInstances(instanceId);
            _uiMethods.PressEnter(isInteractive);

            _uiMethods.DisplayTitle("Allocate Elastic IP address");
            Console.WriteLine(
                "You can allocate an Elastic IP address and associate it
with your instance\nto keep a consistent IP address even when your instance
restarts.");
            var allocationResponse = await _ec2Wrapper.AllocateAddress();
            allocationId = allocationResponse.AllocationId;
            Console.WriteLine(
                "Now we will associate the Elastic IP address with our
instance.");
            associationId = await _ec2Wrapper.AssociateAddress(allocationId,
instanceId);

            // Start the instance again.
            Console.WriteLine("Now let's start the instance again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("Instance information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nHere is the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            Console.WriteLine("Let's stop and start the instance again.");
            _uiMethods.PressEnter(isInteractive);

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("\nThe instance has stopped.");

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);
```

```
            Console.WriteLine("Note that the IP address did not change this
time.");
            _uiMethods.PressEnter(isInteractive);

            await Cleanup();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, starting
cleanup.");
            await Cleanup();
        }

        _uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
        _uiMethods.PressEnter(isInteractive);
    }

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host"></param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<EC2Basics>(loggerFactory);

        // Now the client is available for injection.
        _ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        _ssmWrapper = host.Services.GetRequiredService<SsmWrapper>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Clean up any resources from the scenario.
    /// </summary>
    /// <returns></returns>
    public static async Task Cleanup()
    {
        _uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Now let's clean up the resources we created.");
```

```
        Console.WriteLine("Disassociate the Elastic IP address and release it.");
        // Disassociate the Elastic IP address.
        await _ec2Wrapper.DisassociateIp(associationId);

        // Delete the Elastic IP address.
        await _ec2Wrapper.ReleaseAddress(allocationId);

        // Terminate the instance.
        Console.WriteLine("Terminating the instance we created.");
        await _ec2Wrapper.TerminateInstances(instanceId);

        // Delete the security group.
        Console.WriteLine($"Deleting the Security Group: {groupName}.");
        await _ec2Wrapper.DeleteSecurityGroup(secGroupId);

        // Delete the RSA key pair.
        Console.WriteLine($"Deleting the key pair: {keyPairName}");
        await _ec2Wrapper.DeleteKeyPair(keyPairName);
        Console.WriteLine("Deleting the temporary file with the key
 information.");
        _ec2Wrapper.DeleteTempFile(tempFileName);
        _uiMethods.PressEnter(isInteractive);
    }
}
```

Definieren Sie eine Klasse, die EC2-Aktionen umschließt.

```
/// <summary>
/// Methods of this class perform Amazon Elastic Compute Cloud (Amazon EC2).
/// </summary>
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEC2;
    private readonly ILogger<EC2Wrapper> _logger;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EC2 client.</param>
    /// <param name="logger">The injected logger.</param>
    public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
    {
```

```
        _amazonEC2 = amazonService;
        _logger = logger;
    }

    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP
 address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
 ID {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Associates an Elastic IP address with an instance. When this association
 is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
 {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
 {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
```

```
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }

    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
```

```
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);
```

```
        return pemFileName;
    }

    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and
 description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
 {ec2Exception.Message}");
            }
```

```
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
 {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Create a new Amazon EC2 VPC.
    /// </summary>
    /// <param name="cidrBlock">The CIDR block for the new security group.</
param>
    /// <returns>The VPC Id of the new VPC.</returns>
    public async Task<string?> CreateVPC(string cidrBlock)
    {

        try
        {
            var response = await _amazonEC2.CreateVpcAsync(new CreateVpcRequest
            {
                CidrBlock = cidrBlock,
            });

            Vpc vpc = response.Vpc;
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
            return vpc.VpcId;
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"Couldn't create VPC because: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
```

```
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 VPC.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteVpc(string vpcId)
    {
        var request = new DeleteVpcRequest
        {
            VpcId = vpcId,
        };

        var response = await _amazonEC2.DeleteVpcAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Get information about existing Amazon EC2 images.
    /// </summary>
    /// <returns>A list of image information.</returns>
    public async Task<List<Image>> DescribeImages(List<string>? imageIds)
    {
        var request = new DescribeImagesRequest();
        if (imageIds is not null)
        {
            // If the imageIds list is not null, add the list
            // to the request object.
            request.ImageIds = imageIds;
        }

        var response = await _amazonEC2.DescribeImagesAsync(request);
        return response.Images;
    }

    /// <summary>
    /// Display the information returned by DescribeImages.
    /// </summary>
    /// <param name="images">The list of image information to display.</param>
    public void DisplayImageInfo(List<Image> images)
    {
        images.ForEach(image =>
        {
            Console.WriteLine($"{image.Name} Created on: {image.CreationDate}");
        });

    }

    /// <summary>
    /// Get information about an Amazon EC2 instance.
    /// </summary>
    /// <param name="instanceId">The instance Id of the EC2 instance.</param>
    /// <returns>An EC2 instance.</returns>
    public async Task<Instance> DescribeInstance(string instanceId)
    {
        var response = await _amazonEC2.DescribeInstancesAsync(
            new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
        return response.Reservations[0].Instances[0];
    }

    /// <summary>
```

```
    /// Display EC2 instance information.
    /// </summary>
    /// <param name="instance">The instance Id of the EC2 instance.</param>
    public void DisplayInstanceInformation(Instance instance)
    {
        Console.WriteLine($"ID: {instance.InstanceId}");
        Console.WriteLine($"Image ID: {instance.ImageId}");
        Console.WriteLine($"{instance.InstanceType}");
        Console.WriteLine($"Key Name: {instance.KeyName}");
        Console.WriteLine($"VPC ID: {instance.VpcId}");
        Console.WriteLine($"Public IP: {instance.PublicIpAddress}");
        Console.WriteLine($"State: {instance.State.Name}");
    }

    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
```

```
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));
```

```
            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size
strings conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
```

```
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }

            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
```

```
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups.
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
```

```
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }


    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId =
associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };
```

```
            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
 {ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance
 {ec2InstanceId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId =
 allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
```

```
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };
```

```
            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);
```

```
            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
```

```
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
```

```
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
###############################################################################
# function get_started_with_ec2_instances
#
# Runs an interactive scenario that shows how to get started using EC2 instances.
#
#     "EC2 access" permissions are needed to run this code.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function get_started_with_ec2_instances() {
  # Requires version 4 for mapfile.
  local required_version=4.0

  # Get the current Bash version
  # Check if BASH_VERSION is set
  local current_version
  if [[ -n "$BASH_VERSION" ]]; then
    # Convert BASH_VERSION to a number for comparison
    current_version=$BASH_VERSION
  else
    # Get the current Bash version using the bash command
    current_version=$(bash --version | head -n 1 | awk '{ print $4 }')
  fi

  # Convert version strings to numbers for comparison
  local required_version_num current_version_num
  required_version_num=$(echo "$required_version" | awk -F. '{ print ($1 * 10000)
 + ($2 * 100) + $3 }')
```

```
 current_version_num=$(echo "$current_version" | awk -F. '{ print ($1 * 10000) +
($2 * 100) + $3 }')

 # Compare versions
 if ((current_version_num < required_version_num)); then
   echo "Error: This script requires Bash version $required_version or higher."
   echo "Your current Bash version is number is $current_version."
   exit 1
 fi

 {
   if [ "$EC2_OPERATIONS_SOURCED" != "True" ]; then

     source ./ec2_operations.sh
   fi
 }

 echo_repeat "*" 88
 echo "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
instances demo."
 echo_repeat "*" 88
 echo

 echo "Let's create an RSA key pair that you can be use to securely connect to "
 echo "your EC2 instance."

 echo -n "Enter a unique name for your key: "
 get_input
 local key_name
 key_name=$get_input_result

 local temp_dir
 temp_dir=$(mktemp -d)
 local key_file_name="$temp_dir/${key_name}.pem"

 if ec2_create_keypair -n "${key_name}" -f "${key_file_name}"; then
   echo "Created a key pair $key_name and saved the private key to
$key_file_name"
   echo
 else
   errecho "The key pair failed to create. This demo will exit."
   return 1
 fi
```

```
chmod 400 "${key_file_name}"

if yes_no_input "Do you want to list some of your key pairs? (y/n) "; then
  local keys_and_fingerprints
  keys_and_fingerprints="$(ec2_describe_key_pairs)" && {
    local image_name_and_id
    while IFS=$'\n' read -r image_name_and_id; do
      local entries
      IFS=$'\t' read -ra entries <<<"$image_name_and_id"
      echo "Found rsa key ${entries[0]} with fingerprint:"
      echo "     ${entries[1]}"
    done <<<"$keys_and_fingerprints"

  }
fi

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's create a security group to manage access to your instance."
echo -n "Enter a unique name for your security group: "
get_input
local security_group_name
security_group_name=$get_input_result
local security_group_id
security_group_id=$(ec2_create_security_group -n "$security_group_name" \
  -d "Security group for EC2 instance") || {
  errecho "The security failed to create. This demo will exit."
  clean_up "$key_name" "$key_file_name"
  return 1
}

echo "Security group created with ID $security_group_id"
echo

local public_ip
public_ip=$(curl -s http://checkip.amazonaws.com)

echo "Let's add a rule to allow SSH only from your current IP address."
echo "Your public IP address is $public_ip"
echo -n "press return to add this rule to your security group."
get_input
```

```
  if ! ec2_authorize_security_group_ingress -g "$security_group_id" -i
 "$public_ip" -p tcp -f 22 -t 22; then
    errecho "The security group rules failed to update. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo "Security group rules updated"

  local security_group_description
  security_group_description="$(ec2_describe_security_groups -g
 "${security_group_id}")" || {
    errecho "Failed to describe security groups. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  mapfile -t parameters <<<"$security_group_description"
  IFS=$'\t' read -ra entries <<<"${parameters[0]}"
  echo "Security group: ${entries[0]}"
  echo "    ID: ${entries[1]}"
  echo "    VPC: ${entries[2]}"
  echo "Inbound permissions:"
  IFS=$'\t' read -ra entries <<<"${parameters[1]}"
  echo "    IpProtocol: ${entries[0]}"
  echo "    FromPort: ${entries[1]}"
  echo "    ToPort: ${entries[2]}"
  echo "    CidrIp: ${parameters[2]}"

  local parameters
  parameters="$(ssm_get_parameters_by_path -p "/aws/service/ami-amazon-linux-
latest")" || {
    errecho "Failed to get parameters. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  local image_ids=""
  mapfile -t parameters <<<"$parameters"
  for image_name_and_id in "${parameters[@]}"; do
    IFS=$'\t' read -ra values <<<"$image_name_and_id"
    if [[ "${values[0]}" == *"amzn2"* ]]; then
      image_ids+="${values[1]} "
```

```
    fi
  done

  local images
  images="$(ec2_describe_images -i "$image_ids")" || {
    errecho "Failed to describe images. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  new_line_and_tab_to_list "$images"
  local images=("${list_result[@]}")

  # Get the size of the array
  local images_count=${#images[@]}

  if ((images_count == 0)); then
    errecho "No images found. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
  for ((i = 0; i < images_count; i += 3)); do
    echo "$(((i / 3) + 1)) - ${images[$i]}"
  done

  integer_input "Please enter the number of the AMI you want to use: " 1
 "$((images_count / 3))"
  local choice=$get_input_result
  choice=$(((choice - 1) * 3))

  echo "Great choice."
  echo

  local architecture=${images[$((choice + 1))]}
  local image_id=${images[$((choice + 2))]}
  echo "Here are some instance types that support the ${architecture}
 architecture of the image:"
```

```
  response="$(ec2_describe_instance_types -a "${architecture}" -t
 "*.micro,*.small")" || {
    errecho "Failed to describe instance types. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  local instance_types
  mapfile -t instance_types <<<"$response"

  # Get the size of the array
  local instance_types_count=${#instance_types[@]}

  echo "Here are some options:"
  for ((i = 0; i < instance_types_count; i++)); do
    echo "$((i + 1)) - ${instance_types[$i]}"
  done

  integer_input "Which one do you want to use? " 1 "${#instance_types[@]}
"
  choice=$get_input_result
  local instance_type=${instance_types[$((choice - 1))]}
  echo "Another great choice."
  echo

  echo "Creating your instance and waiting for it to start..."
  local instance_id
  instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k
 "$key_name" -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  ec2_wait_for_instance_running -i "$instance_id"
  echo "Your instance is ready:"
  echo

  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  echo
  print_instance_details "${instance_details}"
```

```
  local public_ip
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  echo
  echo "You can use SSH to connect to your instance"
  echo "If the connection attempt times out, you might have to manually update
 the SSH ingress rule"
  echo "for your IP address in the AWS Management Console."
  connect_to_instance "$key_file_name" "$public_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"

  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  public_ip=$(echo "${instance_details}" | awk '{print $6}')

  echo "Every time your instance is restarted, its public IP address changes"
  connect_to_instance "$key_file_name" "$public_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "You can allocate an Elastic IP address and associate it with your
 instance"
```

```
  echo "to keep a consistent IP address even when your instance restarts."

  local result
  result=$(ec2_allocate_address -d vpc) || {
    errecho "Failed to allocate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
    return 1
  }

  local elastic_ip allocation_id
  elastic_ip=$(echo "$result" | awk '{print $1}')
  allocation_id=$(echo "$result" | awk '{print $2}')

  echo "Allocated static Elastic IP address: $elastic_ip"

  local association_id
  association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id")
 || {
    errecho "Failed to associate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
 "$allocation_id"
    return 1
  }

  echo "Associated your Elastic IP with your instance."
  echo "You can now use SSH to connect to your instance by using the Elastic IP."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"
```

```
  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  echo "Because you have associated an Elastic IP with your instance, you can"
  echo "connect by using a consistent IP address after the instance restarts."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  if yes_no_input "Do you want to delete the resources created in this demo: (y/
n) "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
      "$allocation_id" "$association_id"
  else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
  fi
}

###############################################################################
# function clean_up
#
# This function cleans up the created resources.
#     $1 - The name of the ec2 key pair to delete.
#     $2 - The name of the key file to delete.
#     $3 - The ID of the security group to delete.
#     $4 - The ID of the instance to terminate.
#     $5 - The ID of the elastic IP address to release.
#     $6 - The ID of the elastic IP address to disassociate.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
```

```
###############################################################################
function clean_up() {
  local result=0
  local key_pair_name=$1
  local key_file_name=$2
  local security_group_id=$3
  local instance_id=$4
  local allocation_id=$5
  local association_id=$6

  if [ -n "$association_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_disassociate_address -a "$association_id"); then
      echo "Disassociated elastic IP address with ID $association_id"
    else
      errecho "The elastic IP address disassociation failed."
      result=1
    fi
  fi

  if [ -n "$allocation_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_release_address -a "$allocation_id"); then
      echo "Released elastic IP address with ID $allocation_id"
    else
      errecho "The elastic IP address release failed."
      result=1
    fi
  fi

  if [ -n "$instance_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_terminate_instances -i "$instance_id"); then
      echo "Started terminating instance with ID $instance_id"

      ec2_wait_for_instance_terminated -i "$instance_id"
    else
      errecho "The instance terminate failed."
      result=1
    fi
  fi

  if [ -n "$security_group_id" ]; then
    # bashsupport disable=BP2002
```

```
    if (ec2_delete_security_group -i "$security_group_id"); then
      echo "Deleted security group with ID $security_group_id"
    else
      errecho "The security group delete failed."
      result=1
    fi
  fi

  if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
      echo "Deleted key pair named $key_pair_name"
    else
      errecho "The key pair delete failed."
      result=1
    fi
  fi

  if [ -n "$key_file_name" ]; then
    rm -f "$key_file_name"
  fi

  return $result
}

###############################################################################
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
 Parameter Store
# by specifying a parameter path.
#
# Parameters:
#       -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ssm_get_parameters_by_path() {
  local parameter_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
```

```
  function usage() {
    echo "function ssm_get_parameters_by_path"
    echo "Retrieves one or more parameters from the AWS Systems Manager Parameter
 Store by specifying a parameter path."
    echo "  -p parameter_path - The path of the parameter(s) to retrieve."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "p:h" option; do
    case "${option}" in
      p) parameter_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$parameter_path" ]]; then
    errecho "ERROR: You must provide a parameter path with the -p parameter."
    usage
    return 1
  fi

  response=$(aws ssm get-parameters-by-path \
    --path "$parameter_path" \
    --query "Parameters[*].[Name, Value]" \
    --output text) || {
    aws_cli_error_log $?
    errecho "ERROR: AWS reports get-parameters-by-path operation failed.
$response"
    return 1
  }

  echo "$response"

  return 0
```

```
}

##############################################################################
# function print_instance_details
#
# This function prints the details of an Amazon Elastic Compute Cloud (Amazon
 EC2) instance.
#
# Parameters:
#       instance_details - The instance details in the format "InstanceId ImageId
 InstanceType KeyName VpcId PublicIpAddress State.Name".
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function print_instance_details() {
  local instance_details="$1"

  if [[ -z "${instance_details}" ]]; then
    echo "Error: Missing required instance details argument."
    return 1
  fi

  local instance_id image_id instance_type key_name vpc_id public_ip state
  instance_id=$(echo "${instance_details}" | awk '{print $1}')
  image_id=$(echo "${instance_details}" | awk '{print $2}')
  instance_type=$(echo "${instance_details}" | awk '{print $3}')
  key_name=$(echo "${instance_details}" | awk '{print $4}')
  vpc_id=$(echo "${instance_details}" | awk '{print $5}')
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  state=$(echo "${instance_details}" | awk '{print $7}')

  echo "    ID: ${instance_id}"
  echo "    Image ID: ${image_id}"
  echo "    Instance type: ${instance_type}"
  echo "    Key name: ${key_name}"
  echo "    VPC ID: ${vpc_id}"
  echo "    Public IP: ${public_ip}"
  echo "    State: ${state}"

  return 0
}
```

```
##############################################################################
# function connect_to_instance
#
# This function displays the public IP address of an Amazon Elastic Compute Cloud
 (Amazon EC2) instance and prompts the user to connect to the instance via SSH.
#
# Parameters:
#       $1 - The name of the key file used to connect to the instance.
#       $2 - The public IP address of the instance.
#
# Returns:
#       None
##############################################################################
function connect_to_instance() {
  local key_file_name="$1"
  local public_ip="$2"

  # Validate the input parameters
  if [[ -z "$key_file_name" ]]; then
    echo "ERROR: You must provide a key file name as the first argument." >&2
    return 1
  fi

  if [[ -z "$public_ip" ]]; then
    echo "ERROR: You must provide a public IP address as the second argument."
 >&2
    return 1
  fi

  # Display the public IP address and connection command
  echo "To connect, run the following command:"
  echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

  # Prompt the user to connect to the instance
  if yes_no_input "Do you want to connect now? (y/n) "; then
    echo "After you have connected, you can return to this example by typing
 'exit'"
    ssh -i "${key_file_name}" ec2-user@"${public_ip}"
  fi
}

##############################################################################
# function get_input
#
```

```
# This function gets user input from the command line.
#
# Outputs:
#   User input to stdout.
#
# Returns:
#       0
##############################################################################
function get_input() {

  if [ -z "${mock_input+x}" ]; then
    read -r get_input_result
  else

    if [ "$mock_input_array_index" -lt ${#mock_input_array[@]} ]; then
      get_input_result="${mock_input_array[$mock_input_array_index]}"
      # bashsupport disable=BP2001
      # shellcheck disable=SC2206
      ((mock_input_array_index++))
      echo -n "$get_input_result"
    else
      echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
      return 1
    fi
  fi

  return 0
}

##############################################################################
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#       $1 - The prompt.
#
# Returns:
#       0 - If yes.
#       1 - If no.
##############################################################################
function yes_no_input() {
  if [ -z "$1" ]; then
    echo "Internal error yes_no_input"
```

```
    return 1
  fi

  local index=0
  local response="N"
  while [[ $index -lt 10 ]]; do
    index=$((index + 1))
    echo -n "$1"
    if ! get_input; then
      return 1
    fi
    response=$(echo "$get_input_result" | tr '[:upper:]' '[:lower:]')
    if [ "$response" = "y" ] || [ "$response" = "n" ]; then
      break
    else
      echo -e "\nPlease enter or 'y' or 'n'."
    fi
  done

  echo

  if [ "$response" = "y" ]; then
    return 0
  else
    return 1
  fi
}

##############################################################################
# function integer_input
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#       $1 - The prompt message to display to the user.
#       $2 - The minimum value of the accepted range.
#       $3 - The maximum value of the accepted range.
#
# Returns:
#       The valid integer input from the user.
#       If the input is invalid or out of range, the function will continue
#       prompting the user until a valid input is provided.
##############################################################################
```

```
function integer_input() {
  local prompt="$1"
  local min_value="$2"
  local max_value="$3"
  local input=""

  while true; do
    # Display the prompt message and wait for user input
    echo -n "$prompt"

    if ! get_input; then
      return 1
    fi

    input="$get_input_result"

    # Check if the input is a valid integer
    if [[ "$input" =~ ^-?[0-9]+$ ]]; then
      # Check if the input is within the specified range
      if ((input >= min_value && input <= max_value)); then
        return 0
      else
        echo "Error: Input, $input, must be between $min_value and $max_value."
      fi
    else
      echo "Error: Invalid input- $input. Please enter an integer."
    fi
  done
}
##############################################################################
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#       $1 - The input string containing newlines and tabs.
#
# Returns:
#       The resulting list (array) is stored in the global variable
#       'list_result'.
##############################################################################
function new_line_and_tab_to_list() {
  local input=$1
```

```
  export list_result

  list_result=()
  mapfile -t lines <<<"$input"
  local line
  for line in "${lines[@]}"; do
    IFS=$'\t' read -ra parameters <<<"$line"
    list_result+=("${parameters[@]}")
  done
}

###############################################################################
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#       $1 - The string.
#       $2 - Number of times to print the string.
#
# Outputs:
#   String 'n' times to stdout.
#
# Returns:
#       0
###############################################################################
function echo_repeat() {
  local end=$2
  for ((i = 0; i < end; i++)); do
    echo -n "$1"
  done
  echo
}
```

Die in diesem Szenario verwendeten „DynamoDB“-Funktionen.

```
###############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
```

```
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
```

```
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}

###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 key pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
```

```
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the
 operation fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security
 group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
```

```
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n
 parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe
 (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
 --query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
```

```
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
 failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

```
##############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g.,
 x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
##############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-
t|--type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance
 types (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
```

```
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
  echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
  if [[ $i -lt $((array_size - 1)) ]]; then
    echo -n ',' >>"$tmp_json_file"
  fi
done
echo -n ']},
  {
  "Name": "instance-type",
    "Values": [' >>"$tmp_json_file"
IFS=',' read -ra items <<<"$instance_types"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
```

```
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}

###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
 parameter."
    usage
    return 1
```

```
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_describe_instances
```

```
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
```

```
    return 1
  }

  return 0
}

##############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }

  return 0
}

###############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic
 Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_allocate_address() {
```

```
  local domain response

  # Function to display usage information
  function usage() {
    echo "function ec2_allocate_address"
    echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region."
    echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard')."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "d:h" option; do
    case "${option}" in
      d) domain="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
 or 'standard')."
    return 1
  fi

  if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
  fi

  # Allocate the Elastic IP address
  response=$(aws ec2 allocate-address \
    --domain "$domain" \
```

```
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic
 IP address with."
    echo ""
  }
```

```
  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

```
##############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
 Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
 association of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
```

```
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
```

```
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi
```

```
  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Szenario verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets additional information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI’s
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
```

```
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            Where:
               keyName -  A key pair name (for example, TestKeyPair).\s
               fileName -  A file name where the key information is written to.\s
               groupName - The name of the security group.\s
               groupDesc - The description of the security group.\s
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to
 easily launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.
```

```
            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2
 instances as part of their
            application deployment pipelines. With EC2, teams can focus on
 building and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for
 this service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material
 as a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a
 private key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
```

```
            CreateKeyPairResponse response = future.join();
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error
code {}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " +
(rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving
on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with
amzn2 in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on
an EC2 instance.
            """);
        waitForInputToContinue(scanner);
```

```
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM)
or server image.
           It uniquely identifies a particular version of an EC2 instance,
including its operating system,
           pre-installed software, and any custom configurations. This allows you
to consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
```

```
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future =
ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
```

```
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
 instance type, the security group, and the EC2 AMI value.");
        logger.info("Once the EC2 instance is created, it is placed into a
 running state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
 ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
 ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
 cause.getMessage() : rt.getMessage()));
```

```
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
```

```
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains
the same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to
be accessed consistently from a
            known IP address.

            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you
have purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique
Allocation ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved
Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: "
+allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
```

```
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: "
+associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
```

```
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
```

```
    }
}
```

Definieren Sie eine Klasse, die EC2-Aktionen umschließt.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
```

```
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger =
 LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
```

```
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
              .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .build();

           ec2AsyncClient = Ec2AsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ec2AsyncClient;
   }

   /**
    * Deletes a key pair asynchronously.
    *
    * @param keyPair the name of the key pair to delete
    * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
    *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
    *         that provides the result of the key pair deletion operation.
    */
   public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
keyPair) {
       DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
           .keyName(keyPair)
```

```
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " +
keyPair, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is
deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
```

```
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has
been terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a
waiter to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }

   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous
operation of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse>
releaseEC2AddressAsync(String allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();
```

```
        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP
address", ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

```
    /**
     * Associates an Elastic IP address with an EC2 instance asynchronously.
     *
     * @param instanceId    the ID of the EC2 instance to associate the Elastic
IP address with
     * @param allocationId  the allocation ID of the Elastic IP address to
associate
     * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
     *         or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after
associating address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
```

```
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a
string describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String
newInstanceId) {
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId +
" not found.");
            }
            return publicIpAddress;
```

```
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }

    /**
     * Runs an EC2 instance asynchronously.
     *
     * @param instanceType The instance type to use for the EC2 instance.
     * @param keyName The name of the key pair to associate with the EC2
instance.
     * @param groupName The name of the security group to associate with the EC2
instance.
     * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
     * @return A {@link CompletableFuture} that completes with the ID of the
started EC2 instance.
     * @throws RuntimeException If there is an error running the EC2 instance.
     */
    public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
```

```
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it
returns a
     * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type
string for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
```

```
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }

    /**
     * Asynchronously describes an AWS EC2 image with the specified image ID.
     *
     * @param imageId the ID of the image to be described
     * @return a {@link CompletableFuture} that, when completed, contains the ID
of the described image
     * @throws RuntimeException if no images are found with the provided image
ID, or if an error occurs during the AWS API call
     */
    public CompletableFuture<String> describeImageAsync(String imageId) {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(imageId)
            .build();

        AtomicReference<String> imageIdRef = new AtomicReference<>();
        DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
        return paginator.subscribe(response -> {
            response.images().stream()
                .filter(image -> image.imageId().equals(imageId))
                .findFirst()
                .ifPresent(image -> {
                    logger.info("The description of the image is " +
image.description());
                    logger.info("The name of the image is " + image.name());
```

```
                    imageIdRef.set(image.imageId());
                });
        }).thenApply(v -> {
            String id = imageIdRef.get();
            if (id == null) {
                throw new RuntimeException("No images found with the provided
image ID.");
            }
            return id;
        }).exceptionally(ex -> {
            logger.info("Failed to describe image: " + ex.getMessage());
            throw new RuntimeException("Failed to describe image", ex);
        });
    }

    /**
     * Retrieves the parameter values asynchronously using the AWS Systems
Manager (SSM) API.
     *
     * @return a {@link CompletableFuture} that holds the response from the SSM
API call to get parameters by path
     */
    public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });
```

```
        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
```

```
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
```

```
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
```

```
     *
     * @return a {@link CompletableFuture} that, when completed, contains the
first default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for
the instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has
been stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
```

```
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
```

```
     * @return a {@link CompletableFuture} that completes when the instance has
been started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AllocateAddress
  - AssociateAddress

- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Diese Datei enthält eine Liste häufig verwendeter Aktionen für EC2. Die Schritte sind mit einem Szenario-Framework aufgebaut, das die Ausführung eines interaktiven Beispiels vereinfacht. Den vollständigen Kontext finden Sie im GitHub Repository.

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
```

```
import { join } from "node:path";
import { get } from "node:http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

/**
 * @typedef {{
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,
 *   errors: Error[],
 *   keyPairId?: string,
 *   tmpDirectory?: string,
 *   securityGroupId?: string,
 *   ipAddress?: string,
 *   images?: import('@aws-sdk/client-ec2').Image[],
 *   image?: import('@aws-sdk/client-ec2').Image,
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],
```

```
 *   instanceId?: string,
 *   instanceIpAddress?: string,
 *   allocationId?: string,
 *   allocatedIpAddress?: string,
 *   associationId?: string,
 * }} State
 */

/**
 * A skip function provided to the `skipWhen` of a Step when you want
 * to ignore that step if any errors have occurred.
 * @param {State} state
 */
const skipWhenErrors = (state) => state.errors.length > 0;

const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;

export const confirm = new ScenarioInput("confirmContinue", "Continue?", {
  type: "confirm",
  skipWhen: skipWhenErrors,
});

export const exitOnNoConfirm = new ScenarioAction(
  "exitOnConfirmContinueFalse",
  (/** @type { { earlyExit: boolean } & Record<string, any>} */ state) => {
    if (!state[confirm.name]) {
      state.earlyExit = true;
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

export const greeting = new ScenarioOutput(
  "greeting",
  `

Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
 - A key pair - This is for SSH access to your EC2 instance. You only need to
 provide the name.
```

```
 - A security group - This is used for configuring access to your instance.
 Again, only the name is needed.
 - An IP address - Your public IP address will be fetched.
 - An Amazon Machine Image (AMI)
 - A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyPairName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
      await writeFile(
        `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
        KeyMaterial,
        {
          mode: 0o400,
        },
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidKeyPair.Duplicate"
      ) {
        caught.message = `${caught.message}. Try another key name.`;
      }

      state.errors.push(caught);
```

```
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(
        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
      ) {
        caught.message = `${caught.message}. Did you provide the required value?
`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no key pair to delete or the user chooses
```

```
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.keyPairId || !state[confirmDeleteKeyPair.name],
  },
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",
  { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
  "createSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Create a new security group that will be used to configure ingress/
egress for
      // an EC2 instance.
      const { GroupId } = await state.ec2Client.send(
        new CreateSecurityGroupCommand({
          GroupName: state[provideSecurityGroupName.name],
          Description: "A security group for the Amazon EC2 example.",
        }),
      );
      state.securityGroupId = GroupId;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {
        caught.message = `${caught.message}. Please provide a different name for
 your security group.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);
```

```
export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
  },
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (/** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
```

```
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
      state.ipAddress = ipAddress;
      // Allow ingress from the IP address above to the security group.
      // This will allow you to SSH into the EC2 instance.
      const command = new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.securityGroupId,
        IpPermissions: [
          {
            IpProtocol: "tcp",
            FromPort: 22,
            ToPort: 22,
            IpRanges: [{ CidrIp: `${ipAddress}/32` }],
          },
        ],
      });

      await state.ec2Client.send(command);
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
```

```
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS
 Systems Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon
 EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
 parameters.

    // Create the paginator for getting images. Actions that return multiple
 pages of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
      },
    );

    try {
      for await (const page of getParametersByPathPaginator) {
        for (const param of page.Parameters) {
          // Filter by Amazon Linux 2
          if (param.Name.includes("amzn2")) {
            AMIs.push(param.Value);
          }
        }
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidFilterValue") {
        caught.message = `${caught.message} Please provide a valid filter value
 for paginateGetParametersByPath.`;
      }
      state.errors.push(caught);
```

```
      return;
    }

    const imageDetails = [];
    const describeImagesPaginator = paginateDescribeImages(
      { client: state.ec2Client },
      // The images found from the call to SSM.
      { ImageIds: AMIs },
    );

    try {
      // Get more details for the images found above.
      for await (const page of describeImagesPaginator) {
        imageDetails.push(...(page.Images || []));
      }

      // Store the image details for later use.
      state.images = imageDetails;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
        caught.message = `${caught.message}. Please provide a valid image id.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);
```

```
export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "*.small"] },
        ],
      },
    );

    const instanceTypes = [];

    try {
      for await (const page of paginator) {
        if (page.InstanceTypes.length) {
          instanceTypes.push(...(page.InstanceTypes || []));
        }
      }

      if (!instanceTypes.length) {
        state.errors.push(
          "No instance types matched the instance type filters.",
        );
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check the provided values and
 try again.`;
      }

      state.errors.push(caught);
    }

    state.instanceTypes = instanceTypes;
```

```
  },
  { skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyPairName.name],
        SecurityGroupIds: [state.securityGroupId],
        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your
 ability to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
 least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and
 `MaxCount` set to the same value.
        // If you want to launch up to a certain number of instances, use
 `MaxCount` and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to
 exceed a maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
```

```
      }),
    );

    state.instanceId = Instances[0].InstanceId;

    try {
      // Poll `DescribeInstanceStatus` until status is "ok".
      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logRunInstance = new ScenarioOutput(
  "logRunInstance",
  "The next step is to run your EC2 instance for the first time. This can take a
 few minutes.",
  { header: true, skipWhen: skipWhenErrors },
);

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (/** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
```

```
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );

      for await (const page of paginator) {
        for (const reservation of page.Reservations) {
          instances.push(...reservation.Instances);
        }
      }
      if (instances.length !== 1) {
        throw new Error(`Instance ${state.instanceId} not found.`);
      }

      // The only info we need is the IP address for SSH purposes.
      state.instanceIpAddress = instances[0].PublicIpAddress;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check provided values and try
 again.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);
```

```
export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStopped(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  // Don't try to stop an instance that doesn't exist.
  { skipWhen: (/** @type { State } */ state) => !state.instanceId },
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is
 restarted.",
    "The next step is to stop and restart your instance to demonstrate this
 behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
```

```
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
  "logIpAllocation",
  [
    "It is possible to have a static IP address.",
    "To demonstrate this, an IP will be allocated and associated to your EC2
 instance.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
```

```
  "allocateIp",
  async (/** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours
 until you release it.
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
      state.allocationId = AllocationId;
      state.allocatedIpAddress = PublicIp;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "MissingParameter") {
        caught.message = `${caught.message}. Did you provide these values?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
  "associateIp",
  async (/** @type { State } */ state) => {
    try {
      // Associate an allocated IP address to an EC2 instance. An IP address can
 be allocated
      // with the AllocateAddress action.
      const { AssociationId } = await state.ec2Client.send(
        new AssociateAddressCommand({
          AllocationId: state.allocationId,
          InstanceId: state.instanceId,
        }),
      );
      state.associationId = AssociationId;
      // Update the IP address that is being tracked to match
      // the one just associated.
      state.instanceIpAddress = state.allocatedIpAddress;
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Did you provide the ID of a valid
 Elastic IP address AllocationId?`;
```

```
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(
  "logStaticIpProof",
  "The IP address should remain the same even after stopping and starting the
 instance.",
  { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
  "logCleanUp",
  "That's it! You can choose to clean up the resources now, or clean them up on
 your own later.",
  { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
  "confirmDisassociateAddress",
  "Do you want to disassociate and release the static IP address created
 earlier?",
  {
    type: "confirm",
    skipWhen: (/** @type { State } */ state) => !state.associationId,
  },
);

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
```

```
      ) {
        caught.message = `${caught.message}. Please provide a valid association
 ID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid
 AllocationID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.allocationId,
  },
);

export const confirmTerminateInstance = new ScenarioInput(
  "confirmTerminateInstance",
  "Do you want to terminate the instance?",
  // Don't do anything when an instance was never run.
  {
```

```
    skipWhen: (/** @type { State } */ state) => !state.instanceId,
    type: "confirm",
  },
);

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new TerminateInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );
      await waitUntilInstanceTerminated(
        { client: state.ec2Client },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
 the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no instance to terminate or the
    // use chooses not to terminate.
    skipWhen: (/** @type { State } */ state) =>
      !state.instanceId || !state[confirmTerminateInstance.name],
  },
);

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (/** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
```

```
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances

- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks:

 1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI’s
 architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance
 type.
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
```

```
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as
 a .pem file.")
```

```
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId,
myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security
group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in
the name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
```

```
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)

    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
 instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)
```

```
    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
```

```
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
```

```
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
```

```
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }
```

```
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is
 ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient.fromEnvironment { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
```

```
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() +
 " and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
```

```
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance,
 associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
        Initializes the EC2InstanceScenario with the necessary AWS service
 wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a
 remote execution
                            environment. Defaults to False. If True, the script
 won't prompt
                            for user interaction.
        """
        self.inst_wrapper = inst_wrapper
        self.key_wrapper = key_wrapper
        self.sg_wrapper = sg_wrapper
        self.eip_wrapper = eip_wrapper
        self.ssm_client = ssm_client
        self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
```

```
        Creates an RSA key pair for SSH access to the EC2 instance and lists
available key pairs.
        """
        console.print("**Step 1: Create a Secure Key Pair**", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4)  # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
                if time.time() - start_time > 2:
                    console.print(
                        "Taking longer than expected! Please wait...",
                        style="bold yellow",
                    )

    def create_security_group(self) -> None:
        """
        Creates a security group that controls access to the EC2 instance and
adds a rule
        to allow SSH access from the user's current public IP address.
        """
        console.print("**Step 2: Create a Security Group**", style="bold cyan")
        console.print(
```

```
            "Security groups manage access to your instance. Let's create one."
        )
        sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Security Group Name**: {sg_name}")

        # Create the security group and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Security Group") as bar:
            self.sg_wrapper.create(
                sg_name, "Security group for example: get started with
instances."
            )
            time.sleep(0.5)
            bar()

        console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

        # Get the current public IP to set up SSH access.
        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        console.print(
            "Let's add a rule to allow SSH only from your current IP address."
        )
        console.print(f"- **Your Public IP Address**: {current_ip_address}")
        console.print("- Automatically adding SSH rule...")

        # Update security group rules to allow SSH and simulate with a progress
bar.
        with alive_bar(1, title="Updating Security Group Rules") as bar:
            response = self.sg_wrapper.authorize_ingress(current_ip_address)
            time.sleep(0.4)
            if response and response.get("Return"):
                console.print("- **Security Group Rules Updated**.")
            else:
                console.print(
                    "- **Error**: Couldn't update security group rules.",
                    style="bold red",
                )
            bar()

        self.sg_wrapper.describe(self.sg_wrapper.security_group)

    def create_instance(self) -> None:
        """
```

```
        Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
 pair
        and security group. Displays instance details and SSH connection
 information.
        """
        # Retrieve Amazon Linux 2 AMIs from SSM.
        ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
        ami_options = []
        for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
            ami_options += page["Parameters"]
        amzn2_images = self.inst_wrapper.get_images(
            [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
        )
        console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
        console.print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
        )
        image_choice = 0
        console.print(f"- Selected AMI: {amzn2_images[image_choice]
['ImageId']}\n")

        # Display instance types compatible with the selected AMI
        inst_types = self.inst_wrapper.get_instance_types(
            amzn2_images[image_choice]["Architecture"]
        )
        inst_type_choice = 0
        console.print(
            f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"
        )

        console.print("Creating your instance and waiting for it to start...")
        with alive_bar(1, title="Creating Instance") as bar:
            self.inst_wrapper.create(
                amzn2_images[image_choice]["ImageId"],
                inst_types[inst_type_choice]["InstanceType"],
                self.key_wrapper.key_pair["KeyName"],
                [self.sg_wrapper.security_group],
            )
            time.sleep(21)
            bar()
```

```
        console.print(f"**Success! Your instance is ready:**\n", style="bold
green")
        self.inst_wrapper.display()

        console.print(
            "You can use SSH to connect to your instance. "
            "If the connection attempt times out, you might have to manually
update "
            "the SSH ingress rule for your IP address in the AWS Management
Console."
        )
        self._display_ssh_info()

    def _display_ssh_info(self) -> None:
        """
        Displays SSH connection information for the user to connect to the EC2
instance.
        Handles the case where the instance does or does not have an associated
public IP address.
        """
        if (
            not self.eip_wrapper.elastic_ips
            or not self.eip_wrapper.elastic_ips[0].allocation_id
        ):
            if self.inst_wrapper.instances:
                instance = self.inst_wrapper.instances[0]
                instance_id = instance["InstanceId"]

                waiter =
self.inst_wrapper.ec2_client.get_waiter("instance_running")
                console.print(
                    "Waiting for the instance to be in a running state with a
public IP...",
                    style="bold cyan",
                )

                with alive_bar(1, title="Waiting for Instance to Start") as bar:
                    waiter.wait(InstanceIds=[instance_id])
                    time.sleep(20)
                    bar()

                instance = self.inst_wrapper.ec2_client.describe_instances(
                    InstanceIds=[instance_id]
                )["Reservations"][0]["Instances"][0]
```

```
                public_ip = instance.get("PublicIpAddress")
                if public_ip:
                    console.print(
                        "\nTo connect via SSH, open another command prompt and
 run the following command:",
                        style="bold cyan",
                    )
                    console.print(
                        f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
                    )
                else:
                    console.print(
                        "Instance does not have a public IP address assigned.",
                        style="bold red",
                    )
            else:
                console.print(
                    "No instance available to retrieve public IP address.",
                    style="bold red",
                )
        else:
            elastic_ip = self.eip_wrapper.elastic_ips[0]
            elastic_ip_address = elastic_ip.public_ip
            console.print(
                f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
            )

        if not self.remote_exec:
            console.print("\nOpen a new terminal tab to try the above SSH
 command.")
            input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
 cyan")
        console.print(
```

```
            "You can allocate an Elastic IP address and associate it with your
 instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
 IP."
        )
        self._display_ssh_info()

    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains
        changes that occur when the instance is restarted, such as the potential
 change
        in the public IP address unless an Elastic IP is associated.
        """
        console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
 cyan")
        console.print("Let's stop and start your instance to see what changes.")
        console.print("- **Stopping your instance and waiting until it's
 stopped...**")

        with alive_bar(1, title="Stopping Instance") as bar:
            self.inst_wrapper.stop()
```

```
            time.sleep(360)
            bar()

        console.print("- **Your instance is stopped. Restarting...**")

        with alive_bar(1, title="Starting Instance") as bar:
            self.inst_wrapper.start()
            time.sleep(20)
            bar()

        console.print("**Your instance is running.**", style="bold green")
        self.inst_wrapper.display()

        elastic_ip = (
            self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
        )

        if elastic_ip is None or elastic_ip.allocation_id is None:
            console.print(
                "- **Note**: Every time your instance is restarted, its public IP
address changes."
            )
        else:
            console.print(
                f"Because you have associated an Elastic IP with your instance,
you can \n"
                f"connect by using a consistent IP address after the instance
restarts: {elastic_ip.public_ip}"
            )

        self._display_ssh_info()

    def cleanup(self) -> None:
        """
        Cleans up all the resources created during the scenario, including
disassociating
        and releasing the Elastic IP, terminating the instance, deleting the
security
        group, and deleting the key pair.
        """
        console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
        console.print("Cleaning up resources:")
```

```
        for elastic_ip in self.eip_wrapper.elastic_ips:
            console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

            with alive_bar(1, title="Disassociating Elastic IP") as bar:
                self.eip_wrapper.disassociate(elastic_ip.allocation_id)
                time.sleep(2)
                bar()

            console.print("\t- **Disassociated Elastic IP from the Instance**")

            with alive_bar(1, title="Releasing Elastic IP") as bar:
                self.eip_wrapper.release(elastic_ip.allocation_id)
                time.sleep(1)
                bar()

            console.print("\t- **Released Elastic IP**")

        console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

        with alive_bar(1, title="Terminating Instance") as bar:
            self.inst_wrapper.terminate()
            time.sleep(380)
            bar()

        console.print("\t- **Terminated Instance**")

        console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

        with alive_bar(1, title="Deleting Security Group") as bar:
            self.sg_wrapper.delete(self.sg_wrapper.security_group)
            time.sleep(1)
            bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")
```

```
    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
 groups,
        launches an instance, associates an Elastic IP, and cleans up all
 resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
 %(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
 with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = EC2InstanceScenario(
            EC2InstanceWrapper.from_client(),
            KeyPairWrapper.from_client(),
            SecurityGroupWrapper.from_client(),
            ElasticIpWrapper.from_client(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Schlüsselpaar-Aktionen umschließt.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
 client
        and a temporary directory for storing key files.
```

```
        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.
```

```
        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
```

```
                    f"The key pair '{key_name}' does not exist and cannot be
 deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

Definieren Sie eine Klasse, die Sicherheitsgruppen-Aktionen umschließt.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
a different name."
                )
            raise
        else:
            return self.security_group


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
```

```
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
```

```
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete.
Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an
AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
```

```
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
 because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
 because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

Definieren Sie eine Klasse, die Instance-Aktionen umschließt.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
```

```
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group
of the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
```

```
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Select a different instance type or launch in a
 different availability zone."
                    )
                )
            raise
        return self.instances


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.
```

```
        :param state_filter: The instance state to include in the output. Only
 instances in this state
                             will be displayed. Default is 'running'. Example
 states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match
 the filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
```

```
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
```

```
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str,
instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
```

```
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
"*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
```

```
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types.
 Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support
 the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
 {','.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
 DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

Definieren Sie eine Klasse, die Elastic-IP-Aktionen umschließt.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
```

```
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
```

```
        :return: A response that contains the ID of the association, or None if
no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic
IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
When the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
```

```
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:

self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP
address was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
```

```
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise


    @staticmethod
    def get_elastic_ip_by_allocation(
        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None
if not found.
        """
        return next(
```

```
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die EC2InstanceScenario Implementierung enthält Logik, um das Beispiel als Ganzes auszuführen.

```
//! Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
 Cloud
//! (Amazon EC2) to do the following:
//!
//! * Create a key pair that is used to secure SSH communication between your
 computer and
//!   an EC2 instance.
//! * Create a security group that acts as a virtual firewall for your EC2
 instances to
//!   control incoming and outgoing traffic.
//! * Find an Amazon Machine Image (AMI) and a compatible instance type.
//! * Create an instance that is created from the instance type and AMI you
 select, and
//!   is configured to use the security group and key pair created in this
 example.
//! * Stop and restart the instance.
//! * Create an Elastic IP address and associate it as a consistent IP address
 for your instance.
//! * Connect to your instance with SSH, using both its public IP address and
 your Elastic IP
//!   address.
//! * Clean up all of the resources created by this example.

use std::net::Ipv4Addr;

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
```

```
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
 security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }

    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
        self.stop_and_start_instance().await?;
        Ok(())
    }
```

```
    /// 1. Creates an RSA key pair and saves its private key data as a .pem file
in secure
    ///    temporary storage. The private key data is deleted after the example
completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;

        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
                );
            }
        }

        Ok(())
    }

    /// 1. Creates a security group for the default VPC.
    /// 2. Adds an inbound rule to allow SSH. The SSH rule allows only
    ///    inbound traffic from the current computer’s public IPv4 address.
    /// 3. Displays information about the security group.
```

```
    ///
    /// This function uses <http://checkip.amazonaws.com> to get the current
public IP
    /// address of the computer that is running the example. This method works in
most
    /// cases. However, depending on how your computer connects to the internet,
you
    /// might have to manually add your public IP address to the security group
by using
    /// the AWS Management Console.
    pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
        println!("Let's create a security group to manage access to your
instance.");
        let group_name = self.util.prompt_security_group_name()?;

        self.security_group_manager
            .create(
                &self.ec2,
                &group_name,
                "Security group for example: get started with instances.",
            )
            .await?;

        println!(
            "Created security group {} in your default VPC {}.",
            self.security_group_manager.group_name(),
            self.security_group_manager
                .vpc_id()
                .unwrap_or("(unknown vpc)")
        );

        let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
        let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
            EC2Error::new(format!(
                "Failed to convert response {} to IP Address: {e:?}",
                check_ip
            ))
        })?;

        println!("Your public IP address seems to be {current_ip_address}");
        if self.util.should_add_to_security_group() {
            match self
                .security_group_manager
                .authorize_ingress(&self.ec2, current_ip_address)
```

```
                .await
            {
                Ok(_) => println!("Security group rules updated"),
                Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
            }
        }
        println!("{}", self.security_group_manager);

        Ok(())
    }

    /// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager.
Specifying the
    ///    '/aws/service/ami-amazon-linux-latest' path returns only the latest
AMIs.
    /// 2. Gets and displays information about the available AMIs and lets you
select one.
    /// 3. Gets a list of instance types that are compatible with the selected
AMI and
    ///    lets you select one.
    /// 4. Creates an instance with the previously created key pair and security
group,
    ///    and the selected AMI and instance type.
    /// 5. Waits for the instance to be running and then displays its
information.
    pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
        let ami = self.find_image().await?;

        let instance_types = self
            .ec2
            .list_instance_types(&ami.0)
            .await
            .map_err(|e| e.add_message("Could not find instance types"))?;
        println!(
            "There are several instance types that support the {} architecture of
the image.",
            ami.0
                .architecture
                .as_ref()
                .ok_or_else(|| EC2Error::new(format!("Missing architecture in
{:?}", ami.0)))?
        );
        let instance_type = self.util.select_instance_type(instance_types)?;
```

```
        println!("Creating your instance and waiting for it to start...");
        self.instance_manager
            .create(
                &self.ec2,
                ami.0
                    .image_id()
                    .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
                instance_type,
                self.key_pair_manager.key_pair(),
                self.security_group_manager
                    .security_group()
                    .map(|sg| vec![sg])
                    .ok_or_else(|| EC2Error::new("Could not find security
group"))?,
            )
            .await
            .map_err(|e| e.add_message("Scenario failed to create instance"))?;

        while let Err(err) = self
            .ec2
            .wait_for_instance_ready(self.instance_manager.instance_id(), None)
            .await
        {
            println!("{err}");
            if !self.util.should_continue_waiting() {
                return Err(err);
            }
        }

        println!("Your instance is ready:\n{}", self.instance_manager);

        self.display_ssh_info();

        Ok(())
    }

    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
```

```
            .into_iter()
            .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }

    // 1. Stops the instance and waits for it to stop.
    // 2. Starts the instance and waits for it to start.
    // 3. Displays information about the instance.
    // 4. Displays an SSH connection string. When an Elastic IP address is
associated
    //    with the instance, the IP address stays consistent when the instance
stops
    //    and starts.
    pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
        println!("Let's stop and start your instance to see what changes.");
        println!("Stopping your instance and waiting until it's stopped...");
        self.instance_manager.stop(&self.ec2).await?;
        println!("Your instance is stopped. Restarting...");
        self.instance_manager.start(&self.ec2).await?;
        println!("Your instance is running.");
        println!("{}", self.instance_manager);
        if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
            println!("Every time your instance is restarted, its public IP
address changes.");
        } else {
            println!(
                "Because you have associated an Elastic IP with your instance,
you can connect by using a consistent IP address after the instance restarts."
            );
        }
        self.display_ssh_info();
        Ok(())
```

```
    }

    /// 1. Allocates an Elastic IP address and associates it with the instance.
    /// 2. Displays an SSH connection string that uses the Elastic IP address.
    async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
        self.elastic_ip_manager.allocate(&self.ec2).await?;
        println!(
            "Allocated static Elastic IP address: {}",
            self.elastic_ip_manager.public_ip()
        );

        self.elastic_ip_manager
            .associate(&self.ec2, self.instance_manager.instance_id())
            .await?;
        println!("Associated your Elastic IP with your instance.");
        println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
        self.display_ssh_info();
        Ok(())
    }

    /// Displays an SSH connection string that can be used to connect to a
running
    /// instance.
    fn display_ssh_info(&self) {
        let ip_addr = if self.elastic_ip_manager.has_allocation() {
            self.elastic_ip_manager.public_ip()
        } else {
            self.instance_manager.instance_ip()
        };
        let key_file_path = self.key_pair_manager.key_file_path().unwrap();
        println!("To connect, open another command prompt and run the following
command:");
        println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
        let _ = self.util.enter_to_continue();
    }

    /// 1. Disassociate and delete the previously created Elastic IP.
    /// 2. Terminate the previously created instance.
    /// 3. Delete the previously created security group.
    /// 4. Delete the previously created key pair.
    pub async fn clean_up(self) {
        println!("Let's clean everything up. This example created these
resources:");
```

```
        println!(
            "\tKey pair: {}",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .unwrap_or("(unknown key pair)")
        );
        println!(
            "\tSecurity group: {}",
            self.security_group_manager.group_name()
        );
        println!(
            "\tInstance: {}",
            self.instance_manager.instance_display_name()
        );
        if self.util.should_clean_resources() {
            if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.instance_manager.delete(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.security_group_manager.delete(&self.ec2).await
 {
                eprintln!("{err}");
            }
            if let Err(err) = self.key_pair_manager.delete(&self.ec2,
 &self.util).await {
                eprintln!("{err}");
            }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
("--------------------------------------------------------------------------------");
    println!(
        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
 with instances demo."
    );
```

```
    println!
("--------------------------------------------------------------------------------");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
("--------------------------------------------------------------------------------");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
("--------------------------------------------------------------------------------");
}
```

Die EC2Impl-Struktur dient als Automock-Punkt für Tests und ihre Funktionen umschließen die EC2-SDK-Aufrufe.

```
use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
 KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;

#[cfg(test)]
use mockall::automock;
```

```
#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
 String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key
 material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
            .delete_key_pair()
```

```
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }

    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }

    /// Find a single security group, by ID. Returns Err if multiple groups are
found.
    pub async fn describe_security_group(
        &self,
        group_id: &str,
```

```
    ) -> Result<Option<SecurityGroup>, EC2Error> {
        let group_id: String = group_id.into();
        let describe_output = self
            .client
            .describe_security_groups()
            .group_ids(&group_id)
            .send()
            .await?;

        let mut groups = describe_output.security_groups.unwrap_or_default();

        match groups.len() {
            0 => Ok(None),
            1 => Ok(Some(groups.remove(0))),
            _ => Err(EC2Error::new(format!(
                "Expected single group for {group_id}"
            ))),
        }
    }

    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
```

```
            ))
            .send()
            .await?;
        Ok(())
    }

    pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting security group {group_id}");
        self.client
            .delete_security_group()
            .group_id(group_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
        let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
        let output = self
            .client
            .describe_images()
            .set_image_ids(Some(image_ids))
            .send()
            .await?;

        let images = output.images.unwrap_or_default();
        if images.is_empty() {
            Err(EC2Error::new("No images for selected AMIs"))
        } else {
            Ok(images)
        }
    }

    /// List instance types that match an image's architecture and are free tier
eligible.
    pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
        let architecture = format!(
            "{}",
            image.architecture().ok_or_else(|| EC2Error::new(format!(
                "Image {:?} does not have a listed architecture",
                image.image_id()
```

```
            )))?
        );
        let free_tier_eligible_filter = Filter::builder()
            .name("free-tier-eligible")
            .values("false")
            .build();
        let supported_architecture_filter = Filter::builder()
            .name("processor-info.supported-architecture")
            .values(architecture)
            .build();
        let response = self
            .client
            .describe_instance_types()
            .filters(free_tier_eligible_filter)
            .filters(supported_architecture_filter)
            .send()
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
            .collect())
    }

    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
```

```
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }

    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
```

```
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }
```

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }

    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }
```

```
    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }

    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
```

```
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
        self.client
            .allocate_address()
            .domain(DomainType::Vpc)
            .send()
            .await
            .map_err(EC2Error::from)
    }

    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn associate_ip_address(
        &self,
        allocation_id: &str,
        instance_id: &str,
    ) -> Result<AssociateAddressOutput, EC2Error> {
        let response = self
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }
```

```
    pub async fn disassociate_ip_address(&self, association_id: &str) ->
 Result<(), EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for EC2Error {}

impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
```

```
    }
}
```

Die SSM-Struktur dient als Automock-Punkt für Tests und ihre Funktionen umschließen die SSM-SDK-Aufrufe.

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
 {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
```

```
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
}
```

Das Szenario verwendet mehrere „Manager“-Strukturen, um den Zugriff auf Ressourcen zu verwalten, die im Verlauf des Szenarios erstellt und gelöscht werden.

```
use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}

impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
        "0.0.0.0"
    }
```

```
    pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
        let allocation = ec2.allocate_ip_address().await?;
        self.elastic_ip = Some(allocation);
        Ok(())
    }

    pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
 EC2Error> {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                let association = ec2.associate_ip_address(allocation_id,
 instance_id).await?;
                self.association = Some(association);
                return Ok(());
            }
        }
        Err(EC2Error::new("No ip address allocation to associate"))
    }

    pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(association) = &self.association {
            if let Some(association_id) = association.association_id() {
                ec2.disassociate_ip_address(association_id).await?;
            }
        }
        self.association = None;
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                ec2.deallocate_ip_address(allocation_id).await?;
            }
        }
        self.elastic_ip = None;
        Ok(())
    }
}


use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};

use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
```

```
#[derive(Debug, Default)]
pub struct InstanceManager {
    instance: Option<Instance>,
}

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
 Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{} ({})", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
```

```
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }

    /// Start the managed EC2 instance, if present.
    pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.start_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    /// Stop the managed EC2 instance, if present.
    pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.stop_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }

    /// Terminate and delete the managed EC2 instance, if present.
    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
```

```
            ec2.delete_instance(self.instance_id()).await?;
        }
        Ok(())
    }
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}",
 instance.instance_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            )?;
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            )?;
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            )?;
            writeln!(f, "\tVPC ID: {}",
 instance.vpc_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            )?;
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                        .map(|isn| format!("{isn}"))
                        .unwrap_or("(Unknown)".to_string())
                })
```

```
                .unwrap_or("(Unknown)".to_string());
            writeln!(f, "\tState: {instance_state}")?;
        } else {
            writeln!(f, "\tNo loaded instance")?;
        }
        Ok(())
    }
}


use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }

    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
        &self.key_file_dir
    }
```

```
   /// Creates a key pair that can be used to securely connect to an EC2
instance.
   /// The returned key pair contains private key information that cannot be
retrieved
   /// again. The private key data is stored as a .pem file.
   ///
   /// :param key_name: The name of the key pair to create.
   pub async fn create(
       &mut self,
       ec2: &EC2,
       util: &Util,
       key_name: String,
   ) -> Result<KeyPairInfo, EC2Error> {
       let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
           self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
           e.add_message(format!("Couldn't create key {key_name}"))
       })?;

       let path = self.key_file_dir.join(format!("{key_name}.pem"));

       // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
       self.key_file_path = Some(path.clone());
       self.key_pair = key_pair.clone();

       util.write_secure(&key_name, &path, material)?;

       Ok(key_pair)
   }

   pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
       if let Some(key_name) = self.key_pair.key_name() {
           ec2.delete_key_pair(key_name).await?;
           if let Some(key_path) = self.key_file_path() {
               if let Err(err) = util.remove(key_path) {
                   eprintln!("Failed to remove {key_path:?} ({err:?})");
               }
           }
       }
       Ok(())
   }
```

```
    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }
}

impl Default for KeyPairManager {
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}


use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
```

```
                .map_err(|e| e.add_message("Couldn't create security group"))?,
        );

        Ok(())
    }

    pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
 Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.authorize_security_group_ssh_ingress(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
                vec![ip_address],
            )
            .await?;
        };

        Ok(())
    }

    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.delete_security_group(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            )
            .await?;
        };

        Ok(())
    }

    pub fn group_name(&self) -> &str {
        &self.group_name
    }

    pub fn vpc_id(&self) -> Option<&str> {
        self.security_group.as_ref().and_then(|sg| sg.vpc_id())
    }

    pub fn security_group(&self) -> Option<&SecurityGroup> {
        self.security_group.as_ref()
    }
}
```

```
impl std::fmt::Display for SecurityGroupManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.security_group {
            Some(sg) => {
                writeln!(
                    f,
                    "Security group: {}",
                    sg.group_name().unwrap_or("(unknown group)")
                )?;
                writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
 id)"))?;
                writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
 vpc)"))?;
                if !sg.ip_permissions().is_empty() {
                    writeln!(f, "\tInbound Permissions:")?;
                    for permission in sg.ip_permissions() {
                        writeln!(f, "\t\t{permission:?}")?;
                    }
                }
                Ok(())
            }
            None => writeln!(f, "No security group loaded."),
        }
    }
}
```

Der Haupteinstiegspunkt für das Szenario.

```
use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
```

```
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
    let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
    let util = UtilImpl {};
    let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
    run(scenario).await;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "ec2-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "ec2-scenario",
```

```
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "AWSSSM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Die Datei `entry.swift`

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation
import AWSEC2

// Allow waiters to be used.

import class SmithyWaitersAPI.Waiter
import struct SmithyWaitersAPI.WaiterOptions

import AWSSSM

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
```

```
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "ec2-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon EC2 using
 the
        AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ssmConfig = try await SSMClient.SSMClientConfiguration(region:
 awsRegion)
        let ssmClient = SSMClient(config: ssmConfig)

        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let example = Example(ec2Client: ec2Client, ssmClient: ssmClient)

        await example.run()
    }
}

class Example {
    let ec2Client: EC2Client
    let ssmClient: SSMClient

    // Storage for AWS EC2 properties.

    var keyName: String? = nil
    var securityGroupId: String? = nil
    var instanceId: String? = nil
    var allocationId: String? = nil
    var associationId: String? = nil

    init(ec2Client: EC2Client, ssmClient: SSMClient) {
```

```
        self.ec2Client = ec2Client
        self.ssmClient = ssmClient
    }

    /// The example's main body.
    func run() async {
        //======================================================================
        // 1. Create an RSA key pair, saving the private key as a `.pem` file.
        //    Create a `defer` block that will delete the private key when the
        //    program exits.
        //======================================================================

        print("Creating an RSA key pair...")

        keyName = self.tempName(prefix: "ExampleKeyName")
        let keyUrl = await self.createKeyPair(name: keyName!)

        guard let keyUrl else {
            print("*** Failed to create the key pair!")
            return
        }

        print("Created the private key at: \(keyUrl.absoluteString)")

        // Schedule deleting the private key file to occur automatically when
        // the program exits, no matter how it exits.

        defer {
            do {
                try FileManager.default.removeItem(at: keyUrl)
            } catch {
                print("*** Failed to delete the private key at
\(keyUrl.absoluteString)")
            }
        }

        //======================================================================
        // 2. List the key pairs by calling `DescribeKeyPairs`.
        //======================================================================

        print("Describing available key pairs...")
        await self.describeKeyPairs()

        //======================================================================
```

```
        // 3. Create a security group for the default VPC, and add an inbound
        //    rule to allow SSH from the current computer's public IPv4
        //    address.
        //======================================================================

        print("Creating the security group...")

        let secGroupName = self.tempName(prefix: "ExampleSecurityGroup")
        let ipAddress = self.getMyIPAddress()

        guard let ipAddress else {
            print("*** Unable to get the device's IP address.")
            return
        }

        print("IP address is: \(ipAddress)")

        securityGroupId = await self.createSecurityGroup(
            name: secGroupName,
            description: "An example security group created using the AWS SDK for
Swift"
        )

        if securityGroupId == nil {
            await cleanUp()
            return
        }

        print("Created security group: \(securityGroupId ?? "<unknown>")")

        if !(await self.authorizeSecurityGroupIngress(groupId: securityGroupId!,
ipAddress: ipAddress)) {
            await cleanUp()
            return
        }

        //======================================================================
        // 4. Display security group information for the new security group
        //    using DescribeSecurityGroups.
        //======================================================================

        if !(await self.describeSecurityGroups(groupId: securityGroupId!)) {
            await cleanUp()
            return
```

```
        }

        //=====================================================================
        // 5. Get a list of Amazon Linux 2023 AMIs and pick one (SSM is the
        //    best practice), using path and then filter the list after the
        //    fact to include "al2023" in the Name field
        //    (ssm.GetParametersByPath). Paginate to get all images.
        //=====================================================================

        print("Searching available images for Amazon Linux 2023 images...")

        let options = await self.findAMIsMatchingFilter("al2023")

        //=====================================================================
        // 6. The information in the AMI options isn't great, so make a list
        //    of the image IDs (the "Value" field in the AMI options) and get
        //    more information about them from EC2. Display the Description
        //    field and select one of them (DescribeImages with ImageIds
        //    filter).
        //=====================================================================

        print("Images matching Amazon Linux 2023:")

        var imageIds: [String] = []
        for option in options {
            guard let id = option.value else {
                continue
            }
            imageIds.append(id)
        }

        let images = await self.describeImages(imageIds)

        // This is where you would normally let the user choose which AMI to
        // use. However, for this example, we're just going to use the first
        // one, whatever it is.

        let chosenImage = images[0]

        //=====================================================================
        // 7. Get a list of instance types that are compatible with the
        //    selected AMI's architecture (such as "x86_64") and are either
        //    small or micro. Select one (DescribeInstanceTypes).
        //=====================================================================
```

```
        print("Getting the instance types compatible with the selected image...")

        guard let arch = chosenImage.architecture else {
            print("*** The selected image doesn't have a valid architecture.")
            await cleanUp()
            return
        }

        let imageTypes = await self.getMatchingInstanceTypes(architecture: arch)

        for type in imageTypes {
            guard let instanceType = type.instanceType else {
                continue
            }
            print("    \(instanceType.rawValue)")
        }

        // This example selects the first returned instance type. A real-world
        // application would probably ask the user to select one here.

        let chosenInstanceType = imageTypes[0]

        //=====================================================================
        // 8. Create an instance with the key pair, security group, AMI, and
        //    instance type (RunInstances).
        //=====================================================================

        print("Creating an instance...")

        guard let imageId = chosenImage.imageId else {
            print("*** Cannot start image without a valid image ID.")
            await cleanUp()
            return
        }
        guard let instanceType = chosenInstanceType.instanceType else {
            print("*** Unable to start image without a valid image type.")
            await cleanUp()
            return
        }

        let instance = await self.runInstance(
            imageId: imageId,
            instanceType: instanceType,
```

```
            keyPairName: keyName!,
            securityGroups: [securityGroupId!]
        )

        guard let instance else {
            await cleanUp()
            return
        }

        instanceId = instance.instanceId
        if instanceId == nil {
            print("*** Instance is missing an ID. Canceling.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 9. Wait for the instance to be ready and then display its
        //    information (DescribeInstances).
        //=====================================================================

        print("Waiting a few seconds to let the instance come up...")

        do {
            try await Task.sleep(for: .seconds(20))
        } catch {
            print("*** Error pausing the task.")
        }
        print("Success! Your new instance is ready:")

        //=====================================================================
        // 10. Display SSH connection info for the instance.
        //=====================================================================

        var runningInstance = await self.describeInstance(instanceId:
 instanceId!)

        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command:")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }
```

```
        //=====================================================================
        // 11. Stop the instance and wait for it to stop (StopInstances).
        //=====================================================================

        print("Stopping the instance...")

        if !(await self.stopInstance(instanceId: instanceId!, waitUntilStopped:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Start the instance and wait for it to start (StartInstances).
        //=====================================================================

        print("Starting the instance again...")

        if !(await self.startInstance(instanceId: instanceId!, waitUntilStarted:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 13. Display SSH connection info for the instance. Note that it's
        //     changed.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command.")
            print("This is probably different from when the instance was running
 before.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 14. Allocate an elastic IP and associate it with the instance
        //     (AllocateAddress and AssociateAddress).
        //=====================================================================
```

```
        allocationId = await self.allocateAddress()

        if allocationId == nil {
            await cleanUp()
            return
        }

        associationId = await self.associateAddress(instanceId: instanceId!,
 allocationId: allocationId)

        if associationId == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 15. Display SSH connection info for the connection. Note that the
        //     public IP is now the Elastic IP, which stays constant.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
 {
            print("\nYou can SSH to this instance using the following command.")
            print("This has changed again, and is now the Elastic IP.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // Handle all cleanup tasks
        //=====================================================================

        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items:
    ///
    /// * Elastic IP allocation and association
    /// * Terminate the instance
    /// * Delete the security group
    /// * Delete the key pair
    func cleanUp() async {
        //=====================================================================
```

```
        // 16. Disassociate and delete the Elastic IP (DisassociateAddress and
        //     ReleaseAddress).
        //===========================================================================

        if associationId != nil {
            await self.disassociateAddress(associationId: associationId!)
        }

        if allocationId != nil {
            await self.releaseAddress(allocationId: allocationId!)
        }

        //===========================================================================
        // 17. Terminate the instance and wait for it to terminate
        //     (TerminateInstances).
        //===========================================================================

        if instanceId != nil {
            print("Terminating the instance...")
            _ = await self.terminateInstance(instanceId: instanceId!,
waitUntilTerminated: true)
        }

        //===========================================================================
        // 18. Delete the security group (DeleteSecurityGroup).
        //===========================================================================

        if securityGroupId != nil {
            print("Deleting the security group...")
            _ = await self.deleteSecurityGroup(groupId: securityGroupId!)
        }

        //===========================================================================
        // 19. Delete the key pair (DeleteKeyPair).
        //===========================================================================

        if keyName != nil {
            print("Deleting the key pair...")
            _ = await self.deleteKeyPair(keyPair: keyName!)
        }
    }

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
```

```
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
 String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )
```

```
            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":",
keyPair.keyFingerprint ?? "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair:
\(error.localizedDescription)")
            return false
        }
    }

    /// Return a list of AMI names that contain the specified string.
    ///
    /// - Parameter filter: A string that must be contained in all returned
    ///   AMI names.
    ///
    /// - Returns: An array of the parameters matching the specified substring.
```

```
    func findAMIsMatchingFilter(_ filter: String) async ->
[SSMClientTypes.Parameter] {
        var parameterList: [SSMClientTypes.Parameter] = []
        var matchingAMIs: [SSMClientTypes.Parameter] = []

        do {
            let pages = ssmClient.getParametersByPathPaginated(
                input: GetParametersByPathInput(
                    path: "/aws/service/ami-amazon-linux-latest"
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return matchingAMIs
                }

                for parameter in parameters {
                    parameterList.append(parameter)
                }
            }

            print("Found \(parameterList.count) images total:")
            for parameter in parameterList {
                guard let name = parameter.name else {
                    continue
                }
                print("    \(name)")

                if name.contains(filter) {
                    matchingAMIs.append(parameter)
                }
            }
        } catch {
            return matchingAMIs
        }

        return matchingAMIs
    }

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
```

```
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
 = EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

```
    /// Get the latest information about the specified instance and output it
    /// to the screen, returning the instance details to the caller.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to provide details about.
    ///   - stateFilter: The state to require the instance to be in.
    ///
    /// - Returns: The instance's details as an `EC2ClientTypes.Instance` object.
    func describeInstance(instanceId: String,
                          stateFilter: EC2ClientTypes.InstanceStateName? =
EC2ClientTypes.InstanceStateName.running) async
                          -> EC2ClientTypes.Instance? {
        do {
            let pages = ec2Client.describeInstancesPaginated(
                input: DescribeInstancesInput(
                    instanceIds: [instanceId]
                )
            )

            for try await page in pages {
                guard let reservations = page.reservations else {
                    continue
                }

                for reservation in reservations {
                    guard let instances = reservation.instances else {
                        continue
                    }

                    for instance in instances {
                        guard let state = instance.state else {
                            print("*** Instance is missing its state...")
                            continue
                        }
                        let instanceState = state.name

                        if stateFilter != nil && (instanceState != stateFilter) {
                            continue
                        }

                        let instanceTypeName: String
                        if instance.instanceType == nil {
                            instanceTypeName = "<N/A>"
                        } else {
```

```
                            instanceTypeName = instance.instanceType?.rawValue ??
 "<N/A>"
                        }

                        let instanceStateName: String
                        if instanceState == nil {
                            instanceStateName = "<N/A>"
                        } else {
                            instanceStateName = instanceState?.rawValue ?? "<N/
A>"
                        }

                        print("""
                        Instance: \(instance.instanceId ?? "<N/A>")
                                • Image ID: \(instance.imageId ?? "<N/A>")
                                • Instance type: \(instanceTypeName)
                                • Key name: \(instance.keyName ?? "<N/A>")
                                • VPC ID: \(instance.vpcId ?? "<N/A>")
                                • Public IP: \(instance.publicIpAddress ?? "N/A")
                                • State: \(instanceStateName)
                        """)

                        return instance
                    }
                }
            }
        } catch {
            print("*** Error retrieving instance information to display:
 \(error.localizedDescription)")
            return nil
        }

        return nil
    }

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
```

```
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
 \(error.localizedDescription)")
            return false
        }
    }

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
```

```
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
 -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance:
 \(error.localizedDescription)")
            return false
        }
    }

    /// Terminate the specified instance.
    ///
```

```
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
 \(error.localizedDescription)")
            return false
        }
    }
```

```
    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
```

```
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Return the device's external IP address.
    ///
    /// - Returns: A string containing the device's IP address.
    func getMyIPAddress() -> String? {
        guard let url = URL(string: "http://checkip.amazonaws.com") else {
            print("Couldn't create the URL")
            return nil
        }

        do {
            print("Getting the IP address...")
            return try String(contentsOf: url, encoding:
String.Encoding.utf8).trim()
        } catch {
```

```
            print("*** Unable to get your public IP address.")
            return nil
        }
    }

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description
groupDescription: String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
-> Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
```

```
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
\(group.vpcId ?? "<unknown>")")
            }
            return true
```

```
        } catch {
            print("*** Error getting security group details:
\(error.localizedDescription)")
            return false
        }
    }

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
            return false
        }
    }

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
```

```
        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
```

```
            print("*** Unable to disassociate the IP address:
 \(error.localizedDescription)")
        }
    }

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
 \(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Aktionen für Amazon EC2 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon EC2 EC2-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-EC2-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon EC2 AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Elastic Compute Cloud.

Beispiele

- Verwendung von AcceptVpcPeeringConnection mit einer CLI
- Verwenden Sie mit einem AllocateAddress AWS SDK oder CLI
- Verwendung von AllocateHosts mit einer CLI
- Verwendung von AssignPrivateIpAddresses mit einer CLI
- Verwenden Sie mit einem AssociateAddress AWS SDK oder CLI
- Verwendung von AssociateDhcpOptions mit einer CLI
- Verwendung von AssociateRouteTable mit einer CLI
- Verwendung von AttachInternetGateway mit einer CLI
- Verwendung von AttachNetworkInterface mit einer CLI
- Verwendung von AttachVolume mit einer CLI
- Verwendung von AttachVpnGateway mit einer CLI
- Verwendung von AuthorizeSecurityGroupEgress mit einer CLI
- Verwenden Sie mit einem AuthorizeSecurityGroupIngress AWS SDK oder CLI
- Verwendung von CancelCapacityReservation mit einer CLI
- Verwendung von CancelImportTask mit einer CLI
- Verwendung von CancelSpotFleetRequests mit einer CLI
- Verwendung von CancelSpotInstanceRequests mit einer CLI
- Verwendung von ConfirmProductInstance mit einer CLI
- Verwendung von CopyImage mit einer CLI
- Verwendung von CopySnapshot mit einer CLI

- Verwendung von CreateCapacityReservation mit einer CLI
- Verwendung von CreateCustomerGateway mit einer CLI
- Verwendung von CreateDhcpOptions mit einer CLI
- Verwendung von CreateFlowLogs mit einer CLI
- Verwendung von CreateImage mit einer CLI
- Verwendung von CreateInstanceExportTask mit einer CLI
- Verwendung von CreateInternetGateway mit einer CLI
- Verwenden Sie mit einem CreateKeyPair AWS SDK oder CLI
- Verwenden Sie CreateLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von CreateNetworkAcl mit einer CLI
- Verwendung von CreateNetworkAclEntry mit einer CLI
- Verwendung von CreateNetworkInterface mit einer CLI
- Verwendung von CreatePlacementGroup mit einer CLI
- Verwendung von CreateRoute mit einer CLI
- Verwenden Sie mit einem CreateRouteTable AWS SDK oder CLI
- Verwenden Sie CreateSecurityGroupmit einem AWS SDK oder CLI
- Verwendung von CreateSnapshot mit einer CLI
- Verwendung von CreateSpotDatafeedSubscription mit einer CLI
- Verwenden Sie mit einem CreateSubnet AWS SDK oder CLI
- Verwenden Sie CreateTagsmit einem AWS SDK oder CLI
- Verwendung von CreateVolume mit einer CLI
- Verwenden Sie mit einem CreateVpc AWS SDK oder CLI
- Verwenden Sie CreateVpcEndpointmit einem AWS SDK oder CLI
- Verwendung von CreateVpnConnection mit einer CLI
- Verwendung von CreateVpnConnectionRoute mit einer CLI
- Verwendung von CreateVpnGateway mit einer CLI
- Verwendung von DeleteCustomerGateway mit einer CLI
- Verwendung von DeleteDhcpOptions mit einer CLI
- Verwendung von DeleteFlowLogs mit einer CLI
- Verwendung von DeleteInternetGateway mit einer CLI

- Verwenden Sie mit einem DeleteKeyPair AWS SDK oder CLI
- Verwenden Sie DeleteLaunchTemplatemit einem AWS SDK oder CLI
- Verwendung von DeleteNetworkAcl mit einer CLI
- Verwendung von DeleteNetworkAclEntry mit einer CLI
- Verwendung von DeleteNetworkInterface mit einer CLI
- Verwendung von DeletePlacementGroup mit einer CLI
- Verwendung von DeleteRoute mit einer CLI
- Verwendung von DeleteRouteTable mit einer CLI
- Verwenden Sie mit einem DeleteSecurityGroup AWS SDK oder CLI
- Verwenden Sie DeleteSnapshotmit einem AWS SDK oder CLI
- Verwendung von DeleteSpotDatafeedSubscription mit einer CLI
- Verwendung von DeleteSubnet mit einer CLI
- Verwendung von DeleteTags mit einer CLI
- Verwendung von DeleteVolume mit einer CLI
- Verwenden Sie mit einem DeleteVpc AWS SDK oder CLI
- Verwenden Sie DeleteVpcEndpointsmit einem AWS SDK oder CLI
- Verwendung von DeleteVpnConnection mit einer CLI
- Verwendung von DeleteVpnConnectionRoute mit einer CLI
- Verwendung von DeleteVpnGateway mit einer CLI
- Verwendung von DeregisterImage mit einer CLI
- Verwendung von DescribeAccountAttributes mit einer CLI
- Verwenden Sie mit einem DescribeAddresses AWS SDK oder CLI
- Verwenden Sie DescribeAvailabilityZonesmit einem AWS SDK oder CLI
- Verwendung von DescribeBundleTasks mit einer CLI
- Verwendung von DescribeCapacityReservations mit einer CLI
- Verwendung von DescribeCustomerGateways mit einer CLI
- Verwendung von DescribeDhcpOptions mit einer CLI
- Verwendung von DescribeFlowLogs mit einer CLI
- Verwendung von DescribeHostReservationOfferings mit einer CLI
- Verwendung von DescribeHosts mit einer CLI

- Verwenden Sie mit einem DescribeIamInstanceProfileAssociations AWS SDK oder CLI
- Verwendung von DescribeIdFormat mit einer CLI
- Verwendung von DescribeIdentityIdFormat mit einer CLI
- Verwendung von DescribeImageAttribute mit einer CLI
- Verwenden Sie mit einem DescribeImages AWS SDK oder CLI
- Verwendung von DescribeImportImageTasks mit einer CLI
- Verwendung von DescribeImportSnapshotTasks mit einer CLI
- Verwendung von DescribeInstanceAttribute mit einer CLI
- Verwenden Sie mit einem DescribeInstanceStatus AWS SDK oder CLI
- Verwenden Sie DescribeInstanceTypesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeInstancesmit einem AWS SDK oder CLI
- Verwendung von DescribeInternetGateways mit einer CLI
- Verwenden Sie mit einem DescribeKeyPairs AWS SDK oder CLI
- Verwendung von DescribeNetworkAcls mit einer CLI
- Verwendung von DescribeNetworkInterfaceAttribute mit einer CLI
- Verwendung von DescribeNetworkInterfaces mit einer CLI
- Verwendung von DescribePlacementGroups mit einer CLI
- Verwendung von DescribePrefixLists mit einer CLI
- Verwenden Sie mit einem DescribeRegions AWS SDK oder CLI
- Verwenden Sie DescribeRouteTablesmit einem AWS SDK oder CLI
- Verwendung von DescribeScheduledInstanceAvailability mit einer CLI
- Verwendung von DescribeScheduledInstances mit einer CLI
- Verwenden Sie mit einem DescribeSecurityGroups AWS SDK oder CLI
- Verwendung von DescribeSnapshotAttribute mit einer CLI
- Verwenden Sie mit einem DescribeSnapshots AWS SDK oder CLI
- Verwendung von DescribeSpotDatafeedSubscription mit einer CLI
- Verwendung von DescribeSpotFleetInstances mit einer CLI
- Verwendung von DescribeSpotFleetRequestHistory mit einer CLI
- Verwendung von DescribeSpotFleetRequests mit einer CLI
- Verwendung von DescribeSpotInstanceRequests mit einer CLI

- Verwendung von DescribeSpotPriceHistory mit einer CLI
- Verwenden Sie mit einem DescribeSubnets AWS SDK oder CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeVolumeAttribute mit einer CLI
- Verwendung von DescribeVolumeStatus mit einer CLI
- Verwendung von DescribeVolumes mit einer CLI
- Verwendung von DescribeVpcAttribute mit einer CLI
- Verwendung von DescribeVpcClassicLink mit einer CLI
- Verwendung von DescribeVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von DescribeVpcEndpointServices mit einer CLI
- Verwendung von DescribeVpcEndpoints mit einer CLI
- Verwenden Sie mit einem DescribeVpcs AWS SDK oder CLI
- Verwendung von DescribeVpnConnections mit einer CLI
- Verwendung von DescribeVpnGateways mit einer CLI
- Verwendung von DetachInternetGateway mit einer CLI
- Verwendung von DetachNetworkInterface mit einer CLI
- Verwendung von DetachVolume mit einer CLI
- Verwendung von DetachVpnGateway mit einer CLI
- Verwendung von DisableVgwRoutePropagation mit einer CLI
- Verwendung von DisableVpcClassicLink mit einer CLI
- Verwendung von DisableVpcClassicLinkDnsSupport mit einer CLI
- Verwenden Sie mit einem DisassociateAddress AWS SDK oder CLI
- Verwendung von DisassociateRouteTable mit einer CLI
- Verwendung von EnableVgwRoutePropagation mit einer CLI
- Verwendung von EnableVolumeIo mit einer CLI
- Verwendung von EnableVpcClassicLink mit einer CLI
- Verwendung von EnableVpcClassicLinkDnsSupport mit einer CLI
- Verwendung von GetConsoleOutput mit einer CLI
- Verwendung von GetHostReservationPurchasePreview mit einer CLI
- Verwenden Sie mit einem GetPasswordData AWS SDK oder CLI

- Verwendung von ImportImage mit einer CLI
- Verwendung von ImportKeyPair mit einer CLI
- Verwendung von ImportSnapshot mit einer CLI
- Verwendung von ModifyCapacityReservation mit einer CLI
- Verwendung von ModifyHosts mit einer CLI
- Verwendung von ModifyIdFormat mit einer CLI
- Verwendung von ModifyImageAttribute mit einer CLI
- Verwendung von ModifyInstanceAttribute mit einer CLI
- Verwendung von ModifyInstanceCreditSpecification mit einer CLI
- Verwendung von ModifyNetworkInterfaceAttribute mit einer CLI
- Verwendung von ModifyReservedInstances mit einer CLI
- Verwendung von ModifySnapshotAttribute mit einer CLI
- Verwendung von ModifySpotFleetRequest mit einer CLI
- Verwendung von ModifySubnetAttribute mit einer CLI
- Verwendung von ModifyVolumeAttribute mit einer CLI
- Verwendung von ModifyVpcAttribute mit einer CLI
- Verwenden Sie mit einem MonitorInstances AWS SDK oder CLI
- Verwendung von MoveAddressToVpc mit einer CLI
- Verwendung von PurchaseHostReservation mit einer CLI
- Verwendung von PurchaseScheduledInstances mit einer CLI
- Verwenden Sie mit einem RebootInstances AWS SDK oder CLI
- Verwendung von RegisterImage mit einer CLI
- Verwendung von RejectVpcPeeringConnection mit einer CLI
- Verwenden Sie mit einem ReleaseAddress AWS SDK oder CLI
- Verwendung von ReleaseHosts mit einer CLI
- Verwenden Sie mit einem ReplaceIamInstanceProfileAssociation AWS SDK oder CLI
- Verwendung von ReplaceNetworkAclAssociation mit einer CLI
- Verwendung von ReplaceNetworkAclEntry mit einer CLI
- Verwendung von ReplaceRoute mit einer CLI
- Verwendung von ReplaceRouteTableAssociation mit einer CLI

- Verwendung von ReportInstanceStatus mit einer CLI
- Verwendung von RequestSpotFleet mit einer CLI
- Verwendung von RequestSpotInstances mit einer CLI
- Verwendung von ResetImageAttribute mit einer CLI
- Verwendung von ResetInstanceAttribute mit einer CLI
- Verwendung von ResetNetworkInterfaceAttribute mit einer CLI
- Verwendung von ResetSnapshotAttribute mit einer CLI
- Verwendung von RevokeSecurityGroupEgress mit einer CLI
- Verwendung von RevokeSecurityGroupIngress mit einer CLI
- Verwenden Sie mit einem RunInstances AWS SDK oder CLI
- Verwendung von RunScheduledInstances mit einer CLI
- Verwenden Sie mit einem StartInstances AWS SDK oder CLI
- Verwenden Sie StopInstancesmit einem AWS SDK oder CLI
- Verwenden Sie TerminateInstancesmit einem AWS SDK oder CLI
- Verwendung von UnassignPrivateIpAddresses mit einer CLI
- Verwenden Sie mit einem UnmonitorInstances AWS SDK oder CLI
- Verwendung von UpdateSecurityGroupRuleDescriptionsIngress mit einer CLI

Verwendung von **`AcceptVpcPeeringConnection`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AcceptVpcPeeringConnection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So akzeptieren Sie eine VPC-Peering-Verbindung

In diesem Beispiel wird die angegebene VPC-Peering-Verbindungsanforderung akzeptiert.

Befehl:

```
aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

Ausgabe:

```
{
  "VpcPeeringConnection": {
    "Status": {
      "Message": "Provisioning",
      "Code": "provisioning"
    },
    "Tags": [],
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-44455566",
      "CidrBlock": "10.0.1.0/28"
    },
    "VpcPeeringConnectionId": "pcx-1a2b3c4d",
    "RequesterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-111abc45",
      "CidrBlock": "10.0.0.0/28"
    }
  }
}
```

- Einzelheiten zur API finden Sie AcceptVpcPeeringConnectionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel genehmigt die angeforderte VpcPeeringConnectionId Datei pcx-1dfad234b56ff78be

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Ausgabe:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). AcceptVpcPeeringConnectionAWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel genehmigt die angeforderte Datei pcx-1dfad234b56ff78be VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Ausgabe:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). AcceptVpcPeeringConnectionAWS -Tools für PowerShell

Verwenden Sie mit einem **`AllocateAddress`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AllocateAddress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit virtuellen Maschinen

- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP
 address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
 ID {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic
 Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_allocate_address() {
  local domain response
```

```
# Function to display usage information
function usage() {
  echo "function ec2_allocate_address"
  echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
Cloud (Amazon EC2) instances in a specific AWS Region."
  echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
'standard')."
  echo ""
}

# Parse the command-line arguments
while getopts "d:h" option; do
  case "${option}" in
    d) domain="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$domain" ]]; then
  errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
or 'standard')."
  return 1
fi

if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
  errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
  return 1
fi

# Allocate the Elastic IP address
response=$(aws ec2 allocate-address \
  --domain "$domain" \
  --query "[PublicIp,AllocationId]" \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Allocate an Elastic IP address and associate it with an Amazon Elastic
 Compute Cloud
//! (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param[out] publicIPAddress: String to return the public IP address.
  \param[out] allocationID: String to return the allocation ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
 Aws::String &publicIPAddress,
                                               Aws::String &allocationID,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
```

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
            ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response =
 outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();


    return true;
}
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So weisen Sie eine Elastic-IP-Adresse aus dem Adress-Pool von Amazon zu

Im folgenden Beispiel für `allocate-address` wird eine Elastic-IP-Adresse zugewiesen. Amazon EC2 wählt die Adresse aus dem Adress-Pool von Amazon aus.

```
aws ec2 allocate-address
```

Ausgabe:

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-01435ba59eEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2",
```

```
    "Domain": "vpc"
}
```

Weitere Informationen finden Sie unter Elastische IP-Adressen im Amazon-EC2-Benutzerhandbuch.

Beispiel 2: So weisen Sie eine Elastic-IP-Adresse zu und verknüpfen sie mit einer Netzwerkgrenzgruppe

Im folgenden Beispiel für `allocate-address` wird eine Elastic-IP-Adresse zugewiesen und sie der angegebenen Netzwerkgrenzgruppe zugeordnet.

```
aws ec2 allocate-address \
    --network-border-group us-west-2-lax-1
```

Ausgabe:

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-e03dd489ceEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2-lax-1",
    "Domain": "vpc"
}
```

Weitere Informationen finden Sie unter Elastische IP-Adressen im Amazon-EC2-Benutzerhandbuch.

Beispiel 3: So weisen Sie eine Elastic-IP-Adresse aus einem Adress-Pool zu, der Ihnen gehört

Im folgenden Beispiel für `allocate-address` wird eine Elastic-IP-Adresse aus einem Adress-Pool zugewiesen, den Sie in Ihr Amazon-Web-Services-Konto eingebunden haben. Amazon EC2 wählt die Adresse aus dem Adress-Pool aus.

```
aws ec2 allocate-address \
    --public-ipv4-pool ipv4pool-ec2-1234567890abcdef0
```

Ausgabe:

```
{
```

```
    "AllocationId": "eipalloc-02463d08ceEXAMPLE",
    "NetworkBorderGroup": "us-west-2",
    "CustomerOwnedIp": "18.218.95.81",
    "CustomerOwnedIpv4Pool": "ipv4pool-ec2-1234567890abcdef0",
    "Domain": "vpc"
    "NetworkBorderGroup": "us-west-2",
}
```

Weitere Informationen finden Sie unter Elastische IP-Adressen im Benutzerhandbuch für Amazon EC2.

Beispiel 4: So weisen Sie eine Elastic-IP-Adresse aus einem IPAM-Pool zu

Im folgenden Beispiel für `allocate-address` wird eine bestimmte /32 Elastic-IP-Adresse aus einem Pool von Amazon VPC IP Address Manager (IPAM) zugewiesen.

```
aws ec2 allocate-address \
    --region us-east-1 \
    --ipam-pool-id ipam-pool-1234567890abcdef0 \
    --address 192.0.2.0
```

Ausgabe:

```
{
    "PublicIp": "192.0.2.0",
    "AllocationId": "eipalloc-abcdef01234567890",
    "PublicIpv4Pool": "ipam-pool-1234567890abcdef0",
    "NetworkBorderGroup": "us-east-1",
    "Domain": "vpc"
}
```

Weitere Informationen finden Sie unter Zuweisen sequenzieller Elastic IP-Adressen aus einem IPAM-Pool im Benutzerhandbuch für Amazon VPC IPAM.

- Einzelheiten zur API finden Sie AllocateAddressin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AllocateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Allocates an Elastic IP address to your AWS account.
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon
 EC2. Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- Einzelheiten zur API finden Sie AllocateAddressin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird einer Instance in einer VPC eine Elastic-IP-Adresse zugewiesen.

```
New-EC2Address -Domain Vpc
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Beispiel 2: In diesem Beispiel wird eine Elastic IP-Adresse zugewiesen, die zusammen mit einer Instance verwendet werden soll. EC2-Classic

```
New-EC2Address
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Einzelheiten zur API finden Sie unter AllocateAddress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird einer Instance in einer VPC eine Elastic-IP-Adresse zugewiesen.

```
New-EC2Address -Domain Vpc
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Beispiel 2: In diesem Beispiel wird eine Elastic IP-Adresse zugewiesen, die mit einer Instance verwendet werden soll. EC2-Classic

```
New-EC2Address
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- Einzelheiten zur API finden Sie unter AllocateAddress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
```

```
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
 address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
 maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
```

```
                allocation_id=response["AllocationId"],
 public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
 Support for an increase."
                )
            raise err
        return elastic_ip
```

- Einzelheiten zur API finden Sie AllocateAddressin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
       self.client
           .allocate_address()
           .domain(DomainType::Vpc)
           .send()
           .await
           .map_err(EC2Error::from)
   }
```

- Einzelheiten zur API finden Sie AllocateAddressin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).   " oo_result
 is returned for testing purposes. "
        MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AllocateAddressin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
```

```
        } catch {
            print("*** Unable to allocate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie AllocateAddressin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`AllocateHosts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AllocateHosts` verwendet wird.

CLI

AWS CLI

Beispiel 1: So weisen Sie einen Dedicated Host zu

Im folgenden Beispiel für `allocate-hosts` wird ein einzelner Dedicated Host in der Availability Zone `eu-west-1a` zugewiesen, auf dem Sie `m5.large`-Instances starten können. Standardmäßig akzeptiert der Dedicated Host nur das Starten von Ziel-Instances und unterstützt keine Host-Wiederherstellung.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --quantity 1
```

Ausgabe:

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

Beispiel 2: So weisen Sie einen Dedicated Host mit aktivierter automatischer Platzierung und Host-Wiederherstellung zu

Im folgenden Beispiel für `allocate-hosts` wird ein einzelner Dedicated Host in der Availability Zone `eu-west-1a` zugewiesen, wobei die automatische Platzierung und die Host-Wiederherstellung aktiviert sind.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --auto-placement on \
    --host-recovery on \
    --quantity 1
```

Ausgabe:

```
{
     "HostIds": [
         "h-07879acf49EXAMPLE"
     ]
}
```

Beispiel 3: So weisen Sie einen Dedicated Host mit Tags zu

Im folgenden Beispiel für `allocate-hosts` wird ein einzelner Dedicated Host zugewiesen und mit einem Tag mit dem Schlüssel `purpose` und dem Wert `production` versehen.

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --quantity 1 \
    --tag-specifications 'ResourceType=dedicated-
host,Tags={Key=purpose,Value=production}'
```

Ausgabe:

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

Weitere Informationen finden Sie unter Zuweisen eines Dedicated Hosts im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie AllocateHostsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird Ihrem Konto ein Dedicated Host für den angegebenen Instance-Typ und die angegebene Availability Zone zugewiesen

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType
 m4.xlarge -Quantity 1
```

Ausgabe:

```
h-01e23f4cd567890f3
```

- Einzelheiten zur API finden Sie unter AllocateHosts AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird Ihrem Konto ein Dedicated Host für den angegebenen Instance-Typ und die angegebene Availability Zone zugewiesen

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType
 m4.xlarge -Quantity 1
```

Ausgabe:

```
h-01e23f4cd567890f3
```

- Einzelheiten zur API finden Sie unter AllocateHosts AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AssignPrivateIpAddresses`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AssignPrivateIpAddresses` verwendet wird.

CLI

AWS CLI

So weisen Sie einer Netzwerkschnittstelle eine bestimmte sekundäre private IP-Adresse zu

In diesem Beispiel werden die genannten sekundären privaten IP-Adressen der angegebenen Netzwerkschnittstelle zugewiesen. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

So weisen Sie einer Netzwerkschnittstelle sekundäre private IP-Adressen zu, die Amazon EC2 auswählt

In diesem Beispiel werden zwei sekundäre private IP-Adressen der angegebenen Netzwerkschnittstelle zugewiesen. Amazon EC2 weist diese IP-Adressen automatisch aus den verfügbaren IP-Adressen im CIDR-Blockbereich des Subnetzes zu, dem die Netzwerkschnittstelle zugeordnet ist. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
secondary-private-ip-address-count 2
```

- Einzelheiten zur API finden Sie AssignPrivateIpAddressesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die genannten sekundären privaten IP-Adressen der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Beispiel 2: In diesem Beispiel werden zwei sekundäre private IP-Adressen erstellt und der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Einzelheiten zur API finden Sie unter AssignPrivateIpAddresses AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die genannten sekundären privaten IP-Adressen der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Beispiel 2: In diesem Beispiel werden zwei sekundäre private IP-Adressen erstellt und der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- Einzelheiten zur API finden Sie unter AssignPrivateIpAddresses AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`AssociateAddress`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AssociateAddress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association
 is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
```

```
                    $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
#
##############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic
 IP address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
```

```
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
```

```
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

//! Associate an Elastic IP address with an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param allocationId: An Elastic IP allocation ID.
  \param[out] associationID: String to receive the association ID.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: True if the address was associated with the instance; otherwise,
 false.
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const
 Aws::String &allocationId,
                                    Aws::String &associationID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest request;
    request.SetInstanceId(instanceId);
    request.SetAllocationId(allocationId);

    Aws::EC2::Model::AssociateAddressOutcome outcome =
 ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
                  " with instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
                  " with instance " << instanceId << std::endl;
        associationID = outcome.GetResult().GetAssociationId();
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So ordnen Sie eine Elastic-IP-Adresse einer Instance zu

Im folgenden Beispiel für `associate-address` wird eine Elastic-IP-Adresse der angegebenen EC2-Instance zugeordnet.

```
aws ec2 associate-address \
    --instance-id i-0b263919b6498b123 \
    --allocation-id eipalloc-64d5890a
```

Ausgabe:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Beispiel 2: So ordnen Sie einer Elastic-IP-Adresse eine Netzwerkschnittstelle zu

Im folgenden Beispiel für `associate-address` wird die angegebene Elastic-IP-Adresse der genannten Netzwerkschnittstelle zugeordnet.

```
aws ec2 associate-address
    --allocation-id eipalloc-64d5890a \
    --network-interface-id eni-1a2b3c4d
```

Ausgabe:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Beispiel 3: So weisen Sie einer Elastic-IP-Adresse eine private IP-Adresse zu

Im folgenden Beispiel für `associate-address` wird die angegebene Elastic-IP-Adresse der festgelegten privaten IP-Adresse in der genannten Netzwerkschnittstelle zugeordnet.

```
aws ec2 associate-address \
    --allocation-id eipalloc-64d5890a \
    --network-interface-id eni-1a2b3c4d \
    --private-ip-address 10.0.0.85
```

Ausgabe:

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

Weitere Informationen finden Sie unter Elastische IP-Adressen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie AssociateAddressin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic
IP address with
    * @param allocationId  the allocation ID of the Elastic IP address to
associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
       AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after
associating address.");
```

```
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AssociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that
 are in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
 instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
 with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
```

```
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
 ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Did you provide the ID of a valid Elastic IP address
 AllocationId?`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
```

```
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- Einzelheiten zur API finden Sie AssociateAddressin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse der spezifizierten Instance in einer VPC zugeordnet.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Ausgabe:

```
eipassoc-12345678
```

Beispiel 2: Dieses Beispiel verknüpft die angegebene Elastic IP-Adresse mit der angegebenen Instance in EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Einzelheiten zur API finden Sie unter AssociateAddress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse der spezifizierten Instance in einer VPC zugeordnet.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Ausgabe:

```
eipassoc-12345678
```

Beispiel 2: Dieses Beispiel verknüpft die angegebene Elastic IP-Adresse mit der angegebenen Instance in EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- Einzelheiten zur API finden Sie unter AssociateAddress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
```

```
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association
 is
        created, the Elastic IP's public IP address is immediately used as the
 public
        IP address of the associated instance.
```

```
        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if
no Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance
ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic
IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response
```

- Einzelheiten zur API finden Sie AssociateAddressin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
```

```
end
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
    Ok(response)
}
```

- Einzelheiten zur API finden Sie AssociateAddressin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->associateaddress(                        " oo_result
 is returned for testing purposes. "
            iv_allocationid = iv_allocation_id
            iv_instanceid = iv_instance_id ).
        MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AssociateAddressin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Associate the specified allocated Elastic IP to a given instance.
```

```
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
 String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie AssociateAddressin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`AssociateDhcpOptions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AssociateDhcpOptions` verwendet wird.

CLI

AWS CLI

So ordnen Sie Ihrer VPC einen DHCP-Optionssatz zu

In diesem Beispiel wird der angegebenen VPC der genannte DHCP-Optionssatz zugeordnet. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 associate-dhcp-options --dhcp-options-id dopt-d9070ebb --vpc-id vpc-
a01106c2
```

So ordnen Sie Ihrer VPC den standardmäßigen DHCP-Optionssatz zu

In diesem Beispiel wird der angegebenen VPC der standardmäßige DHCP-Optionssatz zugeordnet. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 associate-dhcp-options --dhcp-options-id default --vpc-id vpc-a01106c2
```

- Einzelheiten zur API finden Sie AssociateDhcpOptionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebenen VPC der genannte DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel wird der angegebenen VPC der standardmäßige DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter AssociateDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen VPC der genannte DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel wird der angegebenen VPC der standardmäßige DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter AssociateDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AssociateRouteTable`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AssociateRouteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So ordnen Sie eine Routing-Tabelle einem Subnetz zu

In diesem Beispiel wird die angegebene Routing-Tabelle dem festgelegten Subnetz zugeordnet.

Befehl:

```
aws ec2 associate-route-table --route-table-id rtb-22574640 --subnet-
id subnet-9d4a7b6c
```

Ausgabe:

```
{
    "AssociationId": "rtbassoc-781d0d1a"
}
```

- Einzelheiten zur API finden Sie AssociateRouteTablein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem festgelegten Subnetz zugeordnet.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
rtbassoc-12345678
```

- Einzelheiten zur API finden Sie unter AssociateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem festgelegten Subnetz zugeordnet.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
rtbassoc-12345678
```

- Einzelheiten zur API finden Sie unter AssociateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachInternetGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachInternetGateway` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So hängen Sie Ihrer VPC ein Internet-Gateway an

Im folgenden Beispiel für `attach-internet-gateway` wird der spezifischen VPC das angegebene Internet-Gateway angehängt.

```
aws ec2 attach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Internet-Gateways im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie AttachInternetGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Im folgenden Beispiel wird der spezifischen VPC das angegebene Internet-Gateway angehängt.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel werden eine VPC und ein Internet-Gateway erstellt und anschließend das Internet-Gateway mit der VPC verbunden.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Einzelheiten zur API finden Sie unter AttachInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Im folgenden Beispiel wird der spezifischen VPC das angegebene Internet-Gateway angehängt.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel werden eine VPC und ein Internet-Gateway erstellt und anschließend das Internet-Gateway mit der VPC verbunden.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- Einzelheiten zur API finden Sie unter AttachInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachNetworkInterface`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachNetworkInterface` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie einer Instance eine Netzwerkschnittstelle an

Im folgenden Beispiel für `attach-network-interface` wird der spezifischen Instance die angegebene Netzwerkschnittstelle angefügt.

```
aws ec2 attach-network-interface \
    --network-interface-id eni-0dc56a8d4640ad10a \
    --instance-id i-1234567890abcdef0 \
    --device-index 1
```

Ausgabe:

```
{
    "AttachmentId": "eni-attach-01a8fc87363f07cf9"
}
```

Weitere Informationen finden Sie unter Elastic Network-Schnittstellen im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So fügen Sie einer Instance mit mehreren Netzwerkkarten eine Netzwerkschnittstelle an

Im folgenden Beispiel für `attach-network-interface` wird der spezifischen Instance und Netzwerkkarte die angegebene Netzwerkschnittstelle angefügt.

```
aws ec2 attach-network-interface \
    --network-interface-id eni-07483b1897541ad83 \
    --instance-id i-01234567890abcdef \
    --network-card-index 1 \
    --device-index 1
```

Ausgabe:

```
{
    "AttachmentId": "eni-attach-0fbd7ee87a88cd06c"
}
```

Weitere Informationen finden Sie unter Elastic Network-Schnittstellen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie AttachNetworkInterfacein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle an die spezifizierte Instance angehängt.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Ausgabe:

```
eni-attach-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter AttachNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle an die spezifizierte Instance angehängt.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Ausgabe:

```
eni-attach-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter AttachNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachVolume`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachVolume` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Blockspeicher-Volumes erstellen und verwalten

CLI

AWS CLI

So hängen Sie einer Instance ein Volume an

Dieser Beispielbefehl hängt ein Volume (`vol-1234567890abcdef0`) an eine Instance (`i-01474ef662b89480`) als `/dev/sdf` an.

Befehl:

```
aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-
id i-01474ef662b89480 --device /dev/sdf
```

Ausgabe:

```
{
    "AttachTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "InstanceId": "i-01474ef662b89480",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "attaching",
```

```
    "Device": "/dev/sdf"
}
```

- Einzelheiten zur API finden Sie AttachVolumein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Volume an die spezifizierte Instance angehängt und mit dem angegebenen Gerätenamen verfügbar gemacht.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Einzelheiten zur API finden Sie unter AttachVolume AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume an die spezifizierte Instance angehängt und mit dem angegebenen Gerätenamen verfügbar gemacht.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- Einzelheiten zur API finden Sie unter AttachVolume AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachVpnGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachVpnGateway` verwendet wird.

CLI

AWS CLI

So fügen Sie Ihrer VPC ein Virtual Private Gateway an

Im folgenden Beispiel für `attach-vpn-gateway` wird der spezifischen VPC das angegebene Virtual Private Gateway angefügt.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id vgw-9a4cacf3 \
    --vpc-id vpc-a01106c2
```

Ausgabe:

```
{
    "VpcAttachment": {
        "State": "attaching",
        "VpcId": "vpc-a01106c2"
    }
}
```

- Einzelheiten zur API finden Sie AttachVpnGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway an die spezifizierte VPC angehängt.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Ausgabe:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Einzelheiten zur API finden Sie unter AttachVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway an die spezifizierte VPC angehängt.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Ausgabe:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- Einzelheiten zur API finden Sie unter AttachVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AuthorizeSecurityGroupEgress`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AuthorizeSecurityGroupEgress` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie eine Regel hinzu, die ausgehenden Datenverkehr in einem bestimmten Adressbereich zulässt

Im folgenden Beispiel für `authorize-security-group-egress` wird eine Regel hinzugefügt, die Zugriff auf die angegebenen Adressbereiche am TCP-Port 80 gewährt.

```
aws ec2 authorize-security-group-egress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0b15794cdb17bf29c",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": true,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "10.0.0.0/16"
        }
    ]
}
```

Beispiel 2: So fügen Sie eine Regel hinzu, die ausgehenden Datenverkehr an eine bestimmte Sicherheitsgruppe zulässt

Im folgenden Beispiel für `authorize-security-group-egress` wird eine Regel hinzugefügt, die Zugriff auf die angegebene Sicherheitsgruppe am TCP-Port 80 gewährt.

```
aws ec2 authorize-security-group-egress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-0aad1c26bbeec5c22}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0b5dd815afcea9cc3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": true,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
```

```
            "ReferencedGroupInfo": {
                "GroupId": "sg-0aad1c26bbeec5c22",
                "UserId": "123456789012"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupEgressin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel definiert eine Ausgangsregel für die angegebene Sicherheitsgruppe für EC2-VPC. Die Regel gewährt Zugriff auf den angegebenen IP-Adressbereich über den TCP-Port 80. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der angegebenen Quellsicherheitsgruppe Zugriff über den TCP-Port 80 gewährt 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupEgress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel definiert eine Ausgangsregel für die angegebene Sicherheitsgruppe für. EC2-VPC Die Regel gewährt Zugriff auf den angegebenen IP-Adressbereich über den TCP-Port 80. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der angegebenen Quellsicherheitsgruppe Zugriff über den TCP-Port 80 gewährt 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupEgress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`AuthorizeSecurityGroupIngress`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AuthorizeSecurityGroupIngress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Eine relationale Datenbankinstanz erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit Dokumentendatenbanken
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit In-Memory-Caching
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen
- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
```

```
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
```

```
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
failed.$response"
```

```
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
/*!
  \param groupID: The EC2 group ID.
  \param clientConfiguration: The ClientConfiguration object.
  \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
 authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
 authorizeSecurityGroupIngressOutcome =
```

```
 ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." <<
 std::endl;
    } else {
        std::cerr << "Error authorizing security group ingress: "
                  << authorizeSecurityGroupIngressOutcome.GetError().GetMessage()
 << std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}
```

Hilfsfunktion zum Erstellen einer Eingangsregel.

```
//! Build a sample ingress rule.
/*!
  \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
  \return void:
 */
void buildSampleIngressRule(
        Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request)
 {
    Aws::String ingressIPRange = "203.0.113.0/24";  // Configure this for your
 allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
```

```
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So fügen Sie eine Regel hinzu, die eingehenden SSH-Datenverkehr zulässt

Im folgenden Beispiel für `authorize-security-group-ingress` wird eine Regel hinzugefügt, die eingehenden Datenverkehr auf TCP-Anschluss 22 (SSH) zulässt.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01afa97ef3e1bedfc",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "203.0.113.0/24"
        }
    ]
}
```

Beispiel 2: So fügen Sie eine Regel hinzu, die eingehenden HTTP-Datenverkehr aus einer anderen Sicherheitsgruppe zulässt

Im folgenden Beispiel für `authorize-security-group-ingress` wird eine Regel hinzugefügt, die eingehenden Zugriff auf TCP-Anschluss 80 von der Quellsicherheitsgruppe `sg-1a2b3c4d` aus ermöglicht. Die Quellgruppe muss sich in derselben VPC oder einer Peer-VPC befinden (dazu ist eine VPC-Peering-Verbindung erforderlich). Eingehender Datenverkehr ist basierend auf den privaten IP-Adressen der Instances erlaubt, die der Quellsicherheitsgruppe zugeordnet sind (nicht die öffentliche IP-Adresse oder die Elastic-IP-Adresse).

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 80 \
    --source-group sg-1a2b3c4d
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01f4be99110f638a7",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "ReferencedGroupInfo": {
                "GroupId": "sg-1a2b3c4d",
                "UserId": "123456789012"
            }
        }
    ]
}
```

Beispiel 3: So fügen Sie mehrere Regeln im selben Aufruf hinzu

Im folgenden `authorize-security-group-ingress` Beispiel werden mithilfe des `ip-permissions` Parameters zwei Regeln für eingehenden Datenverkehr hinzugefügt: eine, die den eingehenden Zugriff auf TCP-Port 3389 (RDP) ermöglicht, und die andere, die aktiviert. ping/ICMP

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=172.31.0.0/16}]'
 'IpProtocol=icmp,FromPort=-1,ToPort=-1,IpRanges=[{CidrIp=172.31.0.0/16}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-00e06e5d3690f29f3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 3389,
            "ToPort": 3389,
            "CidrIpv4": "172.31.0.0/16"
        },
        {
            "SecurityGroupRuleId": "sgr-0a133dd4493944b87",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": -1,
            "ToPort": -1,
            "CidrIpv4": "172.31.0.0/16"
        }
    ]
}
```

Beispiel 4: So fügen Sie eine Regel für ICMP-Datenverkehr hinzu

Im folgenden Beispiel für `authorize-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine eingehende Regel hinzuzufügen, die die ICMP-Nachricht `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set`(Typ 3, Code 4) von überall her zulässt.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0de3811019069b787",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "icmp",
            "FromPort": 3,
            "ToPort": 4,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
```

Beispiel 5: So fügen Sie eine Regel für IPv6-Datenverkehr hinzu

Im folgenden Beispiel für `authorize-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine Regel für eingehenden Datenverkehr hinzuzufügen, die SSH-Zugriff (Anschluss 22) aus dem IPv6-Bereich `2001:db8:1234:1a00::/64` ermöglicht.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=22,ToPort=22,Ipv6Ranges=[{CidrIpv6=2001:db8:1234:1a00::/64}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0455bc68b60805563",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv6": "2001:db8:1234:1a00::/64"
        }
    ]
}
```

Beispiel 6: So fügen Sie eine Regel für ICMPv6-Datenverkehr hinzu

Im folgenden Beispiel für `authorize-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine eingehende Regel hinzuzufügen, die ICMPv6-Datenverkehr von überall her zulässt.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions 'IpProtocol=icmpv6,Ipv6Ranges=[{CidrIpv6=::/0}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-04b612d9363ab6327",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "icmpv6",
            "FromPort": -1,
            "ToPort": -1,
            "CidrIpv6": "::/0"
        }
    ]
```

```
}
```

Beispiel 7: Eine Regel mit einer Beschreibung hinzufügen

Im folgenden Beispiel für `authorize-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine eingehende Regel hinzuzufügen, die RDP-Datenverkehr aus dem angegebenen IPv4-Adressbereich zulässt. Die Regel enthält eine Beschreibung, die Ihnen später hilft, sie zu identifizieren.

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --ip-permissions
 'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=203.0.113.0/24,Description='
 office'}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0397bbcc01e974db3",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 3389,
            "ToPort": 3389,
            "CidrIpv4": "203.0.113.0/24",
            "Description": "RDP access from NY office"
        }
    ]
}
```

Beispiel 8: So fügen Sie eine eingehende Regel hinzu, die eine Präfixliste verwendet

Im folgenden Beispiel für `authorize-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine Regel für eingehenden Datenverkehr hinzuzufügen, die den gesamten Datenverkehr für die CIDR-Bereiche in der angegebenen Präfixliste zulässt.

```
aws ec2 authorize-security-group-ingress \
```

```
    --group-id sg-04a351bfe432d4e71 \
    --ip-permissions
 'IpProtocol=all,PrefixListIds=[{PrefixListId=pl-002dc3ec097de1514}]'
```

Ausgabe:

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09c74b32f677c6c7c",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "-1",
            "FromPort": -1,
            "ToPort": -1,
            "PrefixListId": "pl-0721453c7ac4ec009"
        }
    ]
}
```

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupIngressBefehlsreferenz.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
 description, and VPC ID. It also
```

```
    * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .build();

               IpPermission ipPerm = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(80)
                   .fromPort(80)
                   .ipRanges(ipRange)
                   .build();

               IpPermission ipPerm2 = IpPermission.builder()
                   .ipProtocol("tcp")
                   .toPort(22)
                   .fromPort(22)
                   .ipRanges(ipRange)
                   .build();
```

```
                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  AuthorizeSecurityGroupIngressCommand,
  EC2Client,
```

```
} from "@aws-sdk/client-ec2";

/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
```

```
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}
```

- API-Details finden Sie AuthorizeSecurityGroupIngressin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für EC2-VPC definiert. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Beachten Sie, dass Sie Sicherheitsgruppen identifizieren müssen, um die Sicherheitsgruppen-ID und nicht den Sicherheitsgruppennamen zu EC2-VPC verwenden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
```

```
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 3: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für EC2-Classic definiert. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")
```

```
Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Beispiel 5: Dieses Beispiel gewährt TCP-Port 8081 Zugriff von der angegebenen Quellsicherheitsgruppe (sg-1a2b3c4d) auf die angegebene Sicherheitsgruppe (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Beispiel 6: In diesem Beispiel wird CIDR 5.5.5 hinzugefügt. 5/32 zu den Eingangsregeln der Sicherheitsgruppe sg-1234abcd für TCP-Port 22-Verkehr mit einer Beschreibung.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupIngressCmdlet-Referenz (V4).AWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für definiert. EC2-VPC Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Beachten Sie, dass Sie Sicherheitsgruppen identifizieren müssen, um die Sicherheitsgruppen-ID und nicht den Sicherheitsgruppennamen zu EC2-VPC verwenden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }
```

```
Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 3: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für EC2-Classic definiert. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")
```

```
$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
 @( $ip1, $ip2 )
```

Beispiel 5: Dieses Beispiel gewährt TCP-Port 8081 Zugriff von der angegebenen Quellsicherheitsgruppe (sg-1a2b3c4d) auf die angegebene Sicherheitsgruppe (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Beispiel 6: In diesem Beispiel wird CIDR 5.5.5 hinzugefügt. 5/32 zu den Eingangsregeln der Sicherheitsgruppe sg-1234abcd für TCP-Port 22-Verkehr mit einer Beschreibung.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Einzelheiten zur API finden Sie unter AuthorizeSecurityGroupIngressCmdlet-Referenz (V5).AWS -Tools für PowerShell

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
```

```
    }
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create IP permissions for SSH access (port 22)
    " iv_cidr_ip = '192.0.2.0/24'
    DATA lt_ip_permissions TYPE /aws1/cl_ec2ippermission=>tt_ippermissionlist.
    DATA(lo_ip_permission) = NEW /aws1/cl_ec2ippermission(
      iv_ipprotocol = 'tcp'
      iv_fromport = 22
      iv_toport = 22
      it_ipranges = VALUE /aws1/cl_ec2iprange=>tt_iprangelist(
        ( NEW /aws1/cl_ec2iprange( iv_cidrip = iv_cidr_ip ) )
      )
    ).
    APPEND lo_ip_permission TO lt_ip_permissions.

    TRY.
        oo_result = lo_ec2->authsecuritygroupingress(             " oo_result is
 returned for testing purposes. "
          iv_groupid = iv_group_id
          it_ippermissions = lt_ip_permissions ).
        MESSAGE 'Authorized ingress rule for security group.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
 -> Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
```

```
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`CancelCapacityReservation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelCapacityReservation` verwendet wird.

CLI

AWS CLI

So stornieren Sie eine Kapazitätsreservierung

Im folgenden Beispiel für `cancel-capacity-reservation` wird die angegebene Kapazitätsreservierung storniert.

```
aws ec2 cancel-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE
```

Ausgabe:

```
{
    "Return": true
}
```

Weitere Informationen finden Sie unter Stornieren einer Kapazitätsreservierung im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CancelCapacityReservationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Kapazitätsreservierung cr-0c1f2345db6f7cdba storniert

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
 (CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- Einzelheiten zur API finden Sie unter CancelCapacityReservation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Kapazitätsreservierung cr-0c1f2345db6f7cdba storniert

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
 (CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- Einzelheiten zur API finden Sie unter CancelCapacityReservation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CancelImportTask`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelImportTask` verwendet wird.

CLI

AWS CLI

So brechen Sie eine Importaufgabe ab

Im folgenden Beispiel für `cancel-import-task` wird die angegebene Aufgabe zum Importieren eines Image abgebrochen.

```
aws ec2 cancel-import-task \
    --import-task-id import-ami-1234567890abcdef0
```

Ausgabe:

```
{
    "ImportTaskId": "import-ami-1234567890abcdef0",
    "PreviousState": "active",
    "State": "deleting"
}
```

- Einzelheiten zur API finden Sie CancelImportTaskin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Importaufgabe (entweder Snapshot- oder Bildimport) abgebrochen. Gegebenenfalls kann über den **`-CancelReason`**-Parameter ein Grund angegeben werden.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Einzelheiten zur API finden Sie unter CancelImportTask AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Importaufgabe (entweder Snapshot- oder Bildimport) abgebrochen. Gegebenenfalls kann über den **`-CancelReason`**-Parameter ein Grund angegeben werden.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- Einzelheiten zur API finden Sie unter CancelImportTask AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CancelSpotFleetRequests`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelSpotFleetRequests` verwendet wird.

CLI

AWS CLI

Beispiel 1: So brechen Sie eine Spot-Flottenanforderung ab und beenden die zugehörigen Instances

Im folgenden `cancel-spot-fleet-requests` Beispiel wird eine Spot-Flottenanforderung storniert und die zugehörigen On-Demand Instances und Spot-Instances beendet.

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --terminate-instances
```

Ausgabe:

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_terminating",
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

Beispiel 2: So brechen Sie eine Spot-Flottenanforderung ab, ohne die zugehörigen Instances zu beenden

Im folgenden `cancel-spot-fleet-requests` Beispiel wird eine Spot-Flottenanforderung storniert, ohne die zugehörigen On-Demand Instances und Spot-Instances zu beenden.

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --no-terminate-instances
```

Ausgabe:

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_running",
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

Weitere Informationen finden Sie unter Abbrechen einer Spot-Flottenanforderung im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie unter CancelSpotFleetRequests AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen und die zugehörigen Spot Instances beendet.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Beispiel 2: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen, ohne die zugehörigen Spot Instances zu beenden.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Einzelheiten zur API finden Sie unter CancelSpotFleetRequests AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen und die zugehörigen Spot Instances beendet.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Beispiel 2: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen, ohne die zugehörigen Spot Instances zu beenden.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- Einzelheiten zur API finden Sie unter CancelSpotFleetRequests AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CancelSpotInstanceRequests`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelSpotInstanceRequests` verwendet wird.

CLI

AWS CLI

So brechen Sie Spot-Instance-Anforderungen ab

Dieser Beispielbefehl bricht eine Spot-Instance-Anforderung ab.

Befehl:

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-08b93456
```

Ausgabe:

```
{
    "CancelledSpotInstanceRequests": [
        {
            "State": "cancelled",
            "SpotInstanceRequestId": "sir-08b93456"
        }
    ]
}
```

- Einzelheiten zur API finden Sie CancelSpotInstanceRequestsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung abgebrochen.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- Einzelheiten zur API finden Sie unter CancelSpotInstanceRequests AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung abgebrochen.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
SpotInstanceRequestId    State
---------------------    -----
```

```
sir-12345678            cancelled
```

- Einzelheiten zur API finden Sie unter CancelSpotInstanceRequests AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ConfirmProductInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmProductInstance` verwendet wird.

CLI

AWS CLI

So bestätigen Sie die Produkt-Instance

In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der spezifischen Instance zugeordnet ist.

Befehl:

```
aws ec2 confirm-product-instance --product-code 774F4FF8 --instance-
id i-1234567890abcdef0
```

Ausgabe:

```
{
  "OwnerId": "123456789012"
}
```

- Einzelheiten zur API finden Sie ConfirmProductInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der spezifischen Instance zugeordnet ist.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Einzelheiten zur API finden Sie unter ConfirmProductInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der spezifischen Instance zugeordnet ist.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- Einzelheiten zur API finden Sie unter ConfirmProductInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CopyImage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CopyImage` verwendet wird.

CLI

AWS CLI

Beispiel 1: So kopieren Sie ein AMI in eine andere Region

Im folgenden Beispielbefehl für `copy-image` wird das angegebene AMI von der Region `us-west-2` in die Region `us-east-1` kopiert und eine kurze Beschreibung hinzugefügt.

```
aws ec2 copy-image \
    --region us-east-1 \
    --name ami-name \
    --source-region us-west-2 \
    --source-image-id ami-066877671789bd71b \
    --description "This is my copied image."
```

Ausgabe:

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

Weitere Informationen finden Sie unter Kopieren eines AMI im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So kopieren Sie ein AMI in eine andere Region und verschlüsseln den Sicherungs-Snapshot

Mit dem folgenden Befehl für `copy-image` wird das angegebene AMI von der Region `us-west-2` in die aktuelle Region kopiert und der Sicherungs-Snapshot mit dem angegebenen KMS-Schlüssel verschlüsselt.

```
aws ec2 copy-image \
    --source-region us-west-2 \
    --name ami-name \
    --source-image-id ami-066877671789bd71b \
    --encrypted \
    --kms-key-id alias/my-kms-key
```

Ausgabe:

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

Weitere Informationen finden Sie unter Kopieren eines AMI im Benutzerhandbuch für Amazon EC2.

Beispiel 3: So beziehen Sie Ihre benutzerdefinierten AMI-Tags beim Kopieren eines AMI ein

Im folgenden Befehl für `copy-image` wird der Parameter `--copy-image-tags` verwendet, um Ihre benutzerdefinierten AMI-Tags beim Kopieren des AMI zu kopieren.

```
aws ec2 copy-image \
    --region us-east-1 \
    --name ami-name \
    --source-region us-west-2 \
    --source-image-id ami-066877671789bd71b \
    --description "This is my copied image." \
    --copy-image-tags
```

Ausgabe:

```
{
    "ImageId": "ami-0123456789abcdefg"
```

```
}
```

Weitere Informationen finden Sie unter Kopieren eines AMI im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CopyImagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene AMI in der Region „EU (Irland)“ in die Region „USA West (Oregon)“ kopiert. Wenn -Region nicht angegeben ist, wird die aktuelle Standardregion als Zielregion verwendet.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

Ausgabe:

```
ami-87654321
```

- Einzelheiten zur API finden Sie unter CopyImage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene AMI in der Region „EU (Irland)“ in die Region „USA West (Oregon)“ kopiert. Wenn -Region nicht angegeben ist, wird die aktuelle Standardregion als Zielregion verwendet.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

Ausgabe:

```
ami-87654321
```

- Einzelheiten zur API finden Sie unter CopyImage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CopySnapshot`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CopySnapshot` verwendet wird.

CLI

AWS CLI

Beispiel 1: So kopieren Sie einen Snapshot in eine andere Region

Im folgenden Beispielbefehl für `copy-snapshot` wird der angegebene Snapshot von der Region `us-west-2` in die Region `us-east-1` kopiert und eine kurze Beschreibung hinzugefügt.

```
aws ec2 copy-snapshot \
    --region us-east-1 \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
    --description 'This is my copied snapshot.'
```

Ausgabe:

```
{
    "SnapshotId": "snap-066877671789bd71b"
}
```

Beispiel 2: So kopieren Sie einen unverschlüsselten Snapshot und verschlüsseln den neuen Snapshot

Mit dem folgenden `copy-snapshot`-Befehl wird der angegebene unverschlüsselte Snapshot von der Region `us-west-2` in die aktuelle Region kopiert und der neue Snapshot mit dem angegebenen KMS-Schlüssel verschlüsselt.

```
aws ec2 copy-snapshot \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
    --encrypted \
    --kms-key-id alias/my-kms-key
```

Ausgabe:

```
{
```

```
    "SnapshotId": "snap-066877671789bd71b"
}
```

Weitere Informationen finden Sie unter Kopieren eines Amazon-EBS-Snapshots im Benutzerhandbuch für Amazon EBS.

- Einzelheiten zur API finden Sie CopySnapshotin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot von der Region EU (Irland) in die Region USA West (Oregon) kopiert.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region
 us-west-2
```

Beispiel 2: Wenn Sie eine Standardregion festlegen und den Parameter Region weglassen, ist die Standardzielregion die Standardregion.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Einzelheiten zur API finden Sie unter CopySnapshot AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot von der Region EU (Irland) in die Region USA West (Oregon) kopiert.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region
 us-west-2
```

Beispiel 2: Wenn Sie eine Standardregion festlegen und den Parameter Region weglassen, ist die Standardzielregion die Standardregion.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- Einzelheiten zur API finden Sie unter CopySnapshot AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateCapacityReservation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCapacityReservation` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Kapazitätsreservierung

Im folgenden `create-capacity-reservation` Beispiel wird eine Kapazitätsreservierung in der `eu-west-1a` Availability Zone erstellt, in der Sie drei `t2.medium` Instances starten können, auf denen ein Linux/Unix Betriebssystem ausgeführt wird. Standardmäßig wird die Kapazitätsreservierung mit offenen Instance-Übereinstimmungskriterien und ohne Unterstützung für flüchtigen Speicher erstellt. Sie bleibt aktiv, bis Sie sie manuell stornieren.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type t2.medium \
    --instance-platform Linux/UNIX \
    --instance-count 3
```

Ausgabe:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "open",
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T09:27:35.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "t2.medium"
```

```
    }
}
```

Beispiel 2: Um eine Kapazitätsreservierung zu erstellen, die automatisch zu einem bestimmten Zeitpunkt endet date/time

Im folgenden `create-capacity-reservation` Beispiel wird eine Kapazitätsreservierung in der `eu-west-1a` Availability Zone erstellt, in der Sie drei `m5.large` Instances starten können, auf denen ein Linux/Unix Betriebssystem ausgeführt wird. Diese Kapazitätsreservierung endet automatisch am 08/31 /2019 um 23:59:59.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type m5.large \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --end-date-type limited \
    --end-date 2019-08-31T23:59:59Z
```

Ausgabe:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "limited",
        "AvailabilityZone": "eu-west-1a",
        "EndDate": "2019-08-31T23:59:59.000Z",
        "InstanceMatchCriteria": "open",
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T10:15:53.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "m5.large"
    }
}
```

Beispiel 3: So erstellen Sie eine Kapazitätsreservierung, die nur gezielte Instance-Starts akzeptiert

Im folgenden Beispiel für `create-capacity-reservation` wird eine Kapazitätsreservierung erstellt, die nur gezielte Instance-Starts akzeptiert.

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type m5.large \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --instance-match-criteria targeted
```

Ausgabe:

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "targeted",
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T10:21:57.000Z",
        "AvailableInstanceCount": 3,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 3,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": false,
        "InstanceType": "m5.large"
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer Kapazitätsreservierung im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. CreateCapacityReservationAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Kapazitätsreservierung mit den angegebenen Attributen erstellt

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Ausgabe:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Einzelheiten zur API finden Sie unter CreateCapacityReservation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Kapazitätsreservierung mit den angegebenen Attributen erstellt

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Ausgabe:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
```

```
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- Einzelheiten zur API finden Sie unter CreateCapacityReservation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateCustomerGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCustomerGateway` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein Kunden-Gateway

In diesem Beispiel wird ein Kunden-Gateway mit der angegebenen IP-Adresse für die externe Schnittstelle erstellt.

Befehl:

```
aws ec2 create-customer-gateway --type ipsec.1 --public-ip 12.1.2.3 --bgp-
asn 65534
```

Ausgabe:

```
{
    "CustomerGateway": {
        "CustomerGatewayId": "cgw-0e11f167",
        "IpAddress": "12.1.2.3",
        "State": "available",
        "Type": "ipsec.1",
        "BgpAsn": "65534"
    }
}
```

- Einzelheiten zur API finden Sie CreateCustomerGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway erstellt.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Einzelheiten zur API finden Sie unter CreateCustomerGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway erstellt.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- Einzelheiten zur API finden Sie unter CreateCustomerGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateDhcpOptions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDhcpOptions` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen DHCP-Optionssatz

Im folgenden Beispiel für `create-dhcp-options` wird ein DHCP-Optionssatz erstellt, der den Domainnamen, die Domainnamenserver und den NetBIOS-Knotentyp angibt.

```
aws ec2 create-dhcp-options \
    --dhcp-configuration \
        "Key=domain-name-servers,Values=10.2.5.1,10.2.5.2" \
        "Key=domain-name,Values=example.com" \
        "Key=netbios-node-type,Values=2"
```

Ausgabe:

```
{
    "DhcpOptions": {
        "DhcpConfigurations": [
            {
                "Key": "domain-name",
                "Values": [
                    {
                        "Value": "example.com"
                    }
                ]
            },
            {
                "Key": "domain-name-servers",
                "Values": [
                    {
                        "Value": "10.2.5.1"
                    },
                    {
                        "Value": "10.2.5.2"
                    }
                ]
            },
            {
                "Key": "netbios-node-type",
                "Values": [
                    {
```

```
                        "Value": "2"
                    }
                ]
            }
        ],
        "DhcpOptionsId": "dopt-06d52773eff4c55f3"
    }
}
```

- Einzelheiten zur API finden Sie CreateDhcpOptionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Satz erstellt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jede DHCP-Option verwenden New-Object , um sie zu erstellen.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Einzelheiten zur API finden Sie unter CreateDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Satz erstellt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jede DHCP-Option verwenden New-Object , um sie zu erstellen.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
```

```
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- Einzelheiten zur API finden Sie unter CreateDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateFlowLogs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFlowLogs` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie ein Flow-Protokoll

Im folgenden Beispiel für `create-flow-logs` wird ein Flow-Protokoll erstellt, das den gesamten abgelehnten Datenverkehr für die angegebene Netzwerkschnittstelle erfasst. Die Flow-Logs werden mithilfe der Berechtigungen in der angegebenen IAM-Rolle an eine Protokollgruppe in CloudWatch Logs übermittelt.

```
aws ec2 create-flow-logs \
    --resource-type NetworkInterface \
    --resource-ids eni-11223344556677889 \
    --traffic-type REJECT \
    --log-group-name my-flow-logs \
    --deliver-logs-permission-arn arn:aws:iam::123456789101:role/publishFlowLogs
```

Ausgabe:

```
{
    "ClientToken": "so0eNA2uSHUNlHI0S2cJ305GuIX1CezaRdGtexample",
    "FlowLogIds": [
        "fl-12345678901234567"
    ],
    "Unsuccessful": []
}
```

Weitere Informationen finden Sie unter VPC-Flow-Protokolle im Benutzerhandbuch für Amazon VPC.

Beispiel 2: So erstellen Sie ein Flow-Protokoll mit einem benutzerdefinierten Format

Im folgenden Beispiel für `create-flow-logs` wird ein Flow-Protokoll erstellt, das den gesamten Datenverkehr für die angegebene VPC erfasst und Flow-Protokolle an einen Amazon-S3-Bucket sendet. Der Parameter `--log-format` legt ein benutzerdefiniertes Format für die Flow-Protokolldatensätze fest. Zum Ausführen dieses Befehls unter Windows ändern Sie die einfachen Anführungszeichen (') in doppelte Anführungszeichen (").

```
aws ec2 create-flow-logs \
    --resource-type VPC \
    --resource-ids vpc-00112233344556677 \
    --traffic-type ALL \
    --log-destination-type s3 \
    --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
    --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr}
 ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr}
 ${pkt-dstaddr}'
```

Weitere Informationen finden Sie unter VPC-Flow-Protokolle im Benutzerhandbuch für Amazon VPC.

Beispiel 3: So erstellen Sie ein Flow-Protokoll mit einem maximalen Aggregationsintervall von einer Minute

Im folgenden Beispiel für `create-flow-logs` wird ein Flow-Protokoll erstellt, das den gesamten Datenverkehr für die angegebene VPC erfasst und Flow-Protokolle an einen Amazon-S3-Bucket sendet. Der Parameter `--max-aggregation-interval` gibt ein maximales Aggregationsintervall von 60 Sekunden (1 Minute) an.

```
aws ec2 create-flow-logs \
    --resource-type VPC \
    --resource-ids vpc-00112233344556677 \
    --traffic-type ALL \
    --log-destination-type s3 \
    --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
    --max-aggregation-interval 60
```

Weitere Informationen finden Sie unter VPC-Flow-Protokolle im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie CreateFlowLogsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein EC2-Flow-Protokoll für das Subnetz „subnet-1d234567“ im CloudWatch-Protokoll mit dem Namen „subnet1-log“ für den gesamten „REJECT“-Datenverkehr erstellt, wobei die Berechtigungen der Rolle „Admin“ verwendet werden

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Ausgabe:

```
ClientToken                                  FlowLogIds            Unsuccessful
-----------                                  ----------            ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Einzelheiten zur API finden Sie unter CreateFlowLogs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein EC2-Flow-Protokoll für das Subnetz „subnet-1d234567“ im CloudWatch-Protokoll mit dem Namen „subnet1-log“ für den gesamten „REJECT“-Datenverkehr erstellt, wobei die Berechtigungen der Rolle „Admin“ verwendet werden

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Ausgabe:

```
ClientToken                                  FlowLogIds            Unsuccessful
-----------                                  ----------            ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- Einzelheiten zur API finden Sie unter CreateFlowLogs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateImage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateImage` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um ein AMI aus einer EBS-backed Amazon-Instance zu erstellen

Im folgenden Beispiel für `create-image` wird ein AMI aus der angegebenen Instance erstellt.

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --description "An AMI for my server"
```

Ausgabe:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Weitere Informationen zur Angabe von Blockgerät-Zuweisungen für Ihr AMI finden Sie unter Angeben einer Blockgerät-Zuweisung für ein AMI im Benutzerhandbuch für Amazon EC2.

Beispiel 2: Um ein AMI aus einer EBS-backed Amazon-Instance ohne Neustart zu erstellen

Im folgenden Beispiel für `create-image` wird ein AMI erstellt und der Parameter --no-reboot festgelegt, sodass die Instance nicht neu gestartet wird, bevor das Image erstellt wird.

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --no-reboot
```

Ausgabe:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Weitere Informationen zur Angabe von Blockgerät-Zuweisungen für Ihr AMI finden Sie unter Angeben einer Blockgerät-Zuweisung für ein AMI im Benutzerhandbuch für Amazon EC2.

Beispiel 3: So taggen Sie ein AMI und Snapshots bei der Erstellung

Im folgenden Beispiel für `create-image` wird ein AMI erstellt. Das AMI und die Snapshots werden dann mit demselben Tag markiert: `cost-center=cc123`

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --tag-specifications "ResourceType=image,Tags=[{Key=cost-
center,Value=cc123}]" "ResourceType=snapshot,Tags=[{Key=cost-
center,Value=cc123}]"
```

Ausgabe:

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

Weitere Informationen zum Taggen Ihrer Ressourcen bei der Erstellung finden Sie unter Hinzufügen von Tags bei der Erstellung von Ressourcen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CreateImagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 versucht, die Instance vor dem Erstellen des Images ordnungsgemäß herunterzufahren, und startet die Instance nach Abschluss des Vorgangs neu.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Beispiel 2: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 erstellt das Image, ohne die

Instance herunterzufahren und neu zu starten. Daher kann die Integrität des Dateisystems auf dem erstellten Image nicht garantiert werden.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Beispiel 3: In diesem Beispiel wird ein AMI mit drei Volumes erstellt. Das erste Volume basiert auf einem Amazon-EBS-Snapshot. Bei dem zweiten Volume handelt es sich um ein leeres 100-GiB-Amazon-EBS-Volume. Das dritte Volume ist ein Instance-Speicher-Volume. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- Einzelheiten zur API finden Sie unter CreateImage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 versucht, die Instance vor dem Erstellen des Images ordnungsgemäß herunterzufahren, und startet die Instance nach Abschluss des Vorgangs neu.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Beispiel 2: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 erstellt das Image, ohne die Instance herunterzufahren und neu zu starten. Daher kann die Integrität des Dateisystems auf dem erstellten Image nicht garantiert werden.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Beispiel 3: In diesem Beispiel wird ein AMI mit drei Volumes erstellt. Das erste Volume basiert auf einem Amazon-EBS-Snapshot. Bei dem zweiten Volume handelt es sich um ein leeres 100-GiB-Amazon-EBS-Volume. Das dritte Volume ist ein Instance-Speicher-Volume. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- Einzelheiten zur API finden Sie unter CreateImage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateInstanceExportTask`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateInstanceExportTask` verwendet wird.

CLI

AWS CLI

So exportieren Sie eine Instance

Dieser Beispielbefehl erstellt eine Aufgabe zum Exportieren der Instance i-1234567890abcdef0 in den Amazon-S3-Bucket myexportbucket.

Befehl:

```
aws ec2 create-instance-export-task --description "RHEL5 instance" --
instance-id i-1234567890abcdef0 --target-environment vmware --export-to-s3-
task DiskImageFormat=vmdk,ContainerFormat=ova,S3Bucket=myexportbucket,S3Prefix=RHEL5
```

Ausgabe:

```
{
    "ExportTask": {
        "State": "active",
        "InstanceExportDetails": {
```

```
            "InstanceId": "i-1234567890abcdef0",
            "TargetEnvironment": "vmware"
        },
        "ExportToS3Task": {
            "S3Bucket": "myexportbucket",
            "S3Key": "RHEL5export-i-fh8sjjsq.ova",
            "DiskImageFormat": "vmdk",
            "ContainerFormat": "ova"
        },
        "Description": "RHEL5 instance",
        "ExportTaskId": "export-i-fh8sjjsq"
    }
}
```

- Einzelheiten zur API finden Sie CreateInstanceExportTaskin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine angehaltene Instance, **`i-0800b00a00EXAMPLE`**, als virtuelle Festplatte (VHD) in den S3-Bucket **`testbucket-export-instances-2019`** exportiert. Die Zielumgebung ist**`Microsoft`**, und der Regionsparameter wird hinzugefügt, weil sich die Instanz in der **`us-east-1`** Region befindet, während die AWS Standardregion des Benutzers nicht us-east-1 ist. Um den Status der Exportaufgabe abzurufen, kopieren Sie den **`ExportTaskId`**-Wert aus den Ergebnissen dieses Befehls und führen Sie anschließend **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`** aus.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
 -TargetEnvironment Microsoft -Region us-east-1
```

Ausgabe:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Einzelheiten zur API finden Sie unter CreateInstanceExportTask AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine angehaltene Instance, **`i-0800b00a00EXAMPLE`**, als virtuelle Festplatte (VHD) in den S3-Bucket **`testbucket-export-instances-2019`** exportiert. Die Zielumgebung ist**`Microsoft`**, und der Regionsparameter wird hinzugefügt, weil sich die Instanz in der **`us-east-1`** Region befindet, während die AWS Standardregion des Benutzers nicht us-east-1 ist. Um den Status der Exportaufgabe abzurufen, kopieren Sie den **`ExportTaskId`**-Wert aus den Ergebnissen dieses Befehls und führen Sie anschließend **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`** aus.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
 -TargetEnvironment Microsoft -Region us-east-1
```

Ausgabe:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- Einzelheiten zur API finden Sie unter CreateInstanceExportTask AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateInternetGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateInternetGateway` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So erstellen Sie ein Internet-Gateway

Im folgenden Beispiel für `create-internet-gateway` wird ein Internet-Gateway mit dem Tag `Name=my-igw` erstellt.

```
aws ec2 create-internet-gateway \
    --tag-specifications ResourceType=internet-gateway,Tags=[{Key=Name,Value=my-igw}]
```

Ausgabe:

```
{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-0d0fb496b3994d755",
        "OwnerId": "123456789012",
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-igw"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Internet-Gateways im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie CreateInternetGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Internet-Gateway erstellt.

```
New-EC2InternetGateway
```

Ausgabe:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Einzelheiten zur API finden Sie unter CreateInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Internet-Gateway erstellt.

```
New-EC2InternetGateway
```

Ausgabe:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- Einzelheiten zur API finden Sie unter CreateInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateKeyPair`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateKeyPair` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Einen Containerdienst für VM-Instanzen erstellen
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit Big-Data-Verarbeitungsclustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
```

```
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
```

```
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param keyFilePath: File path where the credentials are stored. Ignored if it
 is an empty string;
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
 &keyFilePath,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
 ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - "  << keyPairName << ". " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
                  keyPairName << std::endl;
        if (!keyFilePath.empty()) {
```

```
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                      keyFilePath << std::endl;
        }

    }

    return outcome.IsSuccess();

}
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Schlüsselpaar

In diesem Beispiel wird eine Schlüsselrolle mit dem Namen `MyKeyPair` erstellt.

Befehl:

```
aws ec2 create-key-pair --key-name MyKeyPair
```

Die Ausgabe ist eine ASCII-Version des privaten Schlüssels und des Schlüsselfingerabdrucks. Sie müssen den Schlüssel in einer Datei speichern.

Weitere Informationen finden Sie unter Verwenden von Schlüsselpaaren im Benutzerhandbuch zu die AWS -Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
               }
           } else {
               throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
           }
```

```
        });

        return responseFuture;
    }
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save
 to a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new CreateKeyPairCommand({
    KeyName: keyName,
  });

  try {
    const { KeyMaterial, KeyName } = await client.send(command);
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
      console.warn(`${caught.message}. Try another key name.`);
    } else {
      throw caught;
```

```
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- API-Details finden Sie CreateKeyPairin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Dieses Beispiel erstellt ein key pair und erfasst den privaten PEM-encoded RSA-Schlüssel in einer Datei mit dem angegebenen Namen. Wenn Sie verwenden PowerShell, muss die Kodierung auf ASCII eingestellt sein, um einen gültigen Schlüssel zu generieren. Weitere Informationen finden Sie unter Erstellen, Anzeigen und Löschen von Amazon EC2

EC2-Schlüsselpaaren (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Einzelheiten zur API finden Sie unter CreateKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel erstellt ein key pair und erfasst den privaten PEM-encoded RSA-Schlüssel in einer Datei mit dem angegebenen Namen. Wenn Sie verwenden PowerShell, muss die Kodierung auf ASCII eingestellt sein, um einen gültigen Schlüssel zu generieren. Weitere Informationen finden Sie unter Erstellen, Anzeigen und Löschen von Amazon EC2 EC2-Schlüsselpaaren (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- Einzelheiten zur API finden Sie unter CreateKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """
```

```
    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
 instance.
```

```
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair
```

- Einzelheiten zur API finden Sie CreateKeyPairin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
```

```
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end
```

```
# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'
```

```
  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rust-Implementierung, die create_key_pair des EC2-Clients aufruft und das zurückgegebene Material extrahiert.

```
    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
 String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
```

```
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
        Ok((info, material))
    }
```

Eine Funktion, die die create_key-Implementierung aufruft und den privaten PEM-Schlüssel sicher speichert.

```
    /// Creates a key pair that can be used to securely connect to an EC2
instance.
    /// The returned key pair contains private key information that cannot be
retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
```

```
    }
```

- Einzelheiten zur API finden Sie CreateKeyPairin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
                 " oo_result is returned for testing purposes. "
        MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateKeyPairin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
 String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeyPairin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateLaunchTemplate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLaunchTemplate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
   /// The launch template specifies a Bash script in its user data field that
runs after
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
```

```
            await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
 {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie CreateLaunchTemplatein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Startvorlage

Das folgende `create-launch-template`-Beispiel erstellt eine Startvorlage, die das Subnetz angibt, in dem die Instance gestartet werden soll, weist der Instance eine öffentliche IP-Adresse und eine IPv6-Adresse zu und erstellt ein Tag für die Instance.

```
aws ec2 create-launch-template \
    --launch-template-name TemplateForWebServer \
    --version-description WebVersion1 \
    --launch-template-data '{"NetworkInterfaces":
[{"AssociatePublicIpAddress":true,"DeviceIndex":0,"Ipv6AddressCount":1,"SubnetId":"subnet
[{"ResourceType":"instance","Tags":[{"Key":"purpose","Value":"webserver"}]}]}'
```

Ausgabe:

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-01238c059e3466abc",
        "LaunchTemplateName": "TemplateForWebServer",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-01-27T09:13:24.000Z"
    }
```

```
}
```

Weitere Informationen finden Sie unter Starten einer Instance von einer Startvorlage im Benutzerhandbuch zu Amazon Elastic Compute Cloud. Informationen zum Zitieren von JSON-formatted Parametern finden Sie unter Quoting Strings im AWS Command Line Interface User Guide.

Beispiel 2: So erstellen Sie eine Startvorlage für Amazon EC2 Auto Scaling

Im folgenden Beispiel für `create-launch-template` wird eine Startvorlage mit mehreren Tags und einer Blockgerät-Zuweisung erstellt, um beim Start einer Instance ein zusätzliches EBS-Volume anzugeben. Geben Sie einen Wert für `Groups` an, der den Sicherheitsgruppen für die VPC entspricht, in dem Ihre Auto-Scaling-Gruppe Instances starten soll. Geben Sie die VPC und Subnetze als Eigenschaften der Auto-Scaling-Gruppe an.

```
aws ec2 create-launch-template \
    --launch-template-name TemplateForAutoScaling \
    --version-description AutoScalingVersion1 \
    --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-7c227019,sg-903004f8"],"DeleteOnTermination":true}],"ImageId":"ami-
b42209de","InstanceType":"m4.large","TagSpecifications":
[{"ResourceType":"instance","Tags":[{"Key":"environment","Value":"production"},
{"Key":"purpose","Value":"webserver"}]},{"ResourceType":"volume","Tags":
[{"Key":"environment","Value":"production"},{"Key":"cost-
center","Value":"cc123"}]}],"BlockDeviceMappings":[{"DeviceName":"/dev/
sda1","Ebs":{"VolumeSize":100}}]}' --region us-east-1
```

Ausgabe:

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0123c79c33a54e0abc",
        "LaunchTemplateName": "TemplateForAutoScaling",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-04-30T18:16:06.000Z"
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer Startvorlage für eine Auto-Scaling-Gruppe im Benutzerhandbuch zu Amazon EC2 Auto Scaling. Informationen zum Zitieren von JSON-formatted Parametern finden Sie unter Quoting Strings im AWS Command Line Interface User Guide.

Beispiel 3: So erstellen Sie eine Startvorlage, die die Verschlüsselung von EBS-Volumes festlegt

Im folgenden Beispiel für `create-launch-template` wird eine Startvorlage erstellt, die verschlüsselte EBS-Volumes enthält, die aus einem unverschlüsselten Snapshot erstellt wurden. Außerdem werden die Volumes bei der Erstellung mit Tags versehen. Wenn die standardmäßige Verschlüsselung deaktiviert ist, müssen Sie die `"Encrypted"`-Option wie im folgenden Beispiel gezeigt angeben. Wenn Sie die `"KmsKeyId"`-Option verwenden, um einen kundenseitig verwalteten Schlüssel (Customer Managed Key; CMK) anzugeben, müssen Sie die `"Encrypted"`-Option auch dann angeben, wenn die standardmäßige Verschlüsselung aktiviert ist.

```
aws ec2 create-launch-template \
  --launch-template-name TemplateForEncryption \
  --launch-template-data file://config.json
```

Inhalt von `config.json`:

```
{
    "BlockDeviceMappings":[
        {
            "DeviceName":"/dev/sda1",
            "Ebs":{
                "VolumeType":"gp2",
                "DeleteOnTermination":true,
                "SnapshotId":"snap-066877671789bd71b",
                "Encrypted":true,
                "KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/abcd1234-
a123-456a-a12b-a123b4cd56ef"
            }
        }
    ],
    "ImageId":"ami-00068cd7555f543d5",
    "InstanceType":"c5.large",
    "TagSpecifications":[
        {
```

```
            "ResourceType":"volume",
            "Tags":[
                {
                    "Key":"encrypted",
                    "Value":"yes"
                }
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0d5bd51bcf8530abc",
        "LaunchTemplateName": "TemplateForEncryption",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2020-01-07T19:08:36.000Z"
    }
}
```

Weitere Informationen finden Sie unter Wiederherstellen eines Amazon-EBS-Volumes aus einem Snapshot und Standardmäßige Verschlüsselung im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie unter CreateLaunchTemplate AWS CLIBefehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const ssmClient = new SSMClient({});
```

```
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
```

- Einzelheiten zur API finden Sie CreateLaunchTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird eine Startvorlage erstellt, die ein Instance-Profil enthält, das der Instance bestimmte Berechtigungen gewährt, und ein Benutzerdaten-Bash-Skript, das nach dem Start auf der Instance ausgeführt wird.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
```

```
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
```

```
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]
```

```
            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template
```

- Einzelheiten zur API finden Sie CreateLaunchTemplatein AWS SDK for Python (Boto3) API Reference.

Verwendung von **`CreateNetworkAcl`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateNetworkAcl` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Netzwerk-ACL

In diesem Beispiel wird eine Netzwerk-ACL für die angegebene VPC erstellt.

Befehl:

```
aws ec2 create-network-acl --vpc-id vpc-a01106c2
```

Ausgabe:

```
{
    "NetworkAcl": {
        "Associations": [],
        "NetworkAclId": "acl-5fb85d36",
        "VpcId": "vpc-a01106c2",
        "Tags": [],
        "Entries": [
            {
                "CidrBlock": "0.0.0.0/0",
                "RuleNumber": 32767,
                "Protocol": "-1",
                "Egress": true,
                "RuleAction": "deny"
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "RuleNumber": 32767,
                "Protocol": "-1",
                "Egress": false,
                "RuleAction": "deny"
            }
        ],
        "IsDefault": false
    }
}
```

- Einzelheiten zur API finden Sie CreateNetworkAclin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Netzwerk-ACL für die angegebene VPC erstellt.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Ausgabe:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateNetworkAcl AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Netzwerk-ACL für die angegebene VPC erstellt.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Ausgabe:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateNetworkAcl AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateNetworkAclEntry`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateNetworkAclEntry` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Eintrag für eine Netzwerk-ACL

In diesem Beispiel wird ein Eintrag für die angegebene Netzwerk-ACL erstellt. Die Regel erlaubt eingehenden Datenverkehr von einer beliebigen IPv4-Adresse (0.0.0. 0/0) auf UDP-Port 53 (DNS) in ein beliebiges zugeordnetes Subnetz. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100 --protocol udp --port-range From=53,To=53 --cidr-block 0.0.0.0/0 --
rule-action allow
```

In diesem Beispiel wird eine Regel für die angegebene Netzwerk-ACL erstellt, die eingehenden Datenverkehr von einer beliebigen IPv6-Adresse (::/0) am TCP-Port 80 (HTTP) zulässt.

Befehl:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 120 --protocol tcp --port-range From=80,To=80 --ipv6-cidr-block ::/0 --
rule-action allow
```

- Einzelheiten zur API finden Sie CreateNetworkAclEntryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Eintrag für die angegebene Netzwerk-ACL erstellt. Die Regel erlaubt eingehenden Verkehr von überall (0.0.0. 0/0) auf UDP-Port 53 (DNS) in ein beliebiges zugeordnetes Subnetz.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -
RuleAction allow
```

- Einzelheiten zur API finden Sie unter CreateNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Eintrag für die angegebene Netzwerk-ACL erstellt. Die Regel erlaubt eingehenden Verkehr von überall (0.0.0. 0/0) auf UDP-Port 53 (DNS) in ein beliebiges zugeordnetes Subnetz.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -
RuleAction allow
```

- Einzelheiten zur API finden Sie unter CreateNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateNetworkInterface`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateNetworkInterface` verwendet wird.

CLI

AWS CLI

Beispiel 1: So geben Sie eine IPv4-Adresse für eine Netzwerkschnittstelle an

Im folgenden Beispiel für `create-network-interface` wird eine Netzwerkschnittstelle für das angegebene Subnetz mit der festgelegten primären IPv4-Adresse erstellt.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my network interface" \
    --groups sg-09dfba7ed20cda78b \
    --private-ip-address 10.0.8.17
```

Ausgabe:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my network interface",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-09dfba7ed20cda78b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "06:6a:0f:9a:49:37",
        "NetworkInterfaceId": "eni-0492b355f0cf3b3f8",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.17",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-17.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.17"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
}
```

Beispiel 2: So erstellen Sie eine Netzwerkschnittstelle mit einer IPv4- und einer IPv6-Adresse

Im folgenden Beispiel für `create-network-interface` wird eine Netzwerkschnittstelle für das angegebene Subnetz mit einer IPv4- und einer IPv6-Adresse erstellt, die von Amazon EC2 ausgewählt wurden.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my dual stack network interface" \
```

```
    --ipv6-address-count 1 \
    --groups sg-09dfba7ed20cda78b
```

Ausgabe:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my dual stack network interface",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-09dfba7ed20cda78b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [
            {
                "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7",
                "IsPrimaryIpv6": false
            }
        ],
        "MacAddress": "06:b8:68:d2:b2:2d",
        "NetworkInterfaceId": "eni-05da417453f9a84bf",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.18",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.18"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b",
        "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7"
    }
```

```
}
```

Beispiel 3: So erstellen Sie eine Netzwerkschnittstelle mit Konfigurationsoptionen für die Verbindungsverfolgung

Im folgenden Beispiel für `create-network-interface` wird eine Netzwerkschnittstelle erstellt und die Timeouts für die Verbindungsverfolgung im Leerlauf werden konfiguriert.

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --groups sg-02e57dbcfe0331c1b \
    --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60
```

Ausgabe:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "ConnectionTrackingConfiguration": {
            "TcpEstablishedTimeout": 86400,
            "UdpTimeout": 60
        },
        "Description": "",
        "Groups": [
            {
                "GroupName": "my-security-group",
                "GroupId": "sg-02e57dbcfe0331c1b"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "06:4c:53:de:6d:91",
        "NetworkInterfaceId": "eni-0c133586e08903d0b",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.94",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
                "PrivateIpAddress": "10.0.8.94"
            }
        ],
```

```
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
}
```

Beispiel 4: So erstellen Sie einen Elastic Fabric Adapter

Im folgenden Beispiel für `create-network-interface` wird ein EFA erstellt.

```
aws ec2 create-network-interface \
    --interface-type efa \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my efa" \
    --groups sg-02e57dbcfe0331c1b
```

Ausgabe:

```
{
    "NetworkInterface": {
        "AvailabilityZone": "us-west-2a",
        "Description": "my efa",
        "Groups": [
            {
                "GroupName": "my-efa-sg",
                "GroupId": "sg-02e57dbcfe0331c1b"
            }
        ],
        "InterfaceType": "efa",
        "Ipv6Addresses": [],
        "MacAddress": "06:d7:a4:f7:4d:57",
        "NetworkInterfaceId": "eni-034acc2885e862b65",
        "OwnerId": "123456789012",
        "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.180",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
```

```
                "PrivateIpAddress": "10.0.8.180"
            }
        ],
        "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-00a24d0d67acf6333",
        "TagSet": [],
        "VpcId": "vpc-02723a0feeeb9d57b"
    }
}
```

Weitere Informationen finden Sie unter Elastic Network-Schnittstellen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CreateNetworkInterfacein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle erstellt.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Ausgabe:

```
Association        :
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
```

```
Status              : pending
SubnetId            : subnet-1a2b3c4d
TagSet              : {}
VpcId               : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle erstellt.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Ausgabe:

```
Association        :
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreatePlacementGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePlacementGroup` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Platzierungsgruppe

Dieser Beispielbefehl erstellt eine Platzierungsgruppe mit dem angegebenen Namen.

Befehl:

```
aws ec2 create-placement-group --group-name my-cluster --strategy cluster
```

So erstellen Sie eine Partition-Placement-Gruppe

Dieser Beispielbefehl erstellt eine Partition-Placement-Gruppe mit dem Namen `HDFS-Group-A` und fünf Partitionen.

Befehl:

```
aws ec2 create-placement-group --group-name HDFS-Group-A --strategy partition --
partition-count 5
```

- Einzelheiten zur API finden Sie CreatePlacementGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Platzierungsgruppe mit dem angegebenen Namen erstellt.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Einzelheiten zur API finden Sie unter CreatePlacementGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Platzierungsgruppe mit dem angegebenen Namen erstellt.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- Einzelheiten zur API finden Sie unter CreatePlacementGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateRoute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRoute` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit Network Transit Gateways
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So erstellen Sie eine Route

Dieses Beispiel erstellt eine Route für die angegebene Routing-Tabelle. Die Route entspricht dem gesamten IPv4-Verkehr (`0.0.0.0/0`) und leitet ihn an das angegebene Internet-Gateway weiter. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-route --route-table-id rtb-22574640 --destination-cidr-
block 0.0.0.0/0 --gateway-id igw-c0a643a9
```

Dieser Beispielbefehl erstellt eine Route in der Routing-Tabelle rtb-g8ff4ea2. Die Route entspricht dem Verkehr für den IPv4 CIDR-Block 10.0.0. 0/16 und leitet es an die VPC-Peering-Verbindung pcx-111aaa22 weiter. Diese Route ermöglicht die Weiterleitung des Datenverkehrs an die Peer-VPC in der VPC-Peering-Verbindung. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-route --route-table-id rtb-g8ff4ea2 --destination-cidr-
block 10.0.0.0/16 --vpc-peering-connection-id pcx-1a2b3c4d
```

In diesem Beispiel wird eine Route in der angegebenen Routing-Tabelle erstellt, die dem gesamten IPv6-Datenverkehr (`::/0`) entspricht und ihn an das angegebene Internet-Gateway nur für ausgehenden Datenverkehr weiterleitet.

Befehl:

```
aws ec2 create-route --route-table-id rtb-dce620b8 --destination-ipv6-cidr-
block ::/0 --egress-only-internet-gateway-id eigw-01eadbd45ecd7943f
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz. CreateRouteAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle erstellt. Die Route entspricht dem gesamten Datenverkehr und leitet ihn an das angegebene Internet-Gateway weiter.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -
GatewayId igw-1a2b3c4d
```

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter CreateRoute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle erstellt. Die Route entspricht dem gesamten Datenverkehr und leitet ihn an das angegebene Internet-Gateway weiter.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -
GatewayId igw-1a2b3c4d
```

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter CreateRoute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateRouteTable`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRouteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So erstellen Sie eine Routing-Tabelle

Dieses Beispiel erstellt eine Routing-Tabelle für die angegebene VPC.

Befehl:

```
aws ec2 create-route-table --vpc-id vpc-a01106c2
```

Ausgabe:

```
{
    "RouteTable": {
        "Associations": [],
        "RouteTableId": "rtb-22574640",
        "VpcId": "vpc-a01106c2",
        "PropagatingVgws": [],
        "Tags": [],
```

```
        "Routes": [
            {
                "GatewayId": "local",
                "DestinationCidrBlock": "10.0.0.0/16",
                "State": "active"
            }
        ]
    }
}
```

- Einzelheiten zur API finden Sie CreateRouteTablein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Routing-Tabelle für die angegebene VPC erstellt.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Ausgabe:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Routing-Tabelle für die angegebene VPC erstellt.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Ausgabe:

```
Associations    : {}
PropagatingVgws : {}
```

```
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
```

```
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
```

```
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
```

```
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts 'Route table created and associated.'
  else
    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateRouteTablein der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`CreateSecurityGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSecurityGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Eine relationale Datenbankinstanz erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen

- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and
 description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
```

```
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
```

```
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the
 operation fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security
 group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  # Validate the input parameters
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n
 parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
```

```
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create a security group.
/*!
  \param groupName: A security group name.
  \param description: A description.
  \param vpcID: A virtual private cloud (VPC) ID.
  \param[out] groupIDResult: A string to receive the group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                      const Aws::String &description,
                                      const Aws::String &vpcID,
                                      Aws::String &groupIDResult,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
    request.SetVpcId(vpcID);

    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
            ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
```

```
    std::cout << "Successfully created security group named " << groupName <<
              std::endl;


    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine Sicherheitsgruppe zu erstellen für EC2-Classic

In diesem Beispiel wird eine Sicherheitsgruppe mit dem Namen `MySecurityGroup` erstellt.

Befehl:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group"
```

Ausgabe:

```
{
    "GroupId": "sg-903004f8"
}
```

Um eine Sicherheitsgruppe zu erstellen für EC2-VPC

In diesem Beispiel wird eine Sicherheitsgruppe mit dem Namen `MySecurityGroup` für die angegebene VPC erstellt.

Befehl:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group" --vpc-id vpc-1a2b3c4d
```

Ausgabe:

```
{
    "GroupId": "sg-903004f8"
}
```

Weitere Informationen finden Sie unter Verwenden von Sicherheitsgruppen im Benutzerhandbuch zu die AWS -Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security
group
    * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
```

```
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
```

```
                    }
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
  const client = new EC2Client({});
  const command = new CreateSecurityGroupCommand({
    // Up to 255 characters in length. Cannot start with sg-.
    GroupName: groupName,
    // Up to 255 characters in length.
    Description: description,
  });

  try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
```

```
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
```

```
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group
 $groupNameVal")
        return resp.groupId
    }
}
```

- API-Details finden Sie CreateSecurityGroupin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird eine Sicherheitsgruppe für die angegebene VPC erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group" -VpcId vpc-12345678
```

Ausgabe:

```
sg-12345678
```

Beispiel 2: In diesem Beispiel wird eine Sicherheitsgruppe für erstellt EC2-Classic.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group"
```

Ausgabe:

```
sg-45678901
```

- Einzelheiten zur API finden Sie unter CreateSecurityGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Sicherheitsgruppe für die angegebene VPC erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group" -VpcId vpc-12345678
```

Ausgabe:

```
sg-12345678
```

Beispiel 2: In diesem Beispiel wird eine Sicherheitsgruppe für EC2-Classic erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group"
```

Ausgabe:

```
sg-45678901
```

- Einzelheiten zur API finden Sie unter CreateSecurityGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
a different name."
                )
            raise
        else:
            return self.security_group
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
```

```
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol
 " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
```

```
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group
 permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
 perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
 perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
  print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

  print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

  print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
 perm.ip_ranges.count.positive?
  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
```

```
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:        #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?

  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
```

```
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http,
 to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
 description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
 from_port_http, to_port_http, cidr_ip_range_http)
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh,
 from_port_ssh, to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
 security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
```

```
  security_group_id = create_security_group(ec2_client, group_name, description,
 vpc_id)
  puts 'Could not create security group. Skipping this step.' if
 security_group_id == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
 to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
 ip_protocol, from_port, to_port,
                                                cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
end

def display_help
  puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
    'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
    'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp',
 '80', '80',
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
```

```
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->createsecuritygroup(                " oo_result is
 returned for testing purposes. "
          iv_description = 'Security group example'
          iv_groupname = iv_security_group_name
          iv_vpcid = iv_vpc_id ).
        MESSAGE 'Security group created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description
 groupDescription: String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
            print("*** Error creating the security group:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`CreateSnapshot`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSnapshot` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

So erstellen Sie einen Snapshot

Dieser Beispielbefehl erstellt ein Snapshot des Volumes mit der Volume-ID `vol-1234567890abcdef0` und einer kurzen Beschreibung zur Identifizierung des Snapshots.

Befehl:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is
 my root volume snapshot"
```

Ausgabe:

```
{
    "Description": "This is my root volume snapshot",
    "Tags": [],
    "Encrypted": false,
    "VolumeId": "vol-1234567890abcdef0",
    "State": "pending",
    "VolumeSize": 8,
    "StartTime": "2018-02-28T21:06:01.000Z",
    "Progress": "",
    "OwnerId": "012345678910",
    "SnapshotId": "snap-066877671789bd71b"
}
```

So erstellen Sie einen Snapshot mit Tags

Dieser Beispielbefehl erstellt einen Snapshot und wendet zwei Tags an: purpose=prod und costcenter=123.

Befehl:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0
 --description 'Prod backup' --tag-specifications
 'ResourceType=snapshot,Tags=[{Key=purpose,Value=prod},
{Key=costcenter,Value=123}]'
```

Ausgabe:

```
{
    "Description": "Prod backup",
    "Tags": [
        {
            "Value": "prod",
            "Key": "purpose"
        },
        {
            "Value": "123",
            "Key": "costcenter"
        }
     ],
     "Encrypted": false,
     "VolumeId": "vol-1234567890abcdef0",
     "State": "pending",
     "VolumeSize": 8,
     "StartTime": "2018-02-28T21:06:06.000Z",
     "Progress": "",
     "OwnerId": "012345678910",
     "SnapshotId": "snap-09ed24a70bc19bbe4"
 }
```

- Einzelheiten zur API finden Sie CreateSnapshotin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Snapshot des angegebenen Volumes erstellt.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- Einzelheiten zur API finden Sie unter CreateSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Snapshot des angegebenen Volumes erstellt.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
```

```
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- Einzelheiten zur API finden Sie unter CreateSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateSpotDatafeedSubscription`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSpotDatafeedSubscription` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Daten-Feed für eine Spot Instance

Im folgenden Beispiel für `create-spot-datafeed-subscription` wird ein Daten-Feed für eine Spot Instance erstellt.

```
aws ec2 create-spot-datafeed-subscription \
    --bucket amzn-s3-demo-bucket \
    --prefix spot-data-feed
```

Ausgabe:

```
{
    "SpotDatafeedSubscription": {
        "Bucket": "amzn-s3-demo-bucket",
        "OwnerId": "123456789012",
        "Prefix": "spot-data-feed",
        "State": "Active"
    }
}
```

Der Daten-Feed wird im Amazon-S3-Bucket gespeichert, den Sie angegeben haben. Die Dateinamen für diesen Daten-Feed haben das folgende Format.

```
amzn-s3-demo-bucket.s3.amazonaws.com/spot-data-feed/123456789012.YYYY-MM-DD-
HH.n.abcd1234.gz
```

Weitere Informationen finden Sie unter Daten-Feed für eine Spot Instance im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CreateSpotDatafeedSubscriptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Daten-Feed für eine Spot Instance erstellt.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Einzelheiten zur API finden Sie unter CreateSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Daten-Feed für eine Spot Instance erstellt.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Einzelheiten zur API finden Sie unter CreateSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateSubnet`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSubnet` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit Network Transit Gateways
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

Beispiel 1: So erstellen Sie ein Subnetz nur mit einem IPv4-CIDR-Block

Das folgende `create-subnet`-Beispiel erstellt ein Subnetz in der angegebenen VPC mit dem angegebenen IPv4-CIDR-Block.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --cidr-block 10.0.0.0/24 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-
subnet}]
```

Ausgabe:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.0.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0e99b93155EXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
```

```
        "OwnerId": "123456789012",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv4-only-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0e99b93155EXAMPLE"
    }
}
```

Beispiel 2: So erstellen Sie ein Subnetz mit sowohl IPv4- als auch IPv6-CIDR-Blöcken

Das folgende `create-subnet`-Beispiel erstellt ein Subnetz in der angegebenen VPC mit den angegebenen IPv4- und IPv6-CIDR-Blöcken.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --cidr-block 10.0.0.0/24 \
    --ipv6-cidr-block 2600:1f16:cfe:3660::/64 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-ipv6-
subnet}]
```

Ausgabe:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.0.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0736441d38EXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
        "OwnerId": "123456789012",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [
            {
```

```
                "AssociationId": "subnet-cidr-assoc-06c5f904499fcc623",
                "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                }
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv4-ipv6-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0736441d38EXAMPLE"
    }
}
```

Beispiel 3: So erstellen Sie ein Subnetz nur mit einem IPv6-CIDR-Block

Das folgende `create-subnet`-Beispiel erstellt ein Subnetz in der angegebenen VPC mit dem angegebenen IPv6-CIDR-Block.

```
aws ec2 create-subnet \
    --vpc-id vpc-081ec835f3EXAMPLE \
    --ipv6-native \
    --ipv6-cidr-block 2600:1f16:115:200::/64 \
    --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv6-only-
subnet}]
```

Ausgabe:

```
{
    "Subnet": {
        "AvailabilityZone": "us-west-2a",
        "AvailabilityZoneId": "usw2-az2",
        "AvailableIpAddressCount": 0,
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-03f720e7deEXAMPLE",
        "VpcId": "vpc-081ec835f3EXAMPLE",
        "OwnerId": "123456789012",
```

```
        "AssignIpv6AddressOnCreation": true,
        "Ipv6CidrBlockAssociationSet": [
            {
                "AssociationId": "subnet-cidr-assoc-01ef639edde556709",
                "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                }
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-ipv6-only-subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-03f720e7deEXAMPLE"
    }
}
```

Weitere Informationen finden Sie unter VPCs und Subnetze im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie CreateSubnetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Subnetz mit dem angegebenen CIDR erstellt.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
```

```
Tag                     : {}
VpcId                   : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateSubnet AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Subnetz mit dem angegebenen CIDR erstellt.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
Tag                     : {}
VpcId                   : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateSubnet AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
```

```
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
```

```
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-west-2a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
```

```
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,
    cidr_block,
    availability_zone,
    tag_key,
    tag_value
  )
    puts 'Subnet created and tagged.'
  else
    puts 'Subnet not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateSubnetin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`CreateTags`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTags` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add or overwrite only the specified tags for the specified Amazon Elastic
 Compute Cloud (Amazon EC2) resource or resources.
/*!
```

```
  \param resources: The resources for the tags.
  \param tags: Vector of tags.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
 ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTagsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 6: So fügen Sie einer Ressource ein Tag hinzu

Im folgenden Beispiel für `create-tags` wird das Tag `Stack=production` zu dem angegebenen Image hinzugefügt oder ein vorhandenes Tag für das AMI überschrieben, wobei der Tag-Schlüssel `Stack` ist.

```
aws ec2 create-tags \
    --resources ami-1234567890abcdef0 \
```

```
    --tags Key=Stack,Value=production
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So fügen Sie Tags mehreren Ressourcen hinzu

Im folgenden Beispiel für `create-tags` werden zwei Tags (Markierungen) für ein AMI und eine Instance hinzugefügt (oder überschreibt). Eines der Tags hat einen Schlüssel (`webserver`), aber keinen Wert (Wert ist auf eine leere Zeichenfolge festgelegt). Das andere Tag hat einen Schlüssel (`stack`) und einen Wert (`Production`).

```
aws ec2 create-tags \
    --resources ami-1a2b3c4d i-1234567890abcdef0 \
    --tags Key=webserver,Value=   Key=stack,Value=Production
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: So fügen Sie Tags mit Sonderzeichen hinzu

In den folgenden Beispielen für `create-tags` wird das Tag `[Group]=test` für eine Instance hinzugefügt. Die eckigen Klammern ([und]) sind Sonderzeichen und müssen mit Escape-Zeichen versehen werden. In den folgenden Beispielen wird auch das Zeilenfortsetzungszeichen verwendet, das für jede Umgebung geeignet ist.

Wenn Sie Windows verwenden, schließen Sie das Element, das Sonderzeichen enthält, in doppelte Anführungszeichen (") ein und stellen Sie jedem doppelten Anführungszeichen wie folgt einen umgekehrten Schrägstrich (\) voran.

```
aws ec2 create-tags ^
    --resources i-1234567890abcdef0 ^
    --tags Key=\"[Group]\",Value=test
```

Wenn Sie Windows verwenden PowerShell, setzen Sie den Wert, der Sonderzeichen enthält, in doppelte Anführungszeichen („), stellen Sie jedem doppelten Anführungszeichen einen umgekehrten Schrägstrich (\) voran und setzen Sie dann die gesamte Schlüssel- und Wertstruktur wie folgt in einfache Anführungszeichen (').

```
aws ec2 create-tags `
    --resources i-1234567890abcdef0 `
```

```
    --tags 'Key=\"[Group]\",Value=test'
```

Wenn Sie Linux oder OS X verwenden, schließen Sie das Element mit den Sonderzeichen mit doppelten Anführungszeichen (") ein und dann die gesamte Schlüssel- und Wertstruktur mit einfachen Anführungszeichen (’) wie folgt.

```
aws ec2 create-tags \
    --resources i-1234567890abcdef0 \
    --tags 'Key="[Group]",Value=test'
```

Weitere Informationen finden Sie unter Markieren Ihrer Amazon-EC2-Ressourcen mit Tags im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie CreateTagsin AWS CLI der Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebenen Ressource ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'my'TagValue. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags aktualisiert oder der angegebenen Ressource hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Einzelheiten zur API finden Sie unter CreateTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Ressource ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist TagValue 'my'. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags aktualisiert oder der angegebenen Ressource hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- Einzelheiten zur API finden Sie unter CreateTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird nach dem Erstellen einer Instance das Tag „Name“ angewendet.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
```

```
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- Einzelheiten zur API finden Sie CreateTagsin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`CreateVolume`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVolume` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Blockspeicher-Volumes erstellen und verwalten
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

So erstellen Sie ein leeres Allzweck-SSD-Volume (gp2)

Im folgenden Beispiel für `create-volume` wird ein Allzweck-SSD-Volume (gp2) mit 80 GiB in der angegebenen Availability Zone erstellt. Beachten Sie, dass die aktuelle Region `us-east-1` sein muss. Sie können auch den Parameter `--region` hinzufügen, um die Region für den Befehl anzugeben.

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --availability-zone us-east-1a
```

Ausgabe:

```
{
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": false,
    "VolumeType": "gp2",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
    "Iops": 240,
    "SnapshotId": "",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 80
}
```

Wenn Sie keinen Volume-Typ angeben, wird der Standard-Volume-Typ gp2 verwendet.

```
aws ec2 create-volume \
    --size 80 \
    --availability-zone us-east-1a
```

Beispiel 2: So erstellen Sie ein bereitgestelltes IOPS-SSD-Volume (io1) aus einem Snapshot

Im folgenden Beispiel für `create-volume` wird mithilfe des angegebenen Snapshots ein bereitgestelltes IOPS-SSD-Volume (io1) mit 1 000 bereitgestellten IOPS in der angegebenen Availability Zone erstellt.

```
aws ec2 create-volume \
    --volume-type io1 \
    --iops 1000 \
    --snapshot-id snap-066877671789bd71b \
    --availability-zone us-east-1a
```

Ausgabe:

```
{
```

```
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": false,
    "VolumeType": "io1",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
    "Iops": 1000,
    "SnapshotId": "snap-066877671789bd71b",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 500
}
```

Beispiel 3: So erstellen Sie ein verschlüsseltes Volume

Im folgenden Beispiel für `create-volume` wird ein verschlüsseltes Volume erstellt, das den Standard-CMK für die EBS-Verschlüsselung verwendet. Wenn die standardmäßige Verschlüsselung deaktiviert ist, müssen Sie den Parameter `--encrypted` wie folgt angeben.

```
aws ec2 create-volume \
    --size 80 \
    --encrypted \
    --availability-zone us-east-1a
```

Ausgabe:

```
{
    "AvailabilityZone": "us-east-1a",
    "Tags": [],
    "Encrypted": true,
    "VolumeType": "gp2",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "creating",
    "Iops": 240,
    "SnapshotId": "",
    "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "Size": 80
}
```

Wenn die Verschlüsselung standardmäßig aktiviert ist, erstellt der folgende Beispielbefehl ein verschlüsseltes Volume, auch ohne den Parameter `--encrypted`.

```
aws ec2 create-volume \
```

```
    --size 80 \
    --availability-zone us-east-1a
```

Wenn Sie den Parameter `--kms-key-id` verwenden, um einen vom Kunden verwalteten Schlüssel anzugeben, müssen Sie den Parameter `--encrypted` auch dann angeben, wenn die standardmäßige Verschlüsselung aktiviert ist.

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --encrypted \
    --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \
    --availability-zone us-east-1a
```

Beispiel 4: So erstellen Sie ein Volume mit Tags

Im folgenden Beispiel für `create-volume` wird ein Volume erstellt und zwei Tags werden hinzugefügt.

```
aws ec2 create-volume \
    --availability-zone us-east-1a \
    --volume-type gp2 \
    --size 80 \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=purpose,Value=production},{Key=cost-
center,Value=cc123}]'
```

- Einzelheiten zur API finden Sie CreateVolumein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Volume erstellt.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2a
```

```
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
```

Beispiel 2: Diese Beispielanforderung erstellt ein Volume und versieht es mit einem Tag mit dem Schlüssel „stack“ und dem Wert „production.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Einzelheiten zur API finden Sie unter CreateVolume AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume erstellt.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
State            : creating
Tags             : {}
```

```
VolumeId         : vol-12345678
VolumeType       : gp2
```

Beispiel 2: Diese Beispielanforderung erstellt ein Volume und versieht es mit einem Tag mit dem Schlüssel „stack“ und dem Wert „production.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- Einzelheiten zur API finden Sie unter CreateVolume AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateVpc`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVpc` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit Network Transit Gateways
- Virtuelles privates Netzwerk mit IP-Adressverwaltung
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine VPC

Im folgenden Beispiel für `create-vpc` wird eine VPC mit dem angegebenen IPv4-CIDR-Block und einem Name-Tag erstellt.

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --tag-specifications ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

Ausgabe:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-5EXAMPLE",
        "State": "pending",
        "VpcId": "vpc-0a60eb65b4EXAMPLE",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-07501b79ecEXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": MyVpc"
            }
        ]
    }
}
```

Beispiel 2: So erstellen Sie eine VPC mit dedizierter Tenancy

Im folgenden Beispiel für `create-vpc` wird eine VPC mit dem angegebenen IPv4-CIDR-Block und einer dedizierten Tenancy erstellt.

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --instance-tenancy dedicated
```

Ausgabe:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-19edf471",
        "State": "pending",
        "VpcId": "vpc-0a53287fa4EXAMPLE",
        "OwnerId": "111122223333",
        "InstanceTenancy": "dedicated",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-00b24cc1c2EXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false
    }
}
```

Beispiel 3: So erstellen Sie eine VPC mit einem IPv6-CIDR-Block

Im folgenden `create-vpc` Beispiel wird eine VPC mit einem Amazon-provided IPv6-CIDR-Block erstellt.

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --amazon-provided-ipv6-cidr-block
```

Ausgabe:

```
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-dEXAMPLE",
        "State": "pending",
        "VpcId": "vpc-0fc5e3406bEXAMPLE",
        "OwnerId": "123456789012",
```

```
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-068432c60bEXAMPLE",
                "Ipv6CidrBlock": "",
                "Ipv6CidrBlockState": {
                    "State": "associating"
                },
                "Ipv6Pool": "Amazon",
                "NetworkBorderGroup": "us-west-2"
            }
        ],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0669f8f9f5EXAMPLE",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false
    }
}
```

Beispiel 4: So erstellen Sie eine VPC mit einer CIDR aus einem IPAM-Pool

Im folgenden Beispiel für `create-vpc` wird eine VPC mit CIDR aus einem Amazon VPC IP Address Manager (IPAM)-Pool erstellt.

Linux und macOS:

```
aws ec2 create-vpc \
    --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 \
    --tag-specifications
 ResourceType=vpc,Tags='[{Key=Environment,Value="Preprod"},
{Key=Owner,Value="Build Team"}]'
```

Windows:

```
aws ec2 create-vpc ^
    --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 ^
```

```
    --tag-specifications
 ResourceType=vpc,Tags=[{Key=Environment,Value="Preprod"},{Key=Owner,Value="Build
 Team"}]
```

Ausgabe:

```
{
    "Vpc": {
        "CidrBlock": "10.0.1.0/24",
        "DhcpOptionsId": "dopt-2afccf50",
        "State": "pending",
        "VpcId": "vpc-010e1791024eb0af9",
        "OwnerId": "123456789012",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0a77de1d803226d4b",
                "CidrBlock": "10.0.1.0/24",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Environment",
                "Value": "Preprod"
            },
            {
                "Key": "Owner",
                "Value": "Build Team"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Eine VPC erstellen, die einen IPAM-Pool CIDR verwendet im Benutzerhandbuch zu Amazon VPC IPAM.

- Einzelheiten zur API finden Sie unter Befehlsreferenz CreateVpc.AWS CLI

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $cidr
     * @return array
     */
    public function createVpc(string $cidr): array
    {
        try {
            $result = $this->ec2Client->createVpc([
                "CidrBlock" => $cidr,
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie CreateVpcin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine VPC mit dem angegebenen CIDR erstellt. Amazon VPC erstellt für die VPC außerdem Folgendes: einen standardmäßigen DHCP-Optionssatz, eine Haupt-Routing-Tabelle und eine Standard-Netzwerk-ACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateVpc AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine VPC mit dem angegebenen CIDR erstellt. Amazon VPC erstellt für die VPC außerdem Folgendes: einen standardmäßigen DHCP-Optionssatz, eine Haupt-Routing-Tabelle und eine Standard-Netzwerk-ACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- Einzelheiten zur API finden Sie unter CreateVpc AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, cidr_block: str) -> str:
        """
        Creates a new Amazon VPC with the specified CIDR block.

        :param cidr_block: The CIDR block for the new VPC, such as '10.0.0.0/16'.
        :return: The ID of the new VPC.
```

```
        """
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr_block)
            vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[vpc_id])
            return vpc_id
        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateVpcin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
```

```
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
```

```
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateVpcin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_cidr_block = '10.0.0.0/16'
    TRY.
        oo_result = lo_ec2->createvpc( iv_cidrblock = iv_cidr_block ).
  " oo_result is returned for testing purposes. "
        DATA(lv_vpc_id) = oo_result->get_vpc( )->get_vpcid( ).
        MESSAGE 'Created VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateVpcin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateVpcEndpoint`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVpcEndpoint` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Gateway-Endpunkt

Im folgenden Beispiel für `create-vpc-endpoint` wird ein Gateway-VPC-Endpunkt zwischen der VPC `vpc-1a2b3c4d` und Amazon S3 in der Region `us-east-1` erstellt und die Routing-Tabelle `rtb-11aa22bb` mit dem Endpunkt verknüpft.

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --service-name com.amazonaws.us-east-1.s3 \
    --route-table-ids rtb-11aa22bb
```

Ausgabe:

```
{
    "VpcEndpoint": {
        "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":
\"\",\"Effect\":\"Allow\",\"Principal\":\"\*\",\"Action\":\"\*\",\"Resource\":
\"\*\"}]}",
        "VpcId": "vpc-1a2b3c4d",
        "State": "available",
        "ServiceName": "com.amazonaws.us-east-1.s3",
        "RouteTableIds": [
            "rtb-11aa22bb"
        ],
        "VpcEndpointId": "vpc-1a2b3c4d",
        "CreationTimestamp": "2015-05-15T09:40:50Z"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Gateway-Endpunkts im AWS PrivateLink Benutzerhandbuch.

Beispiel 2: So erstellen Sie einen Schnittstellenendpunkt

Im folgenden Beispiel für `create-vpc-endpoint` wird ein VPC-Schnittstellenendpunkt zwischen der VPC `vpc-1a2b3c4d` und Amazon S3 in der Region `us-east-1` erstellt. Der Befehl erstellt den Endpunkt im Subnetz `subnet-1a2b3c4d`, ordnet ihn der Sicherheitsgruppe `sg-1a2b3c4d` zu und fügt ein Tag mit dem Schlüssel „Service“ und dem Wert „S3" hinzu.

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.s3 \
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d \
    --tag-specifications ResourceType=vpc-endpoint,Tags=[{Key=service,Value=S3}]
```

Ausgabe:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-1a2b3c4d5e6f1a2b3",
        "VpcEndpointType": "Interface",
```

```
        "VpcId": "vpc-1a2b3c4d",
        "ServiceName": "com.amazonaws.us-east-1.s3",
        "State": "pending",
        "RouteTableIds": [],
        "SubnetIds": [
            "subnet-1a2b3c4d"
        ],
        "Groups": [
            {
                "GroupId": "sg-1a2b3c4d",
                "GroupName": "default"
            }
        ],
        "PrivateDnsEnabled": false,
        "RequesterManaged": false,
        "NetworkInterfaceIds": [
            "eni-0b16f0581c8ac6877"
        ],
        "DnsEntries": [
            {
                "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg.s3.us-
east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg-us-east-1c.s3.us-
east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            }
        ],
        "CreationTimestamp": "2021-03-05T14:46:16.030000+00:00",
        "Tags": [
            {
                "Key": "service",
                "Value": "S3"
            }
        ],
        "OwnerId": "123456789012"
    }
}
```

Weitere Informationen finden Sie im Benutzerhandbuch unter Erstellen eines Schnittstellen-VPC-Endpunkts.AWS PrivateLink

Beispiel 3: So erstellen Sie einen Endpunkt für Gateway Load Balancer

Im folgenden Beispiel für `create-vpc-endpoint` wird ein Endpunkt für Gateway Load Balancer zwischen der VPC `vpc-111122223333aabbc` und einem Service erstellt, der mithilfe eines Gateway Load Balancers konfiguriert ist.

```
aws ec2 create-vpc-endpoint \
    --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
    --vpc-endpoint-type GatewayLoadBalancer \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445
```

Ausgabe:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-aabbaabbaabbaabba",
        "VpcEndpointType": "GatewayLoadBalancer",
        "VpcId": "vpc-111122223333aabbc",
        "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",
        "State": "pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "RequesterManaged": false,
        "NetworkInterfaceIds": [
            "eni-0101012020303040 5"
        ],
        "CreationTimestamp": "2020-11-11T08:06:03.522Z",
        "OwnerId": "123456789012"
    }
}
```

Weitere Informationen finden Sie unter Gateway Load Balancer-Endpoints im AWS PrivateLink Benutzerhandbuch.

Beispiel 4: So erstellen Sie einen Ressourcenendpunkt

Im folgenden Beispiel für `create-vpc-endpoint` wird ein Ressourcenendpunkt erstellt.

```
aws ec2 create-vpc-endpoint \
```

```
    --vpc-endpoint-type Resource \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445 \
    --resource-configuration-arn arn:aws:vpc-lattice-us-
east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432
```

Ausgabe:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-00939a7ed9EXAMPLE",
        "VpcEndpointType": "Resource",
        "VpcId": "vpc-111122223333aabbc",
        "State": "Pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "Groups": [
            {
                "GroupId": "sg-03e2f15fbfc09b000",
                "GroupName": "default"
            }
        ],
        "IpAddressType": "IPV4",
        "PrivateDnsEnabled": false,
        "CreationTimestamp": "2025-02-06T23:38:49.525000+00:00",
        "Tags": [],
        "OwnerId": "123456789012",
        "ResourceConfigurationArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432"
    }
}
```

Weitere Informationen finden Sie im Benutzerhandbuch unter Ressourcenendpunkte.AWS PrivateLink

Beispiel 5: So erstellen Sie einen Service-Netzwerkendpunkt

Im folgenden Beispiel für `create-vpc-endpoint` wird ein Service-Netzwerkendpunkt erstellt.

```
aws ec2 create-vpc-endpoint \
```

```
    --vpc-endpoint-type ServiceNetwork \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445 \
    --service-network-arn arn:aws:vpc-lattice:us-
east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0 \
    --security-group-ids sg-0123456789012abcd
```

Ausgabe:

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-0f00567fa8EXAMPLE",
        "VpcEndpointType": "ServiceNetwork",
        "VpcId": "vpc-111122223333aabbc",
        "State": "Pending",
        "SubnetIds": [
            "subnet-0011aabbcc2233445"
        ],
        "Groups": [
            {
                "GroupId": "sg-0123456789012abcd",
                "GroupName": "my-security-group"
            }
        ],
        "IpAddressType": "IPV4",
        "PrivateDnsEnabled": false,
        "CreationTimestamp": "2025-02-06T23:44:20.449000+00:00",
        "Tags": [],
        "OwnerId": "123456789012",
        "ServiceNetworkArn": "arn:aws:vpc-lattice:us-
east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0"
    }
}
```

Weitere Informationen finden Sie im AWS PrivateLink Benutzerhandbuch unter Service-Netzwerkendpunkte.

- Einzelheiten zur API finden Sie CreateVpcEndpointin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie CreateVpcEndpointin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer VPC-Endpunkt für den Service com.amazonaws.eu-west-1.s3 in der VPC vpc-0fc1ff23f45b678eb erstellt

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Ausgabe:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Einzelheiten zur API finden Sie unter CreateVpcEndpoint AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer VPC-Endpunkt für den Service com.amazonaws.eu-west-1.s3 in der VPC vpc-0fc1ff23f45b678eb erstellt

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Ausgabe:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- Einzelheiten zur API finden Sie unter CreateVpcEndpoint AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create_vpc_endpoint(
        self, vpc_id: str, service_name: str, route_table_ids: list[str]
    ) -> Dict[str, Any]:
        """
        Creates a new VPC endpoint for the specified service and associates it
 with the specified route tables.
```

```
        :param vpc_id: The ID of the VPC to create the endpoint in.
        :param service_name: The name of the service to create the endpoint for.
        :param route_table_ids: A list of IDs of the route tables to associate
with the endpoint.
        :return: A dictionary representing the newly created VPC endpoint.
        """
        try:
            response = self.ec2_client.create_vpc_endpoint(
                VpcId=vpc_id,
                ServiceName=service_name,
                RouteTableIds=route_table_ids,
            )
            return response["VpcEndpoint"]
        except ClientError as err:
            logger.error(
                "Couldn't create VPC endpoint for service %s. Here's why: %s:
%s",
                service_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateVpcEndpointin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_vpc_id = 'vpc-abc123'
    " iv_service_name = 'com.amazonaws.region.service'
```

```
    TRY.
        oo_result = lo_ec2->createvpcendpoint(             " oo_result is
 returned for testing purposes. "
          iv_vpcid = iv_vpc_id
          iv_servicename = iv_service_name
          it_routetableids = it_route_table_ids ).
        DATA(lv_vpc_endpoint_id) = oo_result->get_vpcendpoint( )-
>get_vpcendpointid( ).
        MESSAGE 'Created VPC endpoint.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateVpcEndpointin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`CreateVpnConnection`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVpnConnection` verwendet wird.

CLI

AWS CLI

Beispiel 1: So stellen Sie eine VPN-Verbindung mit dynamischem Routing her

Im folgenden Beispiel für `create-vpn-connection` werden eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem festgelegten Kunden-Gateway hergestellt und Tags auf die VPN-Verbindung angewendet. Die Ausgabe enthält die Konfigurationsinformationen für Ihr Kunden-Gateway-Gerät im XML-Format.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --tag-specification 'ResourceType=vpn-connection,Tags=[{Key=Name,Value=BGP-
VPN}]'
```

Ausgabe:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "...configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {},
                {}
            ]
        },
        "Routes": [],
        "Tags": [
             {
                "Key": "Name",
                "Value": "BGP-VPN"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

Beispiel 2: So stellen Sie eine VPN-Verbindung mit statischem Routing her

Im folgenden Beispiel für `create-vpn-connection` wird eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem festgelegten Kunden-Gateway hergestellt. Die Optionen geben statisches Routing an. Die Ausgabe enthält die Konfigurationsinformationen für Ihr Kunden-Gateway-Gerät im XML-Format.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
```

```
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options "{\"StaticRoutesOnly\":true}"
```

Ausgabe:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": true,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {},
                {}
            ]
        },
        "Routes": [],
        "Tags": []
    }
}
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

Beispiel 3: So stellen Sie eine VPN-Verbindung her und geben Ihr eigenes internes CIDR sowie Ihren im Vorfeld freigegebenen Schlüssel an

Im folgenden Beispiel für `create-vpn-connection` wird eine VPN-Verbindung hergestellt und der CIDR-Block der internen IP-Adresse sowie ein benutzerdefinierter, im Vorfeld freigegebener Schlüssel für jeden Tunnel angegeben. Die angegebenen Werte werden in den `CustomerGatewayConfiguration`-Informationen zurückgegeben.

```
aws ec2 create-vpn-connection \
```

```
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options
 TunnelOptions='[{TunnelInsideCidr=169.254.12.0/30,PreSharedKey=ExamplePreSharedKey1},
{TunnelInsideCidr=169.254.13.0/30,PreSharedKey=ExamplePreSharedKey2}]'
```

Ausgabe:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-123123123123abcab",
        "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv4NetworkCidr": "0.0.0.0/0",
            "RemoteIpv4NetworkCidr": "0.0.0.0/0",
            "TunnelInsideIpVersion": "ipv4",
            "TunnelOptions": [
                {
                    "OutsideIpAddress": "203.0.113.3",
                    "TunnelInsideCidr": "169.254.12.0/30",
                    "PreSharedKey": "ExamplePreSharedKey1"
                },
                {
                    "OutsideIpAddress": "203.0.113.5",
                    "TunnelInsideCidr": "169.254.13.0/30",
                    "PreSharedKey": "ExamplePreSharedKey2"
                }
            ]
        },
        "Routes": [],
        "Tags": []
    }
}
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

Beispiel 4: So stellen Sie eine VPN-Verbindung her, die IPv6-Verkehr unterstützt

Im folgenden Beispiel für `create-vpn-connection` wird eine VPN-Verbindung, die IPv6-Verkehr unterstützt, zwischen dem angegebenen Transit Gateway und dem festgelegten Kunden-Gateway hergestellt. Die Tunneloptionen für beide Tunnel geben an, dass die IKE-Verhandlung initiiert werden AWS muss.

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --transit-gateway-id tgw-12312312312312312 \
    --customer-gateway-id cgw-001122334455aabbc \
    --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
{StartupAction=start}]
```

Ausgabe:

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
        "Category": "VPN",
        "State": "pending",
        "VpnConnectionId": "vpn-11111111122222222",
        "TransitGatewayId": "tgw-12312312312312312",
        "Options": {
            "EnableAcceleration": false,
            "StaticRoutesOnly": false,
            "LocalIpv6NetworkCidr": "::/0",
            "RemoteIpv6NetworkCidr": "::/0",
            "TunnelInsideIpVersion": "ipv6",
            "TunnelOptions": [
                {
                    "OutsideIpAddress": "203.0.113.3",
                    "StartupAction": "start"
                },
                {
                    "OutsideIpAddress": "203.0.113.5",
                    "StartupAction": "start"
                }
            ]
        },
        "Routes": [],
        "Tags": []
```

```
    }
}
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateVpnConnectionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt. Die Ausgabe enthält die Konfigurationsinformationen, die Ihr Netzwerkadministrator benötigt, im XML-Format.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird die VPN-Verbindung hergestellt und die Konfiguration in einer Datei mit dem angegebenen Namen erfasst.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

Beispiel 3: In diesem Beispiel wird eine VPN-Verbindung mit statischem Routing zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Einzelheiten zur API finden Sie unter CreateVpnConnection AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt. Die Ausgabe enthält die Konfigurationsinformationen, die Ihr Netzwerkadministrator benötigt, im XML-Format.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird die VPN-Verbindung hergestellt und die Konfiguration in einer Datei mit dem angegebenen Namen erfasst.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

Beispiel 3: In diesem Beispiel wird eine VPN-Verbindung mit statischem Routing zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- Einzelheiten zur API finden Sie unter CreateVpnConnection AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateVpnConnectionRoute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVpnConnectionRoute` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine statische Route für eine VPN-Verbindung

Dieses Beispiel erstellt eine statische Routing-Tabelle für die angegebene VPN-Verbindung. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- Einzelheiten zur API finden Sie CreateVpnConnectionRoutein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene statische Route für die angegebene VPN-Verbindung erstellt.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Einzelheiten zur API finden Sie unter CreateVpnConnectionRoute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route für die angegebene VPN-Verbindung erstellt.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- Einzelheiten zur API finden Sie unter CreateVpnConnectionRoute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateVpnGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVpnGateway` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit dedizierten Netzwerkverbindungen

CLI

AWS CLI

So erstellen Sie ein Virtual Private Gateway

In diesem Beispiel wird ein Virtual Private Gateway erstellt.

Befehl:

```
aws ec2 create-vpn-gateway --type ipsec.1
```

Ausgabe:

```
{
    "VpnGateway": {
        "AmazonSideAsn": 64512,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
}
```

Um ein virtuelles privates Gateway mit einer bestimmten Amazon-side ASN zu erstellen

In diesem Beispiel wird ein Virtual Private Gateway erstellt und die autonome Systemnummer (ASN) für die Amazon-Seite der BGP-Sitzung angegeben.

Befehl:

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

Ausgabe:

```
{
    "VpnGateway": {
        "AmazonSideAsn": 65001,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
}
```

- Einzelheiten zur API finden Sie CreateVpnGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway erstellt.

```
New-EC2VpnGateway -Type ipsec.1
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter CreateVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway erstellt.

```
New-EC2VpnGateway -Type ipsec.1
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter CreateVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteCustomerGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCustomerGateway` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Kunden-Gateway

In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- Einzelheiten zur API finden Sie DeleteCustomerGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on
 Target "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteCustomerGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on
 Target "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteCustomerGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteDhcpOptions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDhcpOptions` verwendet wird.

CLI

AWS CLI

So löschen Sie einen DHCP-Optionssatz

In diesem Beispiel wird der angegebene DHCP-Optionssatz gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- Einzelheiten zur API finden Sie DeleteDhcpOptionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Optionssatz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Optionssatz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteFlowLogs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFlowLogs` verwendet wird.

CLI

AWS CLI

So löschen Sie ein -Flow-Protokoll

Im folgenden Beispiel für `delete-flow-logs` wird das angegebene Flow-Protokoll gelöscht.

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

Ausgabe:

```
{
    "Unsuccessful": []
}
```

- Einzelheiten zur API finden Sie DeleteFlowLogsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel entfernt den angegebenen Wert FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). DeleteFlowLogsAWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel entfernt den angegebenen FlowLogId Wert fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). DeleteFlowLogsAWS -Tools für PowerShell

Verwendung von **`DeleteInternetGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteInternetGateway` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So löschen Sie ein Internet-Gateway

Im folgenden Beispiel für `delete-internet-gateway` wird das angegebene Internet-Gateway gelöscht.

```
aws ec2 delete-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Internet-Gateways im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie DeleteInternetGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
 Target "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
 Target "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteKeyPair`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteKeyPair` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Einen Containerdienst für VM-Instanzen erstellen
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit verwaltetem Streaming

- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome =
 ec2Client.DeleteKeyPair(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie ein Schlüsselpaar

Im folgenden Beispiel für `delete-key-pair` wird das angegebene Schlüsselpaar gelöscht.

```
aws ec2 delete-key-pair \
    --key-name my-key-pair
```

Ausgabe:

```
{
    "Return": true,
    "KeyPairId": "key-03c8d3aceb53b507"
}
```

Weitere Informationen finden Sie unter Erstellen und Löschen von Schlüsselpaaren im Benutzerhandbuch für die AWS -Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
```

```
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
keyPair) {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " +
keyPair, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
```

```
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
```

```
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- API-Details finden Sie DeleteKeyPairin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
```

```
                         This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                    "Please verify the key pair name and try again."
```

```
            )
        raise
```

- Einzelheiten zur API finden Sie DeleteKeyPairin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wrapper um delete_key, der auch den zugrunde liegenden privaten PEM-Schlüssel entfernt.

```
pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
    if let Some(key_name) = self.key_pair.key_name() {
        ec2.delete_key_pair(key_name).await?;
        if let Some(key_path) = self.key_file_path() {
            if let Err(err) = util.remove(key_path) {
                eprintln!("Failed to remove {key_path:?} ({err:?})");
            }
        }
    }
    Ok(())
}
```

```
pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
    let key_name: String = key_name.into();
    tracing::info!("Deleting key pair {key_name}");
    self.client
        .delete_key_pair()
        .key_name(key_name)
        .send()
        .await?;
```

```
        Ok(())
    }
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
        MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteLaunchTemplate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLaunchTemplate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
```

```
        }
    }
```

- Einzelheiten zur API finden Sie DeleteLaunchTemplatein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie eine Startvorlage

In diesem Beispiel wird die angegebene Startvorlage gelöscht.

Befehl:

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

Ausgabe:

```
{
  "LaunchTemplate": {
      "LatestVersionNumber": 2,
      "LaunchTemplateId": "lt-0abcd290751193123",
      "LaunchTemplateName": "TestTemplate",
      "DefaultVersionNumber": 2,
      "CreatedBy": "arn:aws:iam::123456789012:root",
      "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- Einzelheiten zur API finden Sie DeleteLaunchTemplatein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
```

- Einzelheiten zur API finden Sie DeleteLaunchTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
```

```
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_template(self):
        """
```

```
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DeleteLaunchTemplatein AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DeleteNetworkAcl`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteNetworkAcl` verwendet wird.

CLI

AWS CLI

Um eine Netzwerk-ACL zu löschen

In diesem Beispiel wird die angegebene Netzwerk-ACL gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- Einzelheiten zur API finden Sie DeleteNetworkAclin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAcl AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAcl AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteNetworkAclEntry`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteNetworkAclEntry` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Netzwerk-ACL-Eintrag

In diesem Beispiel wird die Eingangsregel Nummer 100 aus der angegebenen Netzwerk-ACL gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100
```

- Einzelheiten zur API finden Sie DeleteNetworkAclEntryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus der spezifizierten Netzwerk-ACL entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on
 Target "acl-12345678".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus der spezifizierten Netzwerk-ACL entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on
 Target "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteNetworkInterface`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteNetworkInterface` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Netzwerkschnittstelle

In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- Einzelheiten zur API finden Sie DeleteNetworkInterfacein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
 Target "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
 Target "eni-12345678".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeletePlacementGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePlacementGroup` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Platzierungsgruppe

Dieser Beispielbefehl löscht die angegebene Platzierungsgruppe.

Befehl:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- Einzelheiten zur API finden Sie DeletePlacementGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeletePlacementGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeletePlacementGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteRoute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRoute` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Network Transit Gateways

CLI

AWS CLI

So löschen Sie eine Route

In diesem Beispiel wird die angegebene Route aus der festgelegten Routing-Tabelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-
block 0.0.0.0/0
```

- Einzelheiten zur API finden Sie DeleteRoutein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Route aus der spezifizierten Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteRoute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route aus der spezifizierten Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteRoute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteRouteTable`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRouteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So löschen Sie eine Routing-Tabelle

In diesem Beispiel wird die angegebene Routing-Tabelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- Einzelheiten zur API finden Sie DeleteRouteTablein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteSecurityGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSecurityGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Eine relationale Datenbankinstanz erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen
- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
 group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
```

```
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
```

```
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a security group.
/*!
  \param securityGroupID: A security group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;
```

```
    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
 ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

[EC2-Classic] Um eine Sicherheitsgruppe zu löschen

In diesem Beispiel wird die Sicherheitsgruppe mit dem Namen `MySecurityGroup` gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] Um eine Sicherheitsgruppe zu löschen

In diesem Beispiel wird die Sicherheitsgruppe mit der ID `sg-903004f8` gelöscht. Beachten Sie, dass Sie nicht anhand des Namens auf eine Sicherheitsgruppe verweisen können. EC2-VPC Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

Weitere Informationen finden Sie unter Verwenden von Sicherheitsgruppen im Benutzerhandbuch zu die AWS -Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie DeleteSecurityGroupunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an EC2 security group asynchronously.
    *
    * @param groupId the ID of the security group to delete
    * @return a CompletableFuture that completes when the security group is
deleted
    */
   public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
       DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
           .groupId(groupId)
           .build();

       CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
           } else if (resp == null) {
               throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
           }
       }).thenApply(resp -> null);
   }
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2-VPC gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: In diesem Beispiel wird die angegebene Sicherheitsgruppe für gelöscht. EC2-Classic

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Einzelheiten zur API finden Sie unter DeleteSecurityGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für gelöscht. EC2-VPC Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: In diesem Beispiel wird die angegebene Sicherheitsgruppe für gelöscht. EC2-Classic

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- Einzelheiten zur API finden Sie unter DeleteSecurityGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete.
Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an
AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
       tracing::info!("Deleting security group {group_id}");
       self.client
           .delete_security_group()
           .group_id(group_id)
           .send()
           .await?;
       Ok(())
   }
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).
        MESSAGE 'Security group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteSnapshot`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSnapshot` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

So löschen Sie einen Snapshot

Dieser Beispielbefehl löscht einen Snapshot mit der Snapshot-ID von `snap-1234567890abcdef0`. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- Einzelheiten zur API finden Sie DeleteSnapshotin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteSnapshotin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DeleteSpotDatafeedSubscription`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSpotDatafeedSubscription` verwendet wird.

CLI

AWS CLI

So kündigen Sie ein Daten-Feed-Abonnement für eine Spot Instance

Dieser Beispielbefehl löscht ein Daten-Feed-Abonnement einer Spot Instance für das Konto. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-spot-datafeed-subscription
```

- Einzelheiten zur API finden Sie DeleteSpotDatafeedSubscriptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Daten-Feed für die Spot Instance gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Daten-Feed für die Spot Instance gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteSubnet`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSubnet` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken

- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So löschen Sie ein Subnetz

In diesem Beispiel wird das angegebene Subnetz gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- Einzelheiten zur API finden Sie DeleteSubnetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
 "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSubnet AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
 "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteSubnet AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTags` verwendet wird.

CLI

AWS CLI

Beispiel 1: So löschen Sie ein Tag aus einer Ressource

Im folgenden Beispiel für `delete-tags` wird das Tag `Stack=Test` aus dem angegebenen Image gelöscht. Wenn Sie sowohl einen Wert als auch einen Schlüsselnamen angeben, wird das Tag nur gelöscht, wenn der Wert des Tags dem angegebenen Wert entspricht.

```
aws ec2 delete-tags \
    --resources ami-1234567890abcdef0 \
    --tags Key=Stack,Value=Test
```

Die Angabe des Werts für ein Tag ist optional. Im folgenden Beispiel für `delete-tags` wird das Tag mit dem Schlüsselnamen `purpose` aus der angegebenen Instance gelöscht, unabhängig vom Tag-Wert des Tags.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=purpose
```

Wenn Sie die leere Zeichenfolge als Tag-Wert angeben, wird das Tag nur gelöscht, wenn der Tag-Wert eine leere Zeichenfolge ist. Im folgenden Beispiel für `delete-tags` wird die leere Zeichenfolge als Tag-Wert für das zu löschende Tag angegeben.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=Name,Value=
```

Beispiel 2: So löschen Sie ein Tag aus mehreren Ressourcen

Im folgenden Beispiel für `delete-tags` wird das Tag ``Purpose=Test`` sowohl aus einer Instance als auch aus einem AMI gelöscht. Wie im vorherigen Beispiel gezeigt, können Sie den Tag-Wert im Befehl weglassen.

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 ami-1234567890abcdef0 \
    --tags Key=Purpose
```

- Einzelheiten zur API finden Sie DeleteTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Beispiel 2: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -
Force
```

Beispiel 3: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Beispiel 4: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Beispiel 2: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -
Force
```

Beispiel 3: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Beispiel 4: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteVolume`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVolume` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Blockspeicher-Volumes erstellen und verwalten
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

So löschen Sie ein Volume

Dieser Beispielbefehl löscht ein verfügbares Volume mit der Volume-ID `vol-049df61146c4d7901`. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- Einzelheiten zur API finden Sie DeleteVolumein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
 "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVolume AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
 "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVolume AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteVpc`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVpc` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit IP-Adressverwaltung
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So löschen Sie eine VPC

In diesem Beispiel wird die angegebene VPC gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- Einzelheiten zur API finden Sie DeleteVpcin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteVpcin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene VPC gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpc AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpc AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, vpc_id: str) -> None:
        """
        Deletes the specified VPC.

        :param vpc_id: The ID of the VPC to delete.
        """
        try:
            self.ec2_client.delete_vpc(VpcId=vpc_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC %s. Here's why: %s: %s",
                vpc_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteVpcin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletevpc( iv_vpcid = iv_vpc_id ).
        MESSAGE 'Deleted VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteVpcin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteVpcEndpoints`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVpcEndpoints` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So löschen Sie einen Endpunkt

In diesem Beispiel werden die Endpunkte vpce-aa22bb33 und vpce-1a2b3c4d gelöscht. Wenn der Befehl teilweise oder gar nicht erfolgreich ist, wird eine Liste der erfolglosen Elemente zurückgegeben. Wenn der Befehl erfolgreich ausgeführt wurde, ist die zurückgegebene Liste leer.

Befehl:

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-aa22bb33 vpce-1a2b3c4d
```

Ausgabe:

```
{
  "Unsuccessful": []
}
```

- Einzelheiten zur API finden Sie DeleteVpcEndpointsin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
```

```
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteVpcEndpointsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.
```

```
        :return: An instance of VpcWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete_vpc_endpoints(self, vpc_endpoint_ids: list[str]) -> None:
        """
        Deletes the specified VPC endpoints.

        :param vpc_endpoint_ids: A list of IDs of the VPC endpoints to delete.
        """
        try:
            self.ec2_client.delete_vpc_endpoints(VpcEndpointIds=vpc_endpoint_ids)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC endpoints %s. Here's why: %s: %s",
                vpc_endpoint_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteVpcEndpointsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        lo_ec2->deletevpcendpoints( it_vpcendpointids = it_vpc_endpoint_ids ).
        MESSAGE 'Deleted VPC endpoint(s).' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteVpcEndpointsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteVpnConnection`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVpnConnection` verwendet wird.

CLI

AWS CLI

So löschen Sie eine VPN-Verbindung

In diesem Beispiel wird die angegebene VPN-Verbindung gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- Einzelheiten zur API finden Sie DeleteVpnConnectionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnection AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnection AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteVpnConnectionRoute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVpnConnectionRoute` verwendet wird.

CLI

AWS CLI

So löschen Sie eine statische Route aus einer VPN-Verbindung

In diesem Beispiel wird die angegebene statische Route aus der festgelegten VPN-Verbindung gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- Einzelheiten zur API finden Sie DeleteVpnConnectionRoutein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN-Verbindung entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnectionRoute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN-Verbindung entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnConnectionRoute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteVpnGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVpnGateway` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit dedizierten Netzwerkverbindungen

CLI

AWS CLI

So löschen Sie ein Virtual Private Gateway

In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- Einzelheiten zur API finden Sie DeleteVpnGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterImage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterImage` verwendet wird.

CLI

AWS CLI

So heben Sie die Registrierung eines AMI auf

Dieses Beispiel hebt die Registrierung des angegebenen AMI auf. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- Einzelheiten zur API finden Sie DeregisterImagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Registrierung des angegebenen AMI aufgehoben.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Einzelheiten zur API finden Sie unter DeregisterImage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung des angegebenen AMI aufgehoben.

```
Unregister-EC2Image -ImageId ami-12345678
```

- Einzelheiten zur API finden Sie unter DeregisterImage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAccountAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAccountAttributes` verwendet wird.

CLI

AWS CLI

Um alle Attribute für Ihr AWS Konto zu beschreiben

In diesem Beispiel werden die Attribute für Ihr AWS Konto beschrieben.

Befehl:

```
aws ec2 describe-account-attributes
```

Ausgabe:

```
{
    "AccountAttributes": [
        {
            "AttributeName": "vpc-max-security-groups-per-interface",
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        },
        {
            "AttributeName": "max-instances",
            "AttributeValues": [
                {
                    "AttributeValue": "20"
                }
            ]
        },
        {
            "AttributeName": "supported-platforms",
            "AttributeValues": [
                {
                    "AttributeValue": "EC2"
                },
                {
                    "AttributeValue": "VPC"
                }
            ]
        },
        {
```

```
            "AttributeName": "default-vpc",
            "AttributeValues": [
                {
                    "AttributeValue": "none"
                }
            ]
        },
        {
            "AttributeName": "max-elastic-ips",
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        },
        {
            "AttributeName": "vpc-max-elastic-ips",
            "AttributeValues": [
                {
                    "AttributeValue": "5"
                }
            ]
        }
    ]
}
```

Um ein einzelnes Attribut für Ihr AWS Konto zu beschreiben

In diesem Beispiel wird das `supported-platforms` Attribut für Ihr AWS Konto beschrieben.

Befehl:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

Ausgabe:

```
{
    "AccountAttributes": [
        {
            "AttributeName": "supported-platforms",
            "AttributeValues": [
                {
                    "AttributeValue": "EC2"
```

```
                },
                {
                    "AttributeValue": "VPC"
                }
            ]
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAccountAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird beschrieben, ob Sie Instances in EC2-Classic und EC2-VPC in der Region oder nur in der Region starten können EC2-VPC.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
EC2
VPC
```

Beispiel 2: In diesem Beispiel Ihre Standard-VPC beschrieben oder es zeigt „none“, wenn Sie in der Region keine Standard-VPC haben.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
vpc-12345678
```

Beispiel 3: Dieses Beispiel beschreibt die maximale Anzahl von On-Demand Instances, die Sie ausführen können.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
20
```

- Einzelheiten zur API finden Sie unter DescribeAccountAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird beschrieben, ob Sie Instances in EC2-Classic und EC2-VPC in der Region oder nur in EC2-VPC der Region starten können.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
EC2
VPC
```

Beispiel 2: In diesem Beispiel Ihre Standard-VPC beschrieben oder es zeigt „none“, wenn Sie in der Region keine Standard-VPC haben.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
vpc-12345678
```

Beispiel 3: Dieses Beispiel beschreibt die maximale Anzahl von On-Demand Instances, die Sie ausführen können.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
20
```

- Einzelheiten zur API finden Sie unter DescribeAccountAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeAddresses`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAddresses` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Elastic IP addresses.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeAddresses(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
 ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
                  std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
                  std::setw(30) << "Allocation ID" << std::setw(25) <<
```

```
                  "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
 outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                    Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                            address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                      address.GetInstanceId() << std::setw(15) <<
                      address.GetPublicIp() << std::setw(10) << domainString <<
                      std::setw(30) << address.GetAllocationId() << std::setw(25)
                      << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeAddressesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So rufen Sie Details über alle Ihre Elastic-IP-Adressen ab

Im folgenden Beispiel für `describe addresses` werden Details zu Ihren Elastic-IP-Adressen angezeigt.

```
aws ec2 describe-addresses
```

Ausgabe:

```
{
    "Addresses": [
        {
```

```
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "198.51.100.0",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        },
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-12345678",
            "AssociationId": "eipassoc-12345678",
            "NetworkInterfaceOwnerId": "123456789012",
            "PublicIp": "203.0.113.0",
            "AllocationId": "eipalloc-12345678",
            "PrivateIpAddress": "10.0.1.241"
        }
    ]
}
```

Beispiel 2: Um Details zu Ihren Elastic-IP-Adressen abzurufen EC2-VPC

Im folgenden Beispiel für `describe-addresses` werden Details zu Ihren Elastic-IP-Adressen für die Verwendung mit Instances in einer VPC angezeigt.

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=vpc"
```

Ausgabe:

```
{
    "Addresses": [
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-12345678",
            "AssociationId": "eipassoc-12345678",
            "NetworkInterfaceOwnerId": "123456789012",
            "PublicIp": "203.0.113.0",
            "AllocationId": "eipalloc-12345678",
            "PrivateIpAddress": "10.0.1.241"
        }
```

```
    ]
}
```

Beispiel 3: So rufen Sie Details über eine durch die Zuweisungs-ID spezifizierte Elastic-IP-Adresse ab

Im folgenden `describe-addresses` Beispiel werden Details zur Elastic IP-Adresse mit der angegebenen Zuweisungs-ID angezeigt, die einer Instance in zugeordnet ist EC2-VPC.

```
aws ec2 describe-addresses \
    --allocation-ids eipalloc-282d9641
```

Ausgabe:

```
{
    "Addresses": [
        {
            "Domain": "vpc",
            "PublicIpv4Pool": "amazon",
            "InstanceId": "i-1234567890abcdef0",
            "NetworkInterfaceId": "eni-1a2b3c4d",
            "AssociationId": "eipassoc-123abc12",
            "NetworkInterfaceOwnerId": "1234567891012",
            "PublicIp": "203.0.113.25",
            "AllocationId": "eipalloc-282d9641",
            "PrivateIpAddress": "10.251.50.12"
        }
    ]
}
```

Beispiel 4: So rufen Sie Details über eine Elastic-IP-Adresse ab, die durch ihre private VPC-IP-Adresse angegeben ist

Im folgenden `describe-addresses` Beispiel werden Details zur Elastic IP-Adresse angezeigt, die einer bestimmten privaten IP-Adresse in zugeordnet ist EC2-VPC.

```
aws ec2 describe-addresses \
    --filters "Name=private-ip-address,Values=10.251.50.12"
```

Beispiel 5: Um Details zu Elastic IP-Adressen abzurufen in EC2-Classic

Im folgenden `describe-addresses` Beispiel werden Details zu Ihren Elastic IP-Adressen zur Verwendung in angezeigt EC2-Classic.

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=standard"
```

Ausgabe:

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
    ]
}
```

Beispiel 6: So rufen Sie Details über eine Elastic-IP-Adresse ab, die durch ihre öffentliche IP-Adresse angegeben ist

Im folgenden `describe-addresses` Beispiel werden Details zur Elastic IP-Adresse mit dem Wert angezeigt`203.0.110.25`, der einer Instance in zugeordnet ist EC2-Classic.

```
aws ec2 describe-addresses \
    --public-ips 203.0.110.25
```

Ausgabe:

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAddressesunter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeAddressesCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP
 addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: [allocationId],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationId.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeAddressesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Ausgabe:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Elastic-IP-Adressen für Instances in einer VPC. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Ausgabe:

```
AllocationId            :
AssociationId           :
```

```
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Beispiel 4: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in EC2-Classic. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Beispiel 5: In diesem Beispiel werden alle Ihre Elastic IP-Adressen beschrieben.

```
Get-EC2Address
```

Beispiel 6: Dieses Beispiel gibt die öffentliche und private IP für die im Filter angegebene Instance-ID zurück

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Ausgabe:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Beispiel 7: In diesem Beispiel werden alle Elastic IPs mit ihrer Zuweisungs-ID, Zuordnungs-ID und Instance-IDs abgerufen

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Ausgabe:

```
InstanceId          AssociationId              AllocationId
 PublicIp
----------          -------------              ------------
 --------
```

```
                                                    eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                                    eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Beispiel 8: Dieses Beispiel ruft eine Liste der EC2-IP-Adressen ab, die mit dem Tag-Schlüssel „Category” mit dem Wert „Prod” übereinstimmen'

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Ausgabe:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- Einzelheiten zur API finden Sie unter DescribeAddresses AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Ausgabe:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Elastic-IP-Adressen für Instances in einer VPC. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Ausgabe:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

Beispiel 4: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in EC2-Classic. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Beispiel 5: In diesem Beispiel werden alle Ihre Elastic IP-Adressen beschrieben.

```
Get-EC2Address
```

Beispiel 6: Dieses Beispiel gibt die öffentliche und private IP für die im Filter angegebene Instance-ID zurück

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Ausgabe:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Beispiel 7: In diesem Beispiel werden alle Elastic IPs mit ihrer Zuweisungs-ID, Zuordnungs-ID und Instance-IDs abgerufen

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Ausgabe:

```
InstanceId          AssociationId              AllocationId
 PublicIp
----------          -------------              ------------
 --------
                                               eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                               eipalloc-012345678eeabcfad
 17.212.225.7
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
```

```
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Beispiel 8: Dieses Beispiel ruft eine Liste der EC2-IP-Adressen ab, die mit dem Tag-Schlüssel „Category” mit dem Wert „Prod” übereinstimmen'

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Ausgabe:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- Einzelheiten zur API finden Sie unter DescribeAddresses AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeaddresses( ).                        "
oo_result is returned for testing purposes. "
```

```
        DATA(lt_addresses) = oo_result->get_addresses( ).
        MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAddressesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeAvailabilityZones`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAvailabilityZones` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Blockspeicher-Volumes erstellen und verwalten
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Network Transit Gateways
- Erste Schritte mit Videotransport-Streaming
- Virtuelles privates Netzwerk mit privaten Servern
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! DescribeAvailabilityZones
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
 ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "ZoneName" <<
                  std::setw(20) << "State" <<
                  std::setw(32) << "Region" << std::endl;

        const auto &zones =
                outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

 Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                            zone.GetState());
            std::cout << std::left <<
                      std::setw(32) << zone.GetZoneName() <<
                      std::setw(20) << stateString <<
                      std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
                  outcome.GetError().GetMessage() << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie Ihre Availability Zones

Das folgende Beispiel `describe-availability-zones` zeigt Details zu den Availability Zones, die für Sie verfügbar sind. Die Antwort umfasst nur Availability Zones für die aktuelle Region. In diesem Beispiel wird die Standardregion `us-west-2` (Oregon) des Profils verwendet.

```
aws ec2 describe-availability-zones
```

Ausgabe:

```
{
    "AvailabilityZones": [
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2a",
            "ZoneId": "usw2-az1",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2b",
            "ZoneId": "usw2-az2",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
```

```
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2c",
            "ZoneId": "usw2-az3",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opt-in-not-required",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2d",
            "ZoneId": "usw2-az4",
            "GroupName": "us-west-2",
            "NetworkBorderGroup": "us-west-2"
        },
        {
            "State": "available",
            "OptInStatus": "opted-in",
            "Messages": [],
            "RegionName": "us-west-2",
            "ZoneName": "us-west-2-lax-1a",
            "ZoneId": "usw2-lax1-az1",
            "GroupName": "us-west-2-lax-1",
            "NetworkBorderGroup": "us-west-2-lax-1"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Availability Zones für die aktuelle Region beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2AvailabilityZone
```

Ausgabe:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Beispiel 2: In diesem Beispiel werden alle Availability Zones beschrieben, die sich in einem beeinträchtigten Zustand befinden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeAvailabilityZones AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zones für die aktuelle Region beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2AvailabilityZone
```

Ausgabe:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Beispiel 2: In diesem Beispiel werden alle Availability Zones beschrieben, die sich in einem beeinträchtigten Zustand befinden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeAvailabilityZones AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
```

```
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.
```

```
        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeavailabilityzones( ).
 " oo_result is returned for testing purposes. "
        DATA(lt_zones) = oo_result->get_availabilityzones( ).
        MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.

      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeBundleTasks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeBundleTasks` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Bündelungsaufgaben

In diesem Beispiel werden alle Ihre Bündelungsaufgaben beschrieben.

Befehl:

```
aws ec2 describe-bundle-tasks
```

Ausgabe:

```
{
  "BundleTasks": [
    {
      "UpdateTime": "2015-09-15T13:26:54.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "Storage": {
        "S3": {
            "Prefix": "winami",
            "Bucket": "bundletasks"
        }
      },
      "State": "bundling",
      "StartTime": "2015-09-15T13:24:35.000Z",
      "Progress": "3%",
      "BundleId": "bun-2a4e041c"
    }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeBundleTasksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Bündelungsaufgabe beschrieben.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Beispiel 2: In diesem Beispiel werden die Bündelungsaufgaben beschrieben, deren Status entweder „Abgeschlossen“ oder „Fehlgeschlagen“ lautet.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeBundleTasks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Bündelungsaufgabe beschrieben.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Beispiel 2: In diesem Beispiel werden die Bündelungsaufgaben beschrieben, deren Status entweder „Abgeschlossen“ oder „Fehlgeschlagen“ lautet.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- Einzelheiten zur API finden Sie unter DescribeBundleTasks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeCapacityReservations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCapacityReservations` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine oder mehrere Ihrer Kapazitätsreservierungen

Im folgenden `describe-capacity-reservations` Beispiel werden Details zu all Ihren Kapazitätsreservierungen in der aktuellen AWS Region angezeigt.

```
aws ec2 describe-capacity-reservations
```

Ausgabe:

```
{
    "CapacityReservations": [
        {
            "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-1234abcd56EXAMPLE",
            "AvailabilityZoneId": "use1-az2",
            "InstanceType": "c5.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "active",
            "StartDate": "2024-10-23T15:00:24+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:00:24+00:00",
            "Tags": [],
            "CapacityAllocations": []
        },
        {
            "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
            "AvailabilityZoneId": "use1-az2",
```

```
            "InstanceType": "c4.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "cancelled",
            "StartDate": "2024-10-23T15:01:03+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:01:02+00:00",
            "Tags": [],
            "CapacityAllocations": []
        }
    ]
}
```

Beispiel 2: So beschreiben Sie eine oder mehrere Ihrer Kapazitätsreservierungen

Im folgenden Beispiel für `describe-capacity-reservations` werden die Details zu der angegebenen Kapazitätsreservierung angezeigt.

```
aws ec2 describe-capacity-reservations \
    --capacity-reservation-ids cr-1234abcd56EXAMPLE
```

Ausgabe:

```
{
    "CapacityReservations": [
        {
            "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
            "OwnerId": "123456789111",
            "CapacityReservationArn": "arn:aws:ec2:us-
east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
            "AvailabilityZoneId": "use1-az2",
            "InstanceType": "c4.large",
            "InstancePlatform": "Linux/UNIX",
            "AvailabilityZone": "us-east-1a",
            "Tenancy": "default",
            "TotalInstanceCount": 1,
```

```
            "AvailableInstanceCount": 1,
            "EbsOptimized": true,
            "EphemeralStorage": false,
            "State": "active",
            "StartDate": "2024-10-23T15:01:03+00:00",
            "EndDateType": "unlimited",
            "InstanceMatchCriteria": "open",
            "CreateDate": "2024-10-23T15:01:02+00:00",
            "Tags": [],
            "CapacityAllocations": []
        }
    ]
}
```

Weitere Informationen dazu finden Sie unter Anzeigen einer Kapazitätsreservierung im Benutzerhandbuch für Amazon Elastic Compute Cloud für Linux-Instances.

- Einzelheiten zur API finden Sie DescribeCapacityReservationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden eine oder mehrere Ihrer Kapazitätsreservierungen für die Region beschrieben

```
Get-EC2CapacityReservation -Region eu-west-1
```

Ausgabe:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
```

```
State                   : active
Tags                    : {}
Tenancy                 : default
TotalInstanceCount      : 2
```

- Einzelheiten zur API finden Sie unter DescribeCapacityReservations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden eine oder mehrere Ihrer Kapazitätsreservierungen für die Region beschrieben

```
Get-EC2CapacityReservation -Region eu-west-1
```

Ausgabe:

```
AvailabilityZone        : eu-west-1b
AvailableInstanceCount  : 2
CapacityReservationId   : cr-0c1f2345db6f7cdba
CreateDate              : 3/28/2019 9:29:41 AM
EbsOptimized            : True
EndDate                 : 1/1/0001 12:00:00 AM
EndDateType             : unlimited
EphemeralStorage        : False
InstanceMatchCriteria   : open
InstancePlatform        : Windows
InstanceType            : m4.xlarge
State                   : active
Tags                    : {}
Tenancy                 : default
TotalInstanceCount      : 2
```

- Einzelheiten zur API finden Sie unter DescribeCapacityReservations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeCustomerGateways`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCustomerGateways` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Kunden-Gateways

Dieses Beispiel beschreibt Ihre Kunden-Gateways.

Befehl:

```
aws ec2 describe-customer-gateways
```

Ausgabe:

```
{
    "CustomerGateways": [
        {
            "CustomerGatewayId": "cgw-b4dc3961",
            "IpAddress": "203.0.113.12",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65000"
        },
        {
            "CustomerGatewayId": "cgw-0e11f167",
            "IpAddress": "12.1.2.3",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65534"
        }
    ]
}
```

So beschreiben Sie ein bestimmtes Kunden-Gateway

In diesem Beispiel wird das angegebene Kunden-Gateway beschrieben.

Befehl:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

Ausgabe:

```
{
    "CustomerGateways": [
        {
            "CustomerGatewayId": "cgw-0e11f167",
            "IpAddress": "12.1.2.3",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65534"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeCustomerGatewaysin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway beschrieben.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

Beispiel 2: In diesem Beispiel wird jedes Kunden-Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Kunden-Gateways beschrieben.

```
Get-EC2CustomerGateway
```

- Einzelheiten zur API finden Sie unter DescribeCustomerGateways AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway beschrieben.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

Beispiel 2: In diesem Beispiel wird jedes Kunden-Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Kunden-Gateways beschrieben.

```
Get-EC2CustomerGateway
```

- Einzelheiten zur API finden Sie unter DescribeCustomerGateways AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeDhcpOptions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDhcpOptions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Ihre DHCP-Optionen

Im folgenden Beispiel für `describe-dhcp-options` werden Details über Ihre VPCs abgerufen.

```
aws ec2 describe-dhcp-options
```

Ausgabe:

```
{
    "DhcpOptions": [
        {
            "DhcpConfigurations": [
                {
                    "Key": "domain-name",
                    "Values": [
                        {
                            "Value": "us-east-2.compute.internal"
                        }
                    ]
                },
                {
                    "Key": "domain-name-servers",
                    "Values": [
                        {
                            "Value": "AmazonProvidedDNS"
                        }
                    ]
                }
            ],
            "DhcpOptionsId": "dopt-19edf471",
            "OwnerId": "111122223333"
        },
        {
            "DhcpConfigurations": [
                {
                    "Key": "domain-name",
                    "Values": [
                        {
```

```
                            "Value": "us-east-2.compute.internal"
                        }
                    ]
                },
                {
                    "Key": "domain-name-servers",
                    "Values": [
                        {
                            "Value": "AmazonProvidedDNS"
                        }
                    ]
                }
            ],
            "DhcpOptionsId": "dopt-fEXAMPLE",
            "OwnerId": "111122223333"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit DHCP-Optionslisten im Benutzerhandbuch für AWS VPC.

Beispiel 2: So beschreiben Sie Ihre DHCP-Optionen und filtern die Ausgabe

Im folgenden Beispiel für `describe-dhcp-options` werden Ihre DHCP-Optionen beschrieben und ein Filter wird verwendet, um nur die DHCP-Optionen zurückzugeben, für die `example.com` als Domainnamenserver festgelegt ist. Im Beispiel wird der Parameter `--query` verwendet, um nur die Konfigurationsinformationen und die ID in der Ausgabe anzuzeigen.

```
aws ec2 describe-dhcp-options \
    --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
    --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"
```

Ausgabe:

```
[
    [
        [
            {
                "Key": "domain-name",
                "Values": [
```

```
                    {
                        "Value": "example.com"
                    }
                ]
            },
            {
                "Key": "domain-name-servers",
                "Values": [
                    {
                        "Value": "172.16.16.16"
                    }
                ]
            }
        ],
        "dopt-00112233445566 7ab"
    ]
]
```

Weitere Informationen finden Sie unter Arbeiten mit DHCP-Optionslisten im Benutzerhandbuch für AWS VPC.

- Einzelheiten zur API finden Sie DescribeDhcpOptionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Ihre DHCP-Optionssätze aufgelistet.

```
Get-EC2DhcpOption
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

Beispiel 2: In diesem Beispiel werden Konfigurationsdetails für den angegebenen DHCP-Optionssatz abgerufen.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Ausgabe:

```
Key                     Values
---                     ------
domain-name             {abc.local}
domain-name-servers     {10.0.0.101, 10.0.0.102}
```

- Einzelheiten zur API finden Sie unter DescribeDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Ihre DHCP-Optionssätze aufgelistet.

```
Get-EC2DhcpOption
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

Beispiel 2: In diesem Beispiel werden Konfigurationsdetails für den angegebenen DHCP-Optionssatz abgerufen.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Ausgabe:

```
Key                     Values
---                     ------
domain-name             {abc.local}
domain-name-servers     {10.0.0.101, 10.0.0.102}
```

- Einzelheiten zur API finden Sie unter DescribeDhcpOptions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeFlowLogs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFlowLogs` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle Ihre Flow-Protokolle

Im folgenden Beispiel für `describe-flow-logs` werden Details für all Ihre Flow-Protokolle angezeigt.

```
aws ec2 describe-flow-logs
```

Ausgabe:

```
{
    "FlowLogs": [
        {
            "CreationTime": "2018-02-21T13:22:12.644Z",
            "DeliverLogsPermissionArn": "arn:aws:iam::123456789012:role/flow-
logs-role",
            "DeliverLogsStatus": "SUCCESS",
            "FlowLogId": "fl-aabbccdd112233445",
            "MaxAggregationInterval": 600,
            "FlowLogStatus": "ACTIVE",
            "LogGroupName": "FlowLogGroup",
            "ResourceId": "subnet-12345678901234567",
            "TrafficType": "ALL",
            "LogDestinationType": "cloud-watch-logs",
            "LogFormat": "${version} ${account-id} ${interface-id} ${srcaddr}
 ${dstaddr} ${srcport} ${dstport} ${protocol} ${packets} ${bytes} ${start} ${end}
 ${action} ${log-status}"
        },
        {
            "CreationTime": "2020-02-04T15:22:29.986Z",
            "DeliverLogsStatus": "SUCCESS",
            "FlowLogId": "fl-01234567890123456",
            "MaxAggregationInterval": 60,
            "FlowLogStatus": "ACTIVE",
            "ResourceId": "vpc-00112233445566778",
            "TrafficType": "ACCEPT",
```

```
            "LogDestinationType": "s3",
            "LogDestination": "arn:aws:s3:::my-flow-log-bucket/custom",
            "LogFormat": "${version} ${vpc-id} ${subnet-id} ${instance-id}
 ${interface-id} ${account-id} ${type} ${srcaddr} ${dstaddr} ${srcport}
 ${dstport} ${pkt-srcaddr} ${pkt-dstaddr} ${protocol} ${bytes} ${packets}
 ${start} ${end} ${action} ${tcp-flags} ${log-status}"
        }
    ]
}
```

Beispiel 2: So beschreiben Sie eine Teilmenge Ihrer Flow-Protokolle

Das folgende `describe-flow-logs` Beispiel verwendet einen Filter, um Details nur für die Flow-Logs anzuzeigen, die sich in der angegebenen Protokollgruppe in Amazon CloudWatch Logs befinden.

```
aws ec2 describe-flow-logs \
    --filter "Name=log-group-name,Values=MyFlowLogs"
```

- Einzelheiten zur API finden Sie DescribeFlowLogsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden ein oder mehrere Flow-Protokolle mit dem Protokollzieltyp „s3“ beschrieben.

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Ausgabe:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
```

```
LogGroupName              :
ResourceId                : eni-01d2dda3456b7e890
TrafficType               : ALL
```

- Einzelheiten zur API finden Sie unter DescribeFlowLogs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden ein oder mehrere Flow-Protokolle mit dem Protokollzieltyp „s3“ beschrieben.

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Ausgabe:

```
CreationTime              : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage   :
DeliverLogsPermissionArn  :
DeliverLogsStatus         : SUCCESS
FlowLogId                 : fl-01b2e3d45f67f8901
FlowLogStatus             : ACTIVE
LogDestination            : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType        : s3
LogGroupName              :
ResourceId                : eni-01d2dda3456b7e890
TrafficType               : ALL
```

- Einzelheiten zur API finden Sie unter DescribeFlowLogs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeHostReservationOfferings`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeHostReservationOfferings` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Reservierungsangebote für Dedicated Hosts

In diesem Beispiel werden die Dedicated-Host-Reservierungen für die M4-Instance-Familie beschrieben, die erworben werden können.

Befehl:

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-
family,Values=m4
```

Ausgabe:

```
{
  "OfferingSet": [
      {
          "HourlyPrice": "1.499",
          "OfferingId": "hro-03f707bf363b6b324",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      },
      {
          "HourlyPrice": "1.045",
          "OfferingId": "hro-0ef9181cabdef7a02",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "UpfrontPrice": "0.000",
          "Duration": 94608000
      },
      {
          "HourlyPrice": "0.714",
          "OfferingId": "hro-04567a15500b92a51",
          "InstanceFamily": "m4",
          "PaymentOption": "PartialUpfront",
          "UpfrontPrice": "6254.000",
          "Duration": 31536000
      },
      {
          "HourlyPrice": "0.484",
          "OfferingId": "hro-0d5d7a9d23ed7fbfe",
          "InstanceFamily": "m4",
          "PaymentOption": "PartialUpfront",
          "UpfrontPrice": "12720.000",
          "Duration": 94608000
```

```
        },
        {
            "HourlyPrice": "0.000",
            "OfferingId": "hro-05da4108ca998c2e5",
            "InstanceFamily": "m4",
            "PaymentOption": "AllUpfront",
            "UpfrontPrice": "23913.000",
            "Duration": 94608000
        },
        {
            "HourlyPrice": "0.000",
            "OfferingId": "hro-0a9f9be3b95a3dc8f",
            "InstanceFamily": "m4",
            "PaymentOption": "AllUpfront",
            "UpfrontPrice": "12257.000",
            "Duration": 31536000
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeHostReservationOfferingsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Dedicated Host-Reservierungen beschrieben, die für den angegebenen Filter „Instance-Familie“ erworben werden können, wobei der Filter „“ PaymentOption lautet NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Ausgabe:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
```

```
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- Einzelheiten zur API finden Sie unter DescribeHostReservationOfferings AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Dedicated Host-Reservierungen beschrieben, die für den angegebenen Filter 'Instance-Family' erworben werden können, wobei PaymentOption '' NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Ausgabe:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- Einzelheiten zur API finden Sie unter DescribeHostReservationOfferings AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeHosts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeHosts` verwendet wird.

CLI

AWS CLI

So zeigen Sie die Details zu Dedicated Hosts an

Im folgenden `describe-hosts` Beispiel werden Details zu den `available` Dedicated Hosts in Ihrem AWS Konto angezeigt.

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

Ausgabe:

```
{
    "Hosts": [
        {
            "HostId": "h-07879acf49EXAMPLE",
            "Tags": [
                {
                    "Value": "production",
                    "Key": "purpose"
                }
            ],
            "HostProperties": {
                "Cores": 48,
                "TotalVCpus": 96,
                "InstanceType": "m5.large",
                "Sockets": 2
            },
            "Instances": [],
            "State": "available",
            "AvailabilityZone": "eu-west-1a",
            "AvailableCapacity": {
                "AvailableInstanceCapacity": [
                    {
                        "AvailableCapacity": 48,
                        "InstanceType": "m5.large",
                        "TotalCapacity": 48
                    }
```

```
                ],
                "AvailableVCpus": 96
            },
            "HostRecovery": "on",
            "AllocationTime": "2019-08-19T08:57:44.000Z",
            "AutoPlacement": "off"
        }
    ]
}
```

Weitere Informationen dazu finden Sie unter Anzeigen von Dedicated Hosts im Benutzerhandbuch für Amazon Elastic Compute Cloud für Linux-Instances.

- Einzelheiten zur API finden Sie DescribeHostsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die EC2-Hostdetails zurückgegeben

```
Get-EC2Host
```

Ausgabe:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Beispiel 2: In diesem Beispiel wird nach dem Host AvailableInstanceCapacity h-01e23f4cd567899f1 abgefragt

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Ausgabe:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). DescribeHostsAWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die EC2-Hostdetails zurückgegeben

```
Get-EC2Host
```

Ausgabe:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Beispiel 2: In diesem Beispiel wird nach dem Host AvailableInstanceCapacity h-01e23f4cd567899f1 abgefragt

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Ausgabe:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). DescribeHostsAWS -Tools für PowerShell

Verwenden Sie mit einem **`DescribeIamInstanceProfileAssociations`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeIamInstanceProfileAssociations` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
 instanceId)
    {
        try
        {
            var response = await
 _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
```

```
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeIamInstanceProfileAssociationsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie die Zuordnungen von IAM-Instance-Profilen

In diesem Beispiel werden alle Ihre IAM-Instance-Profilzuordnungen beschrieben.

Befehl:

```
aws ec2 describe-iam-instance-profile-associations
```

Ausgabe:

```
{
  "IamInstanceProfileAssociations": [
      {
          "InstanceId": "i-09eb09efa73ec1dee",
          "State": "associated",
          "AssociationId": "iip-assoc-0db249b1f25fa24b8",
          "IamInstanceProfile": {
              "Id": "AIPAJVQN4F5WVLGCJDRGM",
              "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
          }
      },
      {
          "InstanceId": "i-0402909a2f4dffd14",
          "State": "associating",
          "AssociationId": "iip-assoc-0d1ec06278d29f44a",
          "IamInstanceProfile": {
              "Id": "AGJAJVQN4F5WVLGCJABCM",
              "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
          }
      }
   ]
}
```

- Einzelheiten zur API finden Sie DescribeIamInstanceProfileAssociationsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
```

```
          ],
        }),
      );
```

- Einzelheiten zur API finden Sie DescribeIamInstanceProfileAssociationsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
```

```
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
```

```
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
 {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DescribeIamInstanceProfileAssociationsin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DescribeIdFormat`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeIdFormat` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie das ID-Format einer Ressource

Das folgende Beispiel für `describe-id-format` beschreibt das ID-Format für Sicherheitsgruppen.

```
aws ec2 describe-id-format \
    --resource security-group
```

In der folgenden Beispielausgabe gibt der Wert `Deadline` an, dass die Frist für den dauerhaften Wechsel dieses Ressourcentyps vom kurzen zum langen ID-Format am 15. August 2018 um 00:00 Uhr UTC abgelaufen ist.

```
{
    "Statuses": [
        {
```

```
            "Deadline": "2018-08-15T00:00:00.000Z",
            "Resource": "security-group",
            "UseLongIds": true
        }
    ]
}
```

Beispiel 2: So beschreiben Sie das ID-Format für alle Ressourcen

Im folgenden Beispiel für `describe-id-format` wird das ID-Format für alle Ressourcentypen beschrieben. Alle Ressourcentypen, die das kurze ID-Format unterstützten, wurden auf das lange ID-Format umgestellt.

```
aws ec2 describe-id-format
```

- Einzelheiten zur API finden Sie DescribeIdFormatin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das ID-Format für den angegebenen Ressourcentyp beschrieben.

```
Get-EC2IdFormat -Resource instance
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Beispiel 2: In diesem Beispiel werden die ID-Formate für alle Ressourcentypen beschrieben, die längere IDs unterstützen.

```
Get-EC2IdFormat
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Einzelheiten zur API finden Sie unter DescribeIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das ID-Format für den angegebenen Ressourcentyp beschrieben.

```
Get-EC2IdFormat -Resource instance
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Beispiel 2: In diesem Beispiel werden die ID-Formate für alle Ressourcentypen beschrieben, die längere IDs unterstützen.

```
Get-EC2IdFormat
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- Einzelheiten zur API finden Sie unter DescribeIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeIdentityIdFormat`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeIdentityIdFormat` verwendet wird.

CLI

AWS CLI

So beschreiben Sie das ID-Format für eine IAM-Rolle

Das folgende `describe-identity-id-format` Beispiel beschreibt das ID-Format, das von Instances empfangen wird, die mit der IAM-Rolle `EC2Role` in Ihrem AWS Konto erstellt wurden.

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:role/my-iam-role \
    --resource instance
```

Die folgende Ausgabe zeigt, dass von dieser Rolle erstellte Instances IDs im langen ID-Format erhalten.

```
{
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "instance",
            "UseLongIds": true
        }
    ]
}
```

So beschreiben Sie das ID-Format für einen IAM-Benutzer

Das folgende `describe-identity-id-format` Beispiel beschreibt das ID-Format, das von Snapshots empfangen wird, die vom IAM-Benutzer `AdminUser` in Ihrem Konto erstellt wurden. AWS

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:user/AdminUser \
    --resource snapshot
```

Die folgende Ausgabe zeigt, dass von diesem Benutzer erstellte Snapshots IDs im langen ID-Format erhalten.

```
{
```

```
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "snapshot",
            "UseLongIds": true
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter DescribeIdentityIdFormat AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt das ID-Format für die Ressource „image“ für die angegebene Rolle zurück

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

Ausgabe:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Einzelheiten zur API finden Sie unter DescribeIdentityIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt das ID-Format für die Ressource „image“ für die angegebene Rolle zurück

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

Ausgabe:

```
Deadline             Resource UseLongIds
```

```
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- Einzelheiten zur API finden Sie unter DescribeIdentityIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeImageAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImageAttribute` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Startberechtigungen für ein AMI

In diesem Beispiel werden die Startberechtigungen für das angegebene AMI beschrieben.

Befehl:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --
attribute launchPermission
```

Ausgabe:

```
{
    "LaunchPermissions": [
        {
            "UserId": "123456789012"
        }
    ],
    "ImageId": "ami-5731123e",
}
```

So beschreiben Sie die Produktcodes für ein AMI

In diesem Beispiel werden die Produktcodes für das angegebene AMI beschrieben. Beachten Sie, dass dieses AMI keine Produktcodes hat.

Befehl:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

Ausgabe:

```
{
    "ProductCodes": [],
    "ImageId": "ami-5731123e",
}
```

- Einzelheiten zur API finden Sie DescribeImageAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 2: In diesem Beispiel werden die Startberechtigungen für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
```

```
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 3: In diesem Beispiel wird überprüft, ob Enhanced Networking aktiviert ist.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- Einzelheiten zur API finden Sie unter DescribeImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 2: In diesem Beispiel werden die Startberechtigungen für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 3: In diesem Beispiel wird überprüft, ob Enhanced Networking aktiviert ist.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- Einzelheiten zur API finden Sie unter DescribeImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeImages`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImages` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Virtuelles privates Netzwerk mit privaten Servern

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine AMI

Im folgenden Beispiel für `describe-images` wird das angegebene AMI in der angegebenen Region beschrieben.

```
aws ec2 describe-images \
    --region us-east-1 \
    --image-ids ami-1234567890EXAMPLE
```

Ausgabe:

```
{
    "Images": [
        {
            "VirtualizationType": "hvm",
            "Description": "Provided by Red Hat, Inc.",
            "PlatformDetails": "Red Hat Enterprise Linux",
            "EnaSupport": true,
            "Hypervisor": "xen",
            "State": "available",
            "SriovNetSupport": "simple",
            "ImageId": "ami-1234567890EXAMPLE",
            "UsageOperation": "RunInstances:0010",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "SnapshotId": "snap-111222333444aaabb",
                        "DeleteOnTermination": true,
                        "VolumeType": "gp2",
                        "VolumeSize": 10,
```

```
                        "Encrypted": false
                    }
                }
            ],
            "Architecture": "x86_64",
            "ImageLocation": "123456789012/RHEL-8.0.0_HVM-20190618-x86_64-1-
Hourly2-GP2",
            "RootDeviceType": "ebs",
            "OwnerId": "123456789012",
            "RootDeviceName": "/dev/sda1",
            "CreationDate": "2019-05-10T13:17:12.000Z",
            "Public": true,
            "ImageType": "machine",
            "Name": "RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-GP2"
        }
    ]
}
```

Weitere Informationen dazu finden Sie unter Amazon Machine Images (AMI) im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So beschreiben Sie AMIs basierend auf Filtern

Im folgenden Beispiel für `describe-images` werden von Amazon bereitgestellte Windows-AMIs beschrieben, die durch Amazon EBS gesichert sind.

```
aws ec2 describe-images \
    --owners amazon \
    --filters "Name=platform,Values=windows" "Name=root-device-type,Values=ebs"
```

Ein Beispiel für die Ausgabe von `describe-images` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Filtern finden Sie unter Auflisten und Filtern Ihrer Ressourcen im Amazon-EC2-Benutzerhandbuch.

Beispiel 3: So beschreiben Sie AMIs anhand von Tags

Das folgende `describe-images`-Beispiel beschreibt alle AMIs, die das Tag `Type=Custom` haben. Das Beispiel verwendet den `--query`-Parameter, um nur die AMI-IDs anzuzeigen.

```
aws ec2 describe-images \
    --filters "Name=tag:Type,Values=Custom" \
```

```
    --query 'Images[*].[ImageId]' \
    --output text
```

Ausgabe:

```
ami-1234567890EXAMPLE
ami-0abcdef1234567890
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter Arbeiten mit Tags im Amazon-EC2-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeImagesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";

/**
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all
 of the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize
 is just sugar for
    // the MaxResults property in the base command.
    { client, pageSize },
    {
```

```
      // There are almost 70,000 images available. Be specific with your
 filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
client-ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: [architecture] }],
    },
  );

  /**
   * @type {import('@aws-sdk/client-ec2').Image[]}
   */
  const images = [];
  let recordsScanned = 0;

  try {
    for await (const page of paginator) {
      recordsScanned += pageSize;
      if (page.Images.length) {
        images.push(...page.Images);
        break;
      }
      console.log(
        `No matching image found yet. Searched ${recordsScanned} records.`,
      );
    }

    if (images.length) {
      console.log(
        `Found ${images.length} images:\n\n${images.map((image) =>
 image.Name).join("\n")}\n`,
      );
    } else {
      console.log(
        `No matching images found. Searched ${recordsScanned} records.\n`,
      );
    }

    return images;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
```

```
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene AMI beschrieben.

```
Get-EC2Image -ImageId ami-12345678
```

Ausgabe:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Beispiel 2: In diesem Beispiel die AMIs beschrieben, die Sie besitzen.

```
Get-EC2Image -owner self
```

Beispiel 3: In diesem Beispiel werden die öffentlichen AMIs beschrieben, auf denen Microsoft Windows Server ausgeführt wird.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Beispiel 4: In diesem Beispiel werden alle öffentlichen AMIs in der Region „us-west-2“ beschrieben.

```
Get-EC2Image -Region us-west-2
```

- Einzelheiten zur API finden Sie unter DescribeImages AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene AMI beschrieben.

```
Get-EC2Image -ImageId ami-12345678
```

Ausgabe:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
```

```
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Beispiel 2: In diesem Beispiel die AMIs beschrieben, die Sie besitzen.

```
Get-EC2Image -owner self
```

Beispiel 3: In diesem Beispiel werden die öffentlichen AMIs beschrieben, auf denen Microsoft Windows Server ausgeführt wird.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Beispiel 4: In diesem Beispiel werden alle öffentlichen AMIs in der Region „us-west-2“ beschrieben.

```
Get-EC2Image -Region us-west-2
```

- Einzelheiten zur API finden Sie unter DescribeImages AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
```

```
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
 IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
 image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
```

```
            raise
        return images
```

- Einzelheiten zur API finden Sie DescribeImagesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
       let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
       let output = self
           .client
           .describe_images()
           .set_image_ids(Some(image_ids))
           .send()
           .await?;

       let images = output.images.unwrap_or_default();
       if images.is_empty() {
           Err(EC2Error::new("No images for selected AMIs"))
       } else {
           Ok(images)
       }
   }
```

Verwenden Sie die Funktion list_images mit SSM zur Einschränkung auf Grundlage Ihrer Umgebung. Weitere Informationen zu SSM finden Sie unter https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ssm_GetParameters_section.html.

```
    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
            .into_iter()
            .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeimages( it_imageids = it_image_ids ).
     " oo_result is returned for testing purposes. "
```

```
        DATA(lt_images) = oo_result->get_images( ).
        MESSAGE 'Retrieved information about Amazon Machine Images (AMIs).' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
```

```
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DescribeImportImageTasks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImportImageTasks` verwendet wird.

CLI

AWS CLI

So überwachen Sie eine Aufgabe zum Importieren von Images

Im folgenden Beispiel für `describe-import-image-tasks` wird der Status der angegebenen Image-Importaufgabe überprüft.

```
aws ec2 describe-import-image-tasks \
    --import-task-ids import-ami-1234567890abcdef0
```

Ausgabe für eine Aufgabe zum Importieren eines Image, die gerade ausgeführt wird.

```
{
```

```
    "ImportImageTasks": [
        {
            "ImportTaskId": "import-ami-1234567890abcdef0",
            "Progress": "28",
            "SnapshotDetails": [
                {
                    "DiskImageSize": 705638400.0,
                    "Format": "ova",
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "my-import-bucket",
                        "S3Key": "vms/my-server-vm.ova"
                    }
                }
            ],
            "Status": "active",
            "StatusMessage": "converting"
        }
    ]
}
```

Ausgabe für eine Aufgabe zum Importieren eines Image, die abgeschlossen ist. Die ID des resultierenden AMI wird von `ImageId` bereitgestellt.

```
{
    "ImportImageTasks": [
        {
            "ImportTaskId": "import-ami-1234567890abcdef0",
            "ImageId": "ami-1234567890abcdef0",
            "SnapshotDetails": [
                {
                    "DiskImageSize": 705638400.0,
                    "Format": "ova",
                    "SnapshotId": "snap-1234567890abcdef0"
                    "Status": "completed",
                    "UserBucket": {
                        "S3Bucket": "my-import-bucket",
                        "S3Key": "vms/my-server-vm.ova"
                    }
                }
            ],
            "Status": "completed"
        }
```

```
    ]
}
```

- Einzelheiten zur API finden Sie DescribeImportImageTasksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Ausgabe:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask
```

Ausgabe:

```
Architecture    :
Description     : Windows Image 1
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
```

```
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- Einzelheiten zur API finden Sie unter DescribeImportImageTasks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Ausgabe:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask
```

Ausgabe:

```
Architecture    :
Description     : Windows Image 1
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- Einzelheiten zur API finden Sie unter DescribeImportImageTasks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeImportSnapshotTasks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImportSnapshotTasks` verwendet wird.

CLI

AWS CLI

So überwachen Sie eine Snapshot-Importaufgabe

Im folgenden Beispiel für `describe-import-snapshot-tasks` wird der Status der angegebenen Aufgabe zum Importieren eines Snapshots überprüft.

```
aws ec2 describe-import-snapshot-tasks \
    --import-task-ids import-snap-1234567890abcdef0
```

Ausgabe für eine Aufgabe zum Importieren eines Snapshots, die gerade ausgeführt wird.

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "My server VMDK",
            "ImportTaskId": "import-snap-1234567890abcdef0",
            "SnapshotTaskDetail": {
                "Description": "My server VMDK",
                "DiskImageSize": "705638400.0",
                "Format": "VMDK",
                "Progress": "42",
                "Status": "active",
                "StatusMessage": "downloading/converting",
                "UserBucket": {
                    "S3Bucket": "my-import-bucket",
                    "S3Key": "vms/my-server-vm.vmdk"
                }
            }
        }
    ]
}
```

Ausgabe für eine Aufgabe zum Importieren eines Snapshots, die abgeschlossen ist. Die ID des resultierenden Snapshots wird von `SnapshotId` bereitgestellt.

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "My server VMDK",
```

```
            "ImportTaskId": "import-snap-1234567890abcdef0",
            "SnapshotTaskDetail": {
                "Description": "My server VMDK",
                "DiskImageSize": "705638400.0",
                "Format": "VMDK",
                "SnapshotId": "snap-1234567890abcdef0"
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "my-import-bucket",
                    "S3Key": "vms/my-server-vm.vmdk"
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeImportSnapshotTasksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren eines Snapshots beschrieben.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Snapshops beschrieben.

```
Get-EC2ImportSnapshotTask
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Einzelheiten zur API finden Sie unter DescribeImportSnapshotTasks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren eines Snapshots beschrieben.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Snapshops beschrieben.

```
Get-EC2ImportSnapshotTask
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------
```

```
Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Einzelheiten zur API finden Sie unter DescribeImportSnapshotTasks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstanceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceAttribute` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Instance-Typ

In diesem Beispiel wird der Instance-Typ der angegebenen Instance beschrieben.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute instanceType
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0"
    "InstanceType": {
        "Value": "t1.micro"
    }
}
```

Um das ApiTermination Disable-Attribut zu beschreiben

In diesem Beispiel wird das `disableApiTermination`-Attribut der angegebenen Instance beschrieben.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute disableApiTermination
```

Ausgabe:

```
{
"InstanceId": "i-1234567890abcdef0"
    "DisableApiTermination": {
        "Value": "false"
    }
}
```

So beschreiben Sie Blockgerät-Zuweisung für eine Instance

In diesem Beispiel wird das `blockDeviceMapping`-Attribut der angegebenen Instance beschrieben.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute blockDeviceMapping
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0"
    "BlockDeviceMappings": [
        {
            "DeviceName": "/dev/sda1",
            "Ebs": {
                "Status": "attached",
                "DeleteOnTermination": true,
                "VolumeId": "vol-049df61146c4d7901",
                "AttachTime": "2013-05-17T22:42:34.000Z"
            }
        },
        {
            "DeviceName": "/dev/sdf",
            "Ebs": {
                "Status": "attached",
                "DeleteOnTermination": false,
```

```
                "VolumeId": "vol-049df61146c4d7901",
                "AttachTime": "2013-09-10T23:07:00.000Z"
            }
        }
    ],
}
```

- Einzelheiten zur API finden Sie DescribeInstanceAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt den Instance-Typ der angegebenen Instance.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Ausgabe:

```
InstanceType                      : t2.micro
```

Beispiel 2: In diesem Beispiel wird beschrieben, ob Enhanced Networking für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
SriovNetSupport                   : simple
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance beschrieben.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Ausgabe:

```
GroupId
-------
```

```
sg-12345678
sg-45678901
```

Beispiel 4: In diesem Beispiel wird beschrieben, ob die EBS-Optimierung für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ausgabe:

```
EbsOptimized                      : False
```

Beispiel 5: Dieses Beispiel beschreibt das ApiTermination 'Disable'-Attribut der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ausgabe:

```
DisableApiTermination             : False
```

Beispiel 6: Dieses Beispiel beschreibt das Attribut „InstanzInitiatedShutdownBehavior“ der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Ausgabe:

```
InstanceInitiatedShutdownBehavior : stop
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Instance-Typ der angegebenen Instance.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Ausgabe:

```
InstanceType                      : t2.micro
```

Beispiel 2: In diesem Beispiel wird beschrieben, ob Enhanced Networking für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
SriovNetSupport                   : simple
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance beschrieben.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Ausgabe:

```
GroupId
-------
sg-12345678
sg-45678901
```

Beispiel 4: In diesem Beispiel wird beschrieben, ob die EBS-Optimierung für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ausgabe:

```
EbsOptimized                      : False
```

Beispiel 5: Dieses Beispiel beschreibt das ApiTermination 'Disable'-Attribut der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ausgabe:

```
DisableApiTermination             : False
```

Beispiel 6: Dieses Beispiel beschreibt das Attribut „InstanzInitiatedShutdownBehavior“ der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Ausgabe:

```
InstanceInitiatedShutdownBehavior : stop
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeInstanceStatus`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceStatus` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Status einer Instance

Das folgende `describe-instance-status`-Beispiel beschreibt den aktuellen Status der angegebenen Instance.

```
aws ec2 describe-instance-status \
    --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
    "InstanceStatuses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "InstanceState": {
```

```
                "Code": 16,
                "Name": "running"
            },
            "AvailabilityZone": "us-east-1d",
            "SystemStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            },
            "InstanceStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Den Status Ihrer Instances überwachen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DescribeInstanceStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance beschrieben.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Ausgabe:

```
AvailabilityZone : us-west-2a
```

```
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Ausgabe:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

- Einzelheiten zur API finden Sie unter DescribeInstanceStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance beschrieben.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Ausgabe:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Ausgabe:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

- Einzelheiten zur API finden Sie unter DescribeInstanceStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str =
 Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider =
 RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}:", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "  Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("    Event ID:     {}",
 event.instance_event_id().unwrap());
                println!("    Description:  {}", event.description().unwrap());
                println!("    Event code:   {}", event.code().unwrap().as_ref());
                println!();
            }
        }
    }

    Ok(())
```

```
}
```

- Einzelheiten zur API finden Sie DescribeInstanceStatusin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`DescribeInstanceTypes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceTypes` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
```

```
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size
strings conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g.,
 x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
###############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-
t|--type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance
 types (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
```

```
    -a | --architecture)
      architecture="$2"
      shift 2
      ;;
    -t | --type)
      instance_types="$2"
      shift 2
      ;;
    -h | --help)
      usage
      return 0
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
  echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
  if [[ $i -lt $((array_size - 1)) ]]; then
```

```
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
```

```
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen Instance-Typ

Im folgenden Beispiel für `describe-instance-types` werden die Details für den angegebenen Instance-Typ angezeigt.

```
aws ec2 describe-instance-types \
    --instance-types t2.micro
```

Ausgabe:

```
{
    "InstanceTypes": [
        {
            "InstanceType": "t2.micro",
            "CurrentGeneration": true,
            "FreeTierEligible": true,
            "SupportedUsageClasses": [
                "on-demand",
                "spot"
            ],
            "SupportedRootDeviceTypes": [
                "ebs"
            ],
            "BareMetal": false,
            "Hypervisor": "xen",
            "ProcessorInfo": {
                "SupportedArchitectures": [
                    "i386",
                    "x86_64"
                ],
                "SustainedClockSpeedInGhz": 2.5
            },
            "VCpuInfo": {
                "DefaultVCpus": 1,
                "DefaultCores": 1,
                "DefaultThreadsPerCore": 1,
                "ValidCores": [
                    1
                ],
```

```
                "ValidThreadsPerCore": [
                    1
                ]
            },
            "MemoryInfo": {
                "SizeInMiB": 1024
            },
            "InstanceStorageSupported": false,
            "EbsInfo": {
                "EbsOptimizedSupport": "unsupported",
                "EncryptionSupport": "supported"
            },
            "NetworkInfo": {
                "NetworkPerformance": "Low to Moderate",
                "MaximumNetworkInterfaces": 2,
                "Ipv4AddressesPerInterface": 2,
                "Ipv6AddressesPerInterface": 2,
                "Ipv6Supported": true,
                "EnaSupport": "unsupported"
            },
            "PlacementGroupInfo": {
                "SupportedStrategies": [
                    "partition",
                    "spread"
                ]
            },
            "HibernationSupported": false,
            "BurstablePerformanceSupported": true,
            "DedicatedHostsSupported": false,
            "AutoRecoverySupported": true
        }
    ]
}
```

Weitere Informationen finden Sie unter Instance-Typen im Benutzerhandbuch zu Amazon Elastic Compute Cloud für Linux-Instances.

Beispiel 2: So filtern Sie die verfügbaren Instance-Typen

Sie können einen Filter angeben, um die Ergebnisse auf Instance-Typen mit einem bestimmten Merkmal zu beschränken. Im folgenden Beispiel für `describe-instance-types` werden die Instance-Typen aufgeführt, die den Ruhezustand unterstützen.

```
aws ec2 describe-instance-types \
    --filters Name=hibernation-supported,Values=true --query
 'InstanceTypes[*].InstanceType'
```

Ausgabe:

```
[
    "m5.8xlarge",
    "r3.large",
    "c3.8xlarge",
    "r5.large",
    "m4.4xlarge",
    "c4.large",
    "m5.xlarge",
    "m4.xlarge",
    "c3.large",
    "c4.8xlarge",
    "c4.4xlarge",
    "c5.xlarge",
    "c5.12xlarge",
    "r5.4xlarge",
    "c5.4xlarge"
]
```

Weitere Informationen finden Sie unter Instance-Typen im Benutzerhandbuch zu Amazon Elastic Compute Cloud für Linux-Instances.

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Asynchronously retrieves the instance types available in the current AWS
region.
    * <p>
    * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
    * and then processes the response. It logs the memory information, network
information,
    * and instance type for each instance type returned. Additionally, it
returns a
    * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
    * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type
string for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });
```

```
        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstanceTypes } from "@aws-sdk/client-ec2";

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }}
 options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the underlying command.
  const paginator = paginateDescribeInstanceTypes(
```

```
    // Without limiting the page size, this call can take a long time. pageSize
 is just sugar for
    // the MaxResults property in the underlying command.
    { client, pageSize },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: supportedArch,
        },
        { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
     */
    const instanceTypes = [];

    for await (const page of paginator) {
      if (page.InstanceTypes.length) {
        instanceTypes.push(...page.InstanceTypes);

        // When we have at least 1 result, we can stop.
        if (instanceTypes.length >= 1) {
          break;
        }
      }
    }
    console.log(
      `Memory size in MiB for matching instance types:\n\n
${instanceTypes.map((it) => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB}
 MiB`).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
```

```
    }
}
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
```

```
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
 "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types.
 Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support
 the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
 {','.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// List instance types that match an image's architecture and are free tier
eligible.
   pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
       let architecture = format!(
           "{}",
           image.architecture().ok_or_else(|| EC2Error::new(format!(
               "Image {:?} does not have a listed architecture",
               image.image_id()
           )))?
       );
       let free_tier_eligible_filter = Filter::builder()
           .name("free-tier-eligible")
           .values("false")
           .build();
       let supported_architecture_filter = Filter::builder()
```

```
            .name("processor-info.supported-architecture")
            .values(architecture)
            .build();
        let response = self
            .client
            .describe_instance_types()
            .filters(free_tier_eligible_filter)
            .filters(supported_architecture_filter)
            .send()
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
            .collect())
    }
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create filters for architecture and instance type patterns
    " iv_architecture = 'x86_64'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'processor-info.supported-architecture'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_architecture ) )
      )
```

```
    ) TO lt_filters.
    " Filter for instance type patterns like '*.micro', '*.small'
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'instance-type'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.micro' ) )
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.small' ) )
      )
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeinstancetypes( it_filters = lt_filters ).
         " oo_result is returned for testing purposes. "
        DATA(lt_instance_types) = oo_result->get_instancetypes( ).
        MESSAGE 'Retrieved information about EC2 instance types.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
```

```
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
 = EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

- Einzelheiten zur API finden Sie DescribeInstanceTypesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeInstances`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstances` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Verwalten eines ausfallsicheren Services
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Einen Containerdienst für VM-Instanzen erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information about EC2 instances with a particular state.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
```

```
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
```

```
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated
 with an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
```

```
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                          std::setw(48) << "Name" <<
                          std::setw(20) << "ID" <<
                          std::setw(25) << "Ami" <<
                          std::setw(15) << "Type" <<
                          std::setw(15) << "State" <<
                          std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                    outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
                        reservation.GetInstances();
                for (const auto &instance: instances) {
                    Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                    instance.GetState().GetName());

                    Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                    instance.GetInstanceType());

                    Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                    instance.GetMonitoring().GetState());
                    Aws::String name = "Unknown";

                    const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                 [](const Aws::EC2::Model::Tag
&tag) {
                                                     return tag.GetKey() ==
"Name";
```

```
                                                    });
                    if (nameIter != tags.cend()) {
                        name = nameIter->GetValue();
                    }
                    std::cout <<
                              std::setw(48) << name <<
                              std::setw(20) << instance.GetInstanceId() <<
                              std::setw(25) << instance.GetImageId() <<
                              std::setw(15) << typeString <<
                              std::setw(15) << instanceStateString <<
                              std::setw(15) << monitorString << std::endl;
                }
            }

            if (!outcome.GetResult().GetNextToken().empty()) {
                request.SetNextToken(outcome.GetResult().GetNextToken());
            } else {
                done = true;
            }
        } else {
            std::cerr << "Failed to describe EC2 instances:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Instance

Das folgende Beispiel für `describe-instances` beschreibt die angegebene Instance.

```
aws ec2 describe-instances \
    --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0abcdef1234567890",
                    "InstanceId": "i-1234567890abcdef0",
                    "InstanceType": "t3.nano",
                    "KeyName": "my-key-pair",
                    "LaunchTime": "2022-11-15T10:48:59+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-2a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
                    "PrivateIpAddress": "10-0-0-157",
                    "ProductCodes": [],
                    "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                    "PublicIpAddress": "34.253.223.13",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-04a636d18e83cfacb",
                    "VpcId": "vpc-1234567890abcdef0",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2022-11-15T10:49:00+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-02e6ccdca7de29cf2"
```

```
                            }
                        }
                    ],
                    "ClientToken": "1234abcd-1234-abcd-1234-d46a8903e9bc",
                    "EbsOptimized": true,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "IamInstanceProfile": {
                        "Arn": "arn:aws:iam::111111111111:instance-profile/
AmazonSSMRoleForInstancesQuickSetup",
                        "Id": "11111111111111111111"
                    },
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
                                "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                                "PublicIp": "34.253.223.13"
                            },
                            "Attachment": {
                                "AttachTime": "2022-11-15T10:48:59+00:00",
                                "AttachmentId": "eni-attach-1234567890abcdefg",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupName": "launch-wizard-146",
                                    "GroupId": "sg-1234567890abcdefg"
                                }
                            ],
                            "Ipv6Addresses": [],
                            "MacAddress": "00:11:22:33:44:55",
                            "NetworkInterfaceId": "eni-1234567890abcdefg",
                            "OwnerId": "104024344472",
                            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                            "PrivateIpAddress": "10-0-0-157",
                            "PrivateIpAddresses": [
                                {
```

```
                                    "Association": {
                                        "IpOwnerId": "amazon",
                                        "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                                        "PublicIp": "34.253.223.13"
                                    },
                                    "Primary": true,
                                    "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                                    "PrivateIpAddress": "10-0-0-157"
                                }
                            ],
                            "SourceDestCheck": true,
                            "Status": "in-use",
                            "SubnetId": "subnet-1234567890abcdefg",
                            "VpcId": "vpc-1234567890abcdefg",
                            "InterfaceType": "interface"
                        }
                    ],
                    "RootDeviceName": "/dev/xvda",
                    "RootDeviceType": "ebs",
                    "SecurityGroups": [
                        {
                            "GroupName": "launch-wizard-146",
                            "GroupId": "sg-1234567890abcdefg"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Tags": [
                        {
                            "Key": "Name",
                            "Value": "my-instance"
                        }
                    ],
                    "VirtualizationType": "hvm",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 2
                    },
                    "CapacityReservationSpecification": {
                        "CapacityReservationPreference": "open"
                    },
                    "HibernationOptions": {
                        "Configured": false
```

```
                    },
                    "MetadataOptions": {
                        "State": "applied",
                        "HttpTokens": "optional",
                        "HttpPutResponseHopLimit": 1,
                        "HttpEndpoint": "enabled",
                        "HttpProtocolIpv6": "disabled",
                        "InstanceMetadataTags": "enabled"
                    },
                    "EnclaveOptions": {
                        "Enabled": false
                    },
                    "PlatformDetails": "Linux/UNIX",
                    "UsageOperation": "RunInstances",
                    "UsageOperationUpdateTime": "2022-11-15T10:48:59+00:00",
                    "PrivateDnsNameOptions": {
                        "HostnameType": "ip-name",
                        "EnableResourceNameDnsARecord": true,
                        "EnableResourceNameDnsAAAARecord": false
                    },
                    "MaintenanceOptions": {
                        "AutoRecovery": "default"
                    }
                }
            ],
            "OwnerId": "111111111111",
            "ReservationId": "r-1234567890abcdefg"
        }
    ]
}
```

Beispiel 2: So filtern Sie nach Instances mit dem angegebenen Typ

Im folgenden Beispiel für `describe-instances` werden Filter verwendet, um die Ergebnisse auf Instances des angegebenen Typs zu beschränken.

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=m5.large
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Mit der CLI auflisten und filtern im Benutzerhandbuch für Amazon EC2.

Beispiel 3: So filtern Sie nach Instances mit dem angegebenen Typ und der angegebenen Availability Zone

Im folgenden Beispiel für `describe-instances` werden mehrere Filter verwendet, um die Ergebnisse auf Instances mit dem angegebenen Typ zu beschränken, die sich ebenfalls in der angegebenen Availability Zone befinden.

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-
zone,Values=us-east-2c
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 4: So filtern Sie mithilfe einer JSON-Datei nach Instances mit dem angegebenen Typ und der angegebenen Availability Zone

Das folgende `describe-instances`-Beispiel verwendet eine JSON-Eingabedatei, um dieselbe Filterung wie im vorherigen Beispiel durchzuführen. Wenn Filter komplizierter werden, können sie einfacher in einer JSON-Datei angegeben werden.

```
aws ec2 describe-instances \
    --filters file://filters.json
```

Inhalt von `filters.json`:

```
[
    {
        "Name": "instance-type",
        "Values": ["t2.micro", "t3.micro"]
    },
    {
        "Name": "availability-zone",
        "Values": ["us-east-2c"]
    }
]
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 5: So filtern Sie nach Instances mit dem angegebenen Owner-Tag

Im folgenden Beispiel für `describe-instances` werden Tag-Filter verwendet, um die Ergebnisse unabhängig vom Tag-Wert auf Instances zu beschränken, die über ein Tag mit dem angegebenen Tag-Schlüssel (Owner) verfügen.

```
aws ec2 describe-instances \
    --filters "Name=tag-key,Values=Owner"
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 6: So filtern Sie nach Instances mit dem angegebenen my-team-Tag-Wert

Im folgenden Beispiel für `describe-instances` werden Tag-Filter verwendet, um die Ergebnisse auf Instances zu beschränken, die ein Tag mit dem angegebenen Tag-Wert (my-team) haben, unabhängig vom Tag-Schlüssel.

```
aws ec2 describe-instances \
    --filters "Name=tag-value,Values=my-team"
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 7: So filtern Sie nach Instances mit dem angegebenen Besitzer-Tag und my-team-Wert

Im folgenden Beispiel für `describe-instances` werden Tag-Filter verwendet, um die Ergebnisse auf Instances zu beschränken, die das angegebene Tag haben (Besitzer=my-team).

```
aws ec2 describe-instances \
    --filters "Name=tag:Owner,Values=my-team"
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 8: So zeigen Sie nur Instance- und Subnetz-IDs für alle Instances an

In den folgenden `describe-instances`-Beispielen wird der `--query`-Parameter verwendet, um nur die Instance- und Subnetz-IDs für alle Instances im JSON-Format anzuzeigen.

Linux und macOS:

```
aws ec2 describe-instances \
    --query 'Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}'
 \
```

```
    --output json
```

Windows:

```
aws ec2 describe-instances ^
    --query "Reservations[*].Instances[*].
{Instance:InstanceId,Subnet:SubnetId}" ^
    --output json
```

Ausgabe:

```
[
    {
        "Instance": "i-057750d42936e468a",
        "Subnet": "subnet-069beee9b12030077"
    },
    {
        "Instance": "i-001efd250faaa6ffa",
        "Subnet": "subnet-0b715c6b7db68927a"
    },
    {
        "Instance": "i-027552a73f021f3bd",
        "Subnet": "subnet-0250c25a1f4e15235"
    }
    ...
]
```

Beispiel 9: So filtern Sie Instances des angegebenen Typs und zeigen nur ihre Instance-IDs an

Im folgenden Beispiel für `describe-instances` werden Filter verwendet, um die Ergebnisse auf Instances des angegebenen Typs zu beschränken und der `--query`-Parameter, um nur die Instance-IDs anzuzeigen.

```
aws ec2 describe-instances \
    --filters "Name=instance-type,Values=t2.micro" \
    --query "Reservations[*].Instances[*].[InstanceId]" \
    --output text
```

Ausgabe:

```
i-031c0dc19de2fb70c
i-00d8bff789a736b75
```

```
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c
```

Beispiel 10: So filtern Sie Instances des angegebenen Typs und zeigen nur deren Instance-IDs, Availability Zone und den angegebenen Tag-Wert an

In den folgenden `describe-instances`-Beispielen werden die Instance-ID, die Availability Zone und der Wert des `Name`-Tags für Instances, die ein Tag mit dem Namen `tag-key` haben, im Tabellenformat angezeigt.

Linux und macOS:

```
aws ec2 describe-instances \
    --filters Name=tag-key,Values=Name \
    --query 'Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key==`Name`]|
[0].Value}' \
    --output table
```

Windows:

```
aws ec2 describe-instances ^
    --filters Name=tag-key,Values=Name ^
    --query "Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name']|
[0].Value}" ^
    --output table
```

Ausgabe:

```
-------------------------------------------------------------
|                     DescribeInstances                     |
+--------------+-----------------------+--------------------+
|      AZ      |       Instance        |        Name        |
+--------------+-----------------------+--------------------+
|  us-east-2b  |  i-057750d42936e468a  |  my-prod-server    |
|  us-east-2a  |  i-001efd250faaa6ffa  |  test-server-1     |
|  us-east-2a  |  i-027552a73f021f3bd  |  test-server-2     |
+--------------+-----------------------+--------------------+
```

Beispiel 11: So beschreiben Sie Instances in einer Partition-Placement-Gruppe

Das folgende `describe-instances`-Beispiel beschreibt die angegebene Instance. Die Ausgabe enthält die Platzierungsinformationen für die Instance, die den Namen der Platzierungsgruppe und die Partitionsnummer für die Instance enthalten.

```
aws ec2 describe-instances \
    --instance-ids i-0123a456700123456 \
    --query "Reservations[*].Instances[*].Placement"
```

Ausgabe:

```
[
    [
        {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 3,
            "Tenancy": "default"
        }

    ]
]
```

Weitere Informationen finden Sie unter Beschreiben von Instances in einer Platzierungsgruppe im Amazon-EC2-Benutzerhandbuch.

Beispiel 12: So filtern Sie auf Instances mit der angegebenen Platzierungsgruppe und Partitionsnummer

Im folgenden Beispiel für `describe-instances` werden die Ergebnisse nur nach den Instances mit der angegebenen Platzierungsgruppe und Partitionsnummer gefiltert.

```
aws ec2 describe-instances \
    --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
partition-number,Values=7"
```

Im Folgenden werden nur die relevanten Informationen aus der Ausgabe angezeigt.

```
"Instances": [
    {
```

```
        "InstanceId": "i-0123a456700123456",
        "InstanceType": "r4.large",
        "Placement": {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 7,
            "Tenancy": "default"
        }
    },
    {
        "InstanceId": "i-9876a543210987654",
        "InstanceType": "r4.large",
        "Placement": {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 7,
            "Tenancy": "default"
        }
    ],
```

Weitere Informationen finden Sie unter Beschreiben von Instances in einer Platzierungsgruppe im Amazon-EC2-Benutzerhandbuch.

Beispiel 13: So filtern Sie nach Instances, die so konfiguriert sind, dass sie den Zugriff auf Tags aus Instance-Metadaten erlauben

Im folgenden Beispiel für `describe-instances` werden die Ergebnisse nur nach den Instances gefiltert, die so konfiguriert sind, dass sie den Zugriff auf Instance-Tags aus den Instance-Metadaten ermöglichen.

```
aws ec2 describe-instances \
    --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \
    --query "Reservations[*].Instances[*].InstanceId" \
    --output text
```

Im Folgenden wird die erwartete Ausgabe dargestellt.

```
i-1234567890abcdefg
i-abcdefg1234567890
i-11111111aaaaaaaaa
i-aaaaaaaa111111111
```

Weitere Informationen finden Sie unter Mit Instance-Tags in Instance-Metadaten arbeiten im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the state of an EC2 instance.
    * The paginator helps you iterate over multiple pages of results.
    *
    * @param newInstanceId the ID of the EC2 instance to describe
    * @return a {@link CompletableFuture} that, when completed, contains a
string describing the state of the EC2 instance
    */
   public CompletableFuture<String> describeEC2InstancesAsync(String
newInstanceId) {
       DescribeInstancesRequest request = DescribeInstancesRequest.builder()
           .instanceIds(newInstanceId)
           .build();

       DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
       AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.reservations().stream()
               .flatMap(reservation -> reservation.instances().stream())
               .filter(instance -> instance.instanceId().equals(newInstanceId))
               .findFirst()
               .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
       }).thenApply(v -> {
           String publicIpAddress = publicIpAddressRef.get();
           if (publicIpAddress == null) {
```

```
                throw new RuntimeException("Instance with ID " + newInstanceId +
 " not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstances } from "@aws-sdk/client-ec2";

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;

  const paginator = paginateDescribeInstances(
    {
```

```
      client,
      pageSize,
    },
    {
      Filters: [
        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
          Name: "launch-time",
          Values: [launchTimePattern],
        },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').Instance[]}
     */
    const instanceList = [];
    for await (const page of paginator) {
      const { Reservations } = page;
      for (const reservation of Reservations) {
        instanceList.push(...reservation.Instances);
      }
    }
    console.log(
      `Running instances launched this month:\n\n${instanceList.map((instance) =>
 instance.InstanceId).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is
 ${instance.monitoring?.state}")
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeInstancesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beschrieben.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Ausgabe:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Instances in der aktuellen Region beschrieben, gruppiert nach Reservierungen. Um die Instance-Details anzuzeigen, erweitern Sie die Sammlung der Instances innerhalb jedes Reservierungsobjekts.

```
Get-EC2Instance
```

Ausgabe:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Beispiel 3: Dieses Beispiel veranschaulicht die Verwendung eines Filters zur Abfrage von EC2-Instances in einem bestimmten Subnetz einer VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 4: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten zur Abfrage von EC2-Instances, die sowohl ausgeführt werden als auch angehalten sind

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 5: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten, um nach EC2-Instances abzufragen, die sowohl ausgeführt als auch gestoppt wurden, und die Verwendung des Select-Object Cmdlets zur Auswahl bestimmter Werte für die Ausgabe.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
```

```
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

- Einzelheiten zur API finden Sie unter DescribeInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beschrieben.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Ausgabe:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
```

```
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Instances in der aktuellen Region beschrieben, gruppiert nach Reservierungen. Um die Instance-Details anzuzeigen, erweitern Sie die Sammlung der Instances innerhalb jedes Reservierungsobjekts.

```
Get-EC2Instance
```

Ausgabe:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c
```

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Beispiel 3: Dieses Beispiel veranschaulicht die Verwendung eines Filters zur Abfrage von EC2-Instances in einem bestimmten Subnetz einer VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 4: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten zur Abfrage von EC2-Instances, die sowohl ausgeführt werden als auch angehalten sind

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
```

```
i-05a9...f6c5f46e18 t3.medium            10.0.1.7                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
         subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 5: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten für die Abfrage von EC2-Instances, die sowohl ausgeführt als auch gestoppt sind, und die Verwendung des Select-Object Cmdlets zur Auswahl bestimmter Werte für die Ausgabe verwendet wird.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
i-05a9...f6c5f46e18 t3.medium            10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
```

```
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

Beispiel 6: In diesem Beispiel werden Berechtigungen zum Abrufen von EC2-Instances mithilfe des DryRun Parameters validiert, ohne sie tatsächlich abzurufen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Get-EC2Tag -DryRun $true
```

Ausgabe:

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- Einzelheiten zur API finden Sie unter DescribeInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
```

```
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
instances in this state
                              will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
```

```
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match
 the filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise
```

- Einzelheiten zur API finden Sie DescribeInstancesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
```

```
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Details einer EC2-Instance ab.

```
   pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
 EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
 {instance_id}")))?
            .instances()
            .first()
```

```
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }
```

Nachdem Sie eine EC2-Instance erstellt haben, rufen Sie deren Details ab und speichern Sie diese.

```
    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }
```

- Einzelheiten zur API finden Sie DescribeInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeinstances( ).                       "
oo_result is returned for testing purposes. "

        " Retrieving details of EC2 instances. "
        DATA: lv_istance_id    TYPE /aws1/ec2string,
              lv_status        TYPE /aws1/ec2instancestatename,
              lv_instance_type TYPE /aws1/ec2instancetype,
              lv_image_id      TYPE /aws1/ec2string.
        LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
          LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_istance_id = lo_instance->get_instanceid( ).
            lv_status = lo_instance->get_state( )->get_name( ).
            lv_instance_type = lo_instance->get_instancetype( ).
            lv_image_id = lo_instance->get_imageid( ).
          ENDLOOP.
        ENDLOOP.
        MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeInternetGateways`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInternetGateways` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk

CLI

AWS CLI

So beschreiben Sie ein Internet-Gateway

Im folgenden Beispiel für `describe-internet-gateways` wird das angegebene Internet-Gateway beschrieben.

```
aws ec2 describe-internet-gateways \
    --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

Ausgabe:

```
{
    "InternetGateways": [
        {
            "Attachments": [
                {
                    "State": "available",
                    "VpcId": "vpc-0a60eb65b4EXAMPLE"
                }
            ],
            "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",
            "OwnerId": "123456789012",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "my-igw"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Internet-Gateways im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie DescribeInternetGatewaysin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway beschrieben.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Beispiel 2: In diesem Beispiel werden alle Ihre Internet-Gateways beschrieben.

```
Get-EC2InternetGateway
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Einzelheiten zur API finden Sie unter DescribeInternetGateways AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway beschrieben.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
```

```
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Beispiel 2: In diesem Beispiel werden alle Ihre Internet-Gateways beschrieben.

```
Get-EC2InternetGateway
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- Einzelheiten zur API finden Sie unter DescribeInternetGateways AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeKeyPairs`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeKeyPairs` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Big-Data-Verarbeitungsclustern
- Erste Schritte mit verwaltetem Streaming

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }

            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 key pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
```

```
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeKeyPairs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
 ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "Name" <<
                  std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
                outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                      std::setw(32) << key_pair.GetKeyName() <<
                      std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So zeigen Sie ein Schlüsselpaar an

Im folgenden Beispiel für `describe-key-pairs` werden Informationen zu dem angegebenen Schlüsselpaar angezeigt.

```
aws ec2 describe-key-pairs \
```

```
    --key-names my-key-pair
```

Ausgabe:

```
{
    "KeyPairs": [
        {
            "KeyPairId": "key-0b94643da6EXAMPLE",
            "KeyFingerprint":
 "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f",
            "KeyName": "my-key-pair",
            "KeyType": "rsa",
            "Tags": [],
            "CreateTime": "2022-05-27T21:51:16.000Z"
        }
    ]
}
```

Weitere Informationen finden Sie unter Öffentliche Schlüssel beschreiben im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the key pairs associated with the current AWS
 account.
     *
     * @return a {@link CompletableFuture} containing the {@link
 DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
```

```
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
 getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
 exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeKeyPairsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
```

```
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar beschrieben.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
KeyFingerprint                                          KeyName
--------------                                          -------
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Beispiel 2: In diesem Beispiel werden alle Ihre Schlüsselpaare beschrieben.

```
Get-EC2KeyPair
```

- Einzelheiten zur API finden Sie unter DescribeKeyPairs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar beschrieben.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
KeyFingerprint                                          KeyName
--------------                                          -------
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Beispiel 2: In diesem Beispiel werden alle Ihre Schlüsselpaare beschrieben.

```
Get-EC2KeyPair
```

- Einzelheiten zur API finden Sie unter DescribeKeyPairs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir
```

```
    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
    let output = self.client.describe_key_pairs().send().await?;
    Ok(output.key_pairs.unwrap_or_default())
}
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_ec2->describekeypairs( ).                        "
oo_result is returned for testing purposes. "
       DATA(lt_key_pairs) = oo_result->get_keypairs( ).
       MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )

            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":",
 keyPair.keyFingerprint ?? "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeKeyPairsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DescribeNetworkAcls`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNetworkAcls` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Netzwerk-ACLs

Im folgenden Beispiel für `describe-network-acls` werden Details über Ihre Netzwerk-ACLs abgerufen.

```
aws ec2 describe-network-acls
```

Ausgabe:

```
{
    "NetworkAcls": [
        {
            "Associations": [
                {
                    "NetworkAclAssociationId": "aclassoc-0c1679dc41EXAMPLE",
                    "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
                    "SubnetId": "subnet-0931fc2fa5EXAMPLE"
                }
            ],
            "Entries": [
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 100
                },
                {
```

```
                "CidrBlock": "0.0.0.0/0",
                "Egress": true,
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32767
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": false,
                "Protocol": "-1",
                "RuleAction": "allow",
                "RuleNumber": 100
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": false,
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32767
            }
        ],
        "IsDefault": true,
        "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
        "Tags": [],
        "VpcId": "vpc-06e4ab6c6cEXAMPLE",
        "OwnerId": "111122223333"
    },
    {
        "Associations": [],
        "Entries": [
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": true,
                "Protocol": "-1",
                "RuleAction": "allow",
                "RuleNumber": 100
            },
            {
                "Egress": true,
                "Ipv6CidrBlock": "::/0",
                "Protocol": "-1",
                "RuleAction": "allow",
                "RuleNumber": 101
            },
```

```
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": true,
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32767
                },
                {
                    "Egress": true,
                    "Ipv6CidrBlock": "::/0",
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32768
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": false,
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 100
                },
                {
                    "Egress": false,
                    "Ipv6CidrBlock": "::/0",
                    "Protocol": "-1",
                    "RuleAction": "allow",
                    "RuleNumber": 101
                },
                {
                    "CidrBlock": "0.0.0.0/0",
                    "Egress": false,
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32767
                },
                {
                    "Egress": false,
                    "Ipv6CidrBlock": "::/0",
                    "Protocol": "-1",
                    "RuleAction": "deny",
                    "RuleNumber": 32768
                }
            ],
            "IsDefault": true,
```

```
            "NetworkAclId": "acl-0e2a78e4e2EXAMPLE",
            "Tags": [],
            "VpcId": "vpc-03914afb3eEXAMPLE",
            "OwnerId": "111122223333"
        }
    ]
}
```

Weitere Informationen finden Sie unter Netzwerk-ACLs im Benutzerhandbuch für AWS VPC.

- Einzelheiten zur API finden Sie DescribeNetworkAclsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL beschrieben.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden die Regeln für die angegebene Netzwerk-ACL beschrieben.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Ausgabe:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
```

```
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Beispiel 3: In diesem Beispiel werden alle Ihre Netzwerk-ACLs beschrieben.

```
Get-EC2NetworkAcl
```

- Einzelheiten zur API finden Sie unter DescribeNetworkAcls AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL beschrieben.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden die Regeln für die angegebene Netzwerk-ACL beschrieben.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Ausgabe:

```
CidrBlock    : 0.0.0.0/0
```

```
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Beispiel 3: In diesem Beispiel werden alle Ihre Netzwerk-ACLs beschrieben.

```
Get-EC2NetworkAcl
```

- Einzelheiten zur API finden Sie unter DescribeNetworkAcls AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeNetworkInterfaceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNetworkInterfaceAttribute` verwendet wird.

CLI

AWS CLI

So beschreiben Sie das attachment-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `attachment`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute attachment
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Attachment": {
      "Status": "attached",
      "DeviceIndex": 0,
      "AttachTime": "2015-05-21T20:02:20.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "DeleteOnTermination": true,
      "AttachmentId": "eni-attach-43348162",
      "InstanceOwnerId": "123456789012"
  }
}
```

So beschreiben Sie das description-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `description`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute description
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
      "Value": "My description"
  }
}
```

So beschreiben Sie das groupSet-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `groupSet`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute groupSet
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
      {
          "GroupName": "my-security-group",
          "GroupId": "sg-903004f8"
      }
  ]
}
```

Um das DestCheck Quellattribut einer Netzwerkschnittstelle zu beschreiben

Dieser Beispielbefehl beschreibt das `sourceDestCheck`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute sourceDestCheck
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
      "Value": true
  }
}
```

- Einzelheiten zur API finden Sie DescribeNetworkInterfaceAttributeunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Ausgabe:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Beispiel 2: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Ausgabe:

```
Description        : My description
```

Beispiel 3: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Ausgabe:

```
Groups             : {my-security-group}
```

Beispiel 4: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Ausgabe:

```
SourceDestCheck    : True
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Ausgabe:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Beispiel 2: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Ausgabe:

```
Description        : My description
```

Beispiel 3: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Ausgabe:

```
Groups             : {my-security-group}
```

Beispiel 4: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Ausgabe:

```
SourceDestCheck    : True
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeNetworkInterfaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNetworkInterfaces` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

CLI

AWS CLI

So beschreiben Sie Ihre Netzwerkschnittstellen

In diesem Beispiel werden alle Ihre Netzwerkschnittstellen beschrieben.

Befehl:

```
aws ec2 describe-network-interfaces
```

Ausgabe:

```
{
  "NetworkInterfaces": [
      {
          "Status": "in-use",
          "MacAddress": "02:2f:8f:b0:cf:75",
          "SourceDestCheck": true,
          "VpcId": "vpc-a01106c2",
          "Description": "my network interface",
          "Association": {
              "PublicIp": "203.0.113.12",
              "AssociationId": "eipassoc-0fbb766a",
              "PublicDnsName": "ec2-203-0-113-12.compute-1.amazonaws.com",
              "IpOwnerId": "123456789012"
          },
          "NetworkInterfaceId": "eni-e5aa89a3",
          "PrivateIpAddresses": [
              {
                  "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
                  "Association": {
                      "PublicIp": "203.0.113.12",
```

```
                        "AssociationId": "eipassoc-0fbb766a",
                        "PublicDnsName":
"ec2-203-0-113-12.compute-1.amazonaws.com",
                        "IpOwnerId": "123456789012"
                    },
                    "Primary": true,
                    "PrivateIpAddress": "10.0.1.17"
                }
            ],
            "RequesterManaged": false,
            "Ipv6Addresses": [],
            "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
            "AvailabilityZone": "us-east-1d",
            "Attachment": {
                "Status": "attached",
                "DeviceIndex": 1,
                "AttachTime": "2013-11-30T23:36:42.000Z",
                "InstanceId": "i-1234567890abcdef0",
                "DeleteOnTermination": false,
                "AttachmentId": "eni-attach-66c4350a",
                "InstanceOwnerId": "123456789012"
            },
            "Groups": [
                {
                    "GroupName": "default",
                    "GroupId": "sg-8637d3e3"
                }
            ],
            "SubnetId": "subnet-b61f49f0",
            "OwnerId": "123456789012",
            "TagSet": [],
            "PrivateIpAddress": "10.0.1.17"
        },
        {
            "Status": "in-use",
            "MacAddress": "02:58:f5:ef:4b:06",
            "SourceDestCheck": true,
            "VpcId": "vpc-a01106c2",
            "Description": "Primary network interface",
            "Association": {
                "PublicIp": "198.51.100.0",
                "IpOwnerId": "amazon"
            },
            "NetworkInterfaceId": "eni-f9ba99bf",
```

```
            "PrivateIpAddresses": [
                {
                    "Association": {
                        "PublicIp": "198.51.100.0",
                        "IpOwnerId": "amazon"
                    },
                    "Primary": true,
                    "PrivateIpAddress": "10.0.1.149"
                }
            ],
            "RequesterManaged": false,
            "Ipv6Addresses": [],
            "AvailabilityZone": "us-east-1d",
            "Attachment": {
                "Status": "attached",
                "DeviceIndex": 0,
                "AttachTime": "2013-11-30T23:35:33.000Z",
                "InstanceId": "i-0598c7d356eba48d7",
                "DeleteOnTermination": true,
                "AttachmentId": "eni-attach-1b9db777",
                "InstanceOwnerId": "123456789012"
            },
            "Groups": [
                {
                    "GroupName": "default",
                    "GroupId": "sg-8637d3e3"
                }
            ],
            "SubnetId": "subnet-b61f49f0",
            "OwnerId": "123456789012",
            "TagSet": [],
            "PrivateIpAddress": "10.0.1.149"
        }
    ]
}
```

In diesem Beispiel werden Netzwerkschnittstellen beschrieben, die ein Tag mit dem Schlüssel `Purpose` und dem Wert `Prod` besitzen.

Befehl:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

Ausgabe:

```
{
  "NetworkInterfaces": [
      {
          "Status": "available",
          "MacAddress": "12:2c:bd:f9:bf:17",
          "SourceDestCheck": true,
          "VpcId": "vpc-8941ebec",
          "Description": "ProdENI",
          "NetworkInterfaceId": "eni-b9a5ac93",
          "PrivateIpAddresses": [
              {
                  "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
                  "Primary": true,
                  "PrivateIpAddress": "10.0.1.55"
              },
              {
                  "PrivateDnsName": "ip-10-0-1-117.ec2.internal",
                  "Primary": false,
                  "PrivateIpAddress": "10.0.1.117"
              }
          ],
          "RequesterManaged": false,
          "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
          "AvailabilityZone": "us-east-1d",
          "Ipv6Addresses": [],
          "Groups": [
              {
                  "GroupName": "MySG",
                  "GroupId": "sg-905002f5"
              }
          ],
          "SubnetId": "subnet-31d6c219",
          "OwnerId": "123456789012",
          "TagSet": [
              {
                  "Value": "Prod",
                  "Key": "Purpose"
              }
          ],
          "PrivateIpAddress": "10.0.1.55"
      }
  ]
```

```
}
```

- Einzelheiten zur API finden Sie DescribeNetworkInterfacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Netzwerkschnittstellen beschrieben.

```
Get-EC2NetworkInterface
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Netzwerkschnittstellen beschrieben.

```
Get-EC2NetworkInterface
```

- Einzelheiten zur API finden Sie unter DescribeNetworkInterfaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribePlacementGroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePlacementGroups` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Platzierungsgruppen

Dieser Beispielbefehl beschreibt alle Ihre Platzierungsgruppen.

Befehl:

```
aws ec2 describe-placement-groups
```

Ausgabe:

```
{
    "PlacementGroups": [
        {
            "GroupName": "my-cluster",
            "State": "available",
            "Strategy": "cluster"
        },
        ...
    ]
}
```

- Einzelheiten zur API finden Sie DescribePlacementGroupsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe beschrieben.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Einzelheiten zur API finden Sie unter DescribePlacementGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe beschrieben.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- Einzelheiten zur API finden Sie unter DescribePlacementGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribePrefixLists`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePrefixLists` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So beschreiben Sie Präfixlisten

In diesem Beispiel werden alle verfügbaren Präfixlisten für die Region aufgeführt.

Befehl:

```
aws ec2 describe-prefix-lists
```

Ausgabe:

```
{
```

```
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- Einzelheiten zur API finden Sie DescribePrefixListsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das AWS-Services in einer Präfixliste verfügbare Format für die Region abgerufen

```
Get-EC2PrefixList
```

Ausgabe:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- Einzelheiten zur API finden Sie unter DescribePrefixLists AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das AWS-Services in einer Präfixliste verfügbare Format für die Region abgerufen

```
Get-EC2PrefixList
```

Ausgabe:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- Einzelheiten zur API finden Sie unter DescribePrefixLists AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeRegions`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRegions` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeRegions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
 ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "RegionName" <<
                  std::setw(64) << "Endpoint" << std::endl;
```

```
        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                      std::setw(32) << region.GetRegionName() <<
                      std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();

}
```

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle von Ihnen aktivierten Regionen

Im folgenden Beispiel für `describe-regions` werden alle Regionen beschrieben, die für Ihr Konto aktiviert sind.

```
aws ec2 describe-regions
```

Ausgabe:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.eu-north-1.amazonaws.com",
            "RegionName": "eu-north-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-south-1.amazonaws.com",
```

```
            "RegionName": "ap-south-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-3.amazonaws.com",
            "RegionName": "eu-west-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-2.amazonaws.com",
            "RegionName": "eu-west-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-1.amazonaws.com",
            "RegionName": "eu-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
            "RegionName": "ap-northeast-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
            "RegionName": "ap-northeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
            "RegionName": "ap-northeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.sa-east-1.amazonaws.com",
            "RegionName": "sa-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ca-central-1.amazonaws.com",
            "RegionName": "ca-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
```

```
            "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
            "RegionName": "ap-southeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
            "RegionName": "ap-southeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-central-1.amazonaws.com",
            "RegionName": "eu-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
            "RegionName": "us-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2",
            "OptInStatus": "opt-in-not-required"
        }
    ]
}
```

Weitere Informationen finden Sie unter Regionen und Zones im Amazon-EC2-Benutzerhandbuch.

Beispiel 2: So beschreiben Sie aktivierte Regionen mit einem Endpunkt, dessen Name eine bestimmte Zeichenfolge enthält

Das folgende `describe-regions`-Beispiel beschreibt alle Regionen, die Sie aktiviert haben und deren Endpunkt die Zeichenfolge „us“ enthält.

```
aws ec2 describe-regions \
    --filters "Name=endpoint,Values=*us*"
```

Ausgabe:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
            "RegionName": "us-east-1"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2"
        },
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2"
        }
    ]
}
```

Weitere Informationen finden Sie unter Regionen und Zones im Amazon-EC2-Benutzerhandbuch.

Beispiel 3: So beschreiben Sie alle Regionen

Das folgende `describe-regions`-Beispiel beschreibt alle verfügbaren Regionen, einschließlich Regionen, die deaktiviert sind.

```
aws ec2 describe-regions \
    --all-regions
```

Ausgabe:

```
{
    "Regions": [
        {
            "Endpoint": "ec2.eu-north-1.amazonaws.com",
            "RegionName": "eu-north-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-south-1.amazonaws.com",
            "RegionName": "ap-south-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-3.amazonaws.com",
            "RegionName": "eu-west-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-2.amazonaws.com",
            "RegionName": "eu-west-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-west-1.amazonaws.com",
            "RegionName": "eu-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
            "RegionName": "ap-northeast-3",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.me-south-1.amazonaws.com",
            "RegionName": "me-south-1",
            "OptInStatus": "not-opted-in"
        },
        {
            "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
            "RegionName": "ap-northeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
```

```
            "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
            "RegionName": "ap-northeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.sa-east-1.amazonaws.com",
            "RegionName": "sa-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ca-central-1.amazonaws.com",
            "RegionName": "ca-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-east-1.amazonaws.com",
            "RegionName": "ap-east-1",
            "OptInStatus": "not-opted-in"
        },
        {
            "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
            "RegionName": "ap-southeast-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
            "RegionName": "ap-southeast-2",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.eu-central-1.amazonaws.com",
            "RegionName": "eu-central-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-1.amazonaws.com",
            "RegionName": "us-east-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-east-2.amazonaws.com",
            "RegionName": "us-east-2",
            "OptInStatus": "opt-in-not-required"
        },
```

```
        {
            "Endpoint": "ec2.us-west-1.amazonaws.com",
            "RegionName": "us-west-1",
            "OptInStatus": "opt-in-not-required"
        },
        {
            "Endpoint": "ec2.us-west-2.amazonaws.com",
            "RegionName": "us-west-2",
            "OptInStatus": "opt-in-not-required"
        }
    ]
}
```

Weitere Informationen finden Sie unter Regionen und Zones im Amazon-EC2-Benutzerhandbuch.

Beispiel 4: So listen Sie nur die Namen der Regionen auf

Im folgenden Beispiel für `describe-regions` wird der `--query`-Parameter verwendet, um die Ausgabe zu filtern und nur die Namen der Regionen als Text zurückzugeben.

```
aws ec2 describe-regions \
    --all-regions \
    --query "Regions[].{Name:RegionName}" \
    --output text
```

Ausgabe:

```
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-3
ap-northeast-2
me-south-1
ap-northeast-1
sa-east-1
ca-central-1
ap-east-1
ap-southeast-1
ap-southeast-2
```

```
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

Weitere Informationen finden Sie unter Regionen und Zonen im Amazon-EC2-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeRegionsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
 returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
            // You can specify multiple values for a filter.
            // You can also use '*' as a wildcard. This will return all
            // of the regions that start with `us-east-`.
```

```
            Values: regionNames,
          },
        ]
      : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Regionen beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2Region
```

Ausgabe:

```
Endpoint                            RegionName
--------                            ----------
ec2.eu-west-1.amazonaws.com         eu-west-1
ec2.ap-southeast-1.amazonaws.com    ap-southeast-1
ec2.ap-southeast-2.amazonaws.com    ap-southeast-2
ec2.eu-central-1.amazonaws.com      eu-central-1
ec2.ap-northeast-1.amazonaws.com    ap-northeast-1
```

```
ec2.us-east-1.amazonaws.com         us-east-1
ec2.sa-east-1.amazonaws.com         sa-east-1
ec2.us-west-1.amazonaws.com         us-west-1
ec2.us-west-2.amazonaws.com         us-west-2
```

- Einzelheiten zur API finden Sie unter DescribeRegions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Regionen beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2Region
```

Ausgabe:

```
Endpoint                            RegionName
--------                            ----------
ec2.eu-west-1.amazonaws.com         eu-west-1
ec2.ap-southeast-1.amazonaws.com    ap-southeast-1
ec2.ap-southeast-2.amazonaws.com    ap-southeast-2
ec2.eu-central-1.amazonaws.com      eu-central-1
ec2.ap-northeast-1.amazonaws.com    ap-northeast-1
ec2.us-east-1.amazonaws.com         us-east-1
ec2.sa-east-1.amazonaws.com         sa-east-1
ec2.us-west-1.amazonaws.com         us-west-1
ec2.us-west-2.amazonaws.com         us-west-2
```

- Einzelheiten zur API finden Sie unter DescribeRegions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
```

```
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
```

```
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS
 Region '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeRegionsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeregions( ).                        "
 oo_result is returned for testing purposes. "
        DATA(lt_regions) = oo_result->get_regions( ).
        MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeRegionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeRouteTables`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRouteTables` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Network Transit Gateways

CLI

AWS CLI

So beschreiben Sie Ihre Routing-Tabellen

Im folgenden Beispiel für `describe-route-tables` werden Details über Ihre Routing-Tabellen abgerufen.

```
aws ec2 describe-route-tables
```

Ausgabe:

```
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-0df3f54e06EXAMPLE",
                    "RouteTableId": "rtb-09ba434c1bEXAMPLE"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-09ba434c1bEXAMPLE",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.0.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "NatGatewayId": "nat-06c018cbd8EXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "blackhole"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-0065acced4EXAMPLE",
            "OwnerId": "111122223333"
        },
```

```
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-9EXAMPLE",
                    "RouteTableId": "rtb-a1eec7de"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-a1eec7de",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.31.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-fEXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "111122223333"
        },
        {
            "Associations": [
                {
                    "Main": false,
                    "RouteTableAssociationId": "rtbassoc-0b100c28b2EXAMPLE",
                    "RouteTableId": "rtb-07a98f76e5EXAMPLE",
                    "SubnetId": "subnet-0d3d002af8EXAMPLE"
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-07a98f76e5EXAMPLE",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.0.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
```

```
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-06cf664d80EXAMPLE",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-0065acced4EXAMPLE",
            "OwnerId": "111122223333"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Routing-Tabellen im Benutzerhandbuch für AWS VPC.

- Einzelheiten zur API finden Sie DescribeRouteTablesin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param array $routeTableIds
     * @param array $filters
     * @return array
     */
    public function describeRouteTables(array $routeTableIds = [], array $filters
 = []): array
    {
        $parameters = [];
        if($routeTableIds){
```

```
            $parameters['RouteTableIds'] = $routeTableIds;
        }
        if($filters){
            $parameters['Filters'] = $filters;
        }
        try {
            $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
 $parameters);
            $contents = [];
            foreach ($paginator as $result) {
                foreach ($result['RouteTables'] as $object) {
                    $contents[] = $object['RouteTableId'];
                }
            }
        }catch (Ec2Exception $caught){
            echo "There was a problem paginating the results of
 DescribeRouteTables: {$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
        return $contents;
    }
```

- Einzelheiten zur API finden Sie DescribeRouteTablesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Ihre Routing-Tabellen beschrieben.

```
Get-EC2RouteTable
```

Ausgabe:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
```

```
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Beispiel 2: In diesem Beispiel werden Details für die angegebene Routing-Tabelle zurückgegeben.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Beispiel 3: In diesem Beispiel werden die Routing-Tabellen für die angegebene VPC beschrieben.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Ausgabe:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeRouteTables AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Ihre Routing-Tabellen beschrieben.

```
Get-EC2RouteTable
```

Ausgabe:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Beispiel 2: In diesem Beispiel werden Details für die angegebene Routing-Tabelle zurückgegeben.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Beispiel 3: In diesem Beispiel werden die Routing-Tabellen für die angegebene VPC beschrieben.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Ausgabe:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeRouteTables AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
 Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe_route_tables(self, vpc_ids: list[str]) -> None:
        """
```

```
        Displays information about the route tables in the specified VPC.

        :param vpc_ids: A list of VPC IDs.
        """
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": vpc_ids}]
            )
            pp(response["RouteTables"])
        except ClientError as err:
            logger.error(
                "Couldn't describe route tables for VPCs %s. Here's why: %s: %s",
                vpc_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeRouteTablesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Create filter for VPC ID
" iv_vpc_id = 'vpc-abc123'
DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
APPEND NEW /aws1/cl_ec2filter(
  iv_name = 'vpc-id'
  it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
    ( NEW /aws1/cl_ec2valuestringlist_w( iv_vpc_id ) )
  )
```

```
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeroutetables( it_filters = lt_filters ).
       " oo_result is returned for testing purposes. "
        DATA(lt_route_tables) = oo_result->get_routetables( ).
        MESSAGE 'Retrieved information about route tables.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeRouteTablesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeScheduledInstanceAvailability`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScheduledInstanceAvailability` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen verfügbaren Zeitplan

In diesem Beispiel wird ein Zeitplan beschrieben, der jede Woche am Sonntag beginnt und am angegebenen Datum startet.

Befehl:

```
aws ec2 describe-scheduled-instance-availability --
recurrence Frequency=Weekly,Interval=1,OccurrenceDays=[1] --first-slot-start-
time-range EarliestTime=2016-01-31T00:00:00Z,LatestTime=2016-01-31T04:00:00Z
```

Ausgabe:

```
{
  "ScheduledInstanceAvailabilitySet": [
```

```
    {
        "AvailabilityZone": "us-west-2b",
        "TotalScheduledInstanceHours": 1219,
        "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
        "MinTermDurationInDays": 366,
        "AvailableInstanceCount": 20,
        "Recurrence": {
            "OccurrenceDaySet": [
                1
            ],
            "Interval": 1,
            "Frequency": "Weekly",
            "OccurrenceRelativeToEnd": false
        },
        "Platform": "Linux/UNIX",
        "FirstSlotStartTime": "2016-01-31T00:00:00Z",
        "MaxTermDurationInDays": 366,
        "SlotDurationInHours": 23,
        "NetworkPlatform": "EC2-VPC",
        "InstanceType": "c4.large",
        "HourlyPrice": "0.095"
    },
    ...
  ]
}
```

Zum Eingrenzen der Ergebnisse können Sie Filter hinzufügen, die das Betriebssystem, das Netzwerk und den Instance-Typ angeben.

Befehl:

--filters Name=Plattform, Werte= Name=Netzwerkplattform, Werte= Linux/UNIX Name=Instanztyp, Werte=C4.large EC2-VPC

- Einzelheiten zur API finden Sie in der DescribeScheduledInstanceAvailability AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Zeitplan beschrieben, der jede Woche am Sonntag beginnt und am angegebenen Datum startet.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219

...
```

Beispiel 2: Um die Ergebnisse einzugrenzen, können Sie Filter für Kriterien wie Betriebssystem, Netzwerk und Instance-Typ hinzufügen.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstanceAvailability AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Zeitplan beschrieben, der jede Woche am Sonntag beginnt und am angegebenen Datum startet.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219

...
```

Beispiel 2: Um die Ergebnisse einzugrenzen, können Sie Filter für Kriterien wie Betriebssystem, Netzwerk und Instance-Typ hinzufügen.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstanceAvailability AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeScheduledInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeScheduledInstances` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre geplanten Instances

In diesem Beispiel wird die angegebene geplante Instance beschrieben.

Befehl:

```
aws ec2 describe-scheduled-instances --scheduled-instance-
ids sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
{
  "ScheduledInstanceSet": [
      {
          "AvailabilityZone": "us-west-2b",
          "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
          "HourlyPrice": "0.095",
          "CreateDate": "2016-01-25T21:43:38.612Z",
          "Recurrence": {
              "OccurrenceDaySet": [
                  1
              ],
              "Interval": 1,
              "Frequency": "Weekly",
              "OccurrenceRelativeToEnd": false,
              "OccurrenceUnit": ""
          },
          "Platform": "Linux/UNIX",
          "TermEndDate": "2017-01-31T09:00:00Z",
          "InstanceCount": 1,
          "SlotDurationInHours": 32,
          "TermStartDate": "2016-01-31T09:00:00Z",
          "NetworkPlatform": "EC2-VPC",
          "TotalScheduledInstanceHours": 1696,
          "NextSlotStartTime": "2016-01-31T09:00:00Z",
          "InstanceType": "c4.large"
      }
  ]
}
```

In diesem Beispiel werden alle Ihre geplanten Instances beschrieben.

Befehl:

```
aws ec2 describe-scheduled-instances
```

- Einzelheiten zur API finden Sie DescribeScheduledInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance beschrieben.

```
Get-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Beispiel 2: In diesem Beispiel werden alle Ihre geplanten Instances beschrieben.

```
Get-EC2ScheduledInstance
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance beschrieben.

```
Get-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Beispiel 2: In diesem Beispiel werden alle Ihre geplanten Instances beschrieben.

```
Get-EC2ScheduledInstance
```

- Einzelheiten zur API finden Sie unter DescribeScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeSecurityGroups`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSecurityGroups` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Ein Service Networking Mesh erstellen und verwalten
- Erste Schritte mit In-Memory-Caching
- Erste Schritte mit verwaltetem Streaming

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups.
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
```

```
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_security_groups
#
```

```
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe
 (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"
```

```
  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
 --query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
```

```
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
 specific group.
/*!
  \param groupID: A group ID, ignored if empty.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
 ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
                      std::setw(32) << "Name" <<
                      std::setw(30) << "GroupId" <<
                      std::setw(30) << "VpcId" <<
                      std::setw(64) << "Description" << std::endl;

            const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
                    outcome.GetResult().GetSecurityGroups();

            for (const auto &securityGroup: securityGroups) {
                std::cout << std::left <<
                          std::setw(32) << securityGroup.GetGroupName() <<
                          std::setw(30) << securityGroup.GetGroupId() <<
                          std::setw(30) << securityGroup.GetVpcId() <<
                          std::setw(64) << securityGroup.GetDescription() <<
                          std::endl;
            }
        } else {
            std::cerr << "Failed to describe security groups:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return true;
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Sicherheitsgruppe

Das folgende `describe-security-groups`-Beispiel beschreibt die angegebene Sicherheitsgruppe.

```
aws ec2 describe-security-groups \
    --group-ids sg-903004f8
```

Ausgabe:

```
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "UserIdGroupPairs": [],
                    "PrefixListIds": []
                }
            ],
            "Description": "My security group",
            "Tags": [
```

```
                {
                    "Value": "SG1",
                    "Key": "Name"
                }
            ],
            "IpPermissions": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [],
                    "UserIdGroupPairs": [
                        {
                            "UserId": "123456789012",
                            "GroupId": "sg-903004f8"
                        }
                    ],
                    "PrefixListIds": []
                },
                {
                    "PrefixListIds": [],
                    "FromPort": 22,
                    "IpRanges": [
                        {
                            "Description": "Access from NY office",
                            "CidrIp": "203.0.113.0/24"
                        }
                    ],
                    "ToPort": 22,
                    "IpProtocol": "tcp",
                    "UserIdGroupPairs": []
                    }
            ],
            "GroupName": "MySecurityGroup",
            "VpcId": "vpc-1a2b3c4d",
            "OwnerId": "123456789012",
            "GroupId": "sg-903004f8",
        }
    ]
}
```

Beispiel 2: So beschreiben Sie Sicherheitsgruppen, die bestimmte Regeln haben

Im folgenden Beispiel für `describe-security-groups` werden Filter verwendet, um die Ergebnisse auf Sicherheitsgruppen zu beschränken, die über eine Regel verfügen, die SSH-

Datenverkehr (Port 22) zulässt, und eine Regel, die Datenverkehr von allen Adressen zulässt (`0.0.0.0/0`). Im Beispiel wird der `--query`-Parameter verwendet, um nur die Namen der Sicherheitsgruppen anzuzeigen. Sicherheitsgruppen müssen mit allen Filtern übereinstimmen, die in den Ergebnissen zurückgegeben werden. Eine einzige Regel muss jedoch nicht mit allen Filtern übereinstimmen. Die Ausgabe liefert beispielsweise eine Sicherheitsgruppe mit einer Regel, die SSH-Verkehr von einer bestimmten IP-Adresse zulässt und einer anderen Regel, die HTTP-Verkehr von allen Adressen zulässt.

```
aws ec2 describe-security-groups \
    --filters Name=ip-permission.from-port,Values=22 Name=ip-permission.to-
port,Values=22 Name=ip-permission.cidr,Values='0.0.0.0/0' \
    --query "SecurityGroups[*].[GroupName]" \
    --output text
```

Ausgabe:

```
default
my-security-group
web-servers
launch-wizard-1
```

Beispiel 3: So beschreiben Sie Sicherheitsgruppen anhand von Tags

Im folgenden Beispiel für `describe-security-groups` werden Filter verwendet, um die Ergebnisse auf Sicherheitsgruppen zu beschränken, die `test` im Namen der Sicherheitsgruppe enthalten und die das Tag `Test=To-delete` haben. Im Beispiel wird der `--query`-Parameter verwendet, um nur die Namen und IDs der Sicherheitsgruppen anzuzeigen.

```
aws ec2 describe-security-groups \
    --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
    --query "SecurityGroups[*].{Name:GroupName,ID:GroupId}"
```

Ausgabe:

```
[
    {
        "Name": "testfornewinstance",
        "ID": "sg-33bb22aa"
    },
```

```
    {
        "Name": "newgrouptest",
        "ID": "sg-1a2b3c4d"
    }
]
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter Arbeiten mit Tags im Amazon-EC2-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
```

```
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
 securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
 groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the
 name: " + groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
export const main = async ({ groupIds = [] }) => {
```

```
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {
      console.warn(caught.message);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für eine VPC beschrieben. Wenn Sie mit Sicherheitsgruppen arbeiten, die zu einer VPC gehören, müssen Sie die Sicherheitsgruppen-ID (- GroupId Parameter) und nicht den Namen (- GroupName Parameter) verwenden, um auf die Gruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Description         : default VPC security group
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für EC2-Classic. Wenn Sie mit Sicherheitsgruppen für arbeiten, können EC2-Classic Sie entweder den Gruppennamen (- GroupName Parameter) oder die Gruppen-ID (- GroupId Parameter) verwenden, um auf die Sicherheitsgruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Ausgabe:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission,
 Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Beispiel 3: In diesem Beispiel werden alle Sicherheitsgruppen für die Datei vpc-0fc1ff23456b789eb abgerufen

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Einzelheiten zur API finden Sie unter DescribeSecurityGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für eine VPC beschrieben. Wenn Sie mit Sicherheitsgruppen arbeiten, die zu einer VPC gehören, müssen Sie die Sicherheitsgruppen-ID (- GroupId Parameter) und nicht den Namen (- GroupName Parameter) verwenden, um auf die Gruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Description         : default VPC security group
```

```
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für EC2-Classic. Wenn Sie mit Sicherheitsgruppen für arbeiten, können EC2-Classic Sie entweder den Gruppennamen (- GroupName Parameter) oder die Gruppen-ID (- GroupId Parameter) verwenden, um auf die Sicherheitsgruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Ausgabe:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission,
 Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Beispiel 3: In diesem Beispiel werden alle Sicherheitsgruppen für die Datei vpc-0fc1ff23456b789eb abgerufen

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- Einzelheiten zur API finden Sie unter DescribeSecurityGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
 high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
 Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA lt_group_ids TYPE /aws1/
cl_ec2groupidstrlist_w=>tt_groupidstringlist.
        APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
 lt_group_ids.
        oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
         " oo_result is returned for testing purposes. "
        DATA(lt_security_groups) = oo_result->get_securitygroups( ).
        MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Paginierung mit `describeSecurityGroupsPaginated()`.

```
import AWSEC2

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }
```

```
        return groupNames
    }
```

Ohne Paginierung.

```
import AWSEC2

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
 \(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
 \(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DescribeSnapshotAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSnapshotAttribute` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die snapshot-Attribute für einen Snapshot

Im folgenden Beispiel für `describe-snapshot-attribute` werden die Konten aufgelistet, mit denen ein Snapshot geteilt wird.

```
aws ec2 describe-snapshot-attribute \
    --snapshot-id snap-01234567890abcedf \
    --attribute createVolumePermission
```

Ausgabe:

```
{
    "SnapshotId": "snap-01234567890abcedf",
    "CreateVolumePermissions": [
        {
            "UserId": "123456789012"
        }
    ]
}
```

Weitere Informationen finden Sie unter Teilen eines Amazon-EBS-Snapshots im Benutzerhandbuch für Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie DescribeSnapshotAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Ausgabe:

```
CreateVolumePermissions    ProductCodes    SnapshotId
```

```
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Beispiel 2: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Ausgabe:

```
Group    UserId
-----    ------
all
```

- Einzelheiten zur API finden Sie unter DescribeSnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Ausgabe:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Beispiel 2: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Ausgabe:

```
Group    UserId
-----    ------
```

```
all
```

- Einzelheiten zur API finden Sie unter DescribeSnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeSnapshots`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSnapshots` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen Snapshot

Das folgende `describe-snapshots`-Beispiel beschreibt den angegebenen Snapshot.

```
aws ec2 describe-snapshots \
    --snapshot-ids snap-1234567890abcdef0
```

Ausgabe:

```
{
    "Snapshots": [
        {
            "Description": "This is my snapshot",
            "Encrypted": false,
            "VolumeId": "vol-049df61146c4d7901",
            "State": "completed",
            "VolumeSize": 8,
            "StartTime": "2019-02-28T21:28:32.000Z",
            "Progress": "100%",
            "OwnerId": "012345678910",
            "SnapshotId": "snap-01234567890abcdef",
            "Tags": [
                {
                    "Key": "Stack",
                    "Value": "test"
                }
            ]
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Amazon-EBS-Snapshots im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So beschreiben Sie Snapshots basierend auf Filtern

Im folgenden `describe-snapshots` Beispiel werden Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die Ihrem AWS Konto gehören und sich im `pending` Bundesstaat befinden. Das Beispiel verwendet den `--query`-Parameter, um nur die Snapshot-IDs und die Uhrzeit anzuzeigen, zu der der Snapshot gestartet wurde.

```
aws ec2 describe-snapshots \
    --owner-ids self \
    --filters Name=status,Values=pending \
    --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

Ausgabe:

```
[
    {
        "ID": "snap-1234567890abcdef0",
        "Time": "2019-08-04T12:48:18.000Z"
    },
    {
        "ID": "snap-066877671789bd71b",
        "Time": "2019-08-04T02:45:16.000Z
    },
    ...
]
```

Im folgenden Beispiel für `describe-snapshots` werden Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die aus dem angegebenen Volume erstellt wurden. Das Beispiel verwendet den `--query`-Parameter, um nur die Snapshot-IDs anzuzeigen.

```
aws ec2 describe-snapshots \
    --filters Name=volume-id,Values=049df61146c4d7901 \
    --query "Snapshots[*].[SnapshotId]" \
    --output text
```

Ausgabe:

```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter Auflisten und Filtern Ihrer Ressourcen im Amazon-EC2-Benutzerhandbuch.

Beispiel 3: So beschreiben Sie Snapshots auf der Grundlage von Tags

Im folgenden Beispiel für `describe-snapshots` werden Tag-Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die das Tag `Stack=Prod` enthalten.

```
aws ec2 describe-snapshots \
    --filters Name=tag:Stack,Values=prod
```

Ein Beispiel für die Ausgabe von `describe-snapshots` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter Arbeiten mit Tags im Amazon-EC2-Benutzerhandbuch.

Beispiel 4: So beschreiben Sie Snapshots anhand des Alters

Im folgenden `describe-snapshots` Beispiel werden JMESPath-Ausdrücke verwendet, um alle Snapshots zu beschreiben, die von Ihrem AWS Konto vor dem angegebenen Datum erstellt wurden. Es werden nur die Snapshot-IDs angezeigt.

```
aws ec2 describe-snapshots \
    --owner-ids 012345678910 \
    --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter Auflisten und Filtern Ihrer Ressourcen im Amazon-EC2-Benutzerhandbuch.

Beispiel 5: So zeigen Sie nur archivierte Snapshots an

Im folgenden Beispiel für `describe-snapshots` werden ausschließlich Snapshots aufgeführt, die auf der Archivstufe gespeichert sind.

```
aws ec2 describe-snapshots \
    --filters "Name=storage-tier,Values=archive"
```

Ausgabe:

```
{
    "Snapshots": [
        {
            "Description": "Snap A",
            "Encrypted": false,
            "VolumeId": "vol-01234567890aaaaaa",
            "State": "completed",
            "VolumeSize": 8,
            "StartTime": "2021-09-07T21:00:00.000Z",
            "Progress": "100%",
            "OwnerId": "123456789012",
            "SnapshotId": "snap-01234567890aaaaaa",
            "StorageTier": "archive",
            "Tags": []
        },
    ]
}
```

Weitere Informationen finden Sie unter Archivierte Snapshots anzeigen im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie unter Befehlsreferenz DescribeSnapshots.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot beschrieben.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
```

```
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 8
```

Beispiel 2: In diesem Beispiel werden die Snapshots beschrieben, die ein „Name”-Tag enthalten.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Beispiel 3: Dieses Beispiel beschreibt die Schnappschüsse, die ein 'Name' -Tag mit dem Wert 'TestValue' haben.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Beispiel 4: In diesem Beispiel werden alle Ihre Snapshots beschrieben.

```
Get-EC2Snapshot -Owner self
```

- Einzelheiten zur API finden Sie unter DescribeSnapshots AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot beschrieben.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
```

```
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 8
```

Beispiel 2: In diesem Beispiel werden die Snapshots beschrieben, die ein „Name”-Tag enthalten.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Beispiel 3: Dieses Beispiel beschreibt die Schnappschüsse, die ein 'Name'-Tag mit dem Wert " TestValue haben.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Beispiel 4: In diesem Beispiel werden alle Ihre Snapshots beschrieben.

```
Get-EC2Snapshot -Owner self
```

- Einzelheiten zur API finden Sie unter DescribeSnapshots AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt den Status eines Snapshots an.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
```

```
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots();
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:       {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeSnapshotsin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DescribeSpotDatafeedSubscription`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotDatafeedSubscription` verwendet wird.

CLI

AWS CLI

So beschreiben Sie das Daten-Feed-Abonnement für Spot Instances für ein Konto

Dieser Beispielbefehl beschreibt den Daten-Feed für das Konto.

Befehl:

```
aws ec2 describe-spot-datafeed-subscription
```

Ausgabe:

```
{
    "SpotDatafeedSubscription": {
        "OwnerId": "123456789012",
        "Prefix": "spotdata",
        "Bucket": "amzn-s3-demo-bucket",
        "State": "Active"
    }
}
```

- Einzelheiten zur API finden Sie DescribeSpotDatafeedSubscriptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Daten-Feed für Ihre Spot-Instance beschrieben.

```
Get-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Einzelheiten zur API finden Sie unter DescribeSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Daten-Feed für Ihre Spot-Instance beschrieben.

```
Get-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- Einzelheiten zur API finden Sie unter DescribeSpotDatafeedSubscription AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSpotFleetInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotFleetInstances` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Spot Instances, die einer Spot-Flotte zugeordnet sind

Dieser Beispielbefehl listet die Spot Instances auf, die der angegebenen Spot-Flotte zugeordnet sind.

Befehl:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
  "ActiveInstances": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "InstanceType": "m3.medium",
          "SpotInstanceRequestId": "sir-08b93456"
      },
      ...
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- Einzelheiten zur API finden Sie DescribeSpotFleetInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Instances beschrieben, die der angegebenen Spot-Flottenanforderung zugeordnet sind.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Instances beschrieben, die der angegebenen Spot-Flottenanforderung zugeordnet sind.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSpotFleetRequestHistory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotFleetRequestHistory` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Verlauf der Spot-Flotte

Dieser Beispielbefehl gibt den Verlauf für die angegebene Spot-Flotte ab dem angegebenen Zeitpunkt zurück.

Befehl:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

Die folgende Beispielausgabe zeigt die erfolgreichen Starts von zwei Spot Instances für die Spot-Flotte.

Ausgabe:

```
{
  "HistoryRecords": [
      {
          "Timestamp": "2015-05-26T23:17:20.697Z",
          "EventInformation": {
              "EventSubType": "submitted"
          },
          "EventType": "fleetRequestChange"
      },
      {
          "Timestamp": "2015-05-26T23:17:20.873Z",
          "EventInformation": {
              "EventSubType": "active"
          },
          "EventType": "fleetRequestChange"
      },
      {
          "Timestamp": "2015-05-26T23:21:21.712Z",
          "EventInformation": {
              "InstanceId": "i-1234567890abcdef0",
              "EventSubType": "launched"
          },
          "EventType": "instanceChange"
      },
      {
          "Timestamp": "2015-05-26T23:21:21.816Z",
          "EventInformation": {
              "InstanceId": "i-1234567890abcdef1",
              "EventSubType": "launched"
          },
          "EventType": "instanceChange"
      }
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
  "NextToken": "CpHNsscimcV5oH7bSbub03CI2Qms5+ypNpNm
+53MNlR0YcXAkp0xFlfKf91yVxSExmbtma3awYxMFzNA663ZskT0AHtJ6TCb2Z8bQC2EnZgyELbymtWPfpZ1ZbauV
+P+TfGlWxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
  "StartTime": "2015-05-26T00:00:00Z"
}
```

- Einzelheiten zur API finden Sie DescribeSpotFleetRequestHistoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Verlauf der angegebenen Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Ausgabe:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Ausgabe:

```
EventInformation                    EventType             Timestamp
----------------                    ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:05 AM
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequestHistory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Verlauf der angegebenen Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Ausgabe:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Ausgabe:

```
EventInformation                    EventType             Timestamp
----------------                    ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:05 AM
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequestHistory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSpotFleetRequests`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotFleetRequests` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Spot-Flottenanforderungen

In diesem Beispiel werden alle Ihre Spot-Flottenanforderungen beschrieben.

Befehl:

```
aws ec2 describe-spot-fleet-requests
```

Ausgabe:

```
{
  "SpotFleetRequestConfigs": [
      {
          "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
          "SpotFleetRequestConfig": {
              "TargetCapacity": 20,
              "LaunchSpecifications": [
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
                              "SubnetId": "subnet-a61dafcf",
                              "DeviceIndex": 0,
                              "DeleteOnTermination": false,
                              "AssociatePublicIpAddress": true,
                              "SecondaryPrivateIpAddressCount": 0
                          }
                      ],
                      "InstanceType": "cc2.8xlarge",
                      "ImageId": "ami-1a2b3c4d"
                  },
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
                              "SubnetId": "subnet-a61dafcf",
                              "DeviceIndex": 0,
                              "DeleteOnTermination": false,
                              "AssociatePublicIpAddress": true,
                              "SecondaryPrivateIpAddressCount": 0
                          }
                      ],
                      "InstanceType": "r3.8xlarge",
                      "ImageId": "ami-1a2b3c4d"
                  }
              ],
              "SpotPrice": "0.05",
              "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
          },
```

```
            "SpotFleetRequestState": "active"
        },
        {
            "SpotFleetRequestId": "sfr-306341ed-9739-402e-881b-ce47bEXAMPLE",
            "SpotFleetRequestConfig": {
                "TargetCapacity": 20,
                "LaunchSpecifications": [
                    {
                        "EbsOptimized": false,
                        "NetworkInterfaces": [
                            {
                                "SubnetId": "subnet-6e7f829e",
                                "DeviceIndex": 0,
                                "DeleteOnTermination": false,
                                "AssociatePublicIpAddress": true,
                                "SecondaryPrivateIpAddressCount": 0
                            }
                        ],
                        "InstanceType": "m3.medium",
                        "ImageId": "ami-1a2b3c4d"
                    }
                ],
                "SpotPrice": "0.05",
                "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
            },
            "SpotFleetRequestState": "active"
        }
    ]
}
```

So beschreiben Sie eine Spot-Flottenanforderung

In diesem Beispiel wird die angegebene Spot-Flottenanforderung beschrieben.

Befehl:

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
```

```
  "SpotFleetRequestConfigs": [
      {
          "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
          "SpotFleetRequestConfig": {
              "TargetCapacity": 20,
              "LaunchSpecifications": [
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
                              "SubnetId": "subnet-a61dafcf",
                              "DeviceIndex": 0,
                              "DeleteOnTermination": false,
                              "AssociatePublicIpAddress": true,
                              "SecondaryPrivateIpAddressCount": 0
                          }
                      ],
                      "InstanceType": "cc2.8xlarge",
                      "ImageId": "ami-1a2b3c4d"
                  },
                  {
                      "EbsOptimized": false,
                      "NetworkInterfaces": [
                          {
                              "SubnetId": "subnet-a61dafcf",
                              "DeviceIndex": 0,
                              "DeleteOnTermination": false,
                              "AssociatePublicIpAddress": true,
                              "SecondaryPrivateIpAddressCount": 0
                          }
                      ],
                      "InstanceType": "r3.8xlarge",
                      "ImageId": "ami-1a2b3c4d"
                  }
              ],
              "SpotPrice": "0.05",
              "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
          },
          "SpotFleetRequestState": "active"
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeSpotFleetRequestsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE | format-list
```

Ausgabe:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Flottenanforderungen beschrieben.

```
Get-EC2SpotFleetRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequests AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE | format-list
```

Ausgabe:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Flottenanforderungen beschrieben.

```
Get-EC2SpotFleetRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotFleetRequests AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSpotInstanceRequests`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotInstanceRequests` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Spot Instance-Anforderung

Im folgenden Beispiel für `describe-spot-instance-requests` wird die angegebene Spot-Instance-Anforderung beschrieben.

```
aws ec2 describe-spot-instance-requests \
    --spot-instance-request-ids sir-08b93456
```

Ausgabe:

```
{
    "SpotInstanceRequests": [
        {
            "CreateTime": "2018-04-30T18:14:55.000Z",
            "InstanceId": "i-1234567890abcdef1",
            "LaunchSpecification": {
                "InstanceType": "t2.micro",
                "ImageId": "ami-003634241a8fcdec0",
                "KeyName": "my-key-pair",
                "SecurityGroups": [
                    {
                        "GroupName": "default",
                        "GroupId": "sg-e38f24a7"
                    }
                ],
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sda1",
                        "Ebs": {
                            "DeleteOnTermination": true,
                            "SnapshotId": "snap-0e54a519c999adbbd",
```

```
                            "VolumeSize": 8,
                            "VolumeType": "standard",
                            "Encrypted": false
                        }
                    }
                ],
                "NetworkInterfaces": [
                    {
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "SubnetId": "subnet-049df61146c4d7901"
                    }
                ],
                "Placement": {
                    "AvailabilityZone": "us-east-2b",
                    "Tenancy": "default"
                },
                "Monitoring": {
                    "Enabled": false
                }
            },
            "LaunchedAvailabilityZone": "us-east-2b",
            "ProductDescription": "Linux/UNIX",
            "SpotInstanceRequestId": "sir-08b93456",
            "SpotPrice": "0.010000"
            "State": "active",
            "Status": {
                "Code": "fulfilled",
                "Message": "Your Spot request is fulfilled.",
                "UpdateTime": "2018-04-30T18:16:21.000Z"
            },
            "Tags": [],
            "Type": "one-time",
            "InstanceInterruptionBehavior": "terminate"
        }
    ]
}
```

Beispiel 2: So beschreiben Sie Spot-Instance-Anforderungen basierend auf Filtern

Im folgenden Beispiel für `describe-spot-instance-requests` werden Filter verwendet, um die Ergebnisse auf Spot Instances mit dem angegebenen Instance-Typ in der festgelegten

Availability Zone zu beschränken. Im Beispiel wird der `--query`-Parameter verwendet, um nur die Instance-IDs anzuzeigen.

```
aws ec2 describe-spot-instance-requests \
    --filters Name=launch.instance-type,Values=m3.medium Name=launched-
availability-zone,Values=us-east-2a \
    --query "SpotInstanceRequests[*].[InstanceId]" \
    --output text
```

Ausgabe:

```
i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter Listing and filtering your resources im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

Beispiel 3: So beschreiben Sie Spot-Instance-Anforderungen basierend auf Tags

Im folgenden Beispiel für `describe-spot-instance-requests` werden Tag-Filter verwendet, um die Ergebnisse auf Spot-Instance-Anforderungen zu beschränken, die das Tag `cost-center=cc123` enthalten.

```
aws ec2 describe-spot-instance-requests \
    --filters Name=tag:cost-center,Values=cc123
```

Ein Beispiel für die Ausgabe von `describe-spot-instance-requests` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter Arbeiten mit Tags im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DescribeSpotInstanceRequestsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung beschrieben.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Instance-Anforderungen beschrieben.

```
Get-EC2SpotInstanceRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotInstanceRequests AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung beschrieben.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Instance-Anforderungen beschrieben.

```
Get-EC2SpotInstanceRequest
```

- Einzelheiten zur API finden Sie unter DescribeSpotInstanceRequests AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSpotPriceHistory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSpotPriceHistory` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Spot-Preisverlauf

Dieser Beispielbefehl gibt den Spot-Preisverlauf für m1.xlarge-Instances für einen bestimmten Tag im Januar zurück.

Befehl:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-
time 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

Ausgabe:

```
{
  "SpotPriceHistory": [
          {
              "Timestamp": "2014-01-06T07:10:55.000Z",
              "ProductDescription": "SUSE Linux",
              "InstanceType": "m1.xlarge",
              "SpotPrice": "0.087000",
              "AvailabilityZone": "us-west-1b"
          },
          {
              "Timestamp": "2014-01-06T07:10:55.000Z",
              "ProductDescription": "SUSE Linux",
              "InstanceType": "m1.xlarge",
              "SpotPrice": "0.087000",
              "AvailabilityZone": "us-west-1c"
          },
          {
              "Timestamp": "2014-01-06T05:42:36.000Z",
              "ProductDescription": "SUSE Linux (Amazon VPC)",
              "InstanceType": "m1.xlarge",
              "SpotPrice": "0.087000",
              "AvailabilityZone": "us-west-1a"
      },
      ...
}
```

Um die Spot-Preisentwicklung für Linux/UNIX Amazon VPC zu beschreiben

Dieser Beispielbefehl gibt den Spot-Preisverlauf für m1.xlarge, Linux/UNIX Amazon VPC-Instances für einen bestimmten Tag im Januar zurück.

Befehl:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --product-
description "Linux/UNIX (Amazon VPC)" --start-time 2014-01-06T07:08:09 --end-
time 2014-01-06T08:09:10
```

Ausgabe:

```
{
  "SpotPriceHistory": [
```

```
        {
            "Timestamp": "2014-01-06T04:32:53.000Z",
            "ProductDescription": "Linux/UNIX (Amazon VPC)",
            "InstanceType": "m1.xlarge",
            "SpotPrice": "0.080000",
            "AvailabilityZone": "us-west-1a"
        },
        {
            "Timestamp": "2014-01-05T11:28:26.000Z",
            "ProductDescription": "Linux/UNIX (Amazon VPC)",
            "InstanceType": "m1.xlarge",
            "SpotPrice": "0.080000",
            "AvailabilityZone": "us-west-1c"
        }
    ]
}
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz DescribeSpotPriceHistory.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel ruft die letzten 10 Einträge im Spot-Preisverlauf für den angegebenen Instance-Typ und die angegebene Availability Zone ab. Beachten Sie, dass der für den AvailabilityZone Parameter - angegebene Wert für den Regionswert gültig sein muss, der entweder an den Parameter -Region des Cmdlets übergeben wurde (im Beispiel nicht gezeigt) oder als Standard in der Shell festgelegt wurde. Bei diesem Beispielbefehl wird davon ausgegangen, dass die Standardregion „us-west-2“ in der Umgebung festgelegt wurde.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Ausgabe:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM
```

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- Einzelheiten zur API finden Sie unter DescribeSpotPriceHistory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel ruft die letzten 10 Einträge im Spot-Preisverlauf für den angegebenen Instance-Typ und die angegebene Availability Zone ab. Beachten Sie, dass der für den AvailabilityZone Parameter - angegebene Wert für den Regionswert gültig sein muss, der entweder an den Parameter -Region des Cmdlets übergeben wurde (im Beispiel nicht gezeigt) oder als Standard in der Shell festgelegt wurde. Bei diesem Beispielbefehl wird davon ausgegangen, dass die Standardregion „us-west-2“ in der Umgebung festgelegt wurde.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Ausgabe:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
```

```
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- Einzelheiten zur API finden Sie unter DescribeSpotPriceHistory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeSubnets`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSubnets` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Eine relationale Datenbankinstanz erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Erste Schritte mit Dokumentendatenbanken
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Network Transit Gateways

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
 does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while describing the
 subnets.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeSubnetsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Ihre Subnetze

Im folgenden Beispiel für `describe-subnets` werden die Details zu Ihren Subnetzen angezeigt.

```
aws ec2 describe-subnets
```

Ausgabe:

```
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": true,
            "State": "available",
            "SubnetId": "subnet-0bb1c79de3EXAMPLE",
            "VpcId": "vpc-0ee975135dEXAMPLE",
            "OwnerId": "111122223333",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "CustomerOwnedIpv4Pool:": 'pool-2EXAMPLE',
            "SubnetArn": "arn:aws:ec2:us-east-2:111122223333:subnet/
subnet-0bb1c79de3EXAMPLE",
            "EnableDns64": false,
```

```
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        },
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-8EXAMPLE",
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "1111222233333",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "MySubnet"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit VPCs und Subnetzen im Benutzerhandbuch zu AWS VPC.

Beispiel 2: So beschreiben Sie die Subnetze einer bestimmten VPC

Im folgenden Beispiel für `describe-subnets` wird ein Filter verwendet, um Details für die Subnetze der angegebenen VPC abzurufen.

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

Ausgabe:

```
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1d",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 4089,
            "CidrBlock": "172.31.80.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-8EXAMPLE",
            "VpcId": "vpc-3EXAMPLE",
            "OwnerId": "1111222233333",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "MySubnet"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            }
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Arbeiten mit VPCs und Subnetzen im Benutzerhandbuch zu AWS VPC.

Beispiel 3: So beschreiben Sie die Subnetze mit einem bestimmten Tag

Das folgende `describe-subnets`-Beispiel verwendet einen Filter, um die Details dieser Subnetze mit dem Tag `CostCenter=123` und dem `--query`-Parameter abzurufen, um die Subnetz-IDs der Subnetze mit diesem Tag anzuzeigen.

```
aws ec2 describe-subnets \
    --filters "Name=tag:CostCenter,Values=123" \
    --query "Subnets[*].SubnetId" \
    --output text
```

Ausgabe:

```
subnet-0987a87c8b37348ef
subnet-02a95061c45f372ee
subnet-03f720e7de2788d73
```

Weitere Informationen finden Sie unter Arbeiten mit VPCs und Subnetzen im Benutzerhandbuch zu Amazon VPC.

- Einzelheiten zur API finden Sie DescribeSubnetsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
```

```
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
```

- Einzelheiten zur API finden Sie DescribeSubnetsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz beschrieben.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Subnetze beschrieben.

```
Get-EC2Subnet
```

- Einzelheiten zur API finden Sie unter DescribeSubnets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz beschrieben.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Subnetze beschrieben.

```
Get-EC2Subnet
```

- Einzelheiten zur API finden Sie unter DescribeSubnets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
```

```
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DescribeSubnetsin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle Tags für eine einzelne Ressource

Im folgenden Beispiel für `describe-tags` werden die Tags für die angegebene Instance beschrieben.

```
aws ec2 describe-tags \
    --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Beta Server",
            "Key": "Name"
        }
    ]
}
```

Beispiel 2: So beschreiben Sie alle Tags für einen Ressourcentyp

Im folgenden Beispiel für `describe-tags` werden die Tags für Ihre Volumes beschrieben.

```
aws ec2 describe-tags \
    --filters "Name=resource-type,Values=volume"
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-1234567890abcdef0",
            "Value": "Project1",
            "Key": "Purpose"
        },
        {
            "ResourceType": "volume",
            "ResourceId": "vol-049df61146c4d7901",
            "Value": "Logs",
            "Key": "Purpose"
        }
    ]
}
```

Beispiel 3: So beschreiben Sie alle Ihre Tags

Im folgenden Beispiel für `describe-tags` werden die Tags für alle Ihre Ressourcen beschrieben.

```
aws ec2 describe-tags
```

Beispiel 4: So beschreiben Sie die Tags für Ihre Ressourcen anhand eines Tag-Schlüssels

Im folgenden Beispiel für `describe-tags` werden die Tags für Ihre Ressourcen beschrieben, die über ein Tag mit dem Schlüssel `Stack` verfügen.

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-027552a73f021f3b",
            "Value": "Production",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
    ]
}
```

Beispiel 5: So beschreiben Sie die Tags für Ihre Ressourcen anhand eines Tag-Schlüssels und eines Tag-Werts

Im folgenden Beispiel für `describe-tags` werden die Tags für Ihre Ressourcen beschrieben, die über das Tag `Stack=Test` verfügen.

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack Name=value,Values=Test
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "image",
            "ResourceId": "ami-3ac336533f021f3bd",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
```

```
    ]
}
```

Im folgenden Beispiel für `describe-tags` wird eine alternative Syntax verwendet, um Ressourcen mit dem Tag `Stack=Test` zu beschreiben.

```
aws ec2 describe-tags \
    --filters "Name=tag:Stack,Values=Test"
```

Im folgenden Beispiel für `describe-tags` werden die Tags für alle Ihre Instances beschrieben, die über ein Tag mit dem Schlüssel `Purpose` verfügen und keinen Wert haben.

```
aws ec2 describe-tags \
    --filters "Name=resource-
type,Values=instance" "Name=key,Values=Purpose" "Name=value,Values="
```

Ausgabe:

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef5",
            "Value": null,
            "Key": "Purpose"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags für den Ressourcentyp „image“ abgerufen

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Beispiel 2: In diesem Beispiel werden alle Tags für alle Ressourcen abgerufen und nach Ressourcentyp gruppiert

```
Get-EC2Tag | Group-Object resourcetype
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Beispiel 3: In diesem Beispiel werden alle Ressourcen mit dem Tag „auto-delete“ und dem Wert „no“ für die angegebene Region angezeigt

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Beispiel 4: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' mit dem Wert 'no' und weitere Filter in der nächsten Pipe abgerufen, um nur die Ressourcentypen 'Instanz' zu analysieren, und erstellt schließlich das Tag 'ThisInstance' für jede Instanzressource, wobei der Wert die Instanz-ID selbst ist

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
 | Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -
ResourceId $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Beispiel 5: In diesem Beispiel werden Tags für alle Instance-Ressourcen sowie „Name“-Schlüssel abgerufen und in Tabellenform angezeigt

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Ausgabe:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für den Ressourcentyp „image“ abgerufen

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Beispiel 2: In diesem Beispiel werden alle Tags für alle Ressourcen abgerufen und nach Ressourcentyp gruppiert

```
Get-EC2Tag | Group-Object resourcetype
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
```

```
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Beispiel 3: In diesem Beispiel werden alle Ressourcen mit dem Tag „auto-delete“ und dem Wert „no“ für die angegebene Region angezeigt

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Beispiel 4: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' mit dem Wert 'no' und weitere Filter in der nächsten Pipe abgerufen, um nur die Ressourcentypen 'Instanz' zu analysieren, und erstellt schließlich das Tag 'ThisInstance' für jede Instanzressource, wobei der Wert die Instanz-ID selbst ist

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
 | Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -
ResourceId $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Beispiel 5: In diesem Beispiel werden Tags für alle Instance-Ressourcen sowie „Name“-Schlüssel abgerufen und in Tabellenform angezeigt

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Ausgabe:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

Beispiel 6: In diesem Beispiel werden die Berechtigungen für das Abrufen von EC2-Tags mithilfe des DryRun Parameters validiert, ohne sie tatsächlich abzurufen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Get-EC2Tag -DryRun $true
```

Ausgabe:

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVolumeAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVolumeAttribute` verwendet wird.

CLI

AWS CLI

So beschreiben Sie ein Volume-Attribut

Dieser Beispielbefehl beschreibt das Attribut `autoEnableIo` des Volumes mit der ID `vol-049df61146c4d7901`.

Befehl:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --
attribute autoEnableIO
```

Ausgabe:

```
{
    "AutoEnableIO": {
        "Value": false
    },
    "VolumeId": "vol-049df61146c4d7901"
}
```

- Einzelheiten zur API finden Sie DescribeVolumeAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Volumes beschrieben.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Ausgabe:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Einzelheiten zur API finden Sie unter DescribeVolumeAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Volumes beschrieben.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Ausgabe:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- Einzelheiten zur API finden Sie unter DescribeVolumeAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVolumeStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVolumeStatus` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Status eines einzelnen Volumes

Dieser Beispielbefehl beschreibt den Status des Volumes `vol-1234567890abcdef0`.

Befehl:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

Ausgabe:

```
{
    "VolumeStatuses": [
        {
            "VolumeStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "io-enabled"
                    },
                    {
                        "Status": "not-applicable",
                        "Name": "io-performance"
                    }
                ]
            },
            "AvailabilityZone": "us-east-1a",
            "VolumeId": "vol-1234567890abcdef0",
            "Actions": [],
            "Events": []
        }
    ]
}
```

So beschreiben Sie den Status eines beeinträchtigten Volumes

Dieser Beispielbefehl beschreibt den Status aller Volumes, die beeinträchtigt sind. In dieser Beispielausgabe gibt es keine beeinträchtigten Volumes.

Befehl:

```
aws ec2 describe-volume-status --filters Name=volume-
status.status,Values=impaired
```

Ausgabe:

```
{
    "VolumeStatuses": []
}
```

Informationen zu einem Volume mit nicht bestandener Statusprüfung (der Status lautet „impaired“) finden Sie unter „Arbeiten mit einem beeinträchtigten Volume“ im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DescribeVolumeStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Volumes beschrieben.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Ausgabe:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Ausgabe:

```
Details                        Status
-------                        ------
{io-enabled, io-performance}   ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Ausgabe:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- Einzelheiten zur API finden Sie unter DescribeVolumeStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Volumes beschrieben.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Ausgabe:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Ausgabe:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Ausgabe:

```
Name                            Status
----                            ------
```

```
io-enabled                      passed
io-performance                  not-applicable
```

- Einzelheiten zur API finden Sie unter DescribeVolumeStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVolumes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVolumes` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Blockspeicher-Volumes erstellen und verwalten
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

Beispiel 1: So beschreiben Sie ein Volume

Im folgenden Beispiel für `describe-volumes` werden die angegebenen Volumes in der aktuellen Region beschrieben.

```
aws ec2 describe-volumes \
    --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

Ausgabe:

```
{
    "Volumes": [
        {
            "AvailabilityZone": "us-east-1a",
            "Attachments": [
                {
                    "AttachTime": "2013-12-18T22:35:00.000Z",
                    "InstanceId": "i-1234567890abcdef0",
                    "VolumeId": "vol-049df61146c4d7901",
                    "State": "attached",
                    "DeleteOnTermination": true,
```

```
                    "Device": "/dev/sda1"
                }
            ],
            "Encrypted": true,
            "KmsKeyId": "arn:aws:kms:us-east-2a:123456789012:key/8c5b2c63-
b9bc-45a3-a87a-5513eEXAMPLE,
            "VolumeType": "gp2",
            "VolumeId": "vol-049df61146c4d7901",
            "State": "in-use",
            "Iops": 100,
            "SnapshotId": "snap-1234567890abcdef0",
            "CreateTime": "2019-12-18T22:35:00.084Z",
            "Size": 8
        },
        {
            "AvailabilityZone": "us-east-1a",
            "Attachments": [],
            "Encrypted": false,
            "VolumeType": "gp2",
            "VolumeId": "vol-1234567890abcdef0",
            "State": "available",
            "Iops": 300,
            "SnapshotId": "",
            "CreateTime": "2020-02-27T00:02:41.791Z",
            "Size": 100
        }
    ]
}
```

Beispiel 2: So beschreiben Sie Volumes, die an eine bestimmte Instance angehängt sind

Im folgenden Beispiel für `describe-volumes` werden alle Volumes beschrieben, die sowohl an die angegebene Instance angehängt als auch so eingestellt sind, dass sie gelöscht werden, wenn die Instance beendet wird.

```
aws ec2 describe-volumes \
    --region us-east-1 \
    --filters Name=attachment.instance-
id,Values=i-1234567890abcdef0 Name=attachment.delete-on-termination,Values=true
```

Ein Beispiel für die Ausgabe von `describe-volumes` finden Sie in Beispiel 1.

Beispiel 3: So beschreiben Sie verfügbare Volumes in einer bestimmten Availability Zone

Im folgenden Beispiel für `describe-volumes` werden alle Volumes beschrieben, die den Status `available` haben und sich in der angegebenen Availability Zone befinden.

```
aws ec2 describe-volumes \
    --filters Name=status,Values=available Name=availability-zone,Values=us-
east-1a
```

Ein Beispiel für die Ausgabe von `describe-volumes` finden Sie in Beispiel 1.

Beispiel 4: So beschreiben Sie Volumes anhand von Tags

Im folgenden Beispiel für `describe-volumes` werden alle Volumes beschrieben, die den Tag-Schlüssel `Name` und einen Wert haben, der mit `Test` beginnt. Die Ausgabe wird dann mit einer Abfrage gefiltert, die nur die Tags und IDs der Volumes anzeigt.

```
aws ec2 describe-volumes \
    --filters Name=tag:Name,Values=Test* \
    --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

Ausgabe:

```
[
    {
       "Tag": [
           {
               "Value": "Test2",
               "Key": "Name"
           }
       ],
       "ID": "vol-1234567890abcdef0"
   },
   {
       "Tag": [
           {
               "Value": "Test1",
               "Key": "Name"
           }
       ],
       "ID": "vol-049df61146c4d7901"
    }
]
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter Arbeiten mit Tags im Amazon-EC2-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeVolumesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene EBS-Volume beschrieben.

```
Get-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Beispiel 2: Dieses Beispiel beschreibt Ihre EBS-Volumes, die den Status „Verfügbar" aufweisen.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
```

```
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Beispiel 3: In diesem Beispiel werden alle Ihre EBS-Volumes beschrieben.

```
Get-EC2Volume
```

- Einzelheiten zur API finden Sie unter DescribeVolumes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene EBS-Volume beschrieben.

```
Get-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Beispiel 2: Dieses Beispiel beschreibt Ihre EBS-Volumes, die den Status „Verfügbar" aufweisen.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Beispiel 3: In diesem Beispiel werden alle Ihre EBS-Volumes beschrieben.

```
Get-EC2Volume
```

- Einzelheiten zur API finden Sie unter DescribeVolumes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVpcAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcAttribute` verwendet wird.

CLI

AWS CLI

Um das DnsSupport Enable-Attribut zu beschreiben

In diesem Beispiel wird das Attribut `enableDnsSupport` beschrieben. Dieses Attribut gibt an, ob für die VPC die DNS-Auflösung aktiviert ist. Wenn dieses Attribut `true` ist, löst der Amazon-DNS-Server die DNS-Hostnamen der Instances in die entsprechenden IP-Adressen auf. Andernfalls geschieht das nicht.

Befehl:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

Ausgabe:

```
{
    "VpcId": "vpc-a01106c2",
    "EnableDnsSupport": {
        "Value": true
    }
}
```

Um das DnsHostnames Enable-Attribut zu beschreiben

In diesem Beispiel wird das Attribut `enableDnsHostnames` beschrieben. Dieses Attribut gibt an, ob die in der VPC gestarteten Instances DNS-Hostnamen erhalten. Wenn dieses Attribut `true` ist, erhalten die Instances in der VPC DNS-Hostnamen. Andernfalls ist das nicht der Fall.

Befehl:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --
attribute enableDnsHostnames
```

Ausgabe:

```
{
    "VpcId": "vpc-a01106c2",
    "EnableDnsHostnames": {
        "Value": true
    }
}
```

- Einzelheiten zur API finden Sie DescribeVpcAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt das Attribut „enableDnsSupport“.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Ausgabe:

```
EnableDnsSupport
```

```
----------------
True
```

Beispiel 2: Dieses Beispiel beschreibt das Attribut „enableDnsHostnames“.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Ausgabe:

```
EnableDnsHostnames
------------------
True
```

- Einzelheiten zur API finden Sie unter DescribeVpcAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt das Attribut „enableDnsSupport“.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Ausgabe:

```
EnableDnsSupport
----------------
True
```

Beispiel 2: Dieses Beispiel beschreibt das Attribut „enableDnsHostnames“.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Ausgabe:

```
EnableDnsHostnames
------------------
True
```

- Einzelheiten zur API finden Sie unter DescribeVpcAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVpcClassicLink`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcClassicLink` verwendet wird.

CLI

AWS CLI

Um den ClassicLink Status Ihrer VPCs zu beschreiben

In diesem Beispiel wird der ClassicLink Status von vpc-88888888 aufgeführt.

Befehl:

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Vpcs": [
    {
      "ClassicLinkEnabled": true,
      "VpcId": "vpc-88888888",
      "Tags": [
        {
          "Value": "classiclinkvpc",
          "Key": "Name"
        }
      ]
    }
  ]
}
```

In diesem Beispiel werden nur VPCs aufgeführt, die für Classiclink aktiviert sind (der Filterwert `is-classic-link-enabled` ist auf `true` festgelegt).

Befehl:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-
enabled,Values=true"
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz. DescribeVpcClassicLinkAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Das obige Beispiel gibt alle VPCs mit ihrem ClassicLinkEnabled Status für die Region zurück

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Ausgabe:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLink AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Das obige Beispiel gibt alle VPCs mit ihrem ClassicLinkEnabled Status für die Region zurück

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Ausgabe:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLink AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVpcClassicLinkDnsSupport`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcClassicLinkDnsSupport` verwendet wird.

CLI

AWS CLI

Um die ClassicLink DNS-Unterstützung für Ihre VPCs zu beschreiben

In diesem Beispiel wird der Status der ClassicLink DNS-Unterstützung all Ihrer VPCs beschrieben.

Befehl:

```
aws ec2 describe-vpc-classic-link-dns-support
```

Ausgabe:

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeVpcClassicLinkDnsSupportin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt den ClassicLink DNS-Unterstützungsstatus von VPCs für die Region eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Ausgabe:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLinkDnsSupport AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt den ClassicLink DNS-Unterstützungsstatus von VPCs für die Region eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Ausgabe:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- Einzelheiten zur API finden Sie unter DescribeVpcClassicLinkDnsSupport AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVpcEndpointServices`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcEndpointServices` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle VPC-Endpunktservices

Das folgende `describe-vpc-endpoint-services` Beispiel listet alle VPC-Endpunktdienste für eine AWS Region auf.

```
aws ec2 describe-vpc-endpoint-services
```

Ausgabe:

```
{
    "ServiceDetails": [
        {
            "ServiceType": [
                {
                    "ServiceType": "Gateway"
                }
            ],
            "AcceptanceRequired": false,
            "ServiceName": "com.amazonaws.us-east-1.dynamodb",
            "VpcEndpointPolicySupported": true,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
                "us-east-1e",
                "us-east-1f"
            ],
            "BaseEndpointDnsNames": [
                "dynamodb.us-east-1.amazonaws.com"
            ]
        },
        {
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "PrivateDnsName": "ec2.us-east-1.amazonaws.com",
            "ServiceName": "com.amazonaws.us-east-1.ec2",
            "VpcEndpointPolicySupported": false,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
```

```
                "us-east-1e",
                "us-east-1f"
            ],
            "AcceptanceRequired": false,
            "BaseEndpointDnsNames": [
                "ec2.us-east-1.vpce.amazonaws.com"
            ]
        },
        {
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "PrivateDnsName": "ssm.us-east-1.amazonaws.com",
            "ServiceName": "com.amazonaws.us-east-1.ssm",
            "VpcEndpointPolicySupported": true,
            "Owner": "amazon",
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
                "us-east-1e"
            ],
            "AcceptanceRequired": false,
            "BaseEndpointDnsNames": [
                "ssm.us-east-1.vpce.amazonaws.com"
            ]
        }
    ],
    "ServiceNames": [
        "com.amazonaws.us-east-1.dynamodb",
        "com.amazonaws.us-east-1.ec2",
        "com.amazonaws.us-east-1.ec2messages",
        "com.amazonaws.us-east-1.elasticloadbalancing",
        "com.amazonaws.us-east-1.kinesis-streams",
        "com.amazonaws.us-east-1.s3",
        "com.amazonaws.us-east-1.ssm"
    ]
}
```

Beispiel 2: So beschreiben Sie die Details zu einem Endpunktservice

Im folgenden Beispiel für `describe-vpc-endpoint-services` werden die Details des Endpunktservice der Amazon-S3-Schnittstelle aufgelistet.

```
aws ec2 describe-vpc-endpoint-services \
    --filter 'Name=service-type,Values=Interface' Name=service-
name,Values=com.amazonaws.us-east-1.s3
```

Ausgabe:

```
{
    "ServiceDetails": [
        {
            "ServiceName": "com.amazonaws.us-east-1.s3",
            "ServiceId": "vpce-svc-081d84efcdEXAMPLE",
            "ServiceType": [
                {
                    "ServiceType": "Interface"
                }
            ],
            "AvailabilityZones": [
                "us-east-1a",
                "us-east-1b",
                "us-east-1c",
                "us-east-1d",
                "us-east-1e",
            "us-east-1f"
            ],
            "Owner": "amazon",
            "BaseEndpointDnsNames": [
                "s3.us-east-1.vpce.amazonaws.com"
            ],
            "VpcEndpointPolicySupported": true,
            "AcceptanceRequired": false,
            "ManagesVpcEndpoints": false,
            "Tags": []
        }
    ],
    "ServiceNames": [
        "com.amazonaws.us-east-1.s3"
    ]
}
```

Weitere Informationen finden Sie im AWS PrivateLink Benutzerhandbuch unter Verfügbare AWS Dienstnamen anzeigen.

- Einzelheiten zur API finden Sie DescribeVpcEndpointServicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der EC2 VPC-Endpunktservice mit dem angegebenen Filter beschrieben, in diesem Fall com.amazonaws.eu-west-1.ecs. Darüber hinaus erweitert es auch die ServiceDetails Eigenschaft und zeigt die Details an

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Ausgabe:

```
AcceptanceRequired         : False
AvailabilityZones          : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames       : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                      : amazon
PrivateDnsName             : ecs.eu-west-1.amazonaws.com
ServiceName                : com.amazonaws.eu-west-1.ecs
ServiceType                : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Beispiel 2: In diesem Beispiel werden alle EC2-VPC-Endpunktdienste abgerufen und das ServiceNames passende „ssm“ zurückgegeben

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Ausgabe:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeVpcEndpointServicesV4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der EC2 VPC-Endpunktservice mit dem angegebenen Filter beschrieben, in diesem Fall com.amazonaws.eu-west-1.ecs. Darüber hinaus erweitert es auch die ServiceDetails Eigenschaft und zeigt die Details an

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Ausgabe:

```
AcceptanceRequired         : False
AvailabilityZones          : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames       : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                      : amazon
PrivateDnsName             : ecs.eu-west-1.amazonaws.com
ServiceName                : com.amazonaws.eu-west-1.ecs
ServiceType                : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Beispiel 2: In diesem Beispiel werden alle EC2-VPC-Endpunktdienste abgerufen und das ServiceNames passende „ssm“ zurückgegeben

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Ausgabe:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeVpcEndpointServicesV5).

Verwendung von **`DescribeVpcEndpoints`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcEndpoints` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre VPC-Endpunkte

Im folgenden Beispiel für `describe-vpc-endpoints` werden Details für all Ihre VPC-Endpunkte angezeigt.

```
aws ec2 describe-vpc-endpoints
```

Ausgabe:

```
{
    "VpcEndpoints": [
        {
            "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*
\"}]}",
            "VpcId": "vpc-aabb1122",
            "NetworkInterfaceIds": [],
            "SubnetIds": [],
            "PrivateDnsEnabled": true,
            "State": "available",
            "ServiceName": "com.amazonaws.us-east-1.dynamodb",
            "RouteTableIds": [
                "rtb-3d560345"
            ],
            "Groups": [],
            "VpcEndpointId": "vpce-032a826a",
            "VpcEndpointType": "Gateway",
            "CreationTimestamp": "2017-09-05T20:41:28Z",
            "DnsEntries": [],
            "OwnerId": "123456789012"
        },
        {
            "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\":
 \"*\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n
 \"Resource\": \"*\"\n    }\n  ]\n}",
            "VpcId": "vpc-1a2b3c4d",
            "NetworkInterfaceIds": [
                "eni-2ec2b084",
                "eni-1b4a65cf"
```

```
            ],
            "SubnetIds": [
                "subnet-d6fcaa8d",
                "subnet-7b16de0c"
            ],
            "PrivateDnsEnabled": false,
            "State": "available",
            "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
            "RouteTableIds": [],
            "Groups": [
                {
                    "GroupName": "default",
                    "GroupId": "sg-54e8bf31"
                }
            ],
            "VpcEndpointId": "vpce-0f89a33420c1931d7",
            "VpcEndpointType": "Interface",
            "CreationTimestamp": "2017-09-05T17:55:27.583Z",
            "DnsEntries": [
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
                    "DnsName": "vpce-0f89a33420c1931d7-
bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                },
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
                    "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                },
                {
                    "HostedZoneId": "Z7HUB22UULQXV",
                    "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
                }
            ],
            "OwnerId": "123456789012"
        },
        {
            "VpcEndpointId": "vpce-aabbaabbaabbaabba",
            "VpcEndpointType": "GatewayLoadBalancer",
            "VpcId": "vpc-111122223333aabbc",
            "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-123123a1c43abc123",
            "State": "available",
```

```
            "SubnetIds": [
                "subnet-0011aabbcc2233445"
            ],
            "RequesterManaged": false,
            "NetworkInterfaceIds": [
                "eni-01010120203030405"
            ],
            "CreationTimestamp": "2020-11-11T08:06:03.522Z",
            "Tags": [],
            "OwnerId": "123456789012"
        }
    ]
}
```

Weitere Informationen finden Sie im AWS PrivateLink Benutzerhandbuch unter Konzepte.

- Einzelheiten zur API finden Sie DescribeVpcEndpointsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden ein oder mehrere Ihrer VPC-Endpunkte für die Region eu-west-1 beschrieben. Anschließend leitet es die Ausgabe an den nächsten Befehl weiter, der die VpcEndpointId Eigenschaft auswählt und die Array-VPC-ID als String-Array zurückgibt.

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
 VpcEndpointId
```

Ausgabe:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Beispiel 2: Dieses Beispiel beschreibt alle VPC-Endpunkte für die Region eu-west-1 und wählt VpcEndpointId,, ServiceName und PrivateDnsEnabled Eigenschaften aus VpcId, um sie in einem tabellarischen Format darzustellen

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Ausgabe:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
             True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
             True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
             True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
             True
```

Beispiel 3: Dieses Beispiel exportiert das Richtliniendokument für den VPC-Endpunkt vpce-01a2ab3f4f5cc6f7d in eine JSON-Datei

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
 Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeVpcEndpointsV4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden ein oder mehrere Ihrer VPC-Endpunkte für die Region eu-west-1 beschrieben. Anschließend leitet es die Ausgabe an den nächsten Befehl weiter, der die VpcEndpointId Eigenschaft auswählt und die Array-VPC-ID als String-Array zurückgibt.

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
 VpcEndpointId
```

Ausgabe:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
```

```
vpce-0c123db4567890123
```

Beispiel 2: Dieses Beispiel beschreibt alle VPC-Endpunkte für die Region eu-west-1 und wählt VpcEndpointId,, ServiceName und PrivateDnsEnabled Eigenschaften aus VpcId, um sie in einem tabellarischen Format darzustellen

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Ausgabe:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
             True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
             True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
             True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
             True
```

Beispiel 3: Dieses Beispiel exportiert das Richtliniendokument für den VPC-Endpunkt vpce-01a2ab3f4f5cc6f7d in eine JSON-Datei

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
 Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeVpcEndpointsV5).

Verwenden Sie mit einem **`DescribeVpcs`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpcs` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Eine relationale Datenbankinstanz erstellen
- Ein Service Networking Mesh erstellen und verwalten
- Blockspeicher-Volumes erstellen und verwalten
- Erste Schritte mit Dokumentendatenbanken
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Network Transit Gateways
- Virtuelles privates Netzwerk mit IP-Adressverwaltung
- Arbeiten mit Netzwerk-Peering-Verbindungen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
```

```
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeVpcsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle Ihre VPCs

Im folgenden Beispiel für `describe-vpcs` werden Details über Ihre VPCs abgerufen.

```
aws ec2 describe-vpcs
```

Ausgabe:

```
{
    "Vpcs": [
        {
            "CidrBlock": "30.1.0.0/16",
```

```
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-0e9801d129EXAMPLE",
            "OwnerId": "111122223333",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-062c64cfafEXAMPLE",
                    "CidrBlock": "30.1.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Not Shared"
                }
            ]
        },
        {
            "CidrBlock": "10.0.0.0/16",
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-06e4ab6c6cEXAMPLE",
            "OwnerId": "222222222222",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
                    "CidrBlock": "10.0.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Shared VPC"
                }
```

```
            ]
        }
    ]
}
```

Beispiel 2: So beschreiben Sie eine bestimmte VPC

Im folgenden Beispiel für `describe-vpcs` werden Details für die angegebene VPC abgerufen.

```
aws ec2 describe-vpcs \
    --vpc-ids vpc-06e4ab6c6cEXAMPLE
```

Ausgabe:

```
{
    "Vpcs": [
        {
            "CidrBlock": "10.0.0.0/16",
            "DhcpOptionsId": "dopt-19edf471",
            "State": "available",
            "VpcId": "vpc-06e4ab6c6cEXAMPLE",
            "OwnerId": "111122223333",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
                    "CidrBlock": "10.0.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Shared VPC"
                }
            ]
        }
    ]
```

```
}
```

- Einzelheiten zur API finden Sie DescribeVpcsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new EC2Client({});
const { Vpcs } = await client.send(
  new DescribeVpcsCommand({
    Filters: [{ Name: "is-default", Values: ["true"] }],
  }),
);
```

- Einzelheiten zur API finden Sie DescribeVpcsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene VPC beschrieben.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
```

```
Tags            : {Name}
VpcId           : vpc-12345678
```

Beispiel 2: In diesem Beispiel wird die Standard-VPC beschrieben (es kann nur eine pro Region geben). Wenn Ihr Konto EC2-Classic in dieser Region unterstützt, gibt es keine Standard-VPC.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Ausgabe:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Beispiel 3: In diesem Beispiel werden die VPCs beschrieben, die dem angegebenen Filter entsprechen (d. h. deren CIDR dem Wert „10.0.0“ entspricht). 0/16'und befinden sich im Status' verfügbar ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Beispiel 2: In diesem Beispiel werden alle Ihre VPCs beschrieben.

```
Get-EC2Vpc
```

- Einzelheiten zur API finden Sie unter DescribeVpcs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC beschrieben.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Beispiel 2: In diesem Beispiel wird die Standard-VPC beschrieben (es kann nur eine pro Region geben). Wenn Ihr Konto EC2-Classic in dieser Region unterstützt, gibt es keine Standard-VPC.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Ausgabe:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Beispiel 3: In diesem Beispiel werden die VPCs beschrieben, die dem angegebenen Filter entsprechen (d. h. deren CIDR dem Wert „10.0.0“ entspricht). 0/16'und befinden sich im Status' verfügbar ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Beispiel 2: In diesem Beispiel werden alle Ihre VPCs beschrieben.

```
Get-EC2Vpc
```

- Einzelheiten zur API finden Sie unter DescribeVpcs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

- Einzelheiten zur API finden Sie DescribeVpcsin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DescribeVpnConnections`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpnConnections` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Ihre VPN-Verbindungen

Das folgende `describe-vpn-connections` Beispiel beschreibt all Ihre Site-to-Site VPN-Verbindungen.

```
aws ec2 describe-vpn-connections
```

Ausgabe:

```
{
    "VpnConnections": [
        {
            "CustomerGatewayConfiguration": "...configuration information...",
            "CustomerGatewayId": "cgw-01234567abcde1234",
            "Category": "VPN",
            "State": "available",
            "Type": "ipsec.1",
```

```
            "VpnConnectionId": "vpn-1122334455aabbccd",
            "TransitGatewayId": "tgw-00112233445566aab",
            "Options": {
                "EnableAcceleration": false,
                "StaticRoutesOnly": true,
                "LocalIpv4NetworkCidr": "0.0.0.0/0",
                "RemoteIpv4NetworkCidr": "0.0.0.0/0",
                "TunnelInsideIpVersion": "ipv4"
            },
            "Routes": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "CanadaVPN"
                }
            ],
            "VgwTelemetry": [
                {
                    "AcceptedRouteCount": 0,
                    "LastStatusChange": "2020-07-29T10:35:11.000Z",
                    "OutsideIpAddress": "203.0.113.3",
                    "Status": "DOWN",
                    "StatusMessage": ""
                },
                {
                    "AcceptedRouteCount": 0,
                    "LastStatusChange": "2020-09-02T09:09:33.000Z",
                    "OutsideIpAddress": "203.0.113.5",
                    "Status": "UP",
                    "StatusMessage": ""
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

Beispiel 2: So beschreiben Sie Ihre verfügbaren VPN-Verbindungen

Das folgende `describe-vpn-connections` Beispiel beschreibt Ihre Site-to-Site VPN-Verbindungen mit dem Status`available`.

```
aws ec2 describe-vpn-connections \
    --filters "Name=state,Values=available"
```

Weitere Informationen finden Sie unter So funktioniert AWS Site-to-Site VPN im AWS Site-to-Site VPN-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeVpnConnectionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung beschrieben.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
Type                         : ipsec.1
VgwTelemetry                 : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel werden alle VPN-Verbindungen beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Beispiel 4: In diesem Beispiel werden alle Ihre VPN-Verbindungen beschrieben.

```
Get-EC2VpnConnection
```

- Einzelheiten zur API finden Sie unter DescribeVpnConnections AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung beschrieben.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
Type                         : ipsec.1
VgwTelemetry                 : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel werden alle VPN-Verbindungen beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Beispiel 4: In diesem Beispiel werden alle Ihre VPN-Verbindungen beschrieben.

```
Get-EC2VpnConnection
```

- Einzelheiten zur API finden Sie unter DescribeVpnConnections AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeVpnGateways`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVpnGateways` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Beginnen Sie mit dedizierten Netzwerkverbindungen

CLI

AWS CLI

So beschreiben Sie Ihre Virtual Private Gateways

In diesem Beispiel werden Ihre Virtual Private Gateways beschrieben.

Befehl:

```
aws ec2 describe-vpn-gateways
```

Ausgabe:

```
{
    "VpnGateways": [
        {
            "State": "available",
            "Type": "ipsec.1",
            "VpnGatewayId": "vgw-f211f09b",
            "VpcAttachments": [
                {
                    "State": "attached",
                    "VpcId": "vpc-98eb5ef5"
                }
            ]
        },
        {
            "State": "available",
            "Type": "ipsec.1",
            "VpnGatewayId": "vgw-9a4cacf3",
            "VpcAttachments": [
                {
```

```
                    "State": "attaching",
                    "VpcId": "vpc-a01106c2"
                }
            ]
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeVpnGatewaysin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird jedes Virtual Private Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Virtual Private Gateways beschrieben.

```
Get-EC2VpnGateway
```

- Einzelheiten zur API finden Sie unter DescribeVpnGateways AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird jedes Virtual Private Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Virtual Private Gateways beschrieben.

```
Get-EC2VpnGateway
```

- Einzelheiten zur API finden Sie unter DescribeVpnGateways AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachInternetGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachInternetGateway` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken
- Virtuelles privates Netzwerk mit privaten Servern

CLI

AWS CLI

So trennen Sie ein Internet-Gateway von Ihrer VPC

Im folgenden Beispiel für `detach-internet-gateway` wird das spezifische Internet-Gateway von der angegebenen VPC getrennt.

```
aws ec2 detach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Internet-Gateways im Benutzerhandbuch für Amazon VPC.

- Einzelheiten zur API finden Sie DetachInternetGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter DetachInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter DetachInternetGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachNetworkInterface`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachNetworkInterface` verwendet wird.

CLI

AWS CLI

So trennen Sie eine Netzwerkschnittstelle von Ihrer Instance

In diesem Beispiel wird die angegebene Netzwerkschnittstelle von der spezifischen Instance getrennt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- Einzelheiten zur API finden Sie DetachNetworkInterfacein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Verbindung zwischen einer Netzwerkschnittstelle und einer Instance entfernt.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Einzelheiten zur API finden Sie unter DetachNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Verbindung zwischen einer Netzwerkschnittstelle und einer Instance entfernt.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- Einzelheiten zur API finden Sie unter DetachNetworkInterface AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachVolume`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachVolume` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Blockspeicher-Volumes erstellen und verwalten
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

CLI

AWS CLI

So trennen Sie ein Volume von einer Instance

Dieser Beispielbefehl trennt das Volume (`vol-049df61146c4d7901`) von der Instance, an die es angehängt ist.

Befehl:

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

Ausgabe:

```
{
    "AttachTime": "2014-02-27T19:23:06.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "VolumeId": "vol-049df61146c4d7901",
    "State": "detaching",
    "Device": "/dev/sdb"
}
```

- Einzelheiten zur API finden Sie DetachVolumein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Beispiel 2: Sie können auch die Instance-ID und den Gerätenamen angeben, um sicherzustellen, dass Sie das richtige Volume trennen.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Einzelheiten zur API finden Sie unter DetachVolume AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Beispiel 2: Sie können auch die Instance-ID und den Gerätenamen angeben, um sicherzustellen, dass Sie das richtige Volume trennen.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- Einzelheiten zur API finden Sie unter DetachVolume AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachVpnGateway`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachVpnGateway` verwendet wird.

CLI

AWS CLI

So trennen Sie ein Virtual Private Gateway von Ihrer VPC

In diesem Beispiel wird das angegebene Virtual Private Gateway von der festgelegten VPC getrennt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 detach-vpn-gateway --vpn-gateway-id vgw-9a4cacf3 --vpc-id vpc-a01106c2
```

- Einzelheiten zur API finden Sie DetachVpnGatewayin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter DetachVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- Einzelheiten zur API finden Sie unter DetachVpnGateway AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DisableVgwRoutePropagation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisableVgwRoutePropagation` verwendet wird.

CLI

AWS CLI

So deaktivieren Sie die Routing-Propagierung

In diesem Beispiel wird das Propagieren von statischen Routen an die angegebene Routing-Tabelle durch das festgelegte Virtual Private Gateway deaktiviert. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-
id vgw-9a4cacf3
```

- Einzelheiten zur API finden Sie DisableVgwRoutePropagationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel deaktiviert das VGW, damit es Routen nicht automatisch an die angegebene Routing-Tabelle weitergibt.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisableVgwRoutePropagation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel deaktiviert das VGW, damit es Routen nicht automatisch an die angegebene Routing-Tabelle weitergibt.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisableVgwRoutePropagation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DisableVpcClassicLink`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisableVpcClassicLink` verwendet wird.

CLI

AWS CLI

Zur Deaktivierung ClassicLink für eine VPC

Dieses Beispiel deaktiviert ClassicLink für vpc-8888888.

Befehl:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz. DisableVpcClassicLinkAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel deaktiviert EC2VpcClassicLink für vpc-01e23c4a5d6db78e9. Es wird entweder „True“ oder „False“ zurückgegeben

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Einzelheiten AWS -Tools für PowerShell zur API DisableVpcClassicLinkfinden Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel deaktiviert EC2VpcClassicLink für vpc-01e23c4a5d6db78e9. Es wird entweder „True“ oder „False“ zurückgegeben

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- Einzelheiten AWS -Tools für PowerShell zur API DisableVpcClassicLinkfinden Sie unter Cmdlet-Referenz (V5).

Verwendung von **`DisableVpcClassicLinkDnsSupport`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisableVpcClassicLinkDnsSupport` verwendet wird.

CLI

AWS CLI

So deaktivieren Sie die ClassicLink DNS-Unterstützung für eine VPC

In diesem Beispiel wird die ClassicLink DNS-Unterstützung für deaktiviert. `vpc-88888888`

Befehl:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- Einzelheiten zur API finden Sie DisableVpcClassicLinkDnsSupportin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die ClassicLink DNS-Unterstützung für die Datei vpc-0b12d3456a7e8910d deaktiviert

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Einzelheiten zur API DisableVpcClassicLinkDnsSupportfinden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die ClassicLink DNS-Unterstützung für die Datei vpc-0b12d3456a7e8910d deaktiviert

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- Einzelheiten zur API DisableVpcClassicLinkDnsSupportfinden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V5).

Verwenden Sie mit einem **`DisassociateAddress`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisassociateAddress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId =
associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DisassociateAddressin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
 Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
 association of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DisassociateAddressin der AWS CLI Befehlsreferenz.

CLI

AWS CLI

So trennen Sie die Zuordnung von Elastic IP-Adressen in EC2-Classic

In diesem Beispiel wird die Zuordnung einer Elastic IP-Adresse zu einer Instance in aufgehoben. EC2-Classic Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

Um die Zuordnung einer Elastic IP-Adresse zu trennen EC2-VPC

In diesem Beispiel wird eine Elastic-IP-Adresse von einer Instance in einer VPC getrennt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- Einzelheiten zur API finden Sie DisassociateAddressin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
```

```
     */
    public CompletableFuture<DisassociateAddressResponse>
 disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
 DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
 ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

- Einzelheiten zur API finden Sie DisassociateAddressin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
```

```
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DisassociateAddressin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
```

```
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- Einzelheiten zur API finden Sie DisassociateAddressin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse von der angegebenen Instance in einer VPC getrennt.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die Zuordnung zwischen der angegebenen Elastic IP-Adresse und der angegebenen Instance in EC2-Classic aufgehoben.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Einzelheiten zur API finden Sie unter DisassociateAddress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse von der angegebenen Instance in einer VPC getrennt.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die Zuordnung zwischen der angegebenen Elastic IP-Adresse und der angegebenen Instance in aufgehoben. EC2-Classic

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- Einzelheiten zur API finden Sie unter DisassociateAddress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
```

```
                            access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
When the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:
```

```
self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP
address was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise
```

- Einzelheiten zur API finden Sie DisassociateAddressin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
```

```
            .send()
            .await?;
        Ok(())
    }
```

- Einzelheiten zur API finden Sie DisassociateAddressin der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
            print("*** Unable to disassociate the IP address:
 \(error.localizedDescription)")
        }
    }
```

- Einzelheiten zur API finden Sie DisassociateAddressin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DisassociateRouteTable`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisassociateRouteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Arbeiten mit Netzwerk-Peering-Verbindungen

CLI

AWS CLI

So heben Sie die Zuordnung einer Routing-Tabelle auf

In diesem Beispiel wird die Zuordnung der angegebenen Routing-Tabelle zu dem festgelegten Subnetz aufgehoben. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- Einzelheiten zur API finden Sie DisassociateRouteTablein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Zuordnung zwischen einer Routing-Tabelle und einem Subnetz entfernt.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisassociateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zuordnung zwischen einer Routing-Tabelle und einem Subnetz entfernt.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DisassociateRouteTable AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EnableVgwRoutePropagation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableVgwRoutePropagation` verwendet wird.

CLI

AWS CLI

So aktivieren Sie die Routing-Verteilung

In diesem Beispiel wird das Verteilen von statischen Routen an die angegebene Routing-Tabelle durch das festgelegte Virtual Private Gateway aktiviert. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-
id vgw-9a4cacf3
```

- Einzelheiten zur API finden Sie EnableVgwRoutePropagationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die automatische Weitergabe von Routen durch die angegebene VGW an die angegebene Routing-Tabelle aktiviert.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter EnableVgwRoutePropagation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die automatische Weitergabe von Routen durch die angegebene VGW an die angegebene Routing-Tabelle aktiviert.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter EnableVgwRoutePropagation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EnableVolumeIo`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableVolumeIo` verwendet wird.

CLI

AWS CLI

Zur Aktivierung I/O für ein Volume

In diesem Beispiel wird ein Volume I/O aktiviert`vol-1234567890abcdef0`.

Befehl:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

Ausgabe:

```
{
  "Return": true
}
```

- Einzelheiten zur API finden Sie EnableVolumeIoin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden I/O Operationen für das angegebene Volume aktiviert, wenn I/O Operationen deaktiviert wurden.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Einzelheiten zur API finden Sie unter EnableVolumeIo AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden I/O Operationen für das angegebene Volume aktiviert, wenn I/O Operationen deaktiviert wurden.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- Einzelheiten zur API finden Sie unter EnableVolumeIo AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EnableVpcClassicLink`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableVpcClassicLink` verwendet wird.

CLI

AWS CLI

So aktivieren Sie eine VPC für ClassicLink

In diesem Beispiel wird vpc-8888888 für aktiviert. ClassicLink

Befehl:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
```

```
}
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz EnableVpcClassicLink.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel aktiviert VPC vpc-0123456b789b0d12f für ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Ausgabe:

```
True
```

- Einzelheiten zur API EnableVpcClassicLinkfinden Sie AWS -Tools für PowerShell unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel aktiviert VPC vpc-0123456b789b0d12f für ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Ausgabe:

```
True
```

- Einzelheiten zur API EnableVpcClassicLinkfinden Sie AWS -Tools für PowerShell unter Cmdlet-Referenz (V5).

Verwendung von **`EnableVpcClassicLinkDnsSupport`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableVpcClassicLinkDnsSupport` verwendet wird.

CLI

AWS CLI

So aktivieren Sie die ClassicLink DNS-Unterstützung für eine VPC

In diesem Beispiel wird die ClassicLink DNS-Unterstützung für `vpc-88888888` aktiviert.

Befehl:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- Einzelheiten zur API finden Sie EnableVpcClassicLinkDnsSupportin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel ermöglicht vpc-0b12d3456a7e8910d die Unterstützung der DNS-Hostnamenauflösung für ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Einzelheiten zur API EnableVpcClassicLinkDnsSupportfinden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel ermöglicht vpc-0b12d3456a7e8910d die Unterstützung der DNS-Hostnamenauflösung für ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- Einzelheiten zur API EnableVpcClassicLinkDnsSupportfinden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V5).

Verwendung von **`GetConsoleOutput`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetConsoleOutput` verwendet wird.

CLI

AWS CLI

Beispiel 1: So rufen Sie die Konsolenausgabe ab

Im folgenden Beispiel für `get-console-output` wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen.

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-07-25T21:23:53.000Z",
    "Output": "..."
}
```

Weitere Informationen finden Sie unter Instance-Konsolenausgabe im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So rufen Sie die neueste Konsolenausgabe ab

Im folgenden Beispiel für `get-console-output` wird die neueste Konsolenausgabe für die angegebene Linux-Instance abgerufen.

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0 \
    --latest \
    --output text
```

Ausgabe:

```
i-1234567890abcdef0 [    0.000000] Command line: root=LABEL=/ console=tty1
 console=ttyS0 selinux=0 nvme_core.io_timeout=4294967295
[    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point
 registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[    0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
```

```
...
Cloud-init v. 0.7.6 finished at Wed, 09 May 2018 19:01:13 +0000. Datasource
 DataSourceEc2.  Up 21.50 seconds
Amazon Linux AMI release 2018.03
Kernel 4.14.26-46.32.amzn1.x
```

Weitere Informationen finden Sie unter Instance-Konsolenausgabe im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie GetConsoleOutputin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen. Die Konsolenausgabe ist verschlüsselt.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Ausgabe:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Beispiel 2: In diesem Beispiel wird die verschlüsselte Konsolenausgabe in einer Variablen gespeichert und anschließend entschlüsselt.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded
```

- Einzelheiten zur API finden Sie unter GetConsoleOutput AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen. Die Konsolenausgabe ist verschlüsselt.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Ausgabe:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Beispiel 2: In diesem Beispiel wird die verschlüsselte Konsolenausgabe in einer Variablen gespeichert und anschließend entschlüsselt.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded
```

- Einzelheiten zur API finden Sie unter GetConsoleOutput AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetHostReservationPurchasePreview`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetHostReservationPurchasePreview` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Vorschau zum Kauf einer Dedicated-Host-Reservierung ab

Dieses Beispiel bietet eine Vorschau der Kosten für eine bestimmte Dedicated-Host-Reservierung für den angegebenen Dedicated Host in Ihrem Konto.

Befehl:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324
 --host-id-set h-013abcd2a00cbd123
```

Ausgabe:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
      {
          "HourlyPrice": "1.499",
```

```
            "InstanceFamily": "m4",
            "PaymentOption": "NoUpfront",
            "HostIdSet": [
                "h-013abcd2a00cbd123"
            ],
            "UpfrontPrice": "0.000",
            "Duration": 31536000
        }
    ],
    "TotalUpfrontPrice": "0.000"
}
```

- Einzelheiten zur API finden Sie GetHostReservationPurchasePreviewin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt eine Vorschau einer Reservierung mit Konfigurationen, die denen Ihres Dedicated Host h-01e23f4cd567890f1 entsprechen

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- --------------- -----------------
             {}       1.307           0.000
```

- Einzelheiten zur API finden Sie unter GetHostReservationPurchasePreview AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt eine Vorschau einer Reservierung mit Konfigurationen, die denen Ihres Dedicated Host h-01e23f4cd567890f1 entsprechen

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- --------------- -----------------
             {}       1.307           0.000
```

- Einzelheiten zur API finden Sie unter GetHostReservationPurchasePreview AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetPasswordData`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetPasswordData` verwendet wird.

CLI

AWS CLI

So rufen Sie das verschlüsselte Passwort ab

In diesem Beispiel wird das verschlüsselte Passwort abgerufen.

Befehl:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-07T22:18:38.000Z",
    "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+gaOuNOenX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYP7WmU3TUnhsuBd+p6LVk7T2lKUml6OXbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFigzhniQgDDeO1InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVelVz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSFl4ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yLlf3etP2L4ZR6CvG7K1hx7VKOQVN32Dajw=="
}
```

So rufen Sie das entschlüsselte Passwort ab

In diesem Beispiel wird das entschlüsselte Passwort abgerufen.

Befehl:

```
aws ec2 get-password-data --instance-id  i-1234567890abcdef0 --priv-launch-key C:
\Keys\MyKeyPair.pem
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-30T23:18:05.000Z",
    "PasswordData": "&ViJ652e*u"
}
```

- Einzelheiten zur API finden Sie GetPasswordDatain der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
AWS Management Console.\s
            """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }
       String instanceId = args[0];
       Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
           .region(Region.US_EAST_1)
           .build();

       try {
           CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
instanceId);
           future.join();
       } catch (RuntimeException rte) {
           System.err.println("An exception occurred: " + (rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage()));
       }
   }

   /**
    * Fetches the password data for the specified EC2 instance asynchronously.
    *
    * @param ec2AsyncClient the EC2 asynchronous client to use for the request
    * @param instanceId instanceId the ID of the EC2 instance for which you want
to fetch the password data
    * @return a {@link CompletableFuture} that completes when the password data
has been fetched
    * @throws RuntimeException if there was a failure in fetching the password
data
    */
   public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
ec2AsyncClient, String instanceId) {
       GetPasswordDataRequest getPasswordDataRequest =
GetPasswordDataRequest.builder()
           .instanceId(instanceId)
```

```
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance:
 " + instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- Einzelheiten zur API finden Sie GetPasswordDatain der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Passwort entschlüsselt, das Amazon EC2 dem Administratorkonto für die angegebene Windows-Instance zugeordnet hat. Da eine PEM-Datei angegeben wurde, wird automatisch die Einstellung des Schalters -Decrypt übernommen.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 2: (PowerShell nur Windows) Überprüft die Instanz, um den Namen des Schlüsselpaars zu ermitteln, das zum Starten der Instanz verwendet wurde, und versucht dann, die entsprechenden Schlüsselpaardaten im Konfigurationsspeicher des AWS Toolkit for Visual Studio zu finden. Wenn die Schlüsselpaardaten gefunden werden, wird das Passwort entschlüsselt.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 3: Gibt die verschlüsselten Passwortdaten für die Instance zurück.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Ausgabe:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Einzelheiten zur API finden Sie unter GetPasswordDataCmdlet-Referenz (V4).AWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Passwort entschlüsselt, das Amazon EC2 dem Administratorkonto für die angegebene Windows-Instance zugeordnet hat. Da eine PEM-Datei angegeben wurde, wird automatisch die Einstellung des Schalters -Decrypt übernommen.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 2: (PowerShell nur Windows) Überprüft die Instanz, um den Namen des Schlüsselpaars zu ermitteln, das zum Starten der Instanz verwendet wurde, und versucht

dann, die entsprechenden Schlüsselpaardaten im Konfigurationsspeicher des AWS Toolkit for Visual Studio zu finden. Wenn die Schlüsselpaardaten gefunden werden, wird das Passwort entschlüsselt.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 3: Gibt die verschlüsselten Passwortdaten für die Instance zurück.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Ausgabe:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- Einzelheiten zur API finden Sie unter GetPasswordDataCmdlet-Referenz (V5).AWS -Tools für PowerShell

Verwendung von **`ImportImage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ImportImage` verwendet wird.

CLI

AWS CLI

So importieren Sie eine VM-Image-Datei als AMI

Im folgenden Beispiel für `import-image` wird das angegebene OVA importiert.

```
aws ec2 import-image \
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/
my-server-vm.ova}"
```

Ausgabe:

```
{
```

```
    "ImportTaskId": "import-ami-1234567890abcdef0",
    "Progress": "2",
    "SnapshotDetails": [
        {
            "DiskImageSize": 0.0,
            "Format": "ova",
            "UserBucket": {
                "S3Bucket": "my-import-bucket",
                "S3Key": "vms/my-server-vm.ova"
            }
        }
    ],
    "Status": "active",
    "StatusMessage": "pending"
}
```

- Einzelheiten zur API finden Sie ImportImagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein virtuelles Maschinen-Image mit einem einzigen Datenträger aus dem angegebenen Amazon-S3-Bucket mit einem Idempotenz-Token in Amazon EC2 importiert. Das Beispiel setzt voraus, dass eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ vorhanden ist, deren Richtlinie Amazon EC2 den Zugriff auf den angegebenen Bucket erlaubt. Dies wird im Abschnitt über die Voraussetzungen für den VM-Import erläutert. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters **-RoleName** an.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}
```

```
Import-EC2Image -DiskContainer $container @parms
```

Ausgabe:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Einzelheiten zur API finden Sie unter ImportImage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein virtuelles Maschinen-Image mit einem einzigen Datenträger aus dem angegebenen Amazon-S3-Bucket mit einem Idempotenz-Token in Amazon EC2 importiert. Das Beispiel setzt voraus, dass eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ vorhanden ist, deren Richtlinie Amazon EC2 den Zugriff auf den angegebenen Bucket erlaubt. Dies wird im Abschnitt über die Voraussetzungen für den VM-Import erläutert. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters **-RoleName** an.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}
```

```
Import-EC2Image -DiskContainer $container @parms
```

Ausgabe:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- Einzelheiten zur API finden Sie unter ImportImage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ImportKeyPair`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ImportKeyPair` verwendet wird.

CLI

AWS CLI

So importieren Sie einen öffentlichen Schlüssel

Generieren Sie zunächst ein Schlüsselpaar mit einem Tool Ihrer Wahl. Verwenden Sie beispielsweise den folgenden ssh-keygen-Befehl:

Befehl:

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

Ausgabe:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/my-key.
Your public key has been saved in /home/ec2-user/.ssh/my-key.pub.
...
```

Dieser Beispielbefehl importiert den angegebenen öffentlichen Schlüssel.

Befehl:

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/
my-key.pub
```

Ausgabe:

```
{
  "KeyName": "my-key",
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- Einzelheiten zur API finden Sie ImportKeyPairin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein öffentlicher Schlüssel nach EC2 importiert. Die erste Zeile speichert den Inhalt der öffentlichen Schlüsseldatei (*.pub) in der Variablen **$publickey**. Als Nächstes konvertiert das Beispiel das UTF8-Format der Datei mit dem öffentlichen Schlüssel in eine Base64-encoded Zeichenfolge und speichert die konvertierte Zeichenfolge in der Variablen. **$pkbase64** In der letzten Zeile wird der konvertierte öffentliche Schlüssel in EC2 importiert. Das Cmdlet gibt den Fingerabdruck und den Namen des Schlüssels als Ergebnisse zurück.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Ausgabe:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Einzelheiten zur API finden Sie unter ImportKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein öffentlicher Schlüssel nach EC2 importiert. Die erste Zeile speichert den Inhalt der öffentlichen Schlüsseldatei (*.pub) in der Variablen **$publickey**. Als Nächstes konvertiert das Beispiel das UTF8-Format der Datei mit dem öffentlichen Schlüssel in eine Base64-encoded Zeichenfolge und speichert die konvertierte Zeichenfolge in der Variablen. **$pkbase64** In der letzten Zeile wird der konvertierte öffentliche Schlüssel in EC2 importiert. Das Cmdlet gibt den Fingerabdruck und den Namen des Schlüssels als Ergebnisse zurück.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Ausgabe:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- Einzelheiten zur API finden Sie unter ImportKeyPair AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **ImportSnapshot** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ImportSnapshot` verwendet wird.

CLI

AWS CLI

So importieren Sie einen Snapshot

Im folgenden Beispiel für `import-snapshot` wird der angegebene Datenträger als Snapshot importiert.

```
aws ec2 import-snapshot \
    --description "My server VMDK" \
    --disk-container Format=VMDK,UserBucket={'S3Bucket=my-import-
bucket,S3Key=vms/my-server-vm.vmdk'}
```

Ausgabe:

```
{
    "Description": "My server VMDK",
    "ImportTaskId": "import-snap-1234567890abcdef0",
    "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "0.0",
        "Format": "VMDK",
        "Progress": "3",
        "Status": "active",
        "StatusMessage": "pending"
        "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.vmdk"
        }
    }
}
```

- Einzelheiten zur API finden Sie ImportSnapshotin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein VM-Datenträger-Image im Format „VMDK“ in einen Amazon EBS-Snapshot importiert. Das Beispiel erfordert eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ mit einer Richtlinie, die Amazon EC2 EC2-Zugriff auf den angegebenen Bucket ermöglicht, wie im **`VM Import Prequisites`** Thema unter Um eine benutzerdefinierte Rolle http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. zu verwenden, geben Sie den Rollennamen mithilfe des **`-RoleName`** Parameters an.

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Ausgabe:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Einzelheiten zur API finden Sie unter ImportSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein VM-Datenträger-Image im Format „VMDK“ in einen Amazon EBS-Snapshot importiert. Das Beispiel erfordert eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ mit einer Richtlinie, die Amazon EC2 EC2-Zugriff auf den angegebenen Bucket ermöglicht, wie im **`VM Import Prequisites`** Thema unter Um eine benutzerdefinierte Rolle http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. zu verwenden, geben Sie den Rollennamen mithilfe des **`-RoleName`** Parameters an.

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}
```

```
Import-EC2Snapshot @parms
```

Ausgabe:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       -----------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- Einzelheiten zur API finden Sie unter ImportSnapshot AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyCapacityReservation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyCapacityReservation` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie die Anzahl der Instances, die durch eine bestehende Kapazitätsreservierung reserviert wurden

Im folgenden Beispiel für `modify-capacity-reservation` wird die Anzahl der Instances geändert, für die die Kapazitätsreservierung Kapazität reserviert.

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --instance-count 5
```

Ausgabe:

```
{
    "Return": true
}
```

Weitere Informationen finden Sie unter Ändern einer Kapazitätsreservierung im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So ändern Sie das Enddatum und die Endzeit für eine bestehende Kapazitätsreservierung

Im folgenden Beispiel für `modify-capacity-reservation` wird eine bestehende Kapazitätsreservierung so geändert, dass sie am angegebenen Datum und zur angegebenen Uhrzeit endet.

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --end-date-type limited \
    --end-date 2019-08-31T23:59:59Z
```

Weitere Informationen finden Sie unter Ändern einer Kapazitätsreservierung im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie ModifyCapacityReservationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird CapacityReservationId cr-0c1f2345db6f7cdba geändert, indem die Instanzenanzahl auf 1 geändert wird

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Ausgabe:

```
True
```

- Einzelheiten zur ModifyCapacityReservationAPI finden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird CapacityReservationId cr-0c1f2345db6f7cdba geändert, indem die Anzahl der Instanzen auf 1 geändert wird

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Ausgabe:

```
True
```

- Einzelheiten zur ModifyCapacityReservationAPI finden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V5).

Verwendung von **`ModifyHosts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyHosts` verwendet wird.

CLI

AWS CLI

Beispiel 1: So aktivieren Sie die automatische Platzierung für einen Dedicated Host

Im folgenden Beispiel für `modify-hosts` wird die automatische Platzierung für einen Dedicated Host aktiviert, sodass er alle nicht zielgerichteten Instance-Starts, die mit seiner Instance-Typkonfiguration übereinstimmen, akzeptiert.

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --auto-placement on
```

Ausgabe:

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

Weitere Informationen finden Sie unter Ändern der Einstellung für die automatische Platzierung für einen Dedicated Host im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So aktivieren Sie die Host-Wiederherstellung für einen Dedicated Host

Im folgenden Beispiel für `modify-hosts` wird die Host-Wiederherstellung für den angegebenen Dedicated Host aktiviert.

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --host-recovery on
```

Ausgabe:

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

Weitere Informationen finden Sie unter Ändern der Einstellung für die automatische Platzierung für einen Dedicated Host im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie ModifyHostsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die AutoPlacement Einstellungen für den dedizierten Host h-01e23f4cd567890f3 auf Aus geändert

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Ausgabe:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Einzelheiten zur ModifyHostsAPI finden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die AutoPlacement Einstellungen für den dedizierten Host h-01e23f4cd567890f3 auf Aus geändert

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Ausgabe:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- Einzelheiten zur ModifyHostsAPI finden AWS -Tools für PowerShell Sie unter Cmdlet-Referenz (V5).

Verwendung von **`ModifyIdFormat`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyIdFormat` verwendet wird.

CLI

AWS CLI

So aktivieren Sie das längere ID-Format für eine Ressource

Im folgenden Beispiel für `modify-id-format` wird das längere ID-Format für den `instance`-Ressourcentyp aktiviert.

```
aws ec2 modify-id-format \
    --resource instance \
    --use-long-ids
```

So deaktivieren Sie das längere ID-Format für eine Ressource

Im folgenden Beispiel für `modify-id-format` wird das längere ID-Format für den `instance`-Ressourcentyp deaktiviert.

```
aws ec2 modify-id-format \
    --resource instance \
    --no-use-long-ids
```

Im folgenden Beispiel für `modify-id-format` wird das längere ID-Format für alle unterstützten Ressourcentypen aktiviert, die sich innerhalb des Opt-in-Zeitraums befinden.

```
aws ec2 modify-id-format \
    --resource all-current \
    --use-long-ids
```

- Einzelheiten zur API finden Sie ModifyIdFormatin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp aktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Beispiel 2: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp deaktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Einzelheiten zur API finden Sie unter ModifyIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp aktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Beispiel 2: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp deaktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- Einzelheiten zur API finden Sie unter ModifyIdFormat AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyImageAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyImageAttribute` verwendet wird.

CLI

AWS CLI

Beispiel 1: So machen Sie ein AMI öffentlich zugänglich

Im folgenden Beispiel für `modify-instance-attribute` wird das angegebene AMI öffentlich zugänglich gemacht.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{Group=all}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So machen Sie ein AMI privat zugänglich

Im folgenden Beispiel für `modify-instance-attribute` wird das angegebene AMI privat zugänglich gemacht.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{Group=all}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: Um einem AWS Konto die Startberechtigung zu erteilen

Im folgenden `modify-instance-attribute` Beispiel werden dem angegebenen AWS Konto Startberechtigungen erteilt.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{UserId=123456789012}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 4: Um einem AWS Konto die Startberechtigung zu entziehen

Im folgenden `modify-instance-attribute` Beispiel werden die Startberechtigungen aus dem angegebenen AWS Konto entfernt.

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{UserId=123456789012}]"
```

- Einzelheiten zur API finden Sie ModifyImageAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI aktualisiert.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Beispiel 2: In diesem Beispiel wird das AMI öffentlich gemacht (damit es beispielsweise von jedem verwendet AWS-Konto werden kann).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Beispiel 3: In diesem Beispiel wird das AMI privat gemacht (beispielsweise, damit nur Sie als Besitzer es verwenden können).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Beispiel 4: In diesem Beispiel wird dem angegebenen Benutzer die Startberechtigung erteilt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Beispiel 5: In diesem Beispiel wird die Startberechtigung für das angegebene Objekt entfernt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Einzelheiten zur API finden Sie unter ModifyImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI aktualisiert.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Beispiel 2: In diesem Beispiel wird das AMI öffentlich gemacht (damit es beispielsweise von jedem verwendet AWS-Konto werden kann).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Beispiel 3: In diesem Beispiel wird das AMI privat gemacht (beispielsweise, damit nur Sie als Besitzer es verwenden können).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Beispiel 4: In diesem Beispiel wird dem angegebenen Benutzer die Startberechtigung erteilt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Beispiel 5: In diesem Beispiel wird die Startberechtigung für das angegebene Objekt entfernt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- Einzelheiten zur API finden Sie unter ModifyImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyInstanceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyInstanceAttribute` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie den Instance-Typ

Im folgenden Beispiel für `modify-instance-attribute` wird der Typ der angegebenen Instance geändert. Die Instance muss sich im Status `stopped` befinden.

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --instance-type "{\"Value\": \"m1.small\"}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So aktivieren Sie Enhanced Networking auf einer Instance

Im folgenden Beispiel für `modify-instance-attribute` wird Enhanced Networking für die angegebene Instance aktiviert. Die Instance muss sich im Status `stopped` befinden.

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --sriov-net-support simple
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: Um das DestCheck Quellattribut zu ändern

Im folgenden Beispiel für `modify-instance-attribute` wird das `sourceDestCheck`-Attribut für die angegebene Instance in `true` geändert. Die Instance muss sich in einer VPC befinden.

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-
dest-check "{\"Value\": true}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 4: Um das OnTermination Löschattribut des Root-Volumes zu ändern

Im folgenden `modify-instance-attribute` Beispiel wird das `deleteOnTermination` Attribut für das Root-Volume der angegebenen EBS-backed Amazon-Instance auf festgelegt`false`. Standardmäßig ist dieses Attribut für das Root-Volume auf `true` festgelegt.

Befehl:

```
aws ec2 modify-instance-attribute \
  --instance-id i-1234567890abcdef0 \
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\",\"Ebs\":
{\"DeleteOnTermination\":false}}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 5: So ändern Sie die Benutzerdaten, die einer Instance angehängt sind

Im folgenden `modify-instance-attribute` Beispiel wird der Inhalt der Datei `UserData.txt` als UserData für die angegebene Instance hinzugefügt.

Inhalt der Originaldatei `UserData.txt`:

```
#!/bin/bash
yum update -y
service httpd start
chkconfig httpd on
```

Der Inhalt der Datei muss base64-kodiert sein. Der erste Befehl konvertiert die Textdatei in base64 und speichert sie als neue Datei.

Linux/macOS Version des Befehls:

```
base64 UserData.txt > UserData.base64.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Windows-Version des Befehls:

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
 UserData.base64.txt
```

Ausgabe:

```
Input Length = 67
```

```
Output Length = 152
CertUtil: -encode command completed successfully.
```

Jetzt können Sie im folgenden CLI-Befehl auf diese Datei verweisen:

```
aws ec2 modify-instance-attribute \
    --instance-id=i-09b5a14dbca622e76 \
    --attribute userData --value file://UserData.base64.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Benutzerdaten und AWS CLI im EC2-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter ModifyInstanceAttribute AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Instance-Typ der angegebenen Instance geändert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Beispiel 2: In diesem Beispiel wird Enhanced Networking für die angegebene Instanz aktiviert, indem „simple“ als Wert für den Netzwerkunterstützungsparameter Single Root I/O Virtualization (SR-IOV) angegeben wird, -SriovNetSupport..

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance geändert. Die Instance muss sich in einer VPC befinden. Sie müssen die ID jeder Sicherheitsgruppe angeben, nicht den Namen.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Beispiel 4: Dieses Beispiel aktiviert die I/O EBS-Optimierung für die angegebene Instance. Diese Feature ist nicht für alle Instance-Typen verfügbar. Bei der Nutzung einer EBS-optimized Instance fallen zusätzliche Nutzungsgebühren an.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Beispiel 5: In diesem Beispiel wird die source/destination Suche nach der angegebenen Instanz aktiviert. Damit eine NAT-Instance NAT ausführen kann, muss der Wert „False“ sein.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Beispiel 6: In diesem Beispiel wird die Beendigung für die angegebene Instance deaktiviert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Beispiel 7: In diesem Beispiel wird die angegebene Instance so geändert, dass sie beendet wird, wenn der Shutdown von der Instance aus initiiert wird.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -
InstanceInitiatedShutdownBehavior terminate
```

- Einzelheiten zur API finden Sie unter ModifyInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Instance-Typ der angegebenen Instance geändert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Beispiel 2: In diesem Beispiel wird Enhanced Networking für die angegebene Instanz aktiviert, indem „simple“ als Wert für den Netzwerkunterstützungsparameter Single Root I/O Virtualization (SR-IOV) angegeben wird, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance geändert. Die Instance muss sich in einer VPC befinden. Sie müssen die ID jeder Sicherheitsgruppe angeben, nicht den Namen.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Beispiel 4: Dieses Beispiel aktiviert die I/O EBS-Optimierung für die angegebene Instance. Diese Feature ist nicht für alle Instance-Typen verfügbar. Bei der Nutzung einer EBS-optimized Instance fallen zusätzliche Nutzungsgebühren an.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Beispiel 5: In diesem Beispiel wird die source/destination Suche nach der angegebenen Instanz aktiviert. Damit eine NAT-Instance NAT ausführen kann, muss der Wert „False“ sein.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Beispiel 6: In diesem Beispiel wird die Beendigung für die angegebene Instance deaktiviert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Beispiel 7: In diesem Beispiel wird die angegebene Instance so geändert, dass sie beendet wird, wenn der Shutdown von der Instance aus initiiert wird.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -
InstanceInitiatedShutdownBehavior terminate
```

- Einzelheiten zur API finden Sie unter ModifyInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyInstanceCreditSpecification`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyInstanceCreditSpecification` verwendet wird.

CLI

AWS CLI

So ändern Sie die Guthabenoption für die CPU-Nutzung einer Instance

In diesem Beispiel wird die Guthabenoption für die CPU-Nutzung der angegebenen Instance in der festgelegten Region in „unbegrenzt“ geändert. Gültige Guthabenoptionen sind „Standard“ und „unbegrenzt“.

Befehl:

```
aws ec2 modify-instance-credit-specification --instance-credit-
specification "InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

Ausgabe:

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- Einzelheiten zur API finden Sie ModifyInstanceCreditSpecificationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dadurch werden unbegrenzte T2-Gutschriften für die Instance i-01234567890abcdef aktiviert.

```
$Credit = New-Object -TypeName
 Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Einzelheiten zur API finden Sie unter ModifyInstanceCreditSpecification AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dadurch werden unbegrenzte T2-Gutschriften für die Instance i-01234567890abcdef aktiviert.

```
$Credit = New-Object -TypeName
 Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
```

```
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- Einzelheiten zur API finden Sie unter ModifyInstanceCreditSpecification AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyNetworkInterfaceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyNetworkInterfaceAttribute` verwendet wird.

CLI

AWS CLI

So ändern Sie das attachment-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `attachment`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

So ändern Sie das description-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `description`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
description "My description"
```

So ändern Sie das groupSet-Attribut einer Netzwerkschnittstelle

Dieser Beispielbefehl beschreibt das `groupSet`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
groups sg-903004f8 sg-1a2b3c4d
```

Um das DestCheck Quellattribut einer Netzwerkschnittstelle zu ändern

Dieser Beispielbefehl beschreibt das `sourceDestCheck`-Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
no-source-dest-check
```

- Einzelheiten zur API finden Sie ModifyNetworkInterfaceAttributeunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle so geändert, dass die spezifizierte Anlage beim Beenden gelöscht wird.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Beispiel 2: In diesem Beispiel wird die Beschreibung der angegebenen Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description
 "my description"
```

Beispiel 3: In diesem Beispiel wird die Sicherheitsgruppe für die angegebene Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Beispiel 4: In diesem Beispiel wird die source/destination Überprüfung der angegebenen Netzwerkschnittstelle deaktiviert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck $false
```

- Einzelheiten zur API finden Sie unter ModifyNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle so geändert, dass die spezifizierte Anlage beim Beenden gelöscht wird.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Beispiel 2: In diesem Beispiel wird die Beschreibung der angegebenen Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description
 "my description"
```

Beispiel 3: In diesem Beispiel wird die Sicherheitsgruppe für die angegebene Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Beispiel 4: In diesem Beispiel wird die source/destination Überprüfung der angegebenen Netzwerkschnittstelle deaktiviert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck $false
```

- Einzelheiten zur API finden Sie unter ModifyNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyReservedInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyReservedInstances` verwendet wird.

CLI

AWS CLI

So ändern Sie Reserved Instances

Dieser Beispielbefehl verschiebt eine Reserved Instance in eine andere Availability Zone in derselben Region.

Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=10
```

Ausgabe:

```
{
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"
}
```

So ändern Sie die Netzwerkplattform von Reserved Instances

Dieser Beispielbefehl konvertiert EC2-Classic Reserved Instances in EC2-VPC.

Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-VPC,InstanceCount=5
```

Ausgabe:

```
{
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"
}
```

Weitere Informationen finden Sie unter „Ändern Ihrer Reserved Instances“ im Benutzerhandbuch für Amazon EC2.

So ändern Sie die Instance-Größe von Reserved Instances

Mit diesem Beispielbefehl wird eine Reserved Instance mit 10 Linux/UNIX m1.small-Instances in us-west-1c so geändert, dass aus 8 m1.small-Instances 2 m1.large-Instances werden und die verbleibenden 2 m1.small zu einer m1.medium-Instance in derselben Availability Zone werden. Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-
ids 1ba8e2e3-3556-4264-949e-63ee671405a9 --target-
configurations AvailabilityZone=us-west-1c,Platform=EC2-
Classic,InstanceCount=2,InstanceType=m1.large AvailabilityZone=us-
west-1c,Platform=EC2-Classic,InstanceCount=1,InstanceType=m1.medium
```

Ausgabe:

```
{
    "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-
b3e3-1c6b11fa00b6"
}
```

Weitere Informationen finden Sie unter „Ändern der Instance-Größe Ihrer Reservierungen“ im Benutzerhandbuch für Amazon EC2.

- Einzelheiten ModifyReservedInstances AWS CLIzur API finden Sie in der Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Availability Zone, die Anzahl der Instances und die Plattform für die angegebenen Reserved Instances geändert.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE",
 "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
```

```
-TargetConfiguration $config
```

- Einzelheiten zur API finden Sie unter ModifyReservedInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zone, die Anzahl der Instances und die Plattform für die angegebenen Reserved Instances geändert.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE",
 "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- Einzelheiten zur API finden Sie unter ModifyReservedInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifySnapshotAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifySnapshotAttribute` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie ein snapshot-Attribut

Im folgenden Beispiel für `modify-snapshot-attribute` wird das `createVolumePermission`-Attribut für den angegebenen Snapshot aktualisiert und die Volume-Berechtigungen für den angegebenen Benutzer entfernt.

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type remove \
    --user-ids 123456789012
```

Beispiel 2: So machen Sie einen Snapshot öffentlich zugänglich

Im folgenden Beispiel für `modify-snapshot-attribute` wird der angegebene Snapshot öffentlich zugänglich gemacht.

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type add \
    --group-names all
```

- Einzelheiten zur API finden Sie ModifySnapshotAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot veröffentlicht, indem es sein CreateVolumePermission Attribut festlegt.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Einzelheiten zur API finden Sie unter ModifySnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot öffentlich gemacht, indem es sein CreateVolumePermission Attribut festlegt.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- Einzelheiten zur API finden Sie unter ModifySnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifySpotFleetRequest`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifySpotFleetRequest` verwendet wird.

CLI

AWS CLI

So ändern Sie eine Spot-Flotten-Anforderung

Dieser Beispielbefehl aktualisiert die Zielkapazität der angegebenen Spot-Flottenanforderung.

Befehl:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-
id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
    "Return": true
}
```

Dieser Beispielbefehl verringert die Zielkapazität der angegebenen Spot-Flottenanforderung, ohne dadurch Spot Instances zu beenden.

Befehl:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-
termination-policy NoTermination --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
    "Return": true
}
```

- Einzelheiten zur API finden Sie ModifySpotFleetRequestin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Zielkapazität der angegebenen Spot-Flottenanforderung aktualisiert.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter ModifySpotFleetRequest AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Zielkapazität der angegebenen Spot-Flottenanforderung aktualisiert.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Ausgabe:

```
True
```

- Einzelheiten zur API finden Sie unter ModifySpotFleetRequest AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifySubnetAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifySubnetAttribute` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk

CLI

AWS CLI

So ändern Sie das öffentliche IPv4-Adressierungsverhalten eines Subnetzes

In diesem Beispiel wird subnet-1a2b3c4d dahingehend geändert, dass allen in diesem Subnetz gestarteten Instances eine öffentliche IPv4-Adresse zugewiesen wird. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-
launch
```

So ändern Sie das IPv6-Adressierungsverhalten eines Subnetzes

In diesem Beispiel wird subnet-1a2b3c4d dahingehend geändert, dass allen in diesem Subnetz gestarteten Instances eine IPv6-Adresse aus dem Subnetzbereich zugewiesen wird.

Befehl:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-
address-on-creation
```

Weitere Informationen finden Sie unter „IP-Addressierung in Ihrer VPC“ im Benutzerhandbuch für AWS Virtual Private Cloud.

- Einzelheiten zur API finden Sie ModifySubnetAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz aktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Beispiel 2: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz deaktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Einzelheiten zur API finden Sie unter ModifySubnetAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz aktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Beispiel 2: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz deaktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- Einzelheiten zur API finden Sie unter ModifySubnetAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyVolumeAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyVolumeAttribute` verwendet wird.

CLI

AWS CLI

So ändern Sie ein Volume-Attribut

In diesem Beispiel wird das `autoEnableIo`-Attribut des Volumes mit der ID `vol-1234567890abcdef0` auf `true` festgelegt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-
io
```

- Einzelheiten zur API finden Sie ModifyVolumeAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Volumes geändert. I/O Operationen für das Volume werden automatisch wieder aufgenommen, nachdem sie aufgrund potenziell inkonsistenter Daten unterbrochen wurden.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Einzelheiten zur API finden Sie unter ModifyVolumeAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Volumes geändert. I/O Operationen für das Volume werden automatisch wieder aufgenommen, nachdem sie aufgrund potenziell inkonsistenter Daten unterbrochen wurden.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- Einzelheiten zur API finden Sie unter ModifyVolumeAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyVpcAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyVpcAttribute` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit Graphdatenbanken

CLI

AWS CLI

Um das DnsSupport Enable-Attribut zu ändern

In diesem Beispiel wird das `enableDnsSupport`-Attribut geändert. Dieses Attribut gibt an, ob für die VPC die DNS-Auflösung aktiviert ist. Wenn dieses Attribut `true` ist, löst der Amazon-DNS-Server die DNS-Hostnamen der Instances in die entsprechenden IP-Adressen auf. Andernfalls geschieht das nicht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value
\":false}"
```

Um das DnsHostnames Enable-Attribut zu ändern

In diesem Beispiel wird das `enableDnsHostnames`-Attribut geändert. Dieses Attribut gibt an, ob in der VPC gestartete Instances DNS-Hostnamen erhalten. Wenn dieses Attribut `true` ist, erhalten die Instances in der VPC DNS-Hostnamen. Andernfalls ist das nicht der Fall. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames
 "{\"Value\":false}"
```

- Einzelheiten zur API finden Sie ModifyVpcAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wirdt die Unterstützung von DNS-Hostnamen für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Beispiel 2: In diesem Beispiel wird die Unterstützung für DNS-Hostnamen für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Beispiel 3: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Beispiel 4: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Einzelheiten zur API finden Sie unter ModifyVpcAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wirdt die Unterstützung von DNS-Hostnamen für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Beispiel 2: In diesem Beispiel wird die Unterstützung für DNS-Hostnamen für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Beispiel 3: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Beispiel 4: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- Einzelheiten zur API finden Sie unter ModifyVpcAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`MonitorInstances`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `MonitorInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
 ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to enable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
```

```
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
                != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
 ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
                  instanceId << ": " <<
                  monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
                  instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie MonitorInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktivieren Sie eine detaillierte Überwachung für eine Instance

Dieser Beispielbefehl aktiviert die detaillierte Überwachung für die angegebene Instance.

Befehl:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "InstanceMonitorings": [
```

```
        {
            "InstanceId": "i-1234567890abcdef0",
            "Monitoring": {
                "State": "pending"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie MonitorInstancesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, MonitorInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every
 minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
```

```
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie MonitorInstancesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance aktiviert.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter MonitorInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance aktiviert.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter MonitorInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 monitor the instance without actually making the request. "
        lo_ec2->monitorinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to monitor this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
          " DryRun is set to false to enable detailed monitoring. "
          lo_ec2->monitorinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
```

```
          MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to monitor this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to enable detailed monitoring failed. User does not
 have the permissions to monitor the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Einzelheiten zur API finden Sie MonitorInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`MoveAddressToVpc`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `MoveAddressToVpc` verwendet wird.

CLI

AWS CLI

Um eine Adresse zu verschieben EC2-VPC

In diesem Beispiel wird die Elastic IP-Adresse 54.123.4.56 auf die Plattform verschoben. EC2-VPC

Befehl:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

Ausgabe:

```
{
  "Status": "MoveInProgress"
}
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz MoveAddressToVpc.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine EC2-Instance mit der öffentlichen IP-Adresse 12.345.67.89 auf die EC2-VPC Plattform in der Region USA Ost (Nord-Virginia) verschoben.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Beispiel 2: In diesem Beispiel werden die Ergebnisse eines Befehls über die Pipeline an das Cmdlet übergeben. Get-EC2Instance Move-EC2AddressToVpc Der Get-EC2Instance Befehl ruft eine Instanz ab, die durch die Instanz-ID angegeben ist, und gibt dann die öffentliche IP-Adresseigenschaft der Instanz zurück.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Einzelheiten zur API finden Sie unter MoveAddressToVpc AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine EC2-Instance mit der öffentlichen IP-Adresse 12.345.67.89 auf die EC2-VPC Plattform in der Region USA Ost (Nord-Virginia) verschoben.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Beispiel 2: In diesem Beispiel werden die Ergebnisse eines Befehls über die Pipeline an das Cmdlet übergeben. Get-EC2Instance Move-EC2AddressToVpc Der Get-EC2Instance Befehl ruft eine Instanz ab, die durch die Instanz-ID angegeben ist, und gibt dann die öffentliche IP-Adresseigenschaft der Instanz zurück.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- Einzelheiten zur API finden Sie unter MoveAddressToVpc AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PurchaseHostReservation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PurchaseHostReservation` verwendet wird.

CLI

AWS CLI

So erwerben Sie eine Dedicated-Host-Reservierung

In diesem Beispiel wird die angebotene Dedicated-Host-Reservierung für den festgelegten Dedicated Host in Ihrem Konto erworben.

Befehl:

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set h-013abcd2a00cbd123
```

Ausgabe:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
      {
          "HourlyPrice": "1.499",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "HostIdSet": [
              "h-013abcd2a00cbd123"
          ],
          "HostReservationId": "hr-0d418a3a4ffc669ae",
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- Einzelheiten zur API finden Sie PurchaseHostReservationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Reservierung mit der Nummer hro-0c1f23456789d0ab erworben, deren Konfigurationen mit denen Ihres Dedicated Hosts h-01e23f4cd567890f1 übereinstimmen

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Einzelheiten zur API finden Sie unter PurchaseHostReservation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Reservierung mit der Nummer hro-0c1f23456789d0ab erworben, deren Konfigurationen mit denen Ihres Dedicated Hosts h-01e23f4cd567890f1 übereinstimmen

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- Einzelheiten zur API finden Sie unter PurchaseHostReservation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PurchaseScheduledInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PurchaseScheduledInstances` verwendet wird.

CLI

AWS CLI

Um eine geplante Instanz zu kaufen

In diesem Beispiel wird eine geplante Instance gekauft.

Befehl:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-
request.json
```

Purchase-request.json:

```
[
    {
        "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
        "InstanceCount": 1
    }
]
```

Ausgabe:

```
{
  "ScheduledInstanceSet": [
      {
          "AvailabilityZone": "us-west-2b",
          "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
          "HourlyPrice": "0.095",
          "CreateDate": "2016-01-25T21:43:38.612Z",
          "Recurrence": {
              "OccurrenceDaySet": [
                  1
              ],
              "Interval": 1,
              "Frequency": "Weekly",
              "OccurrenceRelativeToEnd": false,
```

```
                "OccurrenceUnit": ""
            },
            "Platform": "Linux/UNIX",
            "TermEndDate": "2017-01-31T09:00:00Z",
            "InstanceCount": 1,
            "SlotDurationInHours": 32,
            "TermStartDate": "2016-01-31T09:00:00Z",
            "NetworkPlatform": "EC2-VPC",
            "TotalScheduledInstanceHours": 1696,
            "NextSlotStartTime": "2016-01-31T09:00:00Z",
            "InstanceType": "c4.large"
        }
    ]
}
```

- Einzelheiten zur API finden Sie PurchaseScheduledInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine geplante Instance gekauft.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
```

```
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Einzelheiten zur API finden Sie unter PurchaseScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine geplante Instance gekauft.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- Einzelheiten zur API finden Sie unter PurchaseScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`RebootInstances`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RebootInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie eine Instance anhand ihrer ID neu.

```
    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance
{ec2InstanceId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);
```

```
        return hasState;
    }
```

Ersetzen Sie das Profil für eine Instance und starten Sie einen Webserver neu.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
```

```
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
```

```
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
 template.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie RebootInstancesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
 ec2Client.RebootInstances(request);
```

```
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to reboot on instance. A dry run should
 trigger an error."
                <<
                std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
 ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie RebootInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So starten Sie eine Amazon-EC2-Instance neu

In diesem Beispiel wird die angegebene Instance neu gestartet. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

Weitere Informationen finden Sie unter Ihre Instance neu starten im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie RebootInstancesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, RebootInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Please provide the InstanceId of a valid instance to
 reboot.`,
```

```
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie RebootInstancesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Instance neu gestartet.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Einzelheiten zur API finden Sie unter RebootInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance neu gestartet.

```
Restart-EC2Instance -InstanceId i-12345678
```

- Einzelheiten zur API finden Sie unter RebootInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
```

```
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie RebootInstancesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }
```

```
    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }
```

Waiters: Warten, bis sich die Instance in den Status „Angehalten“ und „Bereit“ befindet, mithilfe der Waiters-API. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
```

```
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Einzelheiten zur API finden Sie RebootInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 reboot the instance without actually making the request. "
        lo_ec2->rebootinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to reboot this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
          " DryRun is set to false to make a reboot request. "
          lo_ec2->rebootinstances(
             it_instanceids = lt_instance_ids
             iv_dryrun = abap_false ).
          MESSAGE 'Instance rebooted.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to reboot this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to reboot instance failed. User does not have
 permissions to reboot the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie RebootInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`RegisterImage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterImage` verwendet wird.

CLI

AWS CLI

Beispiel 1: So registrieren Sie ein AMI mithilfe einer Manifestdatei

Im folgenden Beispiel für `register-image` wird ein AMI mithilfe der angegebenen Manifestdatei in Amazon S3 registriert.

```
aws ec2 register-image \
    --name my-image \
    --image-location amzn-s3-demo-bucket/myimage/image.manifest.xml
```

Ausgabe:

```
{
    "ImageId": "ami-1234567890EXAMPLE"
}
```

Weitere Informationen dazu finden Sie unter Amazon Machine Images (AMI) im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So registrieren Sie ein AMI mithilfe eines Snapshots eines Root-Geräts

Im folgenden Beispiel für `register-image` wird ein AMI mit dem angegebenen Snapshot eines EBS-Stamm-Volumes als Gerät `/dev/xvda` registriert. Die Blockgerät-Zuweisung umfasst auch ein leeres EBS-Volume von 100 GiB als Gerät `/dev/xvdf`.

```
aws ec2 register-image \
```

```
    --name my-image \
    --root-device-name /dev/xvda \
    --block-device-mappings DeviceName=/dev/
xvda,Ebs={SnapshotId=snap-0db2cf683925d191f} DeviceName=/dev/
xvdf,Ebs={VolumeSize=100}
```

Ausgabe:

```
{
    "ImageId": "ami-1a2b3c4d5eEXAMPLE"
}
```

Weitere Informationen dazu finden Sie unter Amazon Machine Images (AMI) im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie RegisterImagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein AMI mithilfe der angegebenen Manifestdatei in Amazon S3 registriert.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Einzelheiten zur API finden Sie unter RegisterImage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein AMI mithilfe der angegebenen Manifestdatei in Amazon S3 registriert.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- Einzelheiten zur API finden Sie unter RegisterImage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RejectVpcPeeringConnection`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RejectVpcPeeringConnection` verwendet wird.

CLI

AWS CLI

So lehnen Sie eine VPC-Peering-Verbindung ab

In diesem Beispiel wird die angegebene VPC-Peering-Verbindungsanforderung abgelehnt.

Befehl:

```
aws ec2 reject-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

Ausgabe:

```
{
    "Return": true
}
```

- Einzelheiten zur API finden Sie RejectVpcPeeringConnectionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Das obige Beispiel lehnt die Anfrage nach der Anforderungs-ID VpcPeering pcx-01a2b3ce45fe67eb8 ab

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). RejectVpcPeeringConnectionAWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Das obige Beispiel lehnt die Anfrage nach der Anforderungs-ID VpcPeering pcx-01a2b3ce45fe67eb8 ab

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). RejectVpcPeeringConnectionAWS -Tools für PowerShell

Verwenden Sie mit einem **`ReleaseAddress`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ReleaseAddress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Erste Schritte mit virtuellen Maschinen
- Virtuelles privates Netzwerk mit privaten Servern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
```

```
            var request = new ReleaseAddressRequest { AllocationId =
 allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
 {ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
 {allocationId}.: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_release_address
#
```

```
# This function releases an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to
 release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
```

```
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
```

```
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
```

```
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                  allocationID << ":" << outcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                  allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um Elastic IP-Adressen zu veröffentlichen für EC2-Classic

In diesem Beispiel wird eine Elastic IP-Adresse zur Verwendung mit Instances veröffentlicht EC2-Classic. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 release-address --public-ip 198.51.100.0
```

Um eine Elastic IP-Adresse freizugeben für EC2-VPC

In diesem Beispiel wird eine Elastic-IP-Adresse zur Verwendung mit Instances in einer VPC freigegeben. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous
operation of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse>
releaseEC2AddressAsync(String allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();

       CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to release Elastic IP
address", ex);
           }
       });

       return response;
   }
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ReleaseAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AllocationId: allocationId,
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- Einzelheiten zur API finden Sie ReleaseAddressin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse für Instances in einer VPC freigegeben.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in veröffentlicht EC2-Classic.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Einzelheiten zur API finden Sie unter ReleaseAddress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse für Instances in einer VPC freigegeben.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in EC2-Classic veröffentlicht.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- Einzelheiten zur API finden Sie unter ReleaseAddress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
 = None
        ) -> None:
            """
            Initializes the ElasticIp object.
```

```
            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
 address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
 allocation_id)
        if elastic_ip is None:
```

```
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise
```

- Einzelheiten zur API finden Sie ReleaseAddressin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
```

```
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
 EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }
```

- Einzelheiten zur API finden Sie ReleaseAddressin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
        MESSAGE 'Elastic IP address released.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ReleaseAddressin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2
```

```
    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
\(error.localizedDescription)")
        }
    }
```

- Einzelheiten zur API finden Sie ReleaseAddressin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ReleaseHosts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReleaseHosts` verwendet wird.

CLI

AWS CLI

So geben Sie einen Dedicated Host von Ihrem Konto frei

So geben Sie einen Dedicated Host von Ihrem Konto frei Instances, die sich auf dem Host befinden, müssen angehalten oder beendet werden, bevor der Host freigegeben werden kann.

Befehl:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

Ausgabe:

```
{
```

```
    "Successful":  [
        "h-0029d6e3cacf1b3da"
         ],
  "Unsuccessful": []

 }
```

- Einzelheiten zur API finden Sie ReleaseHostsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Host-ID h-0badafd1dcb2f3456 freigegeben

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- Einzelheiten zur API finden Sie unter ReleaseHosts AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Host-ID h-0badafd1dcb2f3456 freigegeben

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- Einzelheiten zur API finden Sie unter ReleaseHosts AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ReplaceIamInstanceProfileAssociation`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ReplaceIamInstanceProfileAssociation` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
 is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
 ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
```

```
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
```

```
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ReplaceIamInstanceProfileAssociationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So ersetzen Sie ein IAM-Instance-Profil für eine Instance

Dieses Beispiel ersetzt das IAM-Instance-Profil, das durch die Verknüpfung `iip-assoc-060bae234aac2e7fa` mit dem genannten IAM-Instance-Profil `AdminRole` dargestellt wird.

```
aws ec2 replace-iam-instance-profile-association \
    --iam-instance-profile Name=AdminRole \
    --association-id iip-assoc-060bae234aac2e7fa
```

Ausgabe:

```
{
    "IamInstanceProfileAssociation": {
        "InstanceId": "i-087711ddaf98f9489",
        "State": "associating",
        "AssociationId": "iip-assoc-0b215292fab192820",
        "IamInstanceProfile": {
            "Id": "AIPAJLNLDX3AMYZNWYYAY",
            "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"
        }
    }
}
```

- Einzelheiten zur API finden Sie ReplaceIamInstanceProfileAssociationin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );
```

- Einzelheiten zur API finden Sie ReplaceIamInstanceProfileAssociationin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel ersetzt das Instance-Profil einer laufenden Instance, startet die Instance neu und sendet nach dem Start einen Befehl an die Instance.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
```

```
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
```

```
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
 is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie ReplaceIamInstanceProfileAssociationin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`ReplaceNetworkAclAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReplaceNetworkAclAssociation` verwendet wird.

CLI

AWS CLI

So ersetzen Sie die einem Subnetz zugeordnete Netzwerk-ACL

In diesem Beispiel wird die angegebene Netzwerk-ACL mit dem Subnetz für die angegebene Netzwerk-ACL-Zuordnung verknüpft.

Befehl:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --
network-acl-id acl-5fb85d36
```

Ausgabe:

```
{
    "NewAssociationId": "aclassoc-3999875b"
}
```

- Einzelheiten zur API finden Sie ReplaceNetworkAclAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL dem Subnetz für die angegebene Netzwerk-ACL-Zuordnung zugeordnet.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Ausgabe:

```
aclassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL dem Subnetz für die angegebene Netzwerk-ACL-Zuordnung zugeordnet.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Ausgabe:

```
aclassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ReplaceNetworkAclEntry`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReplaceNetworkAclEntry` verwendet wird.

CLI

AWS CLI

So ersetzen Sie einen Netzwerk-ACL-Eintrag

In diesem Beispiel wird ein Eintrag in der angegebenen Netzwerk-ACL ersetzt. Die neue Regel 100 erlaubt eingehenden Datenverkehr ab 203.0.113. 12/24 auf UDP-Port 53 (DNS) in ein beliebiges zugeordnetes Subnetz.

Befehl:

```
aws ec2 replace-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100 --protocol udp --port-range From=53,To=53 --cidr-block 203.0.113.12/24 --rule-action allow
```

- Einzelheiten zur API finden Sie ReplaceNetworkAclEntryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Eintrag für die spezifizierte Netzwerk-ACL ersetzt. Die neue Regel erlaubt eingehenden Verkehr von der angegebenen Adresse zu jedem zugehörigen Subnetz.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -RuleAction allow
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Eintrag für die spezifizierte Netzwerk-ACL ersetzt. Die neue Regel erlaubt eingehenden Verkehr von der angegebenen Adresse zu jedem zugehörigen Subnetz.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -
RuleAction allow
```

- Einzelheiten zur API finden Sie unter ReplaceNetworkAclEntry AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ReplaceRoute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReplaceRoute` verwendet wird.

CLI

AWS CLI

So ersetzen Sie eine Route

In diesem Beispiel wird die angegebene Route in der festgelegten Routing-Tabelle ersetzt. Die neue Route entspricht dem angegebenen CIDR und sendet den Datenverkehr an das angegebene Virtual Private Gateway. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-
block 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- Einzelheiten zur API finden Sie ReplaceRoutein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle ersetzt. Die neue Route leitet den angegebenen Datenverkehr an das spezifizierte Virtual Private Gateway.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -
GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter ReplaceRoute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle ersetzt. Die neue Route leitet den angegebenen Datenverkehr an das spezifizierte Virtual Private Gateway.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -
GatewayId vgw-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter ReplaceRoute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ReplaceRouteTableAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReplaceRouteTableAssociation` verwendet wird.

CLI

AWS CLI

So ersetzen Sie die Routing-Tabelle, die einem Subnetz zugeordnet ist

In diesem Beispiel wird die angegebene Routing-Tabelle dem festgelegten Subnetz für die angegebene Routing-Tabelle zugeordnet.

Befehl:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --
route-table-id rtb-22574640
```

Ausgabe:

```
{
    "NewAssociationId": "rtbassoc-3a1f0f58"
}
```

- Einzelheiten zur API finden Sie ReplaceRouteTableAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routing-Tabelle zugeordnet.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Ausgabe:

```
rtbassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceRouteTableAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routing-Tabelle zugeordnet.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Ausgabe:

```
rtbassoc-87654321
```

- Einzelheiten zur API finden Sie unter ReplaceRouteTableAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ReportInstanceStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ReportInstanceStatus` verwendet wird.

CLI

AWS CLI

So geben Sie Feedback zum Status einer Instance

Dieser Beispielbefehl gibt Feedback zum Status der angegebenen Instance.

Befehl:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired
 --reason-codes unresponsive
```

- Einzelheiten zur API finden Sie ReportInstanceStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird Feedback zum Status der angegebenen Instance gegeben.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Einzelheiten zur API finden Sie unter ReportInstanceStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird Feedback zum Status der angegebenen Instance gegeben.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- Einzelheiten zur API finden Sie unter ReportInstanceStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RequestSpotFleet`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RequestSpotFleet` verwendet wird.

CLI

AWS CLI

So fordern Sie eine Spot-Flotte im Subnetz mit dem niedrigsten Preis an

Dieser Beispielbefehl erstellt eine Spot-Flottenanforderung mit zwei Startspezifikationen, die sich nur im Hinblick auf das Subnetz unterscheiden. Die Spot-Flotte startet die Instances im angegebenen Subnetz mit dem niedrigsten Preis. Wenn die Instances in einer Standard-VPC gestartet werden, erhalten sie standardmäßig eine öffentliche IP-Adresse. Wenn die Instances in einer nicht standardmäßigen VPC gestartet werden, erhalten sie keine öffentliche Adresse.

Beachten Sie, dass Sie in einer Spot-Flotten-Anforderung nicht verschiedene Subnetze in derselben Availability Zone angeben können.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "SecurityGroups": [
              {
                  "GroupId": "sg-1a2b3c4d"
              }
          ],
          "InstanceType": "m3.medium",
          "SubnetId": "subnet-1a2b3c4d, subnet-3c4d5e6f",
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
          }
      }
  ]
}
```

Ausgabe:

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
```

```
}
```

So fordern Sie eine Spot-Flotte in der Availability Zone mit dem niedrigsten Preis an

Dieser Beispielbefehl erstellt eine Spot-Flottenanforderung mit zwei Startspezifikationen, die sich nur im Hinblick auf die Availability Zone unterscheiden. Die Spot-Flotte startet die Instances in der angegebenen Availability Zone mit dem niedrigsten Preis. Wenn Ihr Konto EC2-VPC nur unterstützt, startet Amazon EC2 die Spot-Instances im Standardsubnetz der Availability Zone. Wenn Ihr Konto dies unterstützt EC2-Classic, startet Amazon EC2 die Instances EC2-Classic in der Availability Zone.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "SecurityGroups": [
              {
                  "GroupId": "sg-1a2b3c4d"
              }
          ],
          "InstanceType": "m3.medium",
          "Placement": {
              "AvailabilityZone": "us-west-2a, us-west-2b"
          },
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
          }
      }
  ]
}
```

So starten Sie Spot Instances in einem Subnetz und weisen ihnen öffentliche IP-Adressen zu

Dieser Beispielbefehl weist Instances, die in einer nicht standardmäßigen VPC gestartet werden, öffentliche Adressen zu. Beachten Sie, dass Sie bei Angabe einer Netzwerkschnittstelle die Subnetz-ID und die Sicherheitsgruppen-ID mithilfe der Netzwerkschnittstelle angeben müssen.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "KeyName": "my-key-pair",
          "InstanceType": "m3.medium",
          "NetworkInterfaces": [
              {
                  "DeviceIndex": 0,
                  "SubnetId": "subnet-1a2b3c4d",
                  "Groups": [ "sg-1a2b3c4d" ],
                  "AssociatePublicIpAddress": true
              }
          ],
          "IamInstanceProfile": {
              "Arn": "arn:aws:iam::880185128111:instance-profile/my-iam-role"
          }
      }
  ]
}
```

So fordern eine Spot-Flotte anhand der diversifizierten Zuweisungsstrategie an

Dieser Beispielbefehl erstellt eine Spot-Flottenanforderung, die 30 Instances unter Verwendung der diversifizierten Zuweisungsstrategie startet. Die Startspezifikationen unterscheiden sich je nach Instance-Typ. Die Spot-Flotte verteilt die Instances auf die Startspezifikationen, sodass 10 Instances von jedem Typ vorhanden sind.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

Config.json:

```
{
  "SpotPrice": "0.70",
  "TargetCapacity": 30,
  "AllocationStrategy": "diversified",
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "c4.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      },
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "m3.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      },
      {
          "ImageId": "ami-1a2b3c4d",
          "InstanceType": "r3.2xlarge",
          "SubnetId": "subnet-1a2b3c4d"
      }
  ]
}
```

Weitere Informationen finden Sie unter „Spot-Flottenanforderungen“ im Benutzerhandbuch für Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie RequestSpotFleetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Spot-Flottenanforderung in der Availability Zone mit dem niedrigsten Preis für den angegebenen Instance-Typ erstellt. Wenn Ihr Konto EC2-VPC nur unterstützt, startet die Spot-Flotte die Instances in der Availability Zone mit dem niedrigsten

Preis, die über ein Standardsubnetz verfügt. Wenn Ihr Konto dies unterstützt EC2-Classic, startet die Spot-Flotte die Instances EC2-Classic in der Availability Zone mit dem niedrigsten Preis. Beachten Sie, dass der von Ihnen gezahlte Preis den angegebenen Spot-Preis für die Anforderung nicht überschreitet.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Einzelheiten zur API finden Sie unter RequestSpotFleet AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Spot-Flottenanforderung in der Availability Zone mit dem niedrigsten Preis für den angegebenen Instance-Typ erstellt. Wenn Ihr Konto EC2-VPC nur unterstützt, startet die Spot-Flotte die Instances in der Availability Zone mit dem niedrigsten Preis, die über ein Standardsubnetz verfügt. Wenn Ihr Konto dies unterstützt EC2-Classic, startet die Spot-Flotte die Instances EC2-Classic in der Availability Zone mit dem niedrigsten Preis. Beachten Sie, dass der von Ihnen gezahlte Preis den angegebenen Spot-Preis für die Anforderung nicht überschreitet.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- Einzelheiten zur API finden Sie unter RequestSpotFleet AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RequestSpotInstances`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RequestSpotInstances` verwendet wird.

CLI

AWS CLI

So fordern Sie Spot Instances an

Dieser Beispielbefehl erstellt eine einmalige Spot-Instance-Anforderung für 5 Instances in der angegebenen Availability Zone. Wenn Ihr Konto EC2-VPC nur unterstützt, startet Amazon EC2 die Instances im Standardsubnetz der angegebenen Availability Zone. Wenn Ihr Konto dies unterstützt EC2-Classic, startet Amazon EC2 die Instances EC2-Classic in der angegebenen Availability Zone.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "Placement": {
    "AvailabilityZone": "us-west-2a"
  },
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

Ausgabe:

```
{
  "SpotInstanceRequests": [
      {
          "Status": {
              "UpdateTime": "2014-03-25T20:54:21.000Z",
              "Code": "pending-evaluation",
              "Message": "Your Spot request has been submitted for review, and is
 pending evaluation."
          },
          "ProductDescription": "Linux/UNIX",
          "SpotInstanceRequestId": "sir-df6f405d",
          "State": "open",
          "LaunchSpecification": {
              "Placement": {
                  "AvailabilityZone": "us-west-2a"
              },
              "ImageId": "ami-1a2b3c4d",
              "KeyName": "my-key-pair",
              "SecurityGroups": [
                  {
                      "GroupName": "my-security-group",
                      "GroupId": "sg-1a2b3c4d"
                  }
              ],
              "Monitoring": {
                  "Enabled": false
              },
              "IamInstanceProfile": {
                  "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
              },
              "InstanceType": "m3.medium"
          },
          "Type": "one-time",
          "CreateTime": "2014-03-25T20:54:20.000Z",
          "SpotPrice": "0.050000"
      },
      ...
  ]
}
```

Dieser Beispielbefehl erstellt eine einmalige Spot-Instance-Anforderung für 5 Instances im angegebenen Subnetz. Amazon EC2 startet die Instances im ausgewählten Subnetz. Wenn

es sich bei der VPC um eine nicht standardmäßige VPC handelt, erhalten die Instances standardmäßig keine öffentliche IP-Adresse.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

Ausgabe:

```
{
  "SpotInstanceRequests": [
      {
          "Status": {
             "UpdateTime": "2014-03-25T22:21:58.000Z",
             "Code": "pending-evaluation",
             "Message": "Your Spot request has been submitted for review, and is
 pending evaluation."
          },
          "ProductDescription": "Linux/UNIX",
          "SpotInstanceRequestId": "sir-df6f405d",
          "State": "open",
          "LaunchSpecification": {
             "Placement": {
                 "AvailabilityZone": "us-west-2a"
             }
             "ImageId": "ami-1a2b3c4d"
             "SecurityGroups": [
                 {
                     "GroupName": "my-security-group",
```

```
                    "GroupID": "sg-1a2b3c4d"
                }
            ]
            "SubnetId": "subnet-1a2b3c4d",
            "Monitoring": {
                "Enabled": false
            },
            "IamInstanceProfile": {
                "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
            },
            "InstanceType": "m3.medium",
        },
        "Type": "one-time",
        "CreateTime": "2014-03-25T22:21:58.000Z",
        "SpotPrice": "0.050000"
    },
    ...
  ]
}
```

In diesem Beispiel werden Spot Instances, die Sie in einer nicht standardmäßigen VPC starten, eine öffentliche IP-Adresse zugewiesen. Beachten Sie, dass Sie bei Angabe einer Netzwerkschnittstelle die Subnetz-ID und die Sicherheitsgruppen-ID mithilfe der Netzwerkschnittstelle angeben müssen.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "InstanceType": "m3.medium",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-1a2b3c4d",
      "Groups": [ "sg-1a2b3c4d" ],
      "AssociatePublicIpAddress": true
```

```
    }
  ],
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

- Einzelheiten zur API finden Sie RequestSpotInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine einmalige Spot Instance im angegebenen Subnetz angefordert. Beachten Sie, dass die Sicherheitsgruppe für die VPC erstellt werden muss, die das angegebene Subnetz enthält, und dass sie über die Netzwerkschnittstelle anhand der ID angegeben werden muss. Wenn Sie eine Netzwerkschnittstelle angeben, müssen Sie die Subnetz-ID unter Verwendung der Netzwerkschnittstelle angeben.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
```

```
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
State                    : open
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {}
Type                     : one-time
```

- Einzelheiten zur API finden Sie unter RequestSpotInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalige Spot Instance im angegebenen Subnetz angefordert. Beachten Sie, dass die Sicherheitsgruppe für die VPC erstellt werden muss, die das angegebene Subnetz enthält, und dass sie über die Netzwerkschnittstelle anhand der ID angegeben werden muss. Wenn Sie eine Netzwerkschnittstelle angeben, müssen Sie die Subnetz-ID unter Verwendung der Netzwerkschnittstelle angeben.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
```

```
State                     : open
Status                    : Amazon.EC2.Model.SpotInstanceStatus
Tags                      : {}
Type                      : one-time
```

- Einzelheiten zur API finden Sie unter RequestSpotInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResetImageAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResetImageAttribute` verwendet wird.

CLI

AWS CLI

So setzen Sie das launchPermission-Attribut zurück

In diesem Beispiel wird das `launchPermission`-Attribut für das angegebene AMI auf seinen Standardwert zurückgesetzt. Standardmäßig sind alle AMIs privat. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-image-attribute --image-id ami-5731123e --
attribute launchPermission
```

- Einzelheiten zur API finden Sie ResetImageAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Attribut „LaunchPermission“ auf seinen Standardwert zurückgesetzt. Standardmäßig sind alle AMIs privat.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Einzelheiten zur API finden Sie unter ResetImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut „LaunchPermission“ auf seinen Standardwert zurückgesetzt. Standardmäßig sind alle AMIs privat.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- Einzelheiten zur API finden Sie unter ResetImageAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResetInstanceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResetInstanceAttribute` verwendet wird.

CLI

AWS CLI

Um das DestCheck Quellattribut zurückzusetzen

In diesem Beispiel wird das `sourceDestCheck`-Attribut der angegebenen Instance zurückgesetzt. Die Instance muss sich in einer VPC befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute sourceDestCheck
```

So setzen Sie das kernel-Attribut zurück

In diesem Beispiel wird das `kernel`-Attribut der angegebenen Instance zurückgesetzt. Die Instance muss sich im Status `stopped` befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute kernel
```

So setzen Sie das ramdisk-Attribut zurück

In diesem Beispiel wird das `ramdisk`-Attribut der angegebenen Instance zurückgesetzt. Die Instance muss sich im Status `stopped` befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --
attribute ramdisk
```

- Einzelheiten zur API finden Sie ResetInstanceAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Attribut NetSupport 'sriov' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Beispiel 2: In diesem Beispiel wird das Attribut „ebsOptimized“ für die angegebene Instance zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Beispiel 3: In diesem Beispiel wird das DestCheck 'source'-Attribut für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Beispiel 4: In diesem Beispiel wird das ApiTermination 'Disable'-Attribut für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 disableApiTermination
```

Beispiel 5: In diesem Beispiel wird das Attribut InitiatedShutdownBehavior 'instance' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Einzelheiten zur API finden Sie unter ResetInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut NetSupport 'sriov' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Beispiel 2: In diesem Beispiel wird das Attribut „ebsOptimized“ für die angegebene Instance zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Beispiel 3: In diesem Beispiel wird das DestCheck 'source'-Attribut für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Beispiel 4: In diesem Beispiel wird das ApiTermination 'Disable'-Attribut für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 disableApiTermination
```

Beispiel 5: In diesem Beispiel wird das Attribut InitiatedShutdownBehavior 'instance' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- Einzelheiten zur API finden Sie unter ResetInstanceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResetNetworkInterfaceAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResetNetworkInterfaceAttribute` verwendet wird.

CLI

AWS CLI

So setzen Sie ein Netzwerkschnittstellenattribut zurück

Im folgenden `reset-network-interface-attribute` Beispiel wird der Wert des source/destination Prüfattributs auf `true` zurückgesetzt.

```
aws ec2 reset-network-interface-attribute \
    --network-interface-id eni-686ea200 \
    --source-dest-check sourceDestCheck
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie unter ResetNetworkInterfaceAttribute AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die source/destination Überprüfung auf die angegebene Netzwerkschnittstelle zurückgesetzt.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- Einzelheiten zur API finden Sie unter ResetNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die source/destination Überprüfung für die angegebene Netzwerkschnittstelle zurückgesetzt.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- Einzelheiten zur API finden Sie unter ResetNetworkInterfaceAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResetSnapshotAttribute`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResetSnapshotAttribute` verwendet wird.

CLI

AWS CLI

So setzen Sie ein snapshot-Attribut zurück

In diesem Beispiel werden die Berechtigungen zum Erstellen eines Volumes für den Snapshot `snap-1234567890abcdef0` zurückgesetzt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --
attribute createVolumePermission
```

- Einzelheiten zur API finden Sie ResetSnapshotAttributein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots zurückgesetzt.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Einzelheiten zur API finden Sie unter ResetSnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots zurückgesetzt.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- Einzelheiten zur API finden Sie unter ResetSnapshotAttribute AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RevokeSecurityGroupEgress`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RevokeSecurityGroupEgress` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwaltetem Streaming

CLI

AWS CLI

Beispiel 1: So entfernen Sie die Regel, die ausgehenden Datenverkehr in einem bestimmten Adressbereich zulässt

Mit dem folgenden `revoke-security-group-egress`-Beispielbefehl wird die Regel entfernt, die Zugriff auf die angegebenen Adressbereiche am TCP-Port 80 gewährt.

```
aws ec2 revoke-security-group-egress \
    --group-id sg-026c12253ce15eff7 \
    --ip-
permissions [{IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So entfernen Sie die Regel, die ausgehenden Datenverkehr an eine bestimmte Sicherheitsgruppe zulässt

Mit dem folgenden `revoke-security-group-egress`-Beispielbefehl wird die Regel entfernt, die Zugriff auf die angegebene Sicherheitsgruppe am TCP-Port 80 gewährt.

```
aws ec2 revoke-security-group-egress \
    --group-id sg-026c12253ce15eff7 \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":
 443,"UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie RevokeSecurityGroupEgressin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Regel für die angegebene Sicherheitsgruppe für entfernt EC2-VPC. Dadurch wird der Zugriff auf den angegebenen IP-Adressbereich am TCP-Port 80 aufgehoben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf die angegebene Quellsicherheitsgruppe auf TCP-Port 80 widerrufen.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
```

```
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupEgress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Regel für die angegebene Sicherheitsgruppe für EC2-VPC entfernt. Dadurch wird der Zugriff auf den angegebenen IP-Adressbereich am TCP-Port 80 aufgehoben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf die angegebene Quellsicherheitsgruppe auf TCP-Port 80 widerrufen.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupEgress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RevokeSecurityGroupIngress`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RevokeSecurityGroupIngress` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Dokumentendatenbanken
- Erste Schritte mit In-Memory-Caching
- Erste Schritte mit verwaltetem Streaming

CLI

AWS CLI

Beispiel 1: So entfernen Sie eine Regel aus einer Sicherheitsgruppe

Im folgenden Beispiel für `revoke-security-group-ingress` wird der Zugriff am TCP-Port 22 für den Adressbereich `203.0.113.0/24` aus der angegebenen Sicherheitsgruppe für eine Standard-VPC entfernt.

```
aws ec2 revoke-security-group-ingress \
    --group-name mySecurityGroup
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

Dieser Befehl erzeugt keine Ausgabe, wenn er erfolgreich ist.

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So entfernen Sie eine Regel mithilfe des IP-Berechtigungssatzes

Im folgenden Beispiel für `revoke-security-group-ingress` wird der `ip-permissions`-Parameter verwendet, um eine eingehende Regel zu entfernen, die die ICMP-Nachricht `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Typ 3, Code 4) zulässt.

```
aws ec2 revoke-security-group-ingress \
```

```
    --group-id sg-026c12253ce15eff7 \
    --ip-
permissions IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]
```

Dieser Befehl erzeugt keine Ausgabe, wenn er erfolgreich ist.

Weitere Informationen finden Sie unter Sicherheitsgruppen im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie RevokeSecurityGroupIngressin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Zugriff auf den TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-VPC gesperrt. Beachten Sie, dass Sie Sicherheitsgruppen anhand der Sicherheitsgruppen-ID und nicht EC2-VPC anhand des Sicherheitsgruppennamens identifizieren müssen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf den TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-Classic gesperrt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupIngress AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Zugriff auf TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für gesperrt. EC2-VPC Beachten Sie, dass Sie Sicherheitsgruppen anhand der Sicherheitsgruppen-ID und nicht EC2-VPC anhand des Sicherheitsgruppennamens identifizieren müssen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf den TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-Classic gesperrt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- Einzelheiten zur API finden Sie unter RevokeSecurityGroupIngress AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`RunInstances`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RunInstances` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Einen Containerdienst für VM-Instanzen erstellen
- Blockspeicher-Volumes erstellen und verwalten
- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen

- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;
```

```
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
```

```
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceName: A name for the EC2 instance.
  \param amiId: An Amazon Machine Image (AMI) identifier.
  \param[out] instanceID: String to return the instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
```

```
    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
            runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
 runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So starten Sie eine Instance in einem Standard-Subnetz

Das folgende `run-instances`-Beispiel startet eine einzelne Instance des Typs `t2.micro` im Standardsubnetz für die aktuelle Region und ordnet sie dem Standardsubnetz für die Standard-VPC für die Region zu. Das Schlüsselpaar ist optional, wenn Sie nicht vorhaben, über SSH (Linux) oder RDP (Windows) eine Verbindung zu Ihrer Instance herzustellen.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --key-name MyKeyPair
```

Ausgabe:

```
{
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0abcdef1234567890",
            "InstanceId": "i-1231231230abcdef0",
            "InstanceType": "t2.micro",
            "KeyName": "MyKeyPair",
            "LaunchTime": "2018-05-10T08:05:20.000Z",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-2a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
            "PrivateIpAddress": "10.0.0.157",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-04a636d18e83cfacb",
            "VpcId": "vpc-1234567890abcdef0",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "",
            "EbsOptimized": false,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
```

```
                        "AttachTime": "2018-05-10T08:05:20.000Z",
                        "AttachmentId": "eni-attach-0e325c07e928a0405",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching"
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupName": "MySecurityGroup",
                            "GroupId": "sg-0598c7d356eba48d7"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "0a:ab:58:e0:67:e2",
                    "NetworkInterfaceId": "eni-0c0a29997760baee7",
                    "OwnerId": "123456789012",
                    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
                    "PrivateIpAddress": "10.0.0.157",
                    "PrivateIpAddresses": [
                        {
                            "Primary": true,
                            "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
                            "PrivateIpAddress": "10.0.0.157"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Status": "in-use",
                    "SubnetId": "subnet-04a636d18e83cfacb",
                    "VpcId": "vpc-1234567890abcdef0",
                    "InterfaceType": "interface"
                }
            ],
            "RootDeviceName": "/dev/xvda",
            "RootDeviceType": "ebs",
            "SecurityGroups": [
                {
                    "GroupName": "MySecurityGroup",
                    "GroupId": "sg-0598c7d356eba48d7"
                }
            ],
            "SourceDestCheck": true,
            "StateReason": {
```

```
                "Code": "pending",
                "Message": "pending"
            },
            "Tags": [],
            "VirtualizationType": "hvm",
            "CpuOptions": {
                "CoreCount": 1,
                "ThreadsPerCore": 1
            },
            "CapacityReservationSpecification": {
                "CapacityReservationPreference": "open"
            },
            "MetadataOptions": {
                "State": "pending",
                "HttpTokens": "optional",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "enabled"
            }
        }
    ],
    "OwnerId": "123456789012",
    "ReservationId": "r-02a3f596d91211712"
}
```

Beispiel 2: So starten Sie eine Instance in einem nicht standardmäßigen Subnetz und fügen eine öffentliche IP-Adresse hinzu

Im folgenden Beispiel für `run-instances` wird eine öffentliche IP-Adresse für eine Instance angefordert, die Sie in einem nicht standardmäßigen Subnetz starten. Die Instance ist mit der angegebenen Sicherheitsgruppe verbunden.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --subnet-id subnet-08fc749671b2d077c \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --associate-public-ip-address \
    --key-name MyKeyPair
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Beispiel 3: So starten Sie eine Instance mit zusätzlichen Volumes

Das folgende `run-instances`-Beispiel verwendet eine Blockgerät-Zuweisung, die in mapping.json angegeben ist, um beim Start zusätzliche Volumes anzufügen. Eine Blockgerät-Zuweisung kann EBS-Volumes, Instance-Speicher-Volumes oder sowohl EBS-Volumes als auch Instance-Speicher-Volumes angeben.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --subnet-id subnet-08fc749671b2d077c \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --key-name MyKeyPair \
    --block-device-mappings file://mapping.json
```

Inhalt von `mapping.json`. In diesem Beispiel wird `/dev/sdh` ein leeres EBS-Volume mit einer Größe von 100 GiB hinzugefügt.

```
[
    {
        "DeviceName": "/dev/sdh",
        "Ebs": {
            "VolumeSize": 100
        }
    }
]
```

Inhalt von `mapping.json`. In diesem Beispiel wird `ephemeral1` als Instance-Speicher-Volume hinzugefügt.

```
[
    {
        "DeviceName": "/dev/sdc",
        "VirtualName": "ephemeral1"
    }
]
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen zu Blockgerät-Zuweisungen finden Sie unter Blockgerät-Zuweisungen im Benutzerhandbuch für Amazon EC2.

Beispiel 4: So starten Sie eine Instance und fügen bei der Erstellung Tags hinzu

Im folgenden Beispiel für `run-instances` wird ein Tag mit dem Schlüssel `webserver` und dem Wert `production` zur Instance hinzugefügt. Der folgende Befehl wendet ein Tag mit einem Schlüssel von `cost-center` und einem Wert von `cc123` auf ein erstelltes EBS-Volume an (in diesem Fall das Root-Volume).

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --tag-specifications
 'ResourceType=instance,Tags=[{Key=webserver,Value=production}]'
 'ResourceType=volume,Tags=[{Key=cost-center,Value=cc123}]'
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Beispiel 5: So starten Sie eine Instance mit Benutzerdaten

Im folgenden Beispiel für `run-instances` werden Benutzerdaten in eine Datei mit dem Namen `my_script.txt` übergeben, die ein Konfigurationsskript für Ihre Instance enthält. Das Skript wird beim Start ausgeführt.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --user-data file://my_script.txt
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen zu Instance-Benutzerdaten finden Sie unter Arbeiten mit Instance-Benutzerdaten im Benutzerhandbuch für Amazon EC2.

Beispiel 6: So starten Sie eine Instance mit Spitzenlastleistung

Im folgenden Beispiel für `run-instances` wird eine t2.micro-Instance mit der `unlimited`-Kreditoption gestartet. Wenn Sie eine T2-Instance starten und keinen `--credit-`

`specification` angeben, wird standardmäßig die Kreditoption `standard` verwendet. Wenn Sie eine T3-Instance starten, ist die Standardeinstellung die Kreditoption `unlimited`.

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --count 1 \
    --subnet-id subnet-08fc749671b2d077c \
    --key-name MyKeyPair \
    --security-group-ids sg-0b0384b66d7d692f9 \
    --credit-specification CpuCredits=unlimited
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen über Burstable Performance Instances finden Sie unter Burstable Performance Instances im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie RunInstancesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Runs an EC2 instance asynchronously.
     *
     * @param instanceType The instance type to use for the EC2 instance.
     * @param keyName The name of the key pair to associate with the EC2
 instance.
     * @param groupName The name of the security group to associate with the EC2
 instance.
     * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
 instance.
     * @return A {@link CompletableFuture} that completes with the ID of the
 started EC2 instance.
```

```
     * @throws RuntimeException If there is an error running the EC2 instance.
     */
    public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, RunInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Create new EC2 instances.
 * @param {{
 *  keyName: string,
 *  securityGroupIds: string[],
 *  imageId: string,
 *  instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *  minCount?: number,
 *  maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,
  imageId,
  instanceType,
  minCount = "1",
  maxCount = "1",
}) => {
  const client = new EC2Client({});
  minCount = Number.parseInt(minCount);
  maxCount = Number.parseInt(maxCount);
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: keyName,
    // Your security group.
    SecurityGroupIds: securityGroupIds,
    // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
 For more information, see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
    // An instance type describing the resources provided to your instance. There
 are multiple
    // ways to search for instance types. For more information see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-
discovery.html
    InstanceType: instanceType,
    // Availability Zones have capacity limitations that may impact your ability
 to launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at least
 the `MinCount` of instances.
```

```
    // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
    // If you need a specific number of instances, use `MinCount` and `MaxCount`
 set to the same value.
    // If you want to launch up to a certain number of instances, use `MaxCount`
 and let EC2 provision as many as possible.
    // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
    MinCount: minCount,
    MaxCount: maxCount,
  });

  try {
    const { Instances } = await client.send(command);
    const instanceList = Instances.map(
      (instance) => `• ${instance.InstanceId}`,
    ).join("\n");
    console.log(`Launched instances:\n${instanceList}`);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2Instance(
    name: String,
```

```
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiId")
        return instanceId
    }
}
```

- Einzelheiten zur API finden Sie RunInstancesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in EC2-Classic oder einer Standard-VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Beispiel 2: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in einer VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Beispiel 3: Um ein EBS-Volume oder ein Instance-Speicher-Volume hinzuzufügen, definieren Sie eine Blockgerät-Zuweisung und fügen Sie diese dem Befehl hinzu. In diesem Beispiel wird als Instance-Speicher-Volume hinzugefügt.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Beispiel 4: Um eines der aktuellen Windows-AMIs anzugeben, rufen Sie dessen AMI-ID mithilfe von ab Get-EC2ImageByName. In diesem Beispiel wird eine Instance aus dem aktuellen Basis-AMI für Windows Server 2016 gestartet 2016.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

Beispiel 5: Startet eine Instance in der angegebenen Dedicated-Host-Umgebung.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Beispiel 6: Diese Anforderung startet zwei Instances und weist diesen Instances ein Tag mit dem Schlüssel „webserver” und dem Wert „production” zu. Die Anfrage wendet außerdem ein Tag mit dem Schlüssel „cost-center“ und dem Wert „cc123“ auf die erstellten Volumes an (in diesem Fall das Stammvolume für jede Instance).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

- Einzelheiten zur API finden Sie unter RunInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in EC2-Classic oder einer Standard-VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Beispiel 2: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in einer VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Beispiel 3: Um ein EBS-Volume oder ein Instance-Speicher-Volume hinzuzufügen, definieren Sie eine Blockgerät-Zuweisung und fügen Sie diese dem Befehl hinzu. In diesem Beispiel wird als Instance-Speicher-Volume hinzugefügt.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"
```

```
New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Beispiel 4: Um eines der aktuellen Windows-AMIs anzugeben, rufen Sie dessen AMI-ID mithilfe von ab Get-SSMLatestEC2Image. In diesem Beispiel wird eine Instance aus dem aktuellen Basis-AMI für Windows Server 2016 gestartet 2016.

```
$ami = (Get-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
 'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

Beispiel 5: Startet eine Instance in der angegebenen Dedicated-Host-Umgebung.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Beispiel 6: Diese Anforderung startet zwei Instances und weist diesen Instances ein Tag mit dem Schlüssel „webserver” und dem Wert „production” zu. Die Anfrage wendet außerdem ein Tag mit dem Schlüssel „cost-center“ und dem Wert „cc123“ auf die erstellten Volumes an (in diesem Fall das Stammvolume für jede Instance).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

Beispiel 7: In diesem Beispiel werden die Berechtigungen für den Start einer EC2-Instance mithilfe des DryRun Parameters validiert, ohne die Instance tatsächlich zu erstellen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-
key-pair' -Region 'us-west-2' -DryRun $true
```

- Einzelheiten zur API finden Sie unter RunInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.
```

```
        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group
of the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
```

```
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type
 '{instance_type}'. "
                        "Select a different instance type or launch in a
 different availability zone."
                    )
                )
            raise
        return self.instances
```

- Einzelheiten zur API finden Sie RunInstancesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }
```

```
        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- Einzelheiten zur API finden Sie RunInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create tags for resource created during instance launch. "
    DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
    DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
    ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
      iv_resourcetype = 'instance'
      it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
      ) ).
    APPEND ls_tagspecifications TO lt_tagspecifications.

    TRY.
        " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
        oo_result = lo_ec2->runinstances(                           " oo_result
 is returned for testing purposes. "
          iv_imageid = iv_ami_id
          iv_instancetype = 't3.micro'
          iv_maxcount = 1
          iv_mincount = 1
          it_tagspecifications = lt_tagspecifications
          iv_subnetid = iv_subnet_id ).
        MESSAGE 'EC2 instance created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RunInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
 EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie RunInstancesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **RunScheduledInstances** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RunScheduledInstances` verwendet wird.

CLI

AWS CLI

So starten Sie eine geplante Instance

In diesem Beispiel wird die angegebene geplante Instance in einer VPC gestartet.

Befehl:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json
```

Launch-specification.json:

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "InstanceType": "c4.large",
  "NetworkInterfaces": [
    {
        "DeviceIndex": 0,
        "SubnetId": "subnet-12345678",
        "AssociatePublicIpAddress": true,
        "Groups": ["sg-12345678"]
    }
  ],
  "IamInstanceProfile": {
      "Name": "my-iam-role"
  }
}
```

Ausgabe:

```
{
  "InstanceIdSet": [
      "i-1234567890abcdef0"
```

```
  ]
}
```

In diesem Beispiel wird die angegebene Scheduled Instance in gestartet EC2-Classic.

Befehl:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json
```

Launch-specification.json:

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
      "Name": "my-iam-role"
  }
}
```

Ausgabe:

```
{
  "InstanceIdSet": [
      "i-1234567890abcdef0"
  ]
}
```

- Einzelheiten zur API finden Sie RunScheduledInstancesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance gestartet.

```
New-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Einzelheiten zur API finden Sie unter RunScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance gestartet.

```
New-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- Einzelheiten zur API finden Sie unter RunScheduledInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`StartInstances`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartInstances` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
```

```
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
 ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to start instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
```

```
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
 ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So starten Sie eine Amazon-EC2-Instance

In diesem Beispiel wird die angegebene EBS-backed Amazon-Instance gestartet.

Befehl:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
    "StartingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
```

```
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Ihre Instance anhalten und starten im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

- Einzelheiten zur API finden Sie StartInstancesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
    *
    * @param instanceId the ID of the instance to start
    * @return a {@link CompletableFuture} that completes when the instance has
been started and is in the "running" state, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> startInstanceAsync(String instanceId) {
       StartInstancesRequest startRequest = StartInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();
```

```
        logger.info("Starting instance " + instanceId + " and waiting for it to
 run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed
 to start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
```

```
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- Einzelheiten zur API finden Sie StartInstancesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance gestartet.

```
Start-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Beispiel 2: In diesem Beispiel werden die angegebenen Instances gestartet.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Beispiel 3: In diesem Beispiel wird die Gruppe der Instances gestartet, die derzeit angehalten sind. Die von zurückgegebenen Instance-Objekte Get-EC2Instance werden über die Pipeline an Start-EC2Instance übergeben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
 Values="stopped"}).Instances | Start-EC2Instance
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Filter-Parameter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- Einzelheiten zur API finden Sie unter StartInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance gestartet.

```
Start-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Beispiel 2: In diesem Beispiel werden die angegebenen Instances gestartet.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Beispiel 3: In diesem Beispiel wird die Gruppe der Instances gestartet, die derzeit angehalten sind. Die von zurückgegebenen Instance-Objekte Get-EC2Instance werden über die Pipeline an übergeben. Start-EC2Instance Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
 Values="stopped"}).Instances | Start-EC2Instance
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Filter-Parameter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
```

```
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

Beispiel 5: In diesem Beispiel werden die Berechtigungen zum Starten einer EC2-Instance mithilfe des DryRun Parameters validiert, ohne die Instanz tatsächlich zu starten. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- Einzelheiten zur API finden Sie unter StartInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
```

```
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str,
instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
```

```
                    "Ensure the instances are in a stopped state before starting
 them."
                )
            raise
```

- Einzelheiten zur API finden Sie StartInstancesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
```

```
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
```

```
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_started?(ec2_client, instance_id)

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie eine EC2-Instance mithilfe der Instance-ID.

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
 {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
```

```
    }
```

Warten Sie mit der Waiters-API, bis sich die Instance in den Status „Bereit" und „OK" befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- Einzelheiten zur API finden Sie StartInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to
 start the instance without actually making the request. "
        lo_ec2->startinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to start this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
          " DryRun is set to false to start instance. "
          oo_result = lo_ec2->startinstances(           " oo_result is returned
 for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to start instance failed. User does not have
 permissions to start the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
 -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
```

```
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance:
\(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie StartInstancesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`StopInstances`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StopInstances` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit virtuellen Maschinen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
```

```
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
```

```
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Stop an EC2 instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
 ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to stop instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
 ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
              std::endl;
}
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So halten Sie eine Amazon-EC2-Instance an

Das folgende `stop-instances` Beispiel stoppt die angegebene EBS-backed Amazon-Instance.

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
    "StoppingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Ihre Instance anhalten und starten im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

Beispiel 2: So versetzen Sie eine Amazon-EC2-Instance in den Ruhezustand

Im folgenden `stop-instances` Beispiel wird die EBS-backed Amazon-Instance in den Ruhezustand versetzt, wenn die Instance für den Ruhezustand aktiviert ist und die Voraussetzungen für den Ruhezustand erfüllt. Nachdem die Instance in den Ruhezustand versetzt wurde, wird die Instance angehalten.

```
aws ec2 stop-instances \
```

```
    --instance-ids i-1234567890abcdef0 \
    --hibernate
```

Ausgabe:

```
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-1234567890abcdef0",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Hibernate your On-Demand Linux Instance im Amazon Elastic Cloud Compute-Benutzerhandbuch.

- Einzelheiten zur API finden Sie StopInstancesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for
 the instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
```

```
    * @return a {@link CompletableFuture} that completes when the instance has
been stopped, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
       StopInstancesRequest stopRequest = StopInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       CompletableFuture<Void> resultFuture = new CompletableFuture<>();
       logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
       getAsyncClient().stopInstances(stopRequest)
           .thenCompose(response -> {
               if (response.stoppingInstances().isEmpty()) {
                   return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
               }
               return ec2Waiter.waitUntilInstanceStopped(describeRequest);
           })
           .thenAccept(waiterResponse -> {
               logger.info("Successfully stopped instance " + instanceId);
               resultFuture.complete(null);
           })
           .exceptionally(throwable -> {
               logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
               resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
               return null;
           });

       return resultFuture;
   }
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
```

```
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- Einzelheiten zur API finden Sie StopInstancesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance angehalten.

```
Stop-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter StopInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance angehalten.

```
Stop-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter StopInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
 instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None
```

```
        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response
```

- Einzelheiten zur API finden Sie StopInstancesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'
```

```
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end

  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
```

```
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }
```

Warten Sie mithilfe der Waiters API, bis sich eine Instance im Status „Angehalten“ befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }
```

- Einzelheiten zur API finden Sie StopInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to stop
 the instance without actually making the request. "
        lo_ec2->stopinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to stop this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
          " DryRun is set to false to stop instance. "
          oo_result = lo_ec2->stopinstances(           " oo_result is returned
 for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to stop instance failed. User does not have
 permissions to stop the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
```

```
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StopInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )
```

```
            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
\(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie StopInstancesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`TerminateInstances`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TerminateInstances` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie ein grundlegendes virtuelles privates Netzwerk
- Einen Containerdienst für VM-Instanzen erstellen
- Blockspeicher-Volumes erstellen und verwalten

- Beginnen Sie mit dem Kauf auf dem Softwaremarktplatz
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit virtuellen Maschinen
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Terminate an EC2 instance.
   /// </summary>
   /// <param name="ec2InstanceId">The instance Id of the EC2 instance
   /// to terminate.</param>
   /// <returns>Async task.</returns>
   public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
   {
       try
       {
           var request = new TerminateInstancesRequest
           {
               InstanceIds = new List<string> { ec2InstanceId }
           };

           var response = await _amazonEC2.TerminateInstancesAsync(request);
           Console.Write("Waiting for the instance to terminate.");
           await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

           Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
           return response.TerminatingInstances;
       }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
```

```
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
```

```
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
            ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
                  "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
                  ", " <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beenden Sie eine Amazon-EC2-Instance

In diesem Beispiel wird die angegebene Instance beendet.

Befehl:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden von Amazon-EC2-Instances im Benutzerhandbuch zu die AWS -Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has
been terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a
waiter to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
```

```
        });
    }
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
```

```
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is
 ${instance.instanceId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie TerminateInstancesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet (die Instance kann sich im laufenden oder im angehaltenen Status befinden). Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, verwenden Sie den Schalter -Force, um den Prompt zu unterdrücken.

```
Remove-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter TerminateInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet (die Instance kann sich im laufenden oder im angehaltenen Status befinden). Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, verwenden Sie den Schalter -Force, um den Prompt zu unterdrücken.

```
Remove-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- Einzelheiten zur API finden Sie unter TerminateInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
 using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise
```

- Einzelheiten zur API finden Sie TerminateInstancesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
     response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
```

```
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
       tracing::info!("Deleting instance with id {instance_id}");
       self.stop_instance(instance_id).await?;
       self.client
           .terminate_instances()
           .instance_ids(instance_id)
           .send()
           .await?;
       self.wait_for_instance_terminated(instance_id).await?;
       tracing::info!("Terminated instance with id {instance_id}");
       Ok(())
   }
```

Warten Sie mithilfe der Waiters-API, bis sich eine Instance im Status „Beendet“ befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
   async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
       self.client
           .wait_until_instance_terminated()
           .instance_ids(instance_id)
           .wait(Duration::from_secs(60))
           .await
           .map_err(|err| match err {
               WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                   "Exceeded max time ({}s) waiting for instance to terminate.",
                   exceeded.max_wait().as_secs(),
               )),
               _ => EC2Error::from(err),
```

```
            })?;
        Ok(())
    }
```

- Einzelheiten zur API finden Sie TerminateInstancesin der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
```

```
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
\(error.localizedDescription)")
            return false
        }
    }
```

- Einzelheiten zur API finden Sie TerminateInstancesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`UnassignPrivateIpAddresses`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UnassignPrivateIpAddresses` verwendet wird.

CLI

AWS CLI

So heben Sie die Zuweisung einer sekundären privaten IP-Adresse zu einer Netzwerkschnittstelle auf

In diesem Beispiel wird die Zuweisung der genannten privaten IP-Adressen zu der spezifizierten Netzwerkschnittstelle aufgehoben. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 unassign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

- Einzelheiten zur API finden Sie UnassignPrivateIpAddressesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Einzelheiten zur API finden Sie unter UnassignPrivateIpAddresses AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- Einzelheiten zur API finden Sie unter UnassignPrivateIpAddresses AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UnmonitorInstances`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UnmonitorInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Disable monitoring for an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to disable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
```

```
                    instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                    std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
                  << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
                  instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UnmonitorInstancesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So deaktivieren Sie die detaillierte Überwachung für eine Instance

Dieser Beispielbefehl deaktiviert die detaillierte Überwachung für die angegebene Instance.

Befehl:

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "InstanceMonitorings": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "Monitoring": {
```

```
                "State": "disabling"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie UnmonitorInstancesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
```

```
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie UnmonitorInstancesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance deaktiviert.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter UnmonitorInstances AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance deaktiviert.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- Einzelheiten zur API finden Sie unter UnmonitorInstances AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateSecurityGroupRuleDescriptionsIngress`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateSecurityGroupRuleDescriptionsIngress` verwendet wird.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie die Beschreibung einer Sicherheitsgruppenregel für eingehenden Datenverkehr mit einer CIDR-Quelle

Im folgenden Beispiel für `update-security-group-rule-descriptions-ingress` wird die Beschreibung der Sicherheitsgruppenregel für den angegebenen Port und den IPv4-Adressbereich aktualisiert. Die Beschreibung „`SSH access from ABC office`“ ersetzt alle vorhandenen Beschreibungen für die Regel.

```
aws ec2 update-security-group-rule-descriptions-ingress \
    --group-id sg-02f0d35a850ba727f \
    --ip-permissions
 IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{CidrIp=203.0.113.0/16,Description="SSH
 access from corpnet"}]'
```

Ausgabe:

```
{
    "Return": true
}
```

Weitere Informationen finden Sie unter Sicherheitsgruppenregeln im Benutzerhandbuch für Amazon EC2.

Beispiel 2: So aktualisieren Sie die Beschreibung einer Sicherheitsgruppenregel für eingehenden Datenverkehr mit einer Präfixlistenquelle

Im folgenden Beispiel für `update-security-group-rule-descriptions-ingress` wird die Beschreibung der Sicherheitsgruppenregel für den angegebenen Port und die Präfixliste aktualisiert. Die Beschreibung „`SSH access from ABC office`“ ersetzt alle vorhandenen Beschreibungen für die Regel.

```
aws ec2 update-security-group-rule-descriptions-ingress \
    --group-id sg-02f0d35a850ba727f \
    --ip-permissions
 IpProtocol=tcp,FromPort=22,ToPort=22,PrefixListIds='[{PrefixListId=pl-12345678,Descripti
 access from corpnet"}]'
```

Ausgabe:

```
{
    "Return": true
}
```

Weitere Informationen finden Sie unter Sicherheitsgruppenregeln im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie UpdateSecurityGroupRuleDescriptionsIngressin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}
```

```
Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithUpdatedDescription
```

Beispiel 2: Entfernt die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel (indem der Parameter in der Anforderung weggelassen wird).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithoutDescription
```

- Einzelheiten zur API finden Sie unter UpdateSecurityGroupRuleDescriptionsIngress AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithUpdatedDescription
```

Beispiel 2: Entfernt die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel (indem der Parameter in der Anforderung weggelassen wird).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
 $existingInboundRule.GroupId -SecurityGroupRuleDescription
 $ruleWithoutDescription
```

- Einzelheiten zur API finden Sie unter UpdateSecurityGroupRuleDescriptionsIngress AWS - Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Amazon EC2 AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon EC2 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon EC2 oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erste Schritte mit virtuellen privaten Netzwerken
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Container-Service für Instanzen virtueller Maschinen erstellen
- Eine relationale Datenbankinstanz erstellen
- Ein Service Networking Mesh erstellen und verwalten
- Blockspeicher-Volumes erstellen und verwalten
- Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI
- Beginnen Sie mit dem Kauf von Softwaremarktplätzen über die CLI

- Erste Schritte mit Big-Data-Verarbeitungsclustern
- Erste Schritte mit Dokumentendatenbanken
- Erste Schritte mit Graphdatenbanken
- Erste Schritte mit In-Memory-Caching
- Erste Schritte mit dem Lastenausgleich
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Network Transit Gateways
- Erste Schritte mit Videotransport-Streaming
- Erste Schritte mit virtuellen Maschinen
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Erste Schritte mit der IP-Adressverwaltung über die CLI
- Erstellen Sie mithilfe der CLI ein virtuelles privates Netzwerk mit privaten Subnetzen und Gateways zur Netzwerkadressübersetzung
- Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung
- Ich arbeite mit Netzwerk-Peering-Verbindungen

Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
 _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
  calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
```

```
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);
```

```
            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =
```

```
 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
```

```
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
```

```
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
```

```
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
```

```
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
```

```
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();
```

```
            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();
```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
```

```
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
```

```
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
```

```
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
      type: "confirm",
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
```

```
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
```

```
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
```

```
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
```

```
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
```

```
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
```

```
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
```

```
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
```

```
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
```

```
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
```

```
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
```

```
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
```

```
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
```

```
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
```

```
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
```

```
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
```

```
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
```

```
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
```

```
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
```

```
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
```

```
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
```

```
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
```

```
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
```

```
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
```

```
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
```

```
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
```

```
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
```

```
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
```

```
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```

```
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
```

```
    }),
    new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
      if (state.deleteSsmOnlyInstanceProfileError) {
        console.error(state.deleteSsmOnlyInstanceProfileError);
        return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
          "${INSTANCE_PROFILE_NAME}",
          NAMES.ssmOnlyInstanceProfileName,
        );
      }
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }),
    new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
      try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
          new DeletePolicyCommand({
            PolicyArn: ssmOnlyPolicy.Arn,
          }),
        );
      } catch (e) {
        state.deleteSsmOnlyPolicyError = e;
      }
    }),
    new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
      if (state.deleteSsmOnlyPolicyError) {
        console.error(state.deleteSsmOnlyPolicyError);
        return MESSAGES.deleteSsmOnlyPolicyError.replace(
          "${POLICY_NAME}",
          NAMES.ssmOnlyPolicyName,
        );
      }
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }),
    new ScenarioAction("deleteSsmOnlyRole", async (state) => {
      try {
        const iamClient = new IAMClient({});
        await iamClient.send(
```

```
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
```

```
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
```

```
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.

- AttachLoadBalancerTargetGroups
- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")
```

```
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
```

```
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
```

```
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
```

```
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
```

```
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
```

```
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.
```

```
        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
```

```
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
```

```
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
```

```
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
```

```
            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
```

```
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
```

```
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
```

```
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
```

```
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
```

```
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")
```

```
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
```

```
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
```

```
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
```

```
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
```

```
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
 found. "
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
```

```
        Initializes the RecommendationService class with the necessary
parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
```

```
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
```

```
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen

- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)
```

```
# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
```

```
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
```

```
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
```

```
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
```

```
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
```

```
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy

- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erste Schritte mit virtuellen privaten Netzwerken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer VPC
- Erstellen von Subnetzen
- Konfigurieren der Internetverbindung
- Erstellen eines NAT-Gateways
- Konfigurieren der Subnetz-Einstellungen
- Erstellen von Sicherheitsgruppen
- Ihre VPC-Konfiguration überprüfen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Creation Script
# This script creates a VPC with public and private subnets, internet gateway,
 NAT gateway, and security groups

# Set up logging
LOG_FILE="vpc_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to handle errors
handle_error() {
  echo "ERROR: $1"
  echo "Resources created before error:"
  for resource in "${CREATED_RESOURCES[@]}"
  do
    echo "- $resource"
  done

  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to clean up resources
cleanup_resources() {
  echo "Cleaning up resources in reverse order..."

  # Reverse the array to delete in reverse order of creation
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--))
  do
    resource="${CREATED_RESOURCES[$i]}"
    resource_type=$(echo "$resource" | cut -d':' -f1)
    resource_id=$(echo "$resource" | cut -d':' -f2)

    case "$resource_type" in
      "INSTANCE")
        echo "Terminating EC2 instance: $resource_id"
        aws ec2 terminate-instances --instance-ids "$resource_id" || echo "Failed
 to terminate instance: $resource_id"
        # Wait for instance to terminate
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$resource_id" || echo
 "Failed to wait for instance termination: $resource_id"
```

```
        ;;
      "KEY_PAIR")
        echo "Deleting key pair: $resource_id"
        aws ec2 delete-key-pair --key-name "$resource_id" || echo "Failed to
 delete key pair: $resource_id"
        # Remove the .pem file if it exists
        if [ -f "${resource_id}.pem" ]; then
          rm -f "${resource_id}.pem"
        fi
        ;;
      "NAT_GATEWAY")
        echo "Deleting NAT Gateway: $resource_id"
        aws ec2 delete-nat-gateway --nat-gateway-id "$resource_id" || echo
 "Failed to delete NAT Gateway: $resource_id"
        # NAT Gateway deletion takes time, wait for it to complete
        echo "Waiting for NAT Gateway to be deleted..."
        aws ec2 wait nat-gateway-deleted --nat-gateway-ids "$resource_id" || echo
 "Failed to wait for NAT Gateway deletion: $resource_id"
        ;;
      "EIP")
        echo "Releasing Elastic IP: $resource_id"
        aws ec2 release-address --allocation-id "$resource_id" || echo "Failed to
 release Elastic IP: $resource_id"
        ;;
      "ROUTE_TABLE_ASSOCIATION")
        echo "Disassociating Route Table: $resource_id"
        aws ec2 disassociate-route-table --association-id "$resource_id" || echo
 "Failed to disassociate Route Table: $resource_id"
        ;;
      "ROUTE_TABLE")
        echo "Deleting Route Table: $resource_id"
        aws ec2 delete-route-table --route-table-id "$resource_id" || echo
 "Failed to delete Route Table: $resource_id"
        ;;
      "INTERNET_GATEWAY")
        echo "Detaching Internet Gateway: $resource_id from VPC: $VPC_ID"
        aws ec2 detach-internet-gateway --internet-gateway-id "$resource_id" --
vpc-id "$VPC_ID" || echo "Failed to detach Internet Gateway: $resource_id"
        echo "Deleting Internet Gateway: $resource_id"
        aws ec2 delete-internet-gateway --internet-gateway-id "$resource_id" ||
 echo "Failed to delete Internet Gateway: $resource_id"
        ;;
      "SECURITY_GROUP")
        echo "Deleting Security Group: $resource_id"
```

```
        aws ec2 delete-security-group --group-id "$resource_id" || echo "Failed
 to delete Security Group: $resource_id"
        ;;
      "SUBNET")
        echo "Deleting Subnet: $resource_id"
        aws ec2 delete-subnet --subnet-id "$resource_id" || echo "Failed to
 delete Subnet: $resource_id"
        ;;
      "VPC")
        echo "Deleting VPC: $resource_id"
        aws ec2 delete-vpc --vpc-id "$resource_id" || echo "Failed to delete VPC:
 $resource_id"
        ;;
    esac
  done
}

# Initialize array to track created resources
CREATED_RESOURCES=()

echo "Starting VPC creation script at $(date)"

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
aws configure list || handle_error "AWS CLI is not properly configured"

# Verify identity and permissions
echo "Verifying identity and permissions..."
if ! aws sts get-caller-identity; then
  echo "ERROR: Unable to verify AWS identity. This could be due to:"
  echo "  - Expired credentials"
  echo "  - Missing or invalid AWS credentials"
  echo "  - Insufficient permissions"
  echo ""
  echo "Please run 'aws configure' to update your credentials or check your IAM
 permissions."
  exit 1
fi

# Create VPC
echo "Creating VPC with CIDR block 10.0.0.0/16..."
VPC_ID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'Vpc.VpcId' --output text)
```

```
if [ -z "$VPC_ID" ]; then
  handle_error "Failed to create VPC"
fi

CREATED_RESOURCES+=("VPC:$VPC_ID")
echo "VPC created with ID: $VPC_ID"

# Enable DNS support and hostnames
echo "Enabling DNS support and hostnames for VPC..."
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-support ||
 handle_error "Failed to enable DNS support"
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-hostnames ||
 handle_error "Failed to enable DNS hostnames"

# Get available Availability Zones
echo "Getting available Availability Zones..."
AZ1=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[0].ZoneName'
 --output text)
AZ2=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[1].ZoneName'
 --output text)

if [ -z "$AZ1" ] || [ -z "$AZ2" ]; then
  handle_error "Failed to get Availability Zones"
fi

echo "Using Availability Zones: $AZ1 and $AZ2"

# Create public subnets
echo "Creating public subnet in $AZ1..."
PUBLIC_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.0.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ1" ]; then
  handle_error "Failed to create public subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ1")
```

```
echo "Public subnet created in $AZ1 with ID: $PUBLIC_SUBNET_AZ1"

echo "Creating public subnet in $AZ2..."
PUBLIC_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.1.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ2" ]; then
  handle_error "Failed to create public subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ2")
echo "Public subnet created in $AZ2 with ID: $PUBLIC_SUBNET_AZ2"

# Create private subnets
echo "Creating private subnet in $AZ1..."
PRIVATE_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.2.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ1" ]; then
  handle_error "Failed to create private subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ1")
echo "Private subnet created in $AZ1 with ID: $PRIVATE_SUBNET_AZ1"

echo "Creating private subnet in $AZ2..."
PRIVATE_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.3.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
```

```
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ2" ]; then
  handle_error "Failed to create private subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ2")
echo "Private subnet created in $AZ2 with ID: $PRIVATE_SUBNET_AZ2"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-
gateway,Tags=[{Key=Name,Value=MyIGW},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' \
  --query 'InternetGateway.InternetGatewayId' \
  --output text)

if [ -z "$IGW_ID" ]; then
  handle_error "Failed to create Internet Gateway"
fi

CREATED_RESOURCES+=("INTERNET_GATEWAY:$IGW_ID")
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
 "$VPC_ID" || handle_error "Failed to attach Internet Gateway to VPC"

# Create public route table
echo "Creating public route table..."
PUBLIC_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-
RT},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PUBLIC_RT" ]; then
  handle_error "Failed to create public route table"
fi
```

```
CREATED_RESOURCES+=("ROUTE_TABLE:$PUBLIC_RT")
echo "Public route table created with ID: $PUBLIC_RT"

# Add route to Internet Gateway
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID" || handle_error "Failed to add route to
 Internet Gateway"

# Associate public subnets with public route table
echo "Associating public subnet in $AZ1 with public route table..."
PUBLIC_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
 --subnet-id "$PUBLIC_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_1" ]; then
  handle_error "Failed to associate public subnet in AZ1 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_1")

echo "Associating public subnet in $AZ2 with public route table..."
PUBLIC_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
 --subnet-id "$PUBLIC_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_2" ]; then
  handle_error "Failed to associate public subnet in AZ2 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_2")

# Create private route table
echo "Creating private route table..."
PRIVATE_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-
RT},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PRIVATE_RT" ]; then
  handle_error "Failed to create private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PRIVATE_RT")
```

```
echo "Private route table created with ID: $PRIVATE_RT"

# Associate private subnets with private route table
echo "Associating private subnet in $AZ1 with private route table..."
PRIVATE_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
 --subnet-id "$PRIVATE_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_1" ]; then
  handle_error "Failed to associate private subnet in AZ1 with private route
 table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_1")

echo "Associating private subnet in $AZ2 with private route table..."
PRIVATE_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
 --subnet-id "$PRIVATE_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_2" ]; then
  handle_error "Failed to associate private subnet in AZ2 with private route
 table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_2")

# Allocate Elastic IP for NAT Gateway
echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC=$(aws ec2 allocate-address --domain vpc --tag-specifications
 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'AllocationId' --output text)

if [ -z "$EIP_ALLOC" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

CREATED_RESOURCES+=("EIP:$EIP_ALLOC")
echo "Elastic IP allocated with ID: $EIP_ALLOC"

# Create NAT Gateway
echo "Creating NAT Gateway in public subnet in $AZ1..."
NAT_GW=$(aws ec2 create-nat-gateway \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --allocation-id "$EIP_ALLOC" \
```

```
  --tag-specifications
 'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'NatGateway.NatGatewayId' \
  --output text)

if [ -z "$NAT_GW" ]; then
  handle_error "Failed to create NAT Gateway"
fi

CREATED_RESOURCES+=("NAT_GATEWAY:$NAT_GW")
echo "NAT Gateway created with ID: $NAT_GW"

# Wait for NAT Gateway to be available
echo "Waiting for NAT Gateway to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW" || handle_error
 "NAT Gateway did not become available"

# Add route to NAT Gateway in private route table
echo "Adding route to NAT Gateway in private route table..."
aws ec2 create-route --route-table-id "$PRIVATE_RT" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW" || handle_error "Failed to add route to NAT
 Gateway"

# Enable auto-assign public IP for instances in public subnets
echo "Enabling auto-assign public IP for instances in public subnet in $AZ1..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ1" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
 subnet in AZ1"

echo "Enabling auto-assign public IP for instances in public subnet in $AZ2..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ2" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
 subnet in AZ2"

# Create security group for web servers
echo "Creating security group for web servers..."
WEB_SG=$(aws ec2 create-security-group \
  --group-name "WebServerSG-$(date +%s)" \
  --description "Security group for web servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
```

```
  --output text)

if [ -z "$WEB_SG" ]; then
  handle_error "Failed to create security group for web servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$WEB_SG")
echo "Security group for web servers created with ID: $WEB_SG"

# Allow HTTP and HTTPS traffic
echo "Allowing HTTP traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 80 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTP traffic"

echo "Allowing HTTPS traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 443 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTPS traffic"

# Note: In a production environment, you should restrict the source IP ranges for
 security
echo "NOTE: In a production environment, you should restrict the source IP ranges
 for HTTP and HTTPS traffic"

# Create security group for database servers
echo "Creating security group for database servers..."
DB_SG=$(aws ec2 create-security-group \
  --group-name "DBServerSG-$(date +%s)" \
  --description "Security group for database servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$DB_SG" ]; then
  handle_error "Failed to create security group for database servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$DB_SG")
echo "Security group for database servers created with ID: $DB_SG"

# Allow MySQL/Aurora traffic from web servers only
echo "Allowing MySQL/Aurora traffic from web servers to database servers..."
```

```
aws ec2 authorize-security-group-ingress --group-id "$DB_SG" --protocol tcp --
port 3306 --source-group "$WEB_SG" || handle_error "Failed to allow MySQL/Aurora
 traffic"

# Verify VPC configuration
echo "Verifying VPC configuration..."
echo "VPC:"
aws ec2 describe-vpcs --vpc-id "$VPC_ID" || handle_error "Failed to describe VPC"

echo "Subnets:"
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe subnets"

echo "Route tables:"
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe route tables"

echo "Internet gateway:"
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=
$VPC_ID" || handle_error "Failed to describe Internet Gateway"

echo "NAT gateway:"
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe NAT Gateway"

echo "Security groups:"
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe security groups"

echo ""
# Summary of created resources
echo "VPC creation completed successfully!"
echo "Summary of created resources:"
echo "- VPC: $VPC_ID"
echo "- Public Subnet in $AZ1: $PUBLIC_SUBNET_AZ1"
echo "- Public Subnet in $AZ2: $PUBLIC_SUBNET_AZ2"
echo "- Private Subnet in $AZ1: $PRIVATE_SUBNET_AZ1"
echo "- Private Subnet in $AZ2: $PRIVATE_SUBNET_AZ2"
echo "- Internet Gateway: $IGW_ID"
echo "- Public Route Table: $PUBLIC_RT"
echo "- Private Route Table: $PRIVATE_RT"
echo "- Elastic IP: $EIP_ALLOC"
echo "- NAT Gateway: $NAT_GW"
echo "- Web Servers Security Group: $WEB_SG"
```

```
echo "- Database Servers Security Group: $DB_SG"

# Deploy EC2 instances
echo ""
echo "Deploying EC2 instances..."

# Create key pair for SSH access
KEY_NAME="vpc-tutorial-key-$(date +%s)"
echo "Creating key pair $KEY_NAME..."
aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-specifications
 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'KeyMaterial' --output text >
 "${KEY_NAME}.pem" || handle_error "Failed to create key pair"
chmod 400 "${KEY_NAME}.pem"
echo "Key pair saved to ${KEY_NAME}.pem"
CREATED_RESOURCES+=("KEY_PAIR:$KEY_NAME")

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text) ||
 handle_error "Failed to get AMI"
echo "Using AMI: $AMI_ID"

# Launch web server in public subnet
echo "Launching web server in public subnet..."
WEB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$WEB_SG" \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --associate-public-ip-address \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y httpd
    systemctl start httpd
    systemctl enable httpd
    echo "<h1>Hello from $(hostname -f) in the public subnet</h1>" > /var/www/
html/index.html' \
```

```
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch web server"
echo "Web server instance created with ID: $WEB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$WEB_INSTANCE")

# Wait for web server to be running
echo "Waiting for web server to be running..."
aws ec2 wait instance-running --instance-ids "$WEB_INSTANCE"

# Get web server public IP
WEB_PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$WEB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "Web server public IP: $WEB_PUBLIC_IP"
echo "You can access the web server at: http://$WEB_PUBLIC_IP"

# Launch database server in private subnet
echo "Launching database server in private subnet..."
DB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$DB_SG" \
  --subnet-id "$PRIVATE_SUBNET_AZ1" \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y mariadb-server
    systemctl start mariadb
    systemctl enable mariadb' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch database server"
echo "Database server instance created with ID: $DB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$DB_INSTANCE")

# Wait for database server to be running
echo "Waiting for database server to be running..."
aws ec2 wait instance-running --instance-ids "$DB_INSTANCE"

# Get database server private IP
DB_PRIVATE_IP=$(aws ec2 describe-instances --instance-ids "$DB_INSTANCE" \
```

```
  --query 'Reservations[0].Instances[0].PrivateIpAddress' --output text)
echo "Database server private IP: $DB_PRIVATE_IP"

echo "EC2 instances deployed successfully!"
echo "- Web Server (Public): $WEB_INSTANCE ($WEB_PUBLIC_IP)"
echo "- Database Server (Private): $DB_INSTANCE ($DB_PRIVATE_IP)"
echo ""
echo "Note: To connect to the web server: ssh -i ${KEY_NAME}.pem ec2-user@
$WEB_PUBLIC_IP"
echo "To connect to the database server, you must first connect to the web
 server, then use it as a bastion host."
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  echo "Cleaning up resources..."
  cleanup_resources
  echo "All resources have been cleaned up."
else
  echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateKeyPair
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup

- CreateSubnet
- CreateVpc
- DeleteInternetGateway
- DeleteKeyPair
- DeleteNatGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeImages
- DescribeInstances
- DescribeInternetGateways
- DescribeNatGateways
- DescribeRouteTables
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachInternetGateway
- DisassociateRouteTable
- ModifySubnetAttribute
- ModifyVpcAttribute
- ReleaseAddress
- RunInstances
- TerminateInstances

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen

- Erstellen Sie den Dienst
- Bereinigen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()
```

```
# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
```

```
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")
```

```
        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
```

```
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
 then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"
```

```
                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
" ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."
```

```
            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
```

```
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

```
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"
```

```
# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
```

```
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
```

```
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi
```

```
echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
```

```
echo "==========================================="
echo "SERVICE INFORMATION"
echo "==========================================="
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "==========================================="
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "==========================================="
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices

- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Einen Dienst erstellen und überwachen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
```

```
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
```

```
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
```

```
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
$TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
```

```
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
```

```
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
```

```
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"
```

```
    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi
```

```
    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
```

```
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
```

```
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
```

```
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
```

```
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}
```

```
# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table
```

```
    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."
```

```
        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
```

```
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster

- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Eine relationale Datenbankinstanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Netzwerkkomponenten einrichten
- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."
```

```
    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
```

```
if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
```

```
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"
```

```
echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi
```

```
echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."
```

```
    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets

- DescribeVpcs
- GetSecretValue
- Wait

Ein Service Networking Mesh erstellen und verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ein Servicenetzwerk erstellen
- Einen Service erstellen
- Verfügbare VPCs auflisten
- Sicherheitsgruppen für die ausgewählte VPC auflisten
- Listet Dienstzuordnungen auf
- VPC-Zuordnungen auflisten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
```

```
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a
 "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a
 "$LOG_FILE"
        return 1
```

```
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
    validate_input "$resource_id" "resource_id" || return 1
    validate_input "$desired_status" "desired_status" || return 1

    echo "Waiting for $resource_type $resource_id to be in state
 $desired_status..." | tee -a "$LOG_FILE"

    while [ "$attempt" -le "$max_attempts" ]; do
        echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

        # Run the command to get the status and capture the output
        status_output=$(eval "$command" 2>&1) || true
        echo "$status_output" >> "$LOG_FILE"

        # For service networks, they do not have a status field in the output
        # We'll consider them active if we can retrieve them
        if [[ "$resource_type" == "Service Network" ]]; then
            if [[ "$status_output" == *"$resource_id"* ]]; then
                echo "$resource_type $resource_id is now active" | tee -a
 "$LOG_FILE"
                return 0
            fi
        else
            # For other resources, extract the status field
            status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
 $4}' || true)
            echo "Current status: $status" >> "$LOG_FILE"

            if [[ "$status" == "$desired_status" ]]; then
```

```
                echo "$resource_type $resource_id is now in state
 $desired_status" | tee -a "$LOG_FILE"
                return 0
            elif [[ "$status" == *"FAIL"* ]]; then
                echo "ERROR: $resource_type $resource_id failed to reach desired
 state. Current status: $status" | tee -a "$LOG_FILE"
                return 1
            fi
        fi

        echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
 "$LOG_FILE"
        sleep 10
        ((attempt++))
    done

    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"
```

```
# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/
dev/null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-
lattice get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --
output json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name
 '$SERVICE_NAME' --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-
vpc-lattice-gs --output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service
 --service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"

SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
service-association --service-identifier '$SERVICE_ID' --service-network-
identifier '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a
 "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"
```

```
VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output
 text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
```

```
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?

        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a
 "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE"
 "aws vpc-lattice get-service-network-vpc-association --service-network-vpc-
association-identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee
 -a "$LOG_FILE"
```

```
echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --
output json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --
service-network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --
service-network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
```

```
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'"
 || true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"

echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation

- GetServiceNetworkVpcAssociation
- ListServiceNetworkServiceAssociations
- ListServiceNetworkVpcAssociations

Blockspeicher-Volumes erstellen und verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EBS-Volume
- Überprüfen Sie den Volume-Status
- Erstellen Sie eine EC2-Instance (optional)
- Hängen Sie ein Volume an eine EC2-Instance an
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create and manage Amazon EBS volumes
# This script demonstrates how to create an EBS volume and attach it to an EC2
 instance
# It can also create a test EC2 instance if needed

# Set up logging
LOG_FILE="ebs-volume-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS volume creation script at $(date)"
echo "================================================"
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "$VOLUME_ID" ]; then
        echo "- EBS Volume: $VOLUME_ID"
    fi
    if [ -n "$INSTANCE_ID" ]; then
        echo "- EC2 Instance: $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "- Security Group: $SG_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Do you want to clean up created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
    else
        echo "Resources were not cleaned up. You will need to delete them
 manually."
    fi

    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "$VOLUME_ID" ] && [ "$ATTACHED" = true ]; then
        echo "Detaching volume $VOLUME_ID..."
        aws ec2 detach-volume --volume-id "$VOLUME_ID"

        # Wait for volume to be detached
        echo "Waiting for volume to be detached..."
        aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
    fi
```

```
    if [ -n "$VOLUME_ID" ]; then
        echo "Deleting volume $VOLUME_ID..."
        aws ec2 delete-volume --volume-id "$VOLUME_ID"
    fi

    if [ -n "$INSTANCE_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Terminating instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
    fi

    # Clean up security group if created
    if [ -n "$SG_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Deleting security group $SG_ID..."
        # Wait a bit for instance termination to complete
        sleep 10
        aws ec2 delete-security-group --group-id "$SG_ID" 2>/dev/null || echo
 "Security group may have dependencies, delete manually if needed"
    fi

    echo "Cleanup completed."
}

# Function to get available instance type
get_available_instance_type() {
    local region=$1

    # Try instance types in order of preference (cheapest first)
    local instance_types=("t3.nano" "t3.micro" "t2.micro" "t2.nano")

    for instance_type in "${instance_types[@]}"; do
        local available=$(aws ec2 describe-instance-type-offerings \
            --region "$region" \
            --filters "Name=instance-type,Values=$instance_type" \
            --query "length(InstanceTypeOfferings)" \
            --output text)

        if [ "$available" -gt 0 ]; then
            echo "$instance_type"
            return 0
        fi
    done
```

```
    # If none of the preferred types are available, get any available type
    local fallback_type=$(aws ec2 describe-instance-type-offerings \
        --region "$region" \
        --query "InstanceTypeOfferings[0].InstanceType" \
        --output text)

    if [ "$fallback_type" != "None" ] && [ -n "$fallback_type" ]; then
        echo "$fallback_type"
        return 0
    fi

    return 1
}

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 "AvailabilityZones[0].RegionName" --output text)
fi

echo "Using region: $REGION"

# Get available Availability Zones
echo "Retrieving available Availability Zones..."
AZ=$(aws ec2 describe-availability-zones --filters "Name=state,Values=available"
 --query "AvailabilityZones[0].ZoneName" --output text)
if [ -z "$AZ" ]; then
    handle_error "Failed to retrieve Availability Zones"
fi
echo "Using Availability Zone: $AZ"

# Create a gp3 volume
echo "Creating a 10 GiB gp3 volume in $AZ..."
VOLUME_ID=$(aws ec2 create-volume \
    --volume-type gp3 \
    --size 10 \
    --availability-zone "$AZ" \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=EBSTutorialVolume},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-gs-volumes}]' \
    --query 'VolumeId' \
```

```
    --output text)

if [ -z "$VOLUME_ID" ]; then
    handle_error "Failed to create EBS volume"
fi

echo "Volume created with ID: $VOLUME_ID"

# Wait for volume to become available
echo "Waiting for volume to become available..."
aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
if [ $? -ne 0 ]; then
    handle_error "Volume did not become available"
fi

# Check volume details
echo "Retrieving volume details..."
aws ec2 describe-volumes --volume-ids "$VOLUME_ID"

# Ask if user wants to attach the volume to an instance
echo ""
echo "==========================================="
echo "VOLUME ATTACHMENT"
echo "==========================================="
echo "Do you want to attach this volume to an EC2 instance? (y/n): "
read -r ATTACH_CHOICE

ATTACHED=false
CREATED_INSTANCE=false
INSTANCE_ID=""
SG_ID=""

if [[ "$ATTACH_CHOICE" =~ ^[Yy]$ ]]; then
    # List available instances in the same AZ
    echo "Retrieving EC2 instances in $AZ..."
    INSTANCES_COUNT=$(aws ec2 describe-instances \
        --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
        --query "length(Reservations[].Instances[])" \
        --output text)

    # Check if there are any running instances in the AZ
    if [ "$INSTANCES_COUNT" -eq 0 ]; then
        echo "No running instances found in $AZ."
```

```
        echo ""
        echo "Would you like to create a test EC2 instance? (y/n): "
        read -r CREATE_INSTANCE_CHOICE

        if [[ "$CREATE_INSTANCE_CHOICE" =~ ^[Yy]$ ]]; then
            # Get available instance type
            echo "Finding available instance type for region $REGION..."
            INSTANCE_TYPE=$(get_available_instance_type "$REGION")
            if [ $? -ne 0 ] || [ -z "$INSTANCE_TYPE" ]; then
                handle_error "No suitable instance type found in region $REGION"
            fi
            echo "Using instance type: $INSTANCE_TYPE"

            # Get the latest Amazon Linux 2 AMI
            echo "Finding the latest Amazon Linux 2 AMI..."
            AMI_ID=$(aws ec2 describe-images \
                --owners amazon \
                --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
"Name=state,Values=available" \
                --query "sort_by(Images, &CreationDate)[-1].ImageId" \
                --output text)

            if [ -z "$AMI_ID" ]; then
                handle_error "Failed to find a suitable AMI"
            fi

            echo "Using AMI: $AMI_ID"

            # Check if a default VPC exists
            DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
                --filters "Name=isDefault,Values=true" \
                --query "Vpcs[0].VpcId" \
                --output text)

            if [ "$DEFAULT_VPC_ID" = "None" ] || [ -z "$DEFAULT_VPC_ID" ]; then
                handle_error "No default VPC found. Please create a VPC and
subnet before running this script."
            fi

            # Get a subnet in the selected AZ
            SUBNET_ID=$(aws ec2 describe-subnets \
                --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID"
"Name=availability-zone,Values=$AZ" \
                --query "Subnets[0].SubnetId" \
```

```
                --output text)

            if [ -z "$SUBNET_ID" ] || [ "$SUBNET_ID" = "None" ]; then
                handle_error "No subnet found in $AZ. Please create a subnet
 before running this script."
            fi

            echo "Using subnet: $SUBNET_ID"

            # Create a security group that allows SSH
            SG_NAME="EBSTutorialSG-$(date +%s)"
            SG_ID=$(aws ec2 create-security-group \
                --group-name "$SG_NAME" \
                --description "Security group for EBS tutorial" \
                --vpc-id "$DEFAULT_VPC_ID" \
                --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "GroupId" \
                --output text)

            if [ -z "$SG_ID" ]; then
                handle_error "Failed to create security group"
            fi

            echo "Created security group: $SG_ID"

            # Add a rule to allow SSH
            aws ec2 authorize-security-group-ingress \
                --group-id "$SG_ID" \
                --protocol tcp \
                --port 22 \
                --cidr 0.0.0.0/0

            echo "Added SSH rule to security group"

            # Create the instance
            echo "Creating EC2 instance in $AZ with instance type
 $INSTANCE_TYPE..."
            INSTANCE_ID=$(aws ec2 run-instances \
                --image-id "$AMI_ID" \
                --instance-type "$INSTANCE_TYPE" \
                --subnet-id "$SUBNET_ID" \
                --security-group-ids "$SG_ID" \
```

```
                --tag-specifications
 'ResourceType=instance,Tags=[{Key=Name,Value=EBSTutorialInstance},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "Instances[0].InstanceId" \
                --output text)

            if [ -z "$INSTANCE_ID" ]; then
                handle_error "Failed to create EC2 instance"
            fi

            CREATED_INSTANCE=true
            echo "Instance created with ID: $INSTANCE_ID"

            # Wait for the instance to be running
            echo "Waiting for instance to be running..."
            aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

            # Wait a bit more for the instance to initialize
            echo "Waiting for instance initialization (30 seconds)..."
            sleep 30
        else
            echo "Skipping instance creation and volume attachment."
            INSTANCE_ID=""
        fi
    else
        # Display available instances
        echo "Available instances in $AZ:"
        aws ec2 describe-instances \
            --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
            --query "Reservations[*].Instances[*].[InstanceId,Tags[?
Key=='Name'].Value|[0],InstanceType]" \
            --output table

        # Ask for instance ID
        echo ""
        echo "Enter the instance ID to attach the volume to (or press Enter to
 skip): "
        read -r INSTANCE_ID
    fi

    if [ -n "$INSTANCE_ID" ]; then
        # Attach volume to the instance
```

```
        echo "Attaching volume $VOLUME_ID to instance $INSTANCE_ID..."
        ATTACH_RESULT=$(aws ec2 attach-volume \
            --volume-id "$VOLUME_ID" \
            --instance-id "$INSTANCE_ID" \
            --device "/dev/sdf" \
            --query 'State' \
            --output text)

        if [ $? -ne 0 ] || [ -z "$ATTACH_RESULT" ]; then
            handle_error "Failed to attach volume to instance"
        fi

        ATTACHED=true
        echo "Volume attached successfully. Device: /dev/sdf"

        # Verify attachment
        echo "Verifying attachment..."
        aws ec2 describe-volumes \
            --volume-ids "$VOLUME_ID" \
            --query "Volumes[0].Attachments"
    else
        echo "Skipping volume attachment."
    fi
else
    echo "Skipping volume attachment."
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCE SUMMARY"
echo "==========================================="
echo "Created resources:"
echo "- EBS Volume: $VOLUME_ID"
if [ "$ATTACHED" = true ]; then
    echo "  - Attached to: $INSTANCE_ID as /dev/sdf"
fi
if [ "$CREATED_INSTANCE" = true ]; then
    echo "- EC2 Instance: $INSTANCE_ID (type: $INSTANCE_TYPE)"
fi
if [ -n "$SG_ID" ]; then
    echo "- Security Group: $SG_ID"
fi
```

```
# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
else
    echo ""
    echo "Resources were not cleaned up. You can manually delete them later."
    if [ -n "$VOLUME_ID" ]; then
        if [ "$ATTACHED" = true ]; then
            echo "To detach the volume:"
            echo "  aws ec2 detach-volume --volume-id $VOLUME_ID"
        fi
        echo "To delete the volume:"
        echo "  aws ec2 delete-volume --volume-id $VOLUME_ID"
    fi
    if [ "$CREATED_INSTANCE" = true ]; then
        echo "To terminate the instance:"
        echo "  aws ec2 terminate-instances --instance-ids $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "To delete the security group (after instance termination):"
        echo "  aws ec2 delete-security-group --group-id $SG_ID"
    fi
fi

echo ""
echo "Script completed at $(date)"
echo "================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachVolume
  - AuthorizeSecurityGroupIngress
  - CreateSecurityGroup
  - CreateVolume

- DeleteSecurityGroup
- DeleteVolume
- DescribeAvailabilityZones
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeSubnets
- DescribeVolumes
- DescribeVpcs
- DetachVolume
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit dedizierten Netzwerkverbindungen über die CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 CreateVpnGateway
- Verwenden Sie ec2 DeleteVpnGateway
- Verwenden Sie ec2 DescribeVpnGateways
- Verwenden Sie Directconnect CreateConnection
- Verwenden Sie Directconnect CreatePrivateVirtualInterface
- Verwenden Sie Directconnect DeleteConnection

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect
 connections using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1
```

```
    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
```

```
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is
 $CONNECTION_STATE, waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
```

```
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID"
 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"'
 -f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual
 interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"
```

```
# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o
 '"virtualInterfaceId": "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi
```

```
# Step 8: Check connection state and provide guidance for public virtual
 interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
```

```
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is
 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
```

```
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways
  - UpdateConnection

Beginnen Sie mit dem Kauf von Softwaremarktplätzen über die CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 AuthorizeSecurityGroupIngress
- Verwenden Sie ec2 CreateKeyPair
- Verwenden Sie ec2 CreateSecurityGroup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Marketplace Buyer Getting Started Script
# This script demonstrates how to search for products in AWS Marketplace,
# launch an EC2 instance with a product AMI, and manage subscriptions.

set -euo pipefail

# Setup logging with secure permissions
LOG_FILE="marketplace-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS Marketplace Buyer Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. List available products in AWS Marketplace"
echo "2. Create resources needed to launch an EC2 instance"
echo "3. Launch an EC2 instance with an Amazon Linux 2 AMI"
echo "4. Show how to manage and terminate the instance"
echo "====================================================="
echo ""

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it first."
    exit 1
fi
```

```
# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured. Please configure them
 first."
    exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
    echo "ERROR: jq is not installed. Please install jq for safe JSON parsing."
    exit 1
fi

# Function to safely extract JSON values using jq
extract_json_value() {
    local json=$1
    local query=$2

    echo "$json" | jq -r "$query" 2>/dev/null || {
        echo "ERROR: Failed to parse JSON with query: $query" >&2
        return 1
    }
}

# Function to validate AWS permissions
validate_aws_permissions() {
    echo "Validating AWS permissions..."

    local identity
    identity=$(aws sts get-caller-identity --output json)
    local account_id
    account_id=$(extract_json_value "$identity" '.Account') || return 1
    local arn
    arn=$(extract_json_value "$identity" '.Arn') || return 1

    echo "AWS Account ID: $account_id"
    echo "AWS Principal ARN: $arn"
    echo "Note: This script requires EC2 permissions for key pair, security
 group, and instance management."
    echo ""
}

# Function to clean up resources
cleanup_resources() {
```

```
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${INSTANCE_ID:-}" ]; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true

        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" 2>/dev/null || true
        echo "Instance terminated successfully."
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting before deleting security group..."
        sleep 5
        echo "Deleting security group: $SECURITY_GROUP_ID"
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true
        echo "Security group deleted."
    fi

    if [ -n "${KEY_NAME:-}" ]; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" --region "$AWS_REGION" > /
dev/null 2>&1 || true

        # Remove the local key file if it exists with secure deletion
        if [ -f "${KEY_NAME}.pem" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "${KEY_NAME}.pem" 2>/dev/null || rm -f
 "${KEY_NAME}.pem"
            else
                rm -f "${KEY_NAME}.pem"
            fi
            echo "Local key file securely deleted."
        fi
    fi

    echo "Cleanup completed."
}
```

```
# Set trap to ensure cleanup on script exit
trap cleanup_resources EXIT

# Get the current AWS region
AWS_REGION=$(aws configure get region || echo "us-east-1")
if [ -z "$AWS_REGION" ] || [ "$AWS_REGION" = "None" ]; then
    AWS_REGION="us-east-1"
fi
echo "Using AWS Region: $AWS_REGION"
echo ""

# Validate permissions
validate_aws_permissions

# Generate random identifier for resource names using cryptographically secure
 method
RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
KEY_NAME="marketplace-key-${RANDOM_ID}"
SECURITY_GROUP_NAME="marketplace-sg-${RANDOM_ID}"

# Initialize variables to track created resources
INSTANCE_ID=""
SECURITY_GROUP_ID=""
AMI_ID=""

# Step 1: List available products in AWS Marketplace
echo "Listing available products in AWS Marketplace..."
echo "Note: In a real scenario, you would use marketplace-catalog commands to
 list and search for products."
echo "However, this requires specific permissions and product knowledge."
echo ""
echo "For this tutorial, we'll use a public Amazon Linux 2 AMI instead of an
 actual marketplace product."
echo "This is because subscribing to marketplace products requires accepting
 terms via the console."
echo ""

# Step 2: Create a key pair for SSH access
echo "Creating key pair: $KEY_NAME"
KEY_OUTPUT=$(aws ec2 create-key-pair \
  --key-name "$KEY_NAME" \
  --region "$AWS_REGION" \
  --query 'KeyMaterial' \
```

```
  --output text) || {
    echo "ERROR: Failed to create key pair" >&2
    exit 1
}

# Securely save the key with restricted permissions
if ! echo "$KEY_OUTPUT" > "${KEY_NAME}.pem" 2>/dev/null; then
    echo "ERROR: Failed to write key file ${KEY_NAME}.pem" >&2
    exit 1
fi
chmod 600 "${KEY_NAME}.pem" || {
    echo "ERROR: Failed to set permissions on key file" >&2
    rm -f "${KEY_NAME}.pem"
    exit 1
}
echo "Key pair created and saved to ${KEY_NAME}.pem with secure permissions
 (600)"

# Step 3: Create a security group
echo "Creating security group: $SECURITY_GROUP_NAME"
SG_OUTPUT=$(aws ec2 create-security-group \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for AWS Marketplace tutorial" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to create security group" >&2
    exit 1
}

# Extract security group ID using jq for safe parsing
SECURITY_GROUP_ID=$(extract_json_value "$SG_OUTPUT" '.GroupId') || exit 1
if [ -z "$SECURITY_GROUP_ID" ] || [ "$SECURITY_GROUP_ID" = "null" ]; then
    echo "ERROR: Could not extract security group ID" >&2
    exit 1
fi
echo "Security group created with ID: $SECURITY_GROUP_ID"

# Add inbound rules for SSH and HTTP in parallel for better performance
echo "Configuring security group rules..."
{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 22 \
```

```
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
SSH_PID=$!

{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 80 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
HTTP_PID=$!

# Wait for both operations to complete
wait $SSH_PID || {
    echo "ERROR: Failed to add SSH ingress rule" >&2
    exit 1
}
wait $HTTP_PID || {
    echo "ERROR: Failed to add HTTP ingress rule" >&2
    exit 1
}

echo "Security group configured with SSH and HTTP access."
echo "WARNING: SSH rule allows access from any IP (0.0.0.0/0). Restrict this in
 production."
echo "WARNING: In a production environment, you should restrict access to
 specific IP ranges."
echo ""

# Step 4: Get the latest Amazon Linux 2 AMI ID - Use pagination to optimize costs
echo "Getting the latest Amazon Linux 2 AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-2.0.*-x86_64-gp2"
 "Name=state,Values=available" \
  --region "$AWS_REGION" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text \
  --max-results 50) || {
    echo "ERROR: Failed to describe images" >&2
    exit 1
```

```
}

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    echo "ERROR: Could not find a suitable AMI ID" >&2
    exit 1
fi

echo "Using AMI ID: $AMI_ID"
echo "Note: In a real marketplace scenario, you would use the AMI ID from your
 subscribed product."
echo ""

# Step 5: Launch an EC2 instance with cost optimization
echo "Launching EC2 instance with the AMI..."
echo "Using t2.micro (eligible for AWS Free Tier if applicable)"
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --count 1 \
  --region "$AWS_REGION" \
  --monitoring Enabled=false \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=marketplace-buyer-gs}]' \
  --output json) || {
    echo "ERROR: Failed to launch instance" >&2
    exit 1
}

# Extract instance ID using jq for safe parsing
INSTANCE_ID=$(extract_json_value "$INSTANCE_OUTPUT" '.Instances[0].InstanceId')
 || exit 1
if [ -z "$INSTANCE_ID" ] || [ "$INSTANCE_ID" = "null" ]; then
    echo "ERROR: Could not extract instance ID" >&2
    exit 1
fi
echo "Instance launched with ID: $INSTANCE_ID"

# Wait for the instance to be running
echo "Waiting for instance to be in running state..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" || {
    echo "ERROR: Instance failed to reach running state" >&2
```

```
    exit 1
}
echo "Instance is now running."
echo ""

# Step 6: Get instance details
echo "Getting instance details..."
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to describe instance" >&2
    exit 1
}

echo "Instance details:"
extract_json_value "$INSTANCE_DETAILS" '.Reservations[0].Instances[0] |
 {InstanceId, State: .State.Name, PublicDnsName, PrivateIpAddress, LaunchTime,
 InstanceType}' || exit 1

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Key Pair: $KEY_NAME"
echo "Security Group: $SECURITY_GROUP_NAME (ID: $SECURITY_GROUP_ID)"
echo "EC2 Instance: $INSTANCE_ID"
echo "Instance Type: t2.micro (cost-optimized)"
echo "AMI ID: $AMI_ID"
echo "Region: $AWS_REGION"
echo ""
echo "COST OPTIMIZATION NOTES:"
echo "- t2.micro instances are eligible for AWS Free Tier (750 hours/month for 12
 months)"
echo "- Detailed monitoring is disabled to reduce costs"
echo "- Consider using Spot Instances for non-production workloads"
echo "- Review AWS Pricing Calculator: https://calculator.aws/"
echo ""
echo "To connect to your instance (once it's fully initialized):"
echo "ssh -i ${KEY_NAME}.pem ec2-user@<public-dns-name>"
echo "Replace <public-dns-name> with the PublicDnsName from the instance details
 above."
echo ""
```

```
# Auto-confirm cleanup of resources
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up all created resources..."

echo ""
echo "Script completed. See $LOG_FILE for the complete log."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - RunInstances
  - TerminateInstances
  - Wait

Erste Schritte mit Big-Data-Verarbeitungsclustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}
```

```
# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi
```

```
# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true
```

```
# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
```

```
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi
```

```
unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
```

```
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
```

```
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
```

```
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv
```

```
echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb

- Rb
- Rm
- SSH
- TerminateClusters
- Wait

Erste Schritte mit Dokumentendatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

###############################################################################
# Configuration
###############################################################################
```

```
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

###############################################################################
# Region pre-check
###############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

###############################################################################
# Error handler
###############################################################################
handle_error() {
    echo ""
```

```
    echo "==========================================="
    echo "ERROR at $1"
    echo "==========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

################################################################################
# Wait function
################################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
```

```
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
```

```
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
```

```
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "============================================"
echo "Step 1: Create master password in Secrets Manager"
echo "============================================"
```

```
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "==========================================="
echo "Step 2: Find default VPC and subnets"
echo "==========================================="
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)
```

```
if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
```

```
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "============================================"
echo "Step 3: Create DocumentDB subnet group"
echo "============================================"
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "============================================"
echo "Step 4: Create DocumentDB cluster"
echo "============================================"
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
```

```
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
```

```
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
##############################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""
```

```
# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi
```

```
echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "==========================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "==========================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

################################################################################
# Step 9: Display connection information
################################################################################
echo "==========================================="
echo "CONNECTION INFORMATION"
echo "==========================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
```

```
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

################################################################################
# Step 10: Cleanup
################################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster

- CreateDbInstance
- CreateDbSubnetGroup
- CreateDefaultVpc
- CreateSecret
- DeleteDbCluster
- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteSecret
- DescribeDbClusters
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- RevokeSecurityGroupIngress
- Wait

Erste Schritte mit Graphdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihre Neptune-Datenbank
- Erstellen Sie Subnetze in mehreren Verfügbarkeitszonen
- Konfigurieren Sie die Sicherheit für Ihre Neptune-Datenbank
- Erstellen Sie eine Neptune DB-Subnetzgruppe
- Erstellen Sie einen Neptune-DB-Cluster und eine Neptune-DB-Instanz
- Fügen Sie Daten zu Ihrer Graph-Datenbank hinzu
- Fragen Sie Ihre Graphdatenbank ab

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
```

```
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
```

```
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -
f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"

log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"
```

```
# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2
 | tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3
 | tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a
 "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
```

```
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' |
 cut -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-
cidr-block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID
 --protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
```

```
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a
 "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
```

```
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name
\\\", \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or
 setting up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a
 "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID
 --skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so
 we'll sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a
 "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-
id $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"
```

```
    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later."
 | tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteInternetGateway

- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Erste Schritte mit In-Memory-Caching

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine Sicherheitsgruppe für den ElastiCache Zugriff ein
- Erstellen Sie einen serverlosen Valkey-Cache
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail
```

```
# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
```

```
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
```

```
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from
 any IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs
 2>&1); then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
```

```
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' |
 awk -F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
```

```
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
```

```
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to
 complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
```

```
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Erste Schritte mit dem Lastenausgleich

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Application Load Balancers
- Erstellen einer Zielgruppe
- Erstellen eines Listeners
- Überprüfen Ihrer Konfiguration
- Fügen Sie einen HTTPS-Listener hinzu (optional)
- Fügen Sie pfadbasiertes Routing hinzu (optional)
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}
```

```
# Function to validate ARN format
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null ||
 true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN"
 2>/dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi
```

```
    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/
dev/null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/
null && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30
 seconds... (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID"
 >&2
            echo "You may need to delete it manually using: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"
```

```
# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
```

```
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=elastic-load-
balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict
 to specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo
 "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
```

```
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo
 "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-
name,Values=running" \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null
 || echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
```

```
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
```

```
echo ""

# Prompt for cleanup confirmation
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
read -p "Do you want to clean up all created resources? (y/n): " -r
 CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers

- DescribeSubnets
- DescribeTargetHealth
- DescribeVpcs
- Hilfe
- RegisterTargets
- Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster
```

```
# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
```

```
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
```

```
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
```

```
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
```

```
        # Try multiple times with longer delays to ensure dependencies are
removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
```

```
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
```

```
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
```

```
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
```

```
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"
```

```
    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
```

```
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi
```

```
aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
```

```
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"
```

```
# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi
```

```
echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
```

```
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)
```

```
if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))
```

```
    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
```

```
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
```

```
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF
```

```
# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
```

```
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeCluster
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DescribeSubnets
  - DescribeVpcs
  - DetachRolePolicy
  - GetBootstrapBrokers
  - GetCallerIdentity

  - GetInstanceProfile
  - GetPolicy

- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit Network Transit Gateways

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Transit-Gateways
- Verbinden Sie Ihre VPCs mit Ihrem Transit-Gateway
- Fügen Sie Routen zwischen dem Transit-Gateway und Ihren VPCs hinzu
- Testen Sie das Transit-Gateway
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon VPC Transit Gateway CLI Script
# This script demonstrates how to create a transit gateway and connect two VPCs
# Modified to work with older AWS CLI versions that don't support transit gateway
 wait commands
```

```
# Security improved: Added input validation, error handling, and credential
 checks

# Error handling
set -euo pipefail
LOG_FILE="transit-gateway-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Security: Check for required AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
  echo "ERROR: AWS credentials not configured or invalid. Please configure AWS
 credentials."
  exit 1
fi

# Function to validate AWS CLI output
validate_aws_output() {
  local output=$1
  local context=$2

  if [ -z "$output" ] || [ "$output" = "None" ]; then
    echo "ERROR: Failed to retrieve $context from AWS API"
    return 1
  fi
}

# Function to validate CIDR blocks
validate_cidr() {
  local cidr=$1
  if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
    echo "ERROR: Invalid CIDR block format: $cidr"
    return 1
  fi
}

# Function to wait for transit gateway to be available
wait_for_tgw() {
  local tgw_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway $tgw_id to become available..."

  while [ $attempt -lt $max_attempts ]; do
```

```
    status=$(aws ec2 describe-transit-gateways \
      --transit-gateway-ids "$tgw_id" \
      --query "TransitGateways[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway creation failed"
      return 1
    fi

    echo "Waiting for transit gateway to become available. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway to become available"
  return 1
}

# Function to wait for transit gateway attachment to be available
wait_for_tgw_attachment() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to become
 available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"
```

```
    if [ "$status" = "available" ]; then
      echo "Transit Gateway Attachment is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway Attachment creation failed"
      return 1
    fi

    echo "Waiting for transit gateway attachment to become available. Current
 state: $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway attachment to become
 available"
  return 1
}

# Function to wait for transit gateway attachment to be deleted
wait_for_tgw_attachment_deleted() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to be deleted..."

  while [ $attempt -lt $max_attempts ]; do
    count=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --filters "Name=transit-gateway-attachment-id,Values=$attachment_id" \
      --query "length(TransitGatewayVpcAttachments)" \
      --output text 2>/dev/null || echo "0")

    if [ "$count" = "0" ]; then
      echo "Transit Gateway Attachment has been deleted"
      return 0
    fi

    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "deleted")
```

```
    if [ "$status" = "deleted" ] || [ "$status" = "deleting" ]; then
      echo "Transit Gateway Attachment is being deleted. Current state: $status"
    fi

    echo "Waiting for transit gateway attachment to be deleted. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "WARNING: Timeout waiting for transit gateway attachment to be deleted"
  return 0
}

# Function to clean up resources
cleanup() {
  local exit_code=$?
  echo "Error occurred (exit code: $exit_code). Cleaning up resources..."

  # Delete resources in reverse order
  if [ -n "${TGW_ATTACHMENT_1_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 1: $TGW_ATTACHMENT_1_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
  fi

  if [ -n "${TGW_ATTACHMENT_2_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 2: $TGW_ATTACHMENT_2_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
  fi

  if [ -n "${TGW_ID:-}" ]; then
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" &>/dev/null ||
 true
  fi

  exit "$exit_code"
}
```

```
# Set up trap for error handling
trap cleanup EXIT

echo "=== Amazon VPC Transit Gateway Tutorial ==="
echo "This script will create a transit gateway and connect two VPCs"
echo ""

# Get a valid availability zone dynamically
echo "Getting available AZ in current region..."
AZ=$(aws ec2 describe-availability-zones \
  --query "AvailabilityZones[0].ZoneName" \
  --output text)
validate_aws_output "$AZ" "availability zone" || exit 1
echo "Using availability zone: $AZ"

# Check if VPCs exist
echo "Checking for existing VPCs..."
VPC1_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC1" \
  --query "Vpcs[0].VpcId" \
  --output text)
VPC2_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC2" \
  --query "Vpcs[0].VpcId" \
  --output text)

if [ "$VPC1_ID" = "None" ] || [ -z "$VPC1_ID" ]; then
  echo "Creating VPC1..."
  VPC1_ID=$(aws ec2 create-vpc \
    --cidr-block 10.1.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC1_ID" "VPC1" || exit 1
  echo "Created VPC1: $VPC1_ID"

  # Create a subnet in VPC1
  echo "Creating subnet in VPC1..."
  SUBNET1_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC1_ID" \
    --cidr-block 10.1.0.0/24 \
    --availability-zone "$AZ" \
```

```
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
  echo "Created subnet in VPC1: $SUBNET1_ID"
else
  echo "Using existing VPC1: $VPC1_ID"
  SUBNET1_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC1_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET1_ID" = "None" ] || [ -z "$SUBNET1_ID" ]; then
    echo "Creating subnet in VPC1..."
    SUBNET1_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC1_ID" \
      --cidr-block 10.1.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
    echo "Created subnet in VPC1: $SUBNET1_ID"
  else
    echo "Using existing subnet in VPC1: $SUBNET1_ID"
  fi
fi

if [ "$VPC2_ID" = "None" ] || [ -z "$VPC2_ID" ]; then
  echo "Creating VPC2..."
  VPC2_ID=$(aws ec2 create-vpc \
    --cidr-block 10.2.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC2_ID" "VPC2" || exit 1
  echo "Created VPC2: $VPC2_ID"

  # Create a subnet in VPC2
```

```
  echo "Creating subnet in VPC2..."
  SUBNET2_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC2_ID" \
    --cidr-block 10.2.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
  echo "Created subnet in VPC2: $SUBNET2_ID"
else
  echo "Using existing VPC2: $VPC2_ID"
  SUBNET2_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC2_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET2_ID" = "None" ] || [ -z "$SUBNET2_ID" ]; then
    echo "Creating subnet in VPC2..."
    SUBNET2_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC2_ID" \
      --cidr-block 10.2.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-
Subnet},{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-
gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
    echo "Created subnet in VPC2: $SUBNET2_ID"
  else
    echo "Using existing subnet in VPC2: $SUBNET2_ID"
  fi
fi

# Get route tables for each VPC
RTB1_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC1_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB1_ID" "VPC1 route table" || exit 1

RTB2_ID=$(aws ec2 describe-route-tables \
```

```
  --filters "Name=vpc-id,Values=$VPC2_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB2_ID" "VPC2 route table" || exit 1

echo "Route table for VPC1: $RTB1_ID"
echo "Route table for VPC2: $RTB2_ID"

# Step 1: Create the transit gateway
echo "Creating Transit Gateway..."
TGW_ID=$(aws ec2 create-transit-gateway \
  --description "My Transit Gateway" \
  --options
 "AmazonSideAsn=64512,AutoAcceptSharedAttachments=disable,DefaultRouteTableAssociation=en
 \
  --tag-specifications 'ResourceType=transit-
gateway,Tags=[{Key=Name,Value=MyTransitGateway},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGateway.TransitGatewayId \
  --output text)
validate_aws_output "$TGW_ID" "Transit Gateway" || exit 1
echo "Created Transit Gateway: $TGW_ID"

# Wait for the transit gateway to become available
wait_for_tgw "$TGW_ID" || exit 1

# Step 2: Attach VPCs to the transit gateway
echo "Attaching VPC1 to Transit Gateway..."
TGW_ATTACHMENT_1_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC1_ID" \
  --subnet-ids "$SUBNET1_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC1-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_1_ID" "Transit Gateway VPC1 Attachment" ||
 exit 1
echo "Created Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"

echo "Attaching VPC2 to Transit Gateway..."
TGW_ATTACHMENT_2_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
```

```
  --vpc-id "$VPC2_ID" \
  --subnet-ids "$SUBNET2_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC2-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_2_ID" "Transit Gateway VPC2 Attachment" ||
 exit 1
echo "Created Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"

# Wait for the attachments to become available
wait_for_tgw_attachment "$TGW_ATTACHMENT_1_ID" || exit 1
wait_for_tgw_attachment "$TGW_ATTACHMENT_2_ID" || exit 1

# Step 3: Add routes between the transit gateway and VPCs
echo "Adding route from VPC1 to VPC2 via Transit Gateway..."
validate_cidr "10.2.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB1_ID" \
  --destination-cidr-block 10.2.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Adding route from VPC2 to VPC1 via Transit Gateway..."
validate_cidr "10.1.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB2_ID" \
  --destination-cidr-block 10.1.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Routes added successfully"

# Step 4: Display information for testing
echo ""
echo "=== Transit Gateway Setup Complete ==="
echo "Transit Gateway ID: $TGW_ID"
echo "VPC1 ID: $VPC1_ID"
echo "VPC2 ID: $VPC2_ID"
echo ""
echo "To test connectivity:"
echo "1. Launch an EC2 instance in each VPC"
echo "2. Configure security groups to allow ICMP traffic"
echo "3. Connect to one instance and ping the other instance's private IP"
echo ""
```

```
echo "Viewing created resources..."

echo ""
echo "=== Resources Created ===="
echo "Transit Gateway: $TGW_ID"
echo "VPC1: $VPC1_ID"
echo "VPC2: $VPC2_ID"
echo "Subnet in VPC1: $SUBNET1_ID"
echo "Subnet in VPC2: $SUBNET2_ID"
echo "Transit Gateway Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
echo "Transit Gateway Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
echo ""

echo "Starting cleanup..."

# Delete routes
echo "Deleting routes..."
aws ec2 delete-route --route-table-id "$RTB1_ID" --destination-cidr-block
 10.2.0.0/16 || true
aws ec2 delete-route --route-table-id "$RTB2_ID" --destination-cidr-block
 10.1.0.0/16 || true

# Delete transit gateway attachments
echo "Deleting Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" || true

echo "Deleting Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" || true

# Wait for attachments to be deleted
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true

# Delete transit gateway
echo "Deleting Transit Gateway: $TGW_ID"
aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" || true

echo "Cleanup completed successfully"

echo "Tutorial completed. See $LOG_FILE for detailed logs."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateRoute
  - CreateSubnet
  - CreateTransitGateway
  - CreateTransitGatewayVpcAttachment
  - CreateVpc
  - DeleteRoute
  - DeleteTransitGateway
  - DeleteTransitGatewayVpcAttachment
  - DescribeAvailabilityZones
  - DescribeRouteTables
  - DescribeSubnets
  - DescribeTransitGatewayAttachments
  - DescribeTransitGatewayVpcAttachments
  - DescribeTransitGateways
  - DescribeVpcs

Erste Schritte mit Videotransport-Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Zugriff auf Elemental MediaConnect
- Erstellen Sie einen Flow
- Fügen Sie eine Ausgabe hinzu
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
```

```
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"'
 -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-
zA-Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output"
 == "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
```

```
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi
```

```
AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
```

```
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/
null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1
 | cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
```

```
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/
dev/null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o
 '"EntitlementArn": "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
```

```
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN"
 --query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "==========================================="
echo "RESOURCE SUMMARY"
echo "==========================================="
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddFlowOutputs
  - CreateFlow

- DeleteFlow
- DescribeAvailabilityZones
- DescribeFlow
- GrantFlowEntitlements
- ListFlows
- StopFlow

Erste Schritte mit virtuellen Maschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Schlüsselpaares
- Eine Sicherheitsgruppe erstellen
- Anhalten und Starten Ihrer Instance
- Testen Sie die Elastic IP-Persistenz
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."
```

```
    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
```

```
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
```

```
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial'
 --output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null
```

```
if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value"
 \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
```

```
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]'
 'ResourceType=volume,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
```

```
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
```

```
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
```

```
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your
 instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - DisassociateAddress
  - GetParametersByPath
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - Wait

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Experimentvorlage
- Führen Sie das Experiment durch
- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
```

```
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true
```

```
        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"
```

```
# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
```

```
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
```

```
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
```

```
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
```

```
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"
```

```
    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
```

```
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
```

```
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.

- AddRoleToInstanceProfile
- AttachRolePolicy
- CreateExperimentTemplate
- CreateInstanceProfile
- CreateRole
- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Erste Schritte mit der IP-Adressverwaltung über die CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Amazon VPC IP Address Manager (IPAM) mit der CLI ein und konfigurieren Sie ihn.

- Erstellen Sie einen IPAM mit Betriebsregionen (z. B. us-east-1, us-west-2).
- Rufen Sie die private Bereichs-ID für den IPAM ab.
- Erstellen Sie eine hierarchische Struktur von IPv4-Pools (Pools der obersten Ebene, regionale Pools und Entwicklungspools).
- Stellen Sie CIDR-Blöcke für jeden Pool bereit (z. B. 10.0.0. 0/8, 10.0.0. 0/16, 10.0.0. 0/24).
- Erstellen Sie eine VPC mit einem CIDR, das aus einem IPAM-Pool zugewiesen wird.
- Überprüfen Sie die IPAM-Poolzuweisungen und die VPC-Erstellung.
- Beheben Sie häufig auftretende Probleme wie Berechtigungsfehler, Fehler bei der CIDR-Zuweisung und Verstöße gegen Abhängigkeiten.
- Bereinigen Sie die IPAM-Ressourcen (VPC, Pools, CIDRs und IPAM), um unnötige Gebühren zu vermeiden.
- Erkunden Sie die nächsten Schritte für erweiterte IPAM-Features.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# IPAM Getting Started CLI Script - Version 7
# This script creates an IPAM, creates a hierarchy of IP address pools, and
 allocates a CIDR to a VPC
# Fixed to correctly identify the private scope ID, wait for resources to be
 available, add locale to development pool,
# use the correct parameter names for VPC creation, and wait for CIDR
 provisioning to complete

# Set up logging
LOG_FILE="ipam_script.log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting IPAM setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED:"
    echo "============================================"

    if [ -n "$VPC_ID" ]; then
        echo "VPC: $VPC_ID"
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Development Pool: $DEV_POOL_ID"
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Regional Pool: $REGIONAL_POOL_ID"
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Top-level Pool: $TOP_POOL_ID"
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "IPAM: $IPAM_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
```

```
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."

        # Delete resources in reverse order of creation to handle dependencies

        if [ -n "$VPC_ID" ]; then
            echo "Deleting VPC: $VPC_ID"
            aws ec2 delete-vpc --vpc-id "$VPC_ID" || echo "Failed to delete VPC"
            echo "Waiting for VPC to be deleted..."
            sleep 10
        fi

        if [ -n "$DEV_POOL_ID" ]; then
            echo "Deleting Development Pool: $DEV_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$DEV_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Development Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$DEV_POOL_ID"
 --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$DEV_POOL_ID" || echo
 "Failed to delete Development Pool"
            echo "Waiting for Development Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$REGIONAL_POOL_ID" ]; then
            echo "Deleting Regional Pool: $REGIONAL_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id
 "$REGIONAL_POOL_ID" --query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Regional Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id
 "$REGIONAL_POOL_ID" --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
```

```
            aws ec2 delete-ipam-pool --ipam-pool-id "$REGIONAL_POOL_ID" || echo
 "Failed to delete Regional Pool"
            echo "Waiting for Regional Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$TOP_POOL_ID" ]; then
            echo "Deleting Top-level Pool: $TOP_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$TOP_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Top-level Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$TOP_POOL_ID"
 --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$TOP_POOL_ID" || echo
 "Failed to delete Top-level Pool"
            echo "Waiting for Top-level Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$IPAM_ID" ]; then
            echo "Deleting IPAM: $IPAM_ID"
            aws ec2 delete-ipam --ipam-id "$IPAM_ID" || echo "Failed to delete
 IPAM"
        fi

        echo "Cleanup completed."
    else
        echo "Cleanup skipped. Resources will remain in your account."
    fi
}

# Function to wait for a pool to be in the 'create-complete' state
wait_for_pool() {
    local pool_id=$1
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for pool $pool_id to be available..."
```

```
    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 describe-ipam-pools --ipam-pool-ids "$pool_id" --query
 'IpamPools[0].State' --output text)

        if [ "$state" = "create-complete" ]; then
            echo "Pool $pool_id is now available (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: Pool $pool_id is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for pool $pool_id to be available"
    return 1
}

# Function to wait for a CIDR to be fully provisioned
wait_for_cidr_provisioning() {
    local pool_id=$1
    local cidr=$2
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for CIDR $cidr to be fully provisioned in pool $pool_id..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$pool_id" --query
 "IpamPoolCidrs[?Cidr=='$cidr'].State" --output text)

        if [ "$state" = "provisioned" ]; then
            echo "CIDR $cidr is now fully provisioned (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: CIDR $cidr is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done
```

```
    echo "Timed out waiting for CIDR $cidr to be provisioned"
    return 1
}

# Step 1: Create an IPAM
echo "Creating IPAM..."
IPAM_RESULT=$(aws ec2 create-ipam \
    --description "My IPAM" \
    --operating-regions RegionName=us-east-1 RegionName=us-west-2 \
    --tag-specifications 'ResourceType=ipam,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create IPAM"
fi

IPAM_ID=$(echo "$IPAM_RESULT" | grep -o '"IpamId": "[^"]*' | cut -d'"' -f4)
echo "IPAM created with ID: $IPAM_ID"

# Wait for IPAM to be created and available
echo "Waiting for IPAM to be available..."
sleep 20

# Step 2: Get the IPAM Scope ID - FIXED to correctly identify the private scope
echo "Getting IPAM Scope ID..."
SCOPE_RESULT=$(aws ec2 describe-ipams --ipam-id "$IPAM_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to get IPAM details"
fi

# Extract the private scope ID directly from the IPAM details
PRIVATE_SCOPE_ID=$(echo "$SCOPE_RESULT" | grep -o '"PrivateDefaultScopeId":
 "[^"]*' | cut -d'"' -f4)
echo "Private Scope ID: $PRIVATE_SCOPE_ID"

if [ -z "$PRIVATE_SCOPE_ID" ]; then
    handle_error "Failed to get Private Scope ID"
fi

# Step 3: Create a Top-Level IPv4 Pool
echo "Creating Top-level IPv4 Pool..."
TOP_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
```

```
    --address-family ipv4 \
    --description "Top-level pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Top-level Pool"
fi

TOP_POOL_ID=$(echo "$TOP_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -
d'"' -f4)
echo "Top-level Pool created with ID: $TOP_POOL_ID"

# Wait for the top-level pool to be available
if ! wait_for_pool "$TOP_POOL_ID"; then
    handle_error "Top-level Pool did not become available in time"
fi

# Provision CIDR to the top-level pool
echo "Provisioning CIDR to Top-level Pool..."
TOP_POOL_CIDR="10.0.0.0/8"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$TOP_POOL_ID" \
    --cidr "$TOP_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Top-level Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$TOP_POOL_ID" "$TOP_POOL_CIDR"; then
    handle_error "CIDR provisioning to Top-level Pool did not complete in time"
fi

# Step 4: Create a Regional IPv4 Pool
echo "Creating Regional IPv4 Pool..."
REGIONAL_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$TOP_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Regional pool in us-east-1" \
```

```
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Regional Pool"
fi

REGIONAL_POOL_ID=$(echo "$REGIONAL_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*'
 | cut -d'"' -f4)
echo "Regional Pool created with ID: $REGIONAL_POOL_ID"

# Wait for the regional pool to be available
if ! wait_for_pool "$REGIONAL_POOL_ID"; then
    handle_error "Regional Pool did not become available in time"
fi

# Provision CIDR to the regional pool
echo "Provisioning CIDR to Regional Pool..."
REGIONAL_POOL_CIDR="10.0.0.0/16"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$REGIONAL_POOL_ID" \
    --cidr "$REGIONAL_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Regional Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$REGIONAL_POOL_ID" "$REGIONAL_POOL_CIDR"; then
    handle_error "CIDR provisioning to Regional Pool did not complete in time"
fi

# Step 5: Create a Development IPv4 Pool - FIXED to include locale
echo "Creating Development IPv4 Pool..."
DEV_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$REGIONAL_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Development pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create Development Pool"
fi

DEV_POOL_ID=$(echo "$DEV_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -
d'"' -f4)
echo "Development Pool created with ID: $DEV_POOL_ID"

# Wait for the development pool to be available
if ! wait_for_pool "$DEV_POOL_ID"; then
    handle_error "Development Pool did not become available in time"
fi

# Provision CIDR to the development pool
echo "Provisioning CIDR to Development Pool..."
DEV_POOL_CIDR="10.0.0.0/24"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$DEV_POOL_ID" \
    --cidr "$DEV_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Development Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$DEV_POOL_ID" "$DEV_POOL_CIDR"; then
    handle_error "CIDR provisioning to Development Pool did not complete in time"
fi

# Step 6: Create a VPC Using an IPAM Pool CIDR - FIXED to use the correct
 parameter names and a smaller netmask length
echo "Creating VPC using IPAM Pool CIDR..."
VPC_RESULT=$(aws ec2 create-vpc \
    --ipv4-ipam-pool-id "$DEV_POOL_ID" \
    --ipv4-netmask-length 26 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=IPAM-VPC},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create VPC"
fi
```

```
VPC_ID=$(echo "$VPC_RESULT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID"

# Step 7: Verify the IPAM Pool Allocation
echo "Verifying IPAM Pool Allocation..."
ALLOCATION_RESULT=$(aws ec2 get-ipam-pool-allocations \
    --ipam-pool-id "$DEV_POOL_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to verify IPAM Pool Allocation"
fi

echo "IPAM Pool Allocation verified:"
echo "$ALLOCATION_RESULT" | grep -A 5 "Allocations"

echo ""
echo "IPAM setup completed successfully!"
echo ""

# Prompt for cleanup
cleanup_resources

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateIpam
  - CreateIpamPool
  - CreateVpc
  - DeleteIpam
  - DeleteIpamPool
  - DeleteVpc
  - DeprovisionIpamPoolCidr
  - DescribeIpamPools
  - DescribeIpams
  - DescribeVpcs

- GetIpamPoolAllocations
- GetIpamPoolCidrs
- ProvisionIpamPoolCidr

Erstellen Sie mithilfe der CLI ein virtuelles privates Netzwerk mit privaten Subnetzen und Gateways zur Netzwerkadressübersetzung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC mit privaten Subnetzen und NAT-Gateways mit der CLI.
- Richten Sie die erforderlichen Komponenten wie VPC, Subnetze, Routing-Tabellen und NAT-Gateways ein.
- Konfigurieren Sie Sicherheitsgruppen und IAM-Rollen für ordnungsgemäßen Zugriff und Sicherheit.
- Verwenden Sie CLI-Befehle, um die Erstellung und Konfiguration dieser Ressourcen zu automatisieren.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC with Private Subnets and NAT Gateways (IMDSv2 Compliant Version)
# This script creates a VPC with public and private subnets in two Availability
 Zones,
# NAT gateways, an internet gateway, route tables, a VPC endpoint for S3,
# security groups, a launch template, an Auto Scaling group, and an Application
 Load Balancer.
```

```
# Set up logging
LOG_FILE="vpc-private-subnets-nat.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Cleanup function to delete all created resources
cleanup_resources() {
  echo "Cleaning up resources..."

  # Delete Auto Scaling group if it exists
  if [ -n "${ASG_NAME:-}" ]; then
    echo "Deleting Auto Scaling group: $ASG_NAME"
    aws autoscaling delete-auto-scaling-group --auto-scaling-group-name
 "$ASG_NAME" --force-delete
    echo "Waiting for Auto Scaling group to be deleted..."
    aws autoscaling wait auto-scaling-groups-deleted --auto-scaling-group-names
 "$ASG_NAME"
  fi

  # Delete load balancer if it exists
  if [ -n "${LB_ARN:-}" ]; then
    echo "Deleting load balancer: $LB_ARN"
    aws elbv2 delete-load-balancer --load-balancer-arn "$LB_ARN"
    # Wait for load balancer to be deleted
    sleep 30
  fi

  # Delete target group if it exists
  if [ -n "${TARGET_GROUP_ARN:-}" ]; then
    echo "Deleting target group: $TARGET_GROUP_ARN"
    aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN"
  fi

  # Delete launch template if it exists
  if [ -n "${LAUNCH_TEMPLATE_NAME:-}" ]; then
    echo "Deleting launch template: $LAUNCH_TEMPLATE_NAME"
    aws ec2 delete-launch-template --launch-template-name "$LAUNCH_TEMPLATE_NAME"
  fi

  # Delete NAT Gateways if they exist
  if [ -n "${NAT_GW1_ID:-}" ]; then
    echo "Deleting NAT Gateway 1: $NAT_GW1_ID"
    aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW1_ID"
  fi
```

```
  if [ -n "${NAT_GW2_ID:-}" ]; then
    echo "Deleting NAT Gateway 2: $NAT_GW2_ID"
    aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW2_ID"
  fi

  # Wait for NAT Gateways to be deleted
  if [ -n "${NAT_GW1_ID:-}" ] || [ -n "${NAT_GW2_ID:-}" ]; then
    echo "Waiting for NAT Gateways to be deleted..."
    sleep 60
  fi

  # Release Elastic IPs if they exist
  if [ -n "${EIP1_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 1: $EIP1_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP1_ALLOC_ID"
  fi

  if [ -n "${EIP2_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 2: $EIP2_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP2_ALLOC_ID"
  fi

  # Delete VPC endpoint if it exists
  if [ -n "${VPC_ENDPOINT_ID:-}" ]; then
    echo "Deleting VPC endpoint: $VPC_ENDPOINT_ID"
    aws ec2 delete-vpc-endpoints --vpc-endpoint-ids "$VPC_ENDPOINT_ID"
  fi

  # Delete security groups if they exist
  if [ -n "${APP_SG_ID:-}" ]; then
    echo "Deleting application security group: $APP_SG_ID"
    aws ec2 delete-security-group --group-id "$APP_SG_ID"
  fi

  if [ -n "${LB_SG_ID:-}" ]; then
    echo "Deleting load balancer security group: $LB_SG_ID"
    aws ec2 delete-security-group --group-id "$LB_SG_ID"
  fi

  # Detach and delete Internet Gateway if it exists
  if [ -n "${IGW_ID:-}" ] && [ -n "${VPC_ID:-}" ]; then
    echo "Detaching Internet Gateway: $IGW_ID from VPC: $VPC_ID"
    aws ec2 detach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID"
```

```
  echo "Deleting Internet Gateway: $IGW_ID"
  aws ec2 delete-internet-gateway --internet-gateway-id "$IGW_ID"
fi

# Delete route table associations and route tables if they exist
if [ -n "${PUBLIC_RT_ASSOC1_ID:-}" ]; then
  echo "Disassociating public route table from subnet 1: $PUBLIC_RT_ASSOC1_ID"
  aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC1_ID"
fi

if [ -n "${PUBLIC_RT_ASSOC2_ID:-}" ]; then
  echo "Disassociating public route table from subnet 2: $PUBLIC_RT_ASSOC2_ID"
  aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC2_ID"
fi

if [ -n "${PRIVATE_RT1_ASSOC_ID:-}" ]; then
  echo "Disassociating private route table 1: $PRIVATE_RT1_ASSOC_ID"
  aws ec2 disassociate-route-table --association-id "$PRIVATE_RT1_ASSOC_ID"
fi

if [ -n "${PRIVATE_RT2_ASSOC_ID:-}" ]; then
  echo "Disassociating private route table 2: $PRIVATE_RT2_ASSOC_ID"
  aws ec2 disassociate-route-table --association-id "$PRIVATE_RT2_ASSOC_ID"
fi

if [ -n "${PUBLIC_RT_ID:-}" ]; then
  echo "Deleting public route table: $PUBLIC_RT_ID"
  aws ec2 delete-route-table --route-table-id "$PUBLIC_RT_ID"
fi

if [ -n "${PRIVATE_RT1_ID:-}" ]; then
  echo "Deleting private route table 1: $PRIVATE_RT1_ID"
  aws ec2 delete-route-table --route-table-id "$PRIVATE_RT1_ID"
fi

if [ -n "${PRIVATE_RT2_ID:-}" ]; then
  echo "Deleting private route table 2: $PRIVATE_RT2_ID"
  aws ec2 delete-route-table --route-table-id "$PRIVATE_RT2_ID"
fi

# Delete subnets if they exist
if [ -n "${PUBLIC_SUBNET1_ID:-}" ]; then
  echo "Deleting public subnet 1: $PUBLIC_SUBNET1_ID"
  aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET1_ID"
```

```
  fi

  if [ -n "${PUBLIC_SUBNET2_ID:-}" ]; then
    echo "Deleting public subnet 2: $PUBLIC_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET2_ID"
  fi

  if [ -n "${PRIVATE_SUBNET1_ID:-}" ]; then
    echo "Deleting private subnet 1: $PRIVATE_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET1_ID"
  fi

  if [ -n "${PRIVATE_SUBNET2_ID:-}" ]; then
    echo "Deleting private subnet 2: $PRIVATE_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET2_ID"
  fi

  # Delete VPC if it exists
  if [ -n "${VPC_ID:-}" ]; then
    echo "Deleting VPC: $VPC_ID"
    aws ec2 delete-vpc --vpc-id "$VPC_ID"
  fi

  echo "Cleanup completed."
}

# Error handling function
handle_error() {
  echo "ERROR: $1"
  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to check command success
check_command() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"
```

```
# Create VPC
echo "Creating VPC..."
VPC_RESULT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC"

VPC_ID=$(echo "$VPC_RESULT" | jq -r '.Vpc.VpcId')
echo "VPC created with ID: $VPC_ID"

# Get Availability Zones
echo "Getting Availability Zones..."
AZ_RESULT=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0:2].ZoneName' --output text)
check_command "Failed to get Availability Zones"

# Convert space-separated output to array
read -r -a AZS <<< "$AZ_RESULT"
AZ1=${AZS[0]}
AZ2=${AZS[1]}
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create subnets
echo "Creating subnets..."
PUBLIC_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.0.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 1"
PUBLIC_SUBNET1_ID=$(echo "$PUBLIC_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.1.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 1"
PRIVATE_SUBNET1_ID=$(echo "$PRIVATE_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PUBLIC_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.2.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 2"
```

```
PUBLIC_SUBNET2_ID=$(echo "$PUBLIC_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.3.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 2"
PRIVATE_SUBNET2_ID=$(echo "$PRIVATE_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

echo "Subnets created with IDs:"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_RESULT=$(aws ec2 create-internet-gateway --tag-specifications
 "ResourceType=internet-gateway,Tags=[{Key=Name,Value=ProductionIGW-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create Internet Gateway"
IGW_ID=$(echo "$IGW_RESULT" | jq -r '.InternetGateway.InternetGatewayId')
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
 "$VPC_ID"
check_command "Failed to attach Internet Gateway to VPC"

# Create route tables
echo "Creating route tables..."
PUBLIC_RT_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-
servers-gs}]")
check_command "Failed to create public route table"
PUBLIC_RT_ID=$(echo "$PUBLIC_RT_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT1_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private route table 1"
```

```
PRIVATE_RT1_ID=$(echo "$PRIVATE_RT1_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT2_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private route table 2"
PRIVATE_RT2_ID=$(echo "$PRIVATE_RT2_RESULT" | jq -r '.RouteTable.RouteTableId')

echo "Route tables created with IDs:"
echo "Public Route Table: $PUBLIC_RT_ID"
echo "Private Route Table 1: $PRIVATE_RT1_ID"
echo "Private Route Table 2: $PRIVATE_RT2_ID"

# Add route to Internet Gateway in public route table
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT_ID" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID"
check_command "Failed to add route to Internet Gateway"

# Associate subnets with route tables
echo "Associating subnets with route tables..."
PUBLIC_RT_ASSOC1_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET1_ID")
check_command "Failed to associate public subnet 1 with route table"
PUBLIC_RT_ASSOC1_ID=$(echo "$PUBLIC_RT_ASSOC1_RESULT" | jq -r '.AssociationId')

PUBLIC_RT_ASSOC2_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET2_ID")
check_command "Failed to associate public subnet 2 with route table"
PUBLIC_RT_ASSOC2_ID=$(echo "$PUBLIC_RT_ASSOC2_RESULT" | jq -r '.AssociationId')

PRIVATE_RT1_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT1_ID" --subnet-id "$PRIVATE_SUBNET1_ID")
check_command "Failed to associate private subnet 1 with route table"
PRIVATE_RT1_ASSOC_ID=$(echo "$PRIVATE_RT1_ASSOC_RESULT" | jq -r '.AssociationId')

PRIVATE_RT2_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT2_ID" --subnet-id "$PRIVATE_SUBNET2_ID")
check_command "Failed to associate private subnet 2 with route table"
PRIVATE_RT2_ASSOC_ID=$(echo "$PRIVATE_RT2_ASSOC_RESULT" | jq -r '.AssociationId')

echo "Route table associations created with IDs:"
echo "Public Subnet 1 Association: $PUBLIC_RT_ASSOC1_ID"
```

```
echo "Public Subnet 2 Association: $PUBLIC_RT_ASSOC2_ID"
echo "Private Subnet 1 Association: $PRIVATE_RT1_ASSOC_ID"
echo "Private Subnet 2 Association: $PRIVATE_RT2_ASSOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateways..."

# Allocate Elastic IPs for NAT Gateways
echo "Allocating Elastic IPs for NAT Gateways..."
EIP1_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 1"
EIP1_ALLOC_ID=$(echo "$EIP1_RESULT" | jq -r '.AllocationId')

EIP2_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 2"
EIP2_ALLOC_ID=$(echo "$EIP2_RESULT" | jq -r '.AllocationId')

echo "Elastic IPs allocated with IDs:"
echo "EIP 1 Allocation ID: $EIP1_ALLOC_ID"
echo "EIP 2 Allocation ID: $EIP2_ALLOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateway in public subnet 1..."
NAT_GW1_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET1_ID" --allocation-id "$EIP1_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 1"
NAT_GW1_ID=$(echo "$NAT_GW1_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "Creating NAT Gateway in public subnet 2..."
NAT_GW2_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET2_ID" --allocation-id "$EIP2_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 2"
NAT_GW2_ID=$(echo "$NAT_GW2_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "NAT Gateways created with IDs:"
echo "NAT Gateway 1: $NAT_GW1_ID"
```

```
echo "NAT Gateway 2: $NAT_GW2_ID"

# Wait for NAT Gateways to be available
echo "Waiting for NAT Gateways to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW1_ID"
check_command "NAT Gateway 1 did not become available"
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW2_ID"
check_command "NAT Gateway 2 did not become available"
echo "NAT Gateways are now available"

# Add routes to NAT Gateways in private route tables
echo "Adding routes to NAT Gateways in private route tables..."
aws ec2 create-route --route-table-id "$PRIVATE_RT1_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW1_ID"
check_command "Failed to add route to NAT Gateway 1"

aws ec2 create-route --route-table-id "$PRIVATE_RT2_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW2_ID"
check_command "Failed to add route to NAT Gateway 2"

# Create VPC Endpoint for S3
echo "Creating VPC Endpoint for S3..."
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
 'PrefixLists[0].PrefixListId' --output text)
check_command "Failed to get S3 prefix list ID"

VPC_ENDPOINT_RESULT=$(aws ec2 create-vpc-endpoint --vpc-id "$VPC_ID" --
service-name "com.amazonaws.$(aws configure get region).s3" --route-table-ids
 "$PRIVATE_RT1_ID" "$PRIVATE_RT2_ID" --tag-specifications "ResourceType=vpc-
endpoint,Tags=[{Key=Name,Value=S3-Endpoint-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC endpoint for S3"
VPC_ENDPOINT_ID=$(echo "$VPC_ENDPOINT_RESULT" | jq -r
 '.VpcEndpoint.VpcEndpointId')
echo "VPC Endpoint created with ID: $VPC_ENDPOINT_ID"

# Create security groups
echo "Creating security groups..."
LB_SG_RESULT=$(aws ec2 create-security-group --group-name "LoadBalancerSG-
$RANDOM_ID" --description "Security group for the load balancer"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
```

```
check_command "Failed to create load balancer security group"
LB_SG_ID=$(echo "$LB_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from anywhere to the load balancer
aws ec2 authorize-security-group-ingress --group-id "$LB_SG_ID" --protocol tcp --
port 80 --cidr 0.0.0.0/0
check_command "Failed to authorize ingress to load balancer security group"

APP_SG_RESULT=$(aws ec2 create-security-group --group-name "AppServerSG-
$RANDOM_ID" --description "Security group for the application servers"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=AppServerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create application server security group"
APP_SG_ID=$(echo "$APP_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from the load balancer security group to the
 application servers
aws ec2 authorize-security-group-ingress --group-id "$APP_SG_ID" --protocol tcp
 --port 80 --source-group "$LB_SG_ID"
check_command "Failed to authorize ingress to application server security group"

echo "Security groups created with IDs:"
echo "Load Balancer Security Group: $LB_SG_ID"
echo "Application Server Security Group: $APP_SG_ID"

# Create a launch template
echo "Creating launch template..."

# Create user data script with IMDSv2 support
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

# Use IMDSv2 with session token
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
AZ=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/
meta-data/placement/availability-zone)
HOSTNAME=$(hostname -f)
```

```
echo "<h1>Hello from $HOSTNAME in $AZ</h1>" > /var/www/html/index.html
EOF

# Encode user data
USER_DATA=$(base64 -w 0 user-data.sh)

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon --filters
 "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2" "Name=state,Values=available" --
query 'sort_by(Images, &CreationDate)[-1].ImageId' --output text)
check_command "Failed to get latest Amazon Linux 2 AMI"
echo "Using AMI: $AMI_ID"

# Create launch template with IMDSv2 required
LAUNCH_TEMPLATE_NAME="AppServerTemplate-$RANDOM_ID"
echo "Creating launch template: $LAUNCH_TEMPLATE_NAME"

aws ec2 create-launch-template \
  --launch-template-name "$LAUNCH_TEMPLATE_NAME" \
  --version-description "Initial version" \
  --tag-specifications "ResourceType=launch-template,Tags=[{Key=Name,Value=
$LAUNCH_TEMPLATE_NAME},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
private-servers-gs}]" \
  --launch-template-data "{
    \"NetworkInterfaces\": [{
      \"DeviceIndex\": 0,
      \"Groups\": [\"$APP_SG_ID\"],
      \"DeleteOnTermination\": true
    }],
    \"ImageId\": \"$AMI_ID\",
    \"InstanceType\": \"t3.micro\",
    \"UserData\": \"$USER_DATA\",
    \"MetadataOptions\": {
      \"HttpTokens\": \"required\",
      \"HttpEndpoint\": \"enabled\"
    },
    \"TagSpecifications\": [{
      \"ResourceType\": \"instance\",
      \"Tags\": [{
        \"Key\": \"Name\",
        \"Value\": \"AppServer-$RANDOM_ID\"
      }]
    }]
```

```
  }"
check_command "Failed to create launch template"

# Create target group
echo "Creating target group..."
TARGET_GROUP_NAME="AppTargetGroup-$RANDOM_ID"
TARGET_GROUP_RESULT=$(aws elbv2 create-target-group \
  --name "$TARGET_GROUP_NAME" \
  --protocol HTTP \
  --port 80 \
  --vpc-id "$VPC_ID" \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path "/" \
  --health-check-port traffic-port \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create target group"
TARGET_GROUP_ARN=$(echo "$TARGET_GROUP_RESULT" | jq -r
 '.TargetGroups[0].TargetGroupArn')
echo "Target group created with ARN: $TARGET_GROUP_ARN"

# Create load balancer
echo "Creating load balancer..."
LB_NAME="AppLoadBalancer-$RANDOM_ID"
LB_RESULT=$(aws elbv2 create-load-balancer \
  --name "$LB_NAME" \
  --subnets "$PUBLIC_SUBNET1_ID" "$PUBLIC_SUBNET2_ID" \
  --security-groups "$LB_SG_ID" \
  --tags "Key=Name,Value=$LB_NAME" Key=project,Value=doc-smith
 Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create load balancer"
LB_ARN=$(echo "$LB_RESULT" | jq -r '.LoadBalancers[0].LoadBalancerArn')
echo "Load balancer created with ARN: $LB_ARN"

# Wait for load balancer to be active
echo "Waiting for load balancer to be active..."
aws elbv2 wait load-balancer-available --load-balancer-arns "$LB_ARN"
check_command "Load balancer did not become available"

# Create listener
echo "Creating listener..."
LISTENER_RESULT=$(aws elbv2 create-listener \
  --load-balancer-arn "$LB_ARN" \
  --protocol HTTP \
```

```
  --port 80 \
  --default-actions "Type=forward,TargetGroupArn=$TARGET_GROUP_ARN")
check_command "Failed to create listener"
LISTENER_ARN=$(echo "$LISTENER_RESULT" | jq -r '.Listeners[0].ListenerArn')
echo "Listener created with ARN: $LISTENER_ARN"

# Create Auto Scaling group
echo "Creating Auto Scaling group..."
ASG_NAME="AppAutoScalingGroup-$RANDOM_ID"
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name "$ASG_NAME" \
  --launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=\$Latest" \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "$PRIVATE_SUBNET1_ID,$PRIVATE_SUBNET2_ID" \
  --target-group-arns "$TARGET_GROUP_ARN" \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags "Key=Name,Value=AppServer-$RANDOM_ID,PropagateAtLaunch=true"
 Key=project,Value=doc-smith,PropagateAtLaunch=true Key=tutorial,Value=vpc-
private-servers-gs,PropagateAtLaunch=true
check_command "Failed to create Auto Scaling group"
echo "Auto Scaling group created with name: $ASG_NAME"

# Get load balancer DNS name
LB_DNS_NAME=$(aws elbv2 describe-load-balancers --load-balancer-arns "$LB_ARN" --
query 'LoadBalancers[0].DNSName' --output text)
check_command "Failed to get load balancer DNS name"

echo ""
echo "============================================"
echo "DEPLOYMENT COMPLETE"
echo "============================================"
echo "VPC ID: $VPC_ID"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID (AZ: $AZ1)"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID (AZ: $AZ1)"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID (AZ: $AZ2)"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID (AZ: $AZ2)"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"
echo "Load Balancer: $LB_NAME"
echo "Auto Scaling Group: $ASG_NAME"
echo ""
```

```
echo "Your application will be available at: http://$LB_DNS_NAME"
echo "It may take a few minutes for the instances to launch and pass health
 checks."
echo ""

# Add health check monitoring
echo "============================================"
echo "MONITORING INSTANCE HEALTH AND LOAD BALANCER"
echo "============================================"
echo "Waiting for instances to launch and pass health checks..."
echo "This may take 3-5 minutes. Checking every 30 seconds..."

# Monitor instance health and load balancer accessibility
MAX_ATTEMPTS=10
ATTEMPT=1
HEALTHY_INSTANCES=0

while [ $ATTEMPT -le $MAX_ATTEMPTS ] && [ $HEALTHY_INSTANCES -lt 2 ]; do
  echo "Check attempt $ATTEMPT of $MAX_ATTEMPTS..."

  # Check Auto Scaling group instances
  echo "Checking Auto Scaling group instances..."
  ASG_INSTANCES=$(aws autoscaling describe-auto-scaling-groups --auto-
scaling-group-names "$ASG_NAME" --query 'AutoScalingGroups[0].Instances[*].
[InstanceId,HealthStatus]' --output json)
  echo "ASG Instances status:"
  echo "$ASG_INSTANCES" | jq -r '.[] | "Instance: \(.[0]), Health: \(.[1])"'

  # Check target group health
  echo "Checking target group health..."
  TARGET_HEALTH=$(aws elbv2 describe-target-health --target-group-arn
 "$TARGET_GROUP_ARN" --output json)
  echo "Target health status:"
  echo "$TARGET_HEALTH" | jq -r '.TargetHealthDescriptions[] | "Instance:
 \(.Target.Id), State: \(.TargetHealth.State), Reason: \(.TargetHealth.Reason //
 "N/A"), Description: \(.TargetHealth.Description // "N/A")"'

  # Count healthy instances
  HEALTHY_INSTANCES=$(echo "$TARGET_HEALTH" | jq -r '[.TargetHealthDescriptions[]
 | select(.TargetHealth.State=="healthy")] | length')
  echo "Number of healthy instances: $HEALTHY_INSTANCES of 2 expected"

  # Check if we have healthy instances
  if [ $HEALTHY_INSTANCES -ge 2 ]; then
```

```
    echo "All instances are healthy!"

    # Test load balancer accessibility
    echo "Testing load balancer accessibility..."
    HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" "http://$LB_DNS_NAME")

    if [ "$HTTP_STATUS" = "200" ]; then
      echo "Load balancer is accessible! HTTP Status: $HTTP_STATUS"
      echo "You can access your application at: http://$LB_DNS_NAME"

      # Try to get the content to verify IMDSv2 is working
      echo "Fetching content to verify IMDSv2 functionality..."
      CONTENT=$(curl -s "http://$LB_DNS_NAME")
      echo "Response from server:"
      echo "$CONTENT"

      # Check if the content contains the expected pattern
      if [[ "$CONTENT" == *"Hello from"* && "$CONTENT" == *"in"* ]]; then
        echo "IMDSv2 is working correctly! The instance was able to access
 metadata using the token-based approach."
      else
        echo "Warning: Content doesn't match expected pattern. IMDSv2
 functionality could not be verified."
      fi

      break
    else
      echo "Load balancer returned HTTP status: $HTTP_STATUS"
      echo "Will try again in 30 seconds..."
    fi
  else
    echo "Waiting for instances to become healthy..."
    echo "Will check again in 30 seconds..."
  fi

  ATTEMPT=$((ATTEMPT+1))

  if [ $ATTEMPT -le $MAX_ATTEMPTS ]; then
    sleep 30
  fi
done

if [ $HEALTHY_INSTANCES -lt 2 ]; then
  echo "Warning: Not all instances are healthy after maximum attempts."
```

```
  echo "You may need to wait longer or check for configuration issues."
fi

echo "To test your application, run:"
echo "curl http://$LB_DNS_NAME"
echo ""
echo "=========================================="
echo "CLEANUP CONFIRMATION"
echo "=========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  cleanup_resources
  echo "All resources have been deleted."
else
  echo "Resources will not be deleted. You can manually delete them later."
  echo "To delete resources, run this script again and choose to clean up."
fi
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateLaunchTemplate
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - CreateVpcEndpoint
  - DeleteAutoScalingGroup

- DeleteInternetGateway
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteNatGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteTargetGroup
- DeleteVpc
- DeleteVpcEndpoints
- DescribeAvailabilityZones
- DescribeImages
- DescribePrefixLists
- DetachInternetGateway
- ReleaseAddress

Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Amazon EBS-Verschlüsselung standardmäßig aktivieren
- Erstellen Sie einen EBS-Snapshot
- Erstellen und initialisieren Sie ein Volume aus einem Snapshot
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script for EBS operations: encryption, snapshots, and volume initialization
# This script demonstrates:
# 1. Enabling EBS encryption by default
# 2. Creating an EBS snapshot
# 3. Creating a volume from a snapshot
# Cost optimizations:
# - Reduced volume size from 1 GiB to 100 MiB for testing
# - Changed volume type to gp3 with cost-optimized IOPS/throughput
# - Added early cleanup to minimize storage duration
# - Removed unnecessary API calls for KMS key retrieval

set -euo pipefail

# Security: Restrict file permissions for log files
umask 0077

# Setup logging with secure permissions
LOG_FILE="ebs-operations-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS operations script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Function to check command status
check_status() {
    local exit_code=$?
    if [ $exit_code -ne 0 ]; then
        echo "ERROR: $1 failed with exit code $exit_code. Exiting."
        cleanup_resources
```

```
        exit 1
    fi
}

# Function to cleanup resources with retry logic
cleanup_resources() {
    echo "Attempting to clean up resources..."
    local retry_count=0
    local max_retries=3

    if [ -n "${NEW_VOLUME_ID:-}" ]; then
        echo "Deleting new volume $NEW_VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$NEW_VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted new volume $NEW_VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $NEW_VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete new volume $NEW_VOLUME_ID
 after $max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${VOLUME_ID:-}" ]; then
        echo "Deleting original volume $VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted original volume $VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $VOLUME_ID..."
                    sleep 2
                else
```

```
                    echo "WARNING: Could not delete original volume $VOLUME_ID
 after $max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${SNAPSHOT_ID:-}" ]; then
        echo "Deleting snapshot $SNAPSHOT_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-snapshot --snapshot-id "$SNAPSHOT_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted snapshot $SNAPSHOT_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $SNAPSHOT_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete snapshot $SNAPSHOT_ID after
 $max_retries attempts"
                fi
            fi
        done
    fi

    if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
        echo "Restoring original encryption setting..."
        if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
            aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION" 2>/
dev/null || echo "WARNING: Could not restore encryption setting"
        fi
    fi

    echo "Cleanup completed."
}

# Set trap for cleanup on exit
trap cleanup_resources EXIT

# Track created resources
VOLUME_ID=""
NEW_VOLUME_ID=""
```

```
SNAPSHOT_ID=""
ENCRYPTION_MODIFIED=false
ORIGINAL_ENCRYPTION=""

# Input validation function
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS credentials are not properly configured"
        exit 1
    fi
}

validate_aws_cli

# Security: Validate AWS region format
validate_region() {
    local region="$1"
    if [[ ! "$region" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
        echo "ERROR: Invalid AWS region format: $region"
        exit 1
    fi
}

# Get the current AWS region
AWS_REGION="${AWS_REGION:-$(aws configure get region)}"
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config. Using default: $AWS_REGION"
fi

validate_region "$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Security: Validate volume ID format before use
validate_volume_id() {
    local volume_id="$1"
    if [[ ! "$volume_id" =~ ^vol-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid volume ID format: $volume_id"
```

```
        exit 1
    fi
}

# Security: Validate snapshot ID format before use
validate_snapshot_id() {
    local snapshot_id="$1"
    if [[ ! "$snapshot_id" =~ ^snap-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid snapshot ID format: $snapshot_id"
        exit 1
    fi
}

# Get availability zones in the region with caching
AVAILABILITY_ZONE=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --
query 'AvailabilityZones[0].ZoneName' --output text)
check_status "Getting availability zone"

# Security: Validate AZ format
if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}[a-z]$ ]]; then
    echo "ERROR: Invalid availability zone format: $AVAILABILITY_ZONE"
    exit 1
fi
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 1: Check and enable EBS encryption by default
echo "Step 1: Checking current EBS encryption by default setting..."
ORIGINAL_ENCRYPTION=$(aws ec2 get-ebs-encryption-by-default --region
 "$AWS_REGION" --query 'EbsEncryptionByDefault' --output text)
check_status "Checking encryption status"
echo "Current encryption by default setting: $ORIGINAL_ENCRYPTION"

if [ "$ORIGINAL_ENCRYPTION" = "False" ]; then
    echo "Enabling EBS encryption by default..."
    aws ec2 enable-ebs-encryption-by-default --region "$AWS_REGION"
    check_status "Enabling encryption by default"
    ENCRYPTION_MODIFIED=true
    echo "Updated encryption by default setting: True"
else
    echo "EBS encryption by default is already enabled."
fi

# Step 2: Create a test volume for snapshot with minimal size for cost
 optimization
```

```
echo "Step 2: Creating a test volume (1 GiB for testing)..."
VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --availability-zone
 "$AVAILABILITY_ZONE" --size 1 --volume-type gp3 --iops 3000 --throughput 125
 --tag-specifications 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-
volume},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating test volume"

# Security: Validate volume ID
validate_volume_id "$VOLUME_ID"
echo "Created test volume: $VOLUME_ID"

# Wait for volume to become available with timeout
echo "Waiting for volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$VOLUME_ID" || {
    echo "ERROR: Volume did not become available within timeout"
    exit 1
}
check_status "Waiting for volume"

# Step 3: Create a snapshot of the volume
echo "Step 3: Creating snapshot of the volume..."
SNAPSHOT_ID=$(aws ec2 create-snapshot --region "$AWS_REGION" --volume-id
 "$VOLUME_ID" --description "Snapshot for EBS tutorial - $(date +%Y-%m-%d)" --
tag-specifications 'ResourceType=snapshot,Tags=[{Key=Name,Value=ebs-tutorial-
snapshot},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'SnapshotId' --output
 text)
check_status "Creating snapshot"

# Security: Validate snapshot ID
validate_snapshot_id "$SNAPSHOT_ID"
echo "Created snapshot: $SNAPSHOT_ID"

# Wait for snapshot to complete with progress indication and timeout
echo "Waiting for snapshot to complete (this may take several minutes)..."
timeout 1800 aws ec2 wait snapshot-completed --region "$AWS_REGION" --snapshot-
ids "$SNAPSHOT_ID" || {
    echo "ERROR: Snapshot did not complete within timeout"
    exit 1
}
check_status "Waiting for snapshot"
echo "Snapshot completed."
```

```
# Step 4: Create a new volume from the snapshot
echo "Step 4: Creating a new volume from the snapshot..."
NEW_VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --
snapshot-id "$SNAPSHOT_ID" --availability-zone "$AVAILABILITY_ZONE"
 --volume-type gp3 --iops 3000 --throughput 125 --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-volume-from-snapshot},
{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating new volume from snapshot"

# Security: Validate new volume ID
validate_volume_id "$NEW_VOLUME_ID"
echo "Created new volume from snapshot: $NEW_VOLUME_ID"

# Wait for new volume to become available with timeout
echo "Waiting for new volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$NEW_VOLUME_ID" || {
    echo "ERROR: New volume did not become available within timeout"
    exit 1
}
check_status "Waiting for new volume"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Original Volume: $VOLUME_ID"
echo "Snapshot: $SNAPSHOT_ID"
echo "New Volume: $NEW_VOLUME_ID"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup process to minimize storage costs..."

# Delete the new volume immediately to reduce storage duration
echo "Deleting new volume $NEW_VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$NEW_VOLUME_ID"
```

```
check_status "Deleting new volume"

# Delete the original volume
echo "Deleting original volume $VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$VOLUME_ID"
check_status "Deleting original volume"

# Delete the snapshot
echo "Deleting snapshot $SNAPSHOT_ID..."
aws ec2 delete-snapshot --region "$AWS_REGION" --snapshot-id "$SNAPSHOT_ID"
check_status "Deleting snapshot"

# Restore original encryption setting if modified
if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
    echo "Restoring original encryption setting..."
    if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
        aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION"
        check_status "Disabling encryption by default"
    fi
fi

echo "Cleanup completed successfully."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateSnapshot
  - CreateVolume
  - DeleteSnapshot
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeVolumes
  - DetachVolume
  - DisableEbsEncryptionByDefault
  - EnableEbsEncryptionByDefault
  - GetEbsDefaultKmsKeyId
  - GetEbsEncryptionByDefault

- Wait

Ich arbeite mit Netzwerk-Peering-Verbindungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie VPCs für Peering
- Erstellen einer VPC-Peering-Verbindung
- Aktualisieren von Routing-Tabellen
- Überprüfen Sie die VPC-Peering-Verbindung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Peering Connection Script - Version 6 (Security Enhanced)
# This script establishes a VPC peering connection between two VPCs,
# creates subnets if needed, and configures the necessary route tables.
# It will use existing VPCs if available, or create new ones if needed.

set -euo pipefail

# Security: Set strict umask
umask 0077

# Initialize log file with restricted permissions
LOG_FILE="./vpc-peering-script-v6.log"
touch "$LOG_FILE"
chmod 0600 "$LOG_FILE"
```

```
echo "Starting VPC Peering script at $(date)" > "$LOG_FILE"

# Configuration
declare -r AWS_REGION="${AWS_REGION:-us-east-1}"
declare -r MAX_RETRIES=3
declare -r RETRY_DELAY=5

# Validate script is run from secure location
if [[ "$LOG_FILE" != /* ]] && [[ "$LOG_FILE" != ./* ]]; then
    echo "ERROR: Log file path must be absolute or relative starting with ./" >&2
    exit 1
fi

# Function to sanitize variable for safe command execution
sanitize_var() {
    local var="$1"
    if [[ ! "$var" =~ ^[a-zA-Z0-9_/.-]+$ ]]; then
        echo "ERROR: Invalid characters in variable: $var" | tee -a "$LOG_FILE"
        return 1
    fi
    echo "$var"
    return 0
}

# Function to escape string for safe use in commands
escape_string() {
    local string="$1"
    printf '%s\n' "$string" | sed -e 's/[\/&]/\\&/g'
}

# Function to log commands and their output securely
log_cmd() {
    local cmd="$1"

    # Validate command doesn't contain suspicious patterns
    if [[ "$cmd" =~ (\$\(|\`|;.*rm|;.*mv|;.*cp) ]]; then
        echo "ERROR: Suspicious command pattern detected" | tee -a "$LOG_FILE"
        return 1
    fi

    echo "$(date): COMMAND: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}
```

```
# Function to check for errors
check_error() {
    local exit_code="$1"
    local error_msg="${2:-Command failed}"
    if [ "$exit_code" -ne 0 ]; then
        echo "ERROR: $error_msg (exit code: $exit_code)" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        cleanup_on_error
        exit "$exit_code"
    fi
}

# Function to validate AWS CLI is available and configured
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Check AWS CLI version
    local aws_version
    aws_version=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2)
    echo "AWS CLI version: $aws_version" >> "$LOG_FILE"

    if ! aws sts get-caller-identity --region "$AWS_REGION" &>/dev/null; then
        echo "ERROR: AWS CLI is not properly configured or credentials are
 invalid" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null)
    if [[ ! "$account_id" =~ ^[0-9]{12}$ ]]; then
        echo "ERROR: Invalid AWS account ID" | tee -a "$LOG_FILE"
        exit 1
    fi
    echo "Authenticated as AWS Account: $account_id" | tee -a "$LOG_FILE"
}

# Function to validate CIDR blocks
validate_cidr() {
```

```
    local cidr="$1"
    if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
        echo "ERROR: Invalid CIDR block format: $cidr" | tee -a "$LOG_FILE"
        return 1
    fi

    # Additional validation for IP octets
    local ip_part="${cidr%/*}"
    local mask_part="${cidr#*/}"

    IFS='.' read -r -a octets <<< "$ip_part"
    for octet in "${octets[@]}"; do
        if (( octet > 255 )); then
            echo "ERROR: Invalid octet value in CIDR: $cidr" | tee -a "$LOG_FILE"
            return 1
        fi
    done

    if (( mask_part > 32 || mask_part < 0 )); then
        echo "ERROR: Invalid CIDR mask value: $mask_part" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    # List created resources
    echo "Resources created:" | tee -a "$LOG_FILE"
    for resource in "${CREATED_RESOURCES[@]:-}"; do
        echo "- $resource" | tee -a "$LOG_FILE"
    done

    # Clean up in reverse order with retry logic
    for ((i=${#CLEANUP_COMMANDS[@]}-1; i>=0; i--)); do
        echo "Executing cleanup: ${CLEANUP_COMMANDS[$i]}" >> "$LOG_FILE"
        local retry_count=0
        while [ $retry_count -lt $MAX_RETRIES ]; do
            if eval "${CLEANUP_COMMANDS[$i]}" 2>&1 >> "$LOG_FILE"; then
                break
```

```
            else
                retry_count=$((retry_count + 1))
                if [ $retry_count -lt $MAX_RETRIES ]; then
                    echo "Cleanup command failed, retrying in ${RETRY_DELAY}s..."
 >> "$LOG_FILE"
                    sleep "$RETRY_DELAY"
                fi
            fi
        done
    done
}

# Array to store created resources and cleanup commands
declare -a CREATED_RESOURCES=()
declare -a CLEANUP_COMMANDS=()

# Trap errors and cleanup
trap cleanup_on_error EXIT

echo "Setting up VPC peering connection..."

# Validate AWS CLI
validate_aws_cli

# Check for existing VPCs
echo "Checking for existing VPCs..."
EXISTING_VPCS=$(aws ec2 describe-vpcs --region "$AWS_REGION" --query 'Vpcs[?
State==`available`].[VpcId,CidrBlock]' --output text 2>/dev/null || echo "")

if [ -z "$EXISTING_VPCS" ]; then
    echo "No existing VPCs found. Creating new VPCs..."
    CREATE_VPCS=true
else
    echo "Found existing VPCs:"
    echo "$EXISTING_VPCS"
    echo ""
    echo "Using existing VPCs..."
    CREATE_VPCS=false
    # Get the first two available VPCs
    VPC1_INFO=$(echo "$EXISTING_VPCS" | head -n 1)
    VPC2_INFO=$(echo "$EXISTING_VPCS" | head -n 2 | tail -n 1)

    if [ -z "$VPC2_INFO" ]; then
        echo "Only one VPC found. Creating a second VPC..."
```

```
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid
 VPC1_CIDR format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        CREATE_VPC2_ONLY=true
    else
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')
        VPC2_ID=$(echo "$VPC2_INFO" | awk '{print $1}')
        VPC2_CIDR=$(echo "$VPC2_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid
 VPC1_CIDR format"
        VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 format"
        VPC2_CIDR=$(sanitize_var "$VPC2_CIDR") || check_error 1 "Invalid
 VPC2_CIDR format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        validate_cidr "$VPC2_CIDR" || check_error 1 "Invalid VPC2 CIDR"
        CREATE_VPC2_ONLY=false
    fi
fi

# Create VPCs if needed
if [ "$CREATE_VPCS" = true ]; then
    echo "Creating VPC1..."
    VPC1_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.1.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC1"
    VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID
 returned"
    VPC1_CIDR="10.1.0.0/16"
```

```
    CREATED_RESOURCES+=("VPC1: $VPC1_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC1_ID'")
    echo "VPC1 created with ID: $VPC1_ID"

    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPCs to be available
    echo "Waiting for VPCs to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids
 '$VPC1_ID' '$VPC2_ID'"
    check_error $? "Timeout waiting for VPCs to become available"

elif [ "$CREATE_VPC2_ONLY" = true ]; then
    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID
 returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPC2 to be available
    echo "Waiting for VPC2 to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids
 '$VPC2_ID'"
```

```
    check_error $? "Timeout waiting for VPC2 to become available"
fi

echo "Using the following VPCs:"
echo "VPC1: $VPC1_ID ($VPC1_CIDR)"
echo "VPC2: $VPC2_ID ($VPC2_CIDR)"

# Verify the VPCs exist and are available
echo "Verifying VPCs..."
log_cmd "aws ec2 describe-vpcs --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID' --query 'Vpcs[*].[VpcId,State,CidrBlock]' --output table"
check_error $? "Failed to verify VPCs"

# Determine subnet CIDR blocks based on VPC CIDR blocks
VPC1_SUBNET_CIDR=$(echo "$VPC1_CIDR" | sed 's/0\.0\/16/1.0\/24/')
VPC2_SUBNET_CIDR=$(echo "$VPC2_CIDR" | sed 's/0\.0\/16/1.0\/24/')

# Sanitize subnet CIDR blocks
VPC1_SUBNET_CIDR=$(sanitize_var "$VPC1_SUBNET_CIDR") || check_error 1 "Invalid
 VPC1_SUBNET_CIDR format"
VPC2_SUBNET_CIDR=$(sanitize_var "$VPC2_SUBNET_CIDR") || check_error 1 "Invalid
 VPC2_SUBNET_CIDR format"

validate_cidr "$VPC1_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC1"
validate_cidr "$VPC2_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC2"

# Create subnets in both VPCs
echo "Creating subnet in VPC1..."
SUBNET1_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC1_ID' --cidr-block '$VPC1_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC1"
SUBNET1_ID=$(sanitize_var "$SUBNET1_ID") || check_error 1 "Invalid SUBNET1_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC1: $SUBNET1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'")
echo "Subnet created in VPC1 with ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"

echo "Creating subnet in VPC2..."
SUBNET2_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC2_ID' --cidr-block '$VPC2_SUBNET_CIDR' --tag-specifications
```

```
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC2"
SUBNET2_ID=$(sanitize_var "$SUBNET2_ID") || check_error 1 "Invalid SUBNET2_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC2: $SUBNET2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'")
echo "Subnet created in VPC2 with ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"

# Create a VPC peering connection
echo "Creating VPC peering connection..."
PEERING_ID=$(log_cmd "aws ec2 create-vpc-peering-connection --region
 '$AWS_REGION' --vpc-id '$VPC1_ID' --peer-vpc-id '$VPC2_ID' --tag-specifications
 'ResourceType=vpc-peering-connection,Tags=[{Key=Name,Value=VPC1-VPC2-Peering},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'VpcPeeringConnection.VpcPeeringConnectionId' --output text")
check_error $? "Failed to create VPC peering connection"
PEERING_ID=$(sanitize_var "$PEERING_ID") || check_error 1 "Invalid PEERING_ID
 returned"
CREATED_RESOURCES+=("VPC Peering Connection: $PEERING_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-vpc-peering-connection --region '$AWS_REGION'
 --vpc-peering-connection-id '$PEERING_ID'")
echo "VPC Peering Connection created with ID: $PEERING_ID"

# Accept the VPC peering connection
echo "Accepting VPC peering connection..."
log_cmd "aws ec2 accept-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'"
check_error $? "Failed to accept VPC peering connection"
echo "VPC Peering Connection accepted"

# Wait for the peering connection to become active
echo "Waiting for peering connection to become active..."
log_cmd "aws ec2 wait vpc-peering-connection-exists --region '$AWS_REGION' --vpc-
peering-connection-ids '$PEERING_ID'"
check_error $? "Timeout waiting for peering connection to become active"

# Create a route table for VPC1
echo "Creating route table for VPC1..."
RTB1_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC1_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC1-RouteTable},{Key=project,Value=doc-smith},
```

```
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output
 text")
check_error $? "Failed to create route table for VPC1"
RTB1_ID=$(sanitize_var "$RTB1_ID") || check_error 1 "Invalid RTB1_ID returned"
CREATED_RESOURCES+=("Route Table for VPC1: $RTB1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-
table-id '$RTB1_ID'")
echo "Route table created for VPC1 with ID: $RTB1_ID"

# Create a route from VPC1 to VPC2
echo "Creating route from VPC1 to VPC2..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB1_ID'
 --destination-cidr-block '$VPC2_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC1 to VPC2"
echo "Route created from VPC1 to VPC2"

# Associate the route table with the subnet in VPC1
echo "Associating route table with subnet in VPC1..."
RTB1_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB1_ID' --subnet-id '$SUBNET1_ID' --query 'AssociationId' --
output text")
check_error $? "Failed to associate route table with subnet in VPC1"
RTB1_ASSOC_ID=$(sanitize_var "$RTB1_ASSOC_ID") || check_error 1 "Invalid
 RTB1_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC1: $RTB1_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'")
echo "Route table associated with subnet in VPC1"

# Create a route table for VPC2
echo "Creating route table for VPC2..."
RTB2_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC2_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC2-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output
 text")
check_error $? "Failed to create route table for VPC2"
RTB2_ID=$(sanitize_var "$RTB2_ID") || check_error 1 "Invalid RTB2_ID returned"
CREATED_RESOURCES+=("Route Table for VPC2: $RTB2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-
table-id '$RTB2_ID'")
echo "Route table created for VPC2 with ID: $RTB2_ID"

# Create a route from VPC2 to VPC1
```

```
echo "Creating route from VPC2 to VPC1..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB2_ID'
 --destination-cidr-block '$VPC1_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC2 to VPC1"
echo "Route created from VPC2 to VPC1"

# Associate the route table with the subnet in VPC2
echo "Associating route table with subnet in VPC2..."
RTB2_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB2_ID' --subnet-id '$SUBNET2_ID' --query 'AssociationId' --
output text")
check_error $? "Failed to associate route table with subnet in VPC2"
RTB2_ASSOC_ID=$(sanitize_var "$RTB2_ASSOC_ID") || check_error 1 "Invalid
 RTB2_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC2: $RTB2_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'")
echo "Route table associated with subnet in VPC2"

# Verify the VPC peering connection
echo "Verifying VPC peering connection..."
log_cmd "aws ec2 describe-vpc-peering-connections --region '$AWS_REGION' --
vpc-peering-connection-ids '$PEERING_ID' --query 'VpcPeeringConnections[0].
[VpcPeeringConnectionId,Status.Code,AccepterVpcInfo.VpcId,RequesterVpcInfo.VpcId]'
 --output table"
check_error $? "Failed to verify VPC peering connection"
echo "VPC peering connection verified"

# Display summary of created resources
echo ""
echo "================================================"
echo "SUMMARY OF CREATED RESOURCES"
echo "================================================"
echo "VPC1 ID: $VPC1_ID"
echo "VPC1 CIDR: $VPC1_CIDR"
echo "Subnet1 ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"
echo "VPC2 ID: $VPC2_ID"
echo "VPC2 CIDR: $VPC2_CIDR"
echo "Subnet2 ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"
echo "Peering Connection ID: $PEERING_ID"
echo "Route Table 1 ID: $RTB1_ID"
echo "Route Table 1 Association ID: $RTB1_ASSOC_ID"
echo "Route Table 2 ID: $RTB2_ID"
echo "Route Table 2 Association ID: $RTB2_ASSOC_ID"
```

```
echo ""
echo "Created resources:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "================================================"
echo ""

# Test connectivity (optional)
echo "================================================"
echo "CONNECTIVITY TEST"
echo "================================================"
echo "To test connectivity between VPCs, you would need to:"
echo "1. Launch EC2 instances in each subnet"
echo "2. Configure security groups to allow traffic"
echo "3. Test ping or other network connectivity"
echo ""

# Automatic cleanup
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Auto-confirming cleanup of all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Clean up in reverse order
    echo "Disassociating route table from subnet in VPC2..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'" || true

    echo "Disassociating route table from subnet in VPC1..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'" || true

    echo "Deleting route table for VPC2..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB2_ID'" || true

    echo "Deleting route table for VPC1..."
```

```
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB1_ID'" || true

    echo "Deleting VPC peering connection..."
    log_cmd "aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'" || true

    echo "Deleting subnet in VPC2..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'" || true

    echo "Deleting subnet in VPC1..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'" || true

    # Delete VPCs if they were created by this script
    if [ "$CREATE_VPCS" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'"
 || true

        echo "Deleting VPC1..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC1_ID'"
 || true
    elif [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'"
 || true
    fi

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo ""
    echo "To manually clean up later, you can delete resources in this order:"
    echo "1. Route table associations"
    echo "2. Route tables"
    echo "3. VPC peering connection"
    echo "4. Subnets"
    if [ "$CREATE_VPCS" = true ] || [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "5. VPCs (if created by this script)"
    fi
fi
```

```
echo "Script execution completed. See $LOG_FILE for detailed logs."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AcceptVpcPeeringConnection
  - AssociateRouteTable
  - CreateRoute
  - CreateRouteTable
  - CreateSubnet
  - CreateVpc
  - CreateVpcPeeringConnection
  - DeleteRouteTable
  - DeleteSubnet
  - DeleteVpc
  - DeleteVpcPeeringConnection
  - DescribeVpcPeeringConnections
  - DescribeVpcs
  - DisassociateRouteTable
  - Wait

Codebeispiele für Amazon ECR mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic Container Registry mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-ECR-Benutzerhandbuch – Weitere Informationen über Amazon ECR.
- API-Referenz für Amazon ECR – Details zu allen verfügbaren Amazon-ECR-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon ECR AWS SDKs
  - Hello Amazon ECR
  - Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK
  - Aktionen für Amazon ECR mit AWS SDKs
    - Verwenden Sie mit einem CreateRepository AWS SDK oder CLI
    - Verwenden Sie DeleteRepositorymit einem AWS SDK oder CLI
    - Verwenden Sie DescribeImagesmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeRepositoriesmit einem AWS SDK oder CLI
    - Verwenden Sie GetAuthorizationTokenmit einem AWS SDK oder CLI
    - Verwenden Sie GetRepositoryPolicymit einem AWS SDK oder CLI
    - Verwenden Sie ListImagesmit einem AWS SDK oder CLI
    - Verwenden Sie PushImageCmdmit einem AWS SDK
    - Verwenden Sie PutLifeCyclePolicymit einem AWS SDK oder CLI
    - Verwenden Sie SetRepositoryPolicymit einem AWS SDK oder CLI
    - Verwenden Sie StartLifecyclePolicyPreviewmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon ECR AWS SDKs
  - Erste Schritte mit Container-Registern

Grundlegende Beispiele für die Verwendung von Amazon ECR AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic Container Registry mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon ECR
- Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK
- Aktionen für Amazon ECR mit AWS SDKs
  - Verwenden Sie mit einem CreateRepository AWS SDK oder CLI
  - Verwenden Sie DeleteRepositorymit einem AWS SDK oder CLI
  - Verwenden Sie DescribeImagesmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeRepositoriesmit einem AWS SDK oder CLI
  - Verwenden Sie GetAuthorizationTokenmit einem AWS SDK oder CLI
  - Verwenden Sie GetRepositoryPolicymit einem AWS SDK oder CLI
  - Verwenden Sie ListImagesmit einem AWS SDK oder CLI
  - Verwenden Sie PushImageCmdmit einem AWS SDK
  - Verwenden Sie PutLifeCyclePolicymit einem AWS SDK oder CLI
  - Verwenden Sie SetRepositoryPolicymit einem AWS SDK oder CLI
  - Verwenden Sie StartLifecyclePolicyPreviewmit einem AWS SDK oder CLI

Hello Amazon ECR

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon ECR beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {
```

```
    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
```

```
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
```

```
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS -SDK für Python (Boto3).

Lernen Sie die Grundlagen von Amazon ECR mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Amazon-ECR-Repository.
- Legen Sie Repository-Richtlinien fest.
- Rufen Sie Repository-URIs ab.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.
- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.

- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 Without a local image,
```

```
 * this Java program will not successfully run. For more information including
 how to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions
 to access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed
 Docker container registry
             service provided by AWS. It allows developers and organizations to
 securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
 a set of methods to
```

```
            programmatically interact with the Amazon ECR service. This allows
developers to
            automate the storage, retrieval, and management of container images
as part of their application
            deployment pipelines. With ECR, teams can focus on building and
deploying their
            applications without having to worry about the underlying
infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for
this service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }

       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println("""
          1. Create an ECR repository.
```

```
            The first task is to ensure we have a local Docker image named echo-
text.
            If this image exists, then an Amazon ECR repository is created.

            An ECR repository is a private Docker container repository provided
            by Amazon Web Services (AWS). It is a managed service that makes it
 easy
            to store, manage, and deploy Docker container images.\s
            """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
 repository: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
 is crucial for maintaining
        the security and integrity of your container images. The repository
 policy allows you to
```

```
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
```

```
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images,
ensuring that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
```

```
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message
indicating that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker
image to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the
 image in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS
 CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=
%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
```

```
        or use the force option (used in this scenario) to delete the repository
and have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " +
e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
          }
```

```
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
```

```
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
     * @throws IllegalArgumentException     If repository name is invalid.
     * @throws RuntimeException             if an error occurs while creating the
 repository.
     */
    public String createECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
 empty");
        }

        CreateRepositoryRequest request = CreateRepositoryRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<CreateRepositoryResponse> response =
 getAsyncClient().createRepository(request);
        try {
            CreateRepositoryResponse result = response.join();
            if (result != null) {
                System.out.println("The " + repoName + " repository was created
 successfully.");
                return result.repository().repositoryArn();
            } else {
                throw new RuntimeException("Unexpected response type");
```

```
            }
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof EcrException ex) {
                if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                    System.out.println("The Amazon ECR repository already exists,
moving on...");
                    DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                        .repositoryNames(repoName)
                        .build();
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return
describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
```

```
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
```

```
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
         It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
```

```
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
 getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
 successfully.");
            } else {
```

```
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                } else {
```

```
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     * @throws EcrException        if there is an error retrieving the repository
 information.
     * @throws CompletionException if the asynchronous operation completes
 exceptionally.
     */
    public void getRepositoryURI(String repoName) {
        DescribeRepositoriesRequest request =
 DescribeRepositoriesRequest.builder()
            .repositoryNames(repoName)
            .build();

        CompletableFuture<DescribeRepositoriesResponse> response =
 getAsyncClient().describeRepositories(request);
        response.whenComplete((describeRepositoriesResponse, ex) -> {
            if (ex != null) {
                Throwable cause = ex.getCause();
                if (cause instanceof InterruptedException) {
                    Thread.currentThread().interrupt();
                    String errorMessage = "Thread interrupted while waiting for
 asynchronous operation: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
```

```
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the
console.
     * If an exception occurs, the method handles the exception and prints the
error message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
```

```
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
```

```
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
     * @throws EcrException                      if there is an unexpected error
setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
            }
```

```
            """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
```

```
                String token =
 response.authorizationData().get(0).authorizationToken();
                String decodedToken = new
 String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

```
    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy

- SetRepositoryPolicy
- StartLifecyclePolicyPreview

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a
 local image,
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
```

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
        service provided by AWS. It allows developers and organizations to
 securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
        from building and testing to production deployment.

        The `EcrClient` service client that is part of the AWS SDK for Kotlin
 provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
 developers to
        automate the storage, retrieval, and management of container images as
 part of their application
```

```
        deployment pipelines. With ECR, teams can focus on building and deploying
 their
        applications without having to worry about the underlying infrastructure
 required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
 service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-
text.
        If this image exists, then an Amazon ECR repository is created.
```

```
        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function
is crucial for maintaining
        the security and integrity of your container images. The repository
policy allows you to
        define specific rules and restrictions for accessing and managing the
images stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
```

```
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is
responsible for obtaining a
        valid authorization token, which is required to authenticate your
requests to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to
deploy a container image to
```

```
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime
to pull the
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker
images from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by
making a call to the AWS SDK.
```

```
        The method uses the authorization token to create an `AuthConfig` object,
 which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
 Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
 repository using the Docker client.
        If the push operation is successful, the method prints a message
 indicating that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon
 ECR by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:

           aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

        3. Run the Docker container and view the output using this command:
```

```
            docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository
 and have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
```

```
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =

 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
            dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactor
        } else {
            dockerClient = DockerClientBuilder.getInstance().build()
        }
```

```
        return dockerClient
    }


    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
policy.
     */
    suspend fun setLifeCyclePolicy(repoName: String): String? {
        val polText =
            """
             {
             "rules": [
                 {
                     "rulePriority": 1,
                     "description": "Expire images older than 14 days",
                     "selection": {
                         "tagStatus": "any",
                         "countType": "sinceImagePushed",
                         "countUnit": "days",
                         "countNumber": 14
                     },
                     "action": {
                         "type": "expire"
                     }
                 }
            ]
            }

            """.trimIndent()
        val lifecyclePolicyPreviewRequest =
            StartLifecyclePolicyPreviewRequest {
                lifecyclePolicyText = polText
                repositoryName = repoName
            }

        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
```

```
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
```

```
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
```

```
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
```

```
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse =
ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
```

```
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }


    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
```

```
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
 empty")
```

```
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
            val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations
 using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
```

```
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
from building and testing to production deployment.

The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of
 their application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required
 to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
```

```
        print(f"Creating a repository named {self.repository_name}")
        self.repository =
 self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for
 this repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
 crucial for maintaining
the security and integrity of your container images. The repository policy allows
 you to
```

```
define specific rules and restrictions for accessing and managing the images
 stored within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )
```

```
        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image
 URI,
which includes the ECR repository URI. This allows the container runtime to pull
 the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored
 in your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
```

```
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
        )
```

```
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you
 need to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS
 --password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )

        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
 resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
 '{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
```

```
            or q.ask(
                f"Would you like to delete the local Docker image
'{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
```

```
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
```

```
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

ECRWrapper-Klasse, die Amazon ECR-Aktionen umschließt.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
```

```
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
```

```
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
```

```
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

```
    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
```

```
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Aktionen für Amazon ECR mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon ECR-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon ECR-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon ECR AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Elastic Container Registry.

Beispiele

- Verwenden Sie mit einem CreateRepository AWS SDK oder CLI
- Verwenden Sie DeleteRepositorymit einem AWS SDK oder CLI
- Verwenden Sie DescribeImagesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeRepositoriesmit einem AWS SDK oder CLI
- Verwenden Sie GetAuthorizationTokenmit einem AWS SDK oder CLI
- Verwenden Sie GetRepositoryPolicymit einem AWS SDK oder CLI
- Verwenden Sie ListImagesmit einem AWS SDK oder CLI
- Verwenden Sie PushImageCmdmit einem AWS SDK
- Verwenden Sie PutLifeCyclePolicymit einem AWS SDK oder CLI
- Verwenden Sie SetRepositoryPolicymit einem AWS SDK oder CLI
- Verwenden Sie StartLifecyclePolicyPreviewmit einem AWS SDK oder CLI

Verwenden Sie mit einem **`CreateRepository`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRepository` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Container-Registern

CLI

AWS CLI

Beispiel 1: So erstellen Sie ein Repository

Im folgenden Beispiel für `create-repository` wird ein Repository innerhalb des angegebenen Namespace im Standard-Registry für ein Konto erstellt.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Repositorys im Benutzerhandbuch für Amazon ECR.

Beispiel 2: So erstellen Sie ein Repository, das mit der Unveränderlichkeit von Image-Tags konfiguriert ist

Im folgenden Beispiel für `create-repository` wird im Standard-Registry für ein Konto ein Repository erstellt, das für die Unveränderlichkeit von Tags konfiguriert ist.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-tag-mutability IMMUTABLE
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageTagMutability": "IMMUTABLE"
    }
}
```

Weitere Informationen finden Sie unter Veränderlichkeit von Image-Tags im Benutzerhandbuch für Amazon ECR.

Beispiel 3: So erstellen Sie ein Repository, das mit einer Scan-Konfiguration konfiguriert ist

Im folgenden Beispiel für `create-repository` wird im Standard-Registry für ein Konto ein Repository erstellt, das für die Durchführung eines Scans auf Schwachstellen beim Pushen von Images konfiguriert ist.

```
aws ecr create-repository \
    --repository-name project-a/sample-repo \
    --image-scanning-configuration scanOnPush=true
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "project-a/sample-repo",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
        "imageScanningConfiguration": {
            "scanOnPush": true
        }
    }
}
```

Weitere Informationen finden Sie unter Scannen von Images im Benutzerhandbuch für Amazon ECR.

- Einzelheiten zur API finden Sie CreateRepositoryin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
```

```
    * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
               if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                   System.out.println("The Amazon ECR repository already exists,
moving on...");
                   DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                       .repositoryNames(repoName)
                       .build();
                   DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                   return
describeResponse.repositories().get(0).repositoryArn();
               } else {
                   throw new RuntimeException(ex);
```

```
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateRepositoryin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
```

```
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- Einzelheiten zur API finden Sie CreateRepositoryin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)
```

```
    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie CreateRepositoryin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->createrepository(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_repository_uri) = oo_result->get_repository( )-
>get_repositoryuri( ).
        MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryalrexex.
        " If repository already exists, retrieve it
        DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.
        APPEND NEW /aws1/cl_ecrrepositorynamels00( iv_value =
 iv_repository_name ) TO lt_repo_names.
        DATA(lo_describe_result) = lo_ecr-
>describerepositories( it_repositorynames = lt_repo_names ).
        DATA(lt_repos) = lo_describe_result->get_repositories( ).
        IF lines( lt_repos ) > 0.
          READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
          oo_result = NEW /aws1/cl_ecrcrerepositoryrsp( io_repository =
 lo_repo ).
          MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
        ENDIF.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateRepositoryin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteRepository`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRepository` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Container-Registern

CLI

AWS CLI

So löschen Sie ein Repository

Der folgende `delete-repository`-Beispielbefehl erzwingt das Löschen des angegebenen Repositorys im Standard-Registry für ein Konto. Das `--force`-Flag ist erforderlich, wenn das Repository Images enthält.

```
aws ecr delete-repository \
    --repository-name ubuntu \
    --force
```

Ausgabe:

```
{
    "repository": {
        "registryId": "123456789012",
        "repositoryName": "ubuntu",
        "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
    }
}
```

Weitere Informationen finden Sie unter Löschen eines Repositorys im Benutzerhandbuch für Amazon ECR.

- Einzelheiten zur API finden Sie DeleteRepositoryin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
```

```
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " +
repoName + " repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteRepositoryin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_repository_name = 'my-repository'
    lo_ecr->deleterepository(
      iv_repositoryname = iv_repository_name
      iv_force = abap_true ).
    MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
  CATCH /aws1/cx_ecrrepositorynotfndex.
    MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteRepositoryin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeImages`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeImages` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie ein Image in einem Repository

Im folgenden `describe-images` Beispiel werden Details zu einem Bild im `cluster-autoscaler` Repository mit dem Tag angezeigt`v1.13.6`.

```
aws ecr describe-images \
    --repository-name cluster-autoscaler \
    --image-ids imageTag=v1.13.6
```

Ausgabe:

```
{
    "imageDetails": [
        {
            "registryId": "012345678910",
            "repositoryName": "cluster-autoscaler",
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTags": [
                "v1.13.6"
            ],
            "imageSizeInBytes": 48318255,
            "imagePushedAt": 1565128275.0
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeImagesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    */
   suspend fun verifyImage(
       repoName: String?,
       imageTagVal: String?,
   ) {
       require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
       require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

       val imageId =
           ImageIdentifier {
               imageTag = imageTagVal
           }
       val request =
           DescribeImagesRequest {
               repositoryName = repoName
               imageIds = listOf(imageId)
           }
```

```
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)
```

```
    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images
for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeImagesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       " iv_repository_name = 'my-repository'
       " it_image_ids = VALUE #( ( NEW /aws1/cl_ecrimageidentifier( iv_imagetag
= 'latest' ) ) )
       IF it_image_ids IS NOT INITIAL.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name
           it_imageids = it_image_ids ).
       ELSE.
         oo_result = lo_ecr->describeimages(
           iv_repositoryname = iv_repository_name ).
       ENDIF.
       DATA(lt_image_details) = oo_result->get_imagedetails( ).
       MESSAGE |Found { lines( lt_image_details ) } images in repository.| TYPE
'I'.
     CATCH /aws1/cx_ecrrepositorynotfndex.
       MESSAGE 'Repository not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrimagenotfoundex.
       MESSAGE 'Image not found.' TYPE 'I'.
     CATCH /aws1/cx_ecrinvalidparameterex.
       MESSAGE 'Invalid parameter provided.' TYPE 'I'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeImagesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeRepositories`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRepositories` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie die Repositorys in einem Registry

In diesem Beispiel werden die Repositorys im Standard-Registry für ein Konto beschrieben.

Befehl:

```
aws ecr describe-repositories
```

Ausgabe:

```
{
    "repositories": [
        {
            "registryId": "012345678910",
            "repositoryName": "ubuntu",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
        },
        {
            "registryId": "012345678910",
            "repositoryName": "test",
            "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " +
cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
```

```
                }
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
```

```
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
 ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
 describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
```

```
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
 aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " it_repository_names = VALUE #( ( NEW /aws1/
cl_ecrrepositorynamels00( iv_value = 'my-repository' ) ) )
        oo_result = lo_ecr->describerepositories(
          it_repositorynames = it_repository_names ).
        DATA(lt_repositories) = oo_result->get_repositories( ).
        MESSAGE |Found { lines( lt_repositories ) } repositories.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeRepositoriesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetAuthorizationToken`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAuthorizationToken` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie ein Autorisierungstoken für Ihr Standard-Registry ab

Mit dem folgenden `get-authorization-token`-Beispielbefehl wird ein Autorisierungstoken für Ihr Standard-Registry abgerufen.

```
aws ecr get-authorization-token
```

Ausgabe:

```
{
    "authorizationData": [
        {
            "authorizationToken": "QVdTOkN...",
```

```
            "expiresAt": 1448875853.241,
            "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
    * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
    * If the operation is successful, the method prints the token to the
console.
    * If an exception occurs, the method handles the exception and prints the
error message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
```

```
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
        response.join();
    }
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
```

```
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.
```

```
        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ecr->getauthorizationtoken( ).
        DATA(lt_auth_data) = oo_result->get_authorizationdata( ).
        IF lines( lt_auth_data ) > 0.
          READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
          DATA(lv_token) = lo_auth_data->get_authorizationtoken( ).
          MESSAGE 'Authorization token retrieved.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ecrserverexception.
        MESSAGE 'Server exception occurred.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetAuthorizationTokenin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetRepositoryPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetRepositoryPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie die Repository-Richtlinie für ein Repository ab

Im folgenden Beispiel für `get-repository-policy` werden Details zur Repository-Richtlinie für das `cluster-autoscaler`-Repository angezeigt.

```
aws ecr get-repository-policy \
    --repository-name cluster-autoscaler
```

Ausgabe:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
```

```
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
```

```
    }
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
```

```
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->getrepositorypolicy(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_policy_text) = oo_result->get_policytext( ).
        MESSAGE 'Repository policy retrieved.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryplynot00.
```

```
        MESSAGE 'Repository policy not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetRepositoryPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListImages`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListImages` verwendet wird.

CLI

AWS CLI

So listen Sie die Images in einem Repository auf

Im folgenden Beispiel für `list-images` wird eine Liste der Images im `cluster-autoscaler`-Repository angezeigt.

```
aws ecr list-images \
    --repository-name cluster-autoscaler
```

Ausgabe:

```
{
    "imageIds": [
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.8"
        },
        {
            "imageDigest":
 "sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
            "imageTag": "v1.13.7"
        },
        {
            "imageDigest":
 "sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
            "imageTag": "v1.13.6"
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListImagesin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListImagesin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **PushImageCmd**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PushImageCmd` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName  the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   public void pushDockerImage(String repoName, String imageName) {
       System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
       CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
           .thenApply(response -> {
               String token =
response.authorizationData().get(0).authorizationToken();
               String decodedToken = new
String(Base64.getDecoder().decode(token));
               String password = decodedToken.substring(4);

               DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
               Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
               assert repoData != null;
               String registryURL = repoData.repositoryUri().split("/")[0];
```

```
                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to
 ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- Einzelheiten zur API finden Sie PushImageCmdin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
    *
    * @param repoName the name of the ECR repository to push the image to.
    * @param imageName the name of the Docker image.
    */
   suspend fun pushDockerImage(
       repoName: String,
       imageName: String,
   ) {
       println("Pushing $imageName to $repoName will take a few seconds")
       val authConfig = getAuthConfig(repoName)

       EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
           val desRequest =
               DescribeRepositoriesRequest {
                   repositoryNames = listOf(repoName)
               }

           val describeRepoResponse = ecrClient.describeRepositories(desRequest)
           val repoData =
               describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
                   ?: throw RuntimeException("Repository not found: $repoName")

           val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
           if (tagImageCmd != null) {
               tagImageCmd.exec()
           }
           val pushImageCmd =
               repoData.repositoryUri?.let {
                   dockerClient?.pushImageCmd(it)
                       // ?.withTag("latest")
                       ?.withAuthConfig(authConfig)
               }

           try {
               if (pushImageCmd != null) {
                   pushImageCmd.start().awaitCompletion()
               }
```

```
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- Einzelheiten zur API finden Sie PushImageCmdin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutLifeCyclePolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutLifeCyclePolicy` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Lebenszyklusrichtlinie

Im folgenden Beispiel für `put-lifecycle-policy` wird eine Lebenszyklusrichtlinie für das angegebene Repository im Standard-Registry für ein Konto erstellt.

```
aws ecr put-lifecycle-policy \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Inhalt von `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
               "countNumber": 14
           },
           "action": {
```

```
                "type": "expire"
            }
        }
    ]
}
```

Ausgabe:

```
{
   "registryId": "<aws_account_id>",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\"rules\":[{\"rulePriority\":1,\"description\":
\"Expire images older than 14 days\",\"selection\":{\"tagStatus\":\"untagged\",
\"countType\":\"sinceImagePushed\",\"countUnit\":\"days\",\"countNumber\":14},
\"action\":{\"type\":\"expire\"}}]}"
}
```

Weitere Informationen finden Sie unter Lebenszyklusrichtlinien im Benutzerhandbuch für Amazon ECR.

- Einzelheiten zur API finden Sie PutLifeCyclePolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.
```

```
        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Beispiel, das eine Richtlinie für ein Ablaufdatum festlegt.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
```

```
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Einzelheiten zur API finden Sie PutLifeCyclePolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
        lo_ecr->putlifecyclepolicy(
          iv_repositoryname = iv_repository_name
          iv_lifecyclepolicytext = iv_lifecycle_policy_text ).
        MESSAGE |Lifecycle policy set for repository { iv_repository_name }.|
 TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_ecrvalidationex.
        MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutLifeCyclePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SetRepositoryPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetRepositoryPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So legen Sie die Repository-Richtlinie für ein Repository fest

Im folgenden Beispiel für `set-repository-policy` wird eine in einer Datei enthaltene Repository-Richtlinie an das `cluster-autoscaler`-Repository angehängt.

```
aws ecr set-repository-policy \
    --repository-name cluster-autoscaler \
    --policy-text file://my-policy.json
```

Inhalt von `my-policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement" : [
        {
            "Sid" : "allow public pull",
            "Effect" : "Allow",
            "Principal" : "*",
            "Action" : [
                "ecr:BatchCheckLayerAvailability",
```

```
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "registryId": "012345678910",
    "repositoryName": "cluster-autoscaler",
    "policyText": "{\n  \"Version\" : \"2008-10-17\",\n  \"Statement\" :
 [ {\n    \"Sid\" : \"allow public pull\",\n    \"Effect\" : \"Allow\",\n
 \"Principal\" : \"*\",\n    \"Action\" : [ \"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\" ]\n  } ]\n}"
}
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole  the IAM role to be granted access to the repository.
     * @throws RepositoryPolicyNotFoundException if the repository policy does
 not exist.
     * @throws EcrException                      if there is an unexpected error
 setting the repository policy.
     */
    public void setRepoPolicy(String repoName, String iamRole) {
```

```
        /*
          This example policy document grants the specified AWS principal the
permission to perform the
          `ecr:BatchGetImage` action. This policy is designed to allow the
specified principal
          to retrieve Docker images from the ECR repository.
         */
        String policyDocumentTemplate = """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "%s"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }
             """;

        String policyDocument = String.format(policyDocumentTemplate, iamRole);
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " +
cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
```

```
            }
        });
        response.join();
    }
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
    val policyDocumentTemplate =
        """
         {
          "Version":"2012-10-17",
          "Statement" : [ {
            "Sid" : "new statement",
            "Effect" : "Allow",
            "Principal" : {
              "AWS" : "$iamRole"
            },
            "Action" : "ecr:BatchGetImage"
```

```
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
 ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.
```

```
        :return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Beispiel, das einer IAM-Rolle Download-Zugriff gewährt.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR
repository.
```

```
        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_policy_text = '{"Version":"2012-10-17",       "Statement":[...]}'
        lo_ecr->setrepositorypolicy(
          iv_repositoryname = iv_repository_name
          iv_policytext = iv_policy_text ).
        MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
```

```
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SetRepositoryPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartLifecyclePolicyPreview`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartLifecyclePolicyPreview` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie eine Lebenszyklus-Richtlinienvorschau

Im folgenden Beispiel für `start-lifecycle-policy-preview` wird eine Lebenszyklus-Richtlinienvorschau, die durch eine JSON-Datei definiert ist, für das angegebene Repository erstellt.

```
aws ecr start-lifecycle-policy-preview \
    --repository-name "project-a/amazon-ecs-sample" \
    --lifecycle-policy-text "file://policy.json"
```

Inhalt von `policy.json`:

```
{
   "rules": [
       {
           "rulePriority": 1,
           "description": "Expire images older than 14 days",
           "selection": {
               "tagStatus": "untagged",
               "countType": "sinceImagePushed",
               "countUnit": "days",
```

```
                "countNumber": 14
            },
            "action": {
                "type": "expire"
            }
        }
    ]
}
```

Ausgabe:

```
{
   "registryId": "012345678910",
   "repositoryName": "project-a/amazon-ecs-sample",
   "lifecyclePolicyText": "{\n    \"rules\": [\n        {\n
 \"rulePriority\": 1,\n            \"description\": \"Expire images older than 14
 days\",\n            \"selection\": {\n                \"tagStatus\": \"untagged
\",\n                \"countType\": \"sinceImagePushed\",\n
 \"countUnit\": \"days\",\n                \"countNumber\": 14\n            },\n
           \"action\": {\n                \"type\": \"expire\"\n            }\n
      }\n    ]\n}\n",
   "status": "IN_PROGRESS"
}
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
```

```
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name
 cannot be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
 cannot be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
```

```
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- Einzelheiten zur API finden Sie StartLifecyclePolicyPreviewin der API-Referenz zum AWS SDK für Kotlin.

Szenarien für die Verwendung von Amazon ECR AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon ECR mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon ECR oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Container-Registern

Erste Schritte mit Container-Registern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Docker-Image
- Erstellen Sie ein Amazon ECR-Repository
- Löschen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
```

```
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "====================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "====================================================="
```

```
    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "==================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
```

```
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
```

```
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
```

```
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
```

```
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository

- GetCallerIdentity
- GetLoginPassword

Codebeispiele für Amazon ECS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic Container Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-ECS-Entwicklerhandbuch – Weitere Informationen zu Amazon ECS.
- API-Referenz für Amazon ECS – Details zu allen verfügbaren Amazon-ECS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon ECS AWS SDKs
  - Hello Amazon ECS
  - Aktionen für Amazon ECS mit AWS SDKs
    - Verwenden Sie CreateClustermit einem AWS SDK oder CLI
    - Verwenden Sie CreateServicemit einem AWS SDK oder CLI
    - Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
    - Verwenden Sie DeleteServicemit einem AWS SDK oder CLI
    - Verwenden Sie mit einem DescribeClusters AWS SDK oder CLI
    - Verwendung von DescribeServices mit einer CLI
    - Verwenden Sie mit einem DescribeTasks AWS SDK oder CLI

    - Verwenden Sie ListClustersmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem ListServices AWS SDK oder CLI
    - Verwenden Sie mit einem ListTasks AWS SDK oder CLI
    - Verwendung von UpdateClusterSettings mit einer CLI
    - Verwenden Sie mit einem UpdateService AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon ECS AWS SDKs
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
  - Einen Container-Service für Instanzen virtueller Maschinen erstellen
  - Rufen Sie ARN-Informationen für Amazon ECS-Cluster, -Services und -Tasks mit einem ab AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon ECS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic Container Service mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon ECS
- Aktionen für Amazon ECS mit AWS SDKs
  - Verwenden Sie CreateClustermit einem AWS SDK oder CLI
  - Verwenden Sie CreateServicemit einem AWS SDK oder CLI
  - Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
  - Verwenden Sie DeleteServicemit einem AWS SDK oder CLI
  - Verwenden Sie mit einem DescribeClusters AWS SDK oder CLI
  - Verwendung von DescribeServices mit einer CLI
  - Verwenden Sie mit einem DescribeTasks AWS SDK oder CLI
  - Verwenden Sie ListClustersmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem ListServices AWS SDK oder CLI
  - Verwenden Sie mit einem ListTasks AWS SDK oder CLI
  - Verwendung von UpdateClusterSettings mit einer CLI
  - Verwenden Sie mit einem UpdateService AWS SDK oder CLI

Hello Amazon ECS

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon ECS beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace ECSActions;

/// <summary>
/// A class that introduces the Amazon ECS Client by listing the
/// cluster ARNs for the account.
/// </summary>
public class HelloECS
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon ECS client.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonECS>()
            )
            .Build();

        var amazonECSClient = host.Services.GetRequiredService<IAmazonECS>();

        Console.WriteLine($"Hello Amazon ECS! Following are some cluster ARNS
 available in the your account");
        Console.WriteLine();

        var clusters = new List<string>();

        var clustersPaginator = amazonECSClient.Paginators.ListClusters(new
 ListClustersRequest());

        await foreach (var response in clustersPaginator.Responses)
        {
            clusters.AddRange(response.ClusterArns);
        }

        if (clusters.Count > 0)
        {
            clusters.ForEach(cluster =>
            {
                Console.WriteLine($"\tARN: {cluster}");
                Console.WriteLine($"Cluster Name: {cluster.Split("/").Last()}");
                Console.WriteLine();
            });
        }
        else
        {
            Console.WriteLine("No clusters were found.");
        }

    }
}
```

- Einzelheiten zur API finden Sie ListClustersin der AWS SDK für .NET API-Referenz.

Aktionen für Amazon ECS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon ECS-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-ECS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon ECS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Elastic Container Service.

Beispiele

- Verwenden Sie CreateClustermit einem AWS SDK oder CLI
- Verwenden Sie CreateServicemit einem AWS SDK oder CLI
- Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
- Verwenden Sie DeleteServicemit einem AWS SDK oder CLI
- Verwenden Sie mit einem DescribeClusters AWS SDK oder CLI
- Verwendung von DescribeServices mit einer CLI
- Verwenden Sie mit einem DescribeTasks AWS SDK oder CLI
- Verwenden Sie ListClustersmit einem AWS SDK oder CLI
- Verwenden Sie mit einem ListServices AWS SDK oder CLI
- Verwenden Sie mit einem ListTasks AWS SDK oder CLI
- Verwendung von UpdateClusterSettings mit einer CLI
- Verwenden Sie mit einem UpdateService AWS SDK oder CLI

Verwenden Sie **`CreateCluster`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCluster` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen neuen Cluster

Im folgenden `create-cluster` Beispiel wird ein Cluster mit dem Namen CloudWatch Container Insights mit verbesserter Beobachtbarkeit erstellt `MyCluster` und aktiviert.

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --settings name=containerInsights,value=enhanced
```

Ausgabe:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enhanced"
            }
        ],
        "tags": []
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Clusters im Entwicklerhandbuch für Amazon ECS.

Beispiel 2: So erstellen Sie einen neuen Cluster mithilfe von Kapazitätsanbietern

Im folgenden Beispiel für `create-cluster` wird ein Cluster erstellt und diesem werden zwei bestehende Kapazitätsanbieter zugeordnet. Mit dem `create-capacity-provider`-Befehl erstellen Sie einen Kapazitätsanbieter. Die Angabe einer Standardstrategie für Kapazitätsanbieter ist optional, wird jedoch empfohlen. In diesem Beispiel erstellen wir einen Cluster mit dem Namen `MyCluster` und ordnen ihm die Kapazitätsanbieter `MyCapacityProvider1` und `MyCapacityProvider2` zu. Eine Standardstrategie für Kapazitätsanbieter wird festgelegt, bei der die Aufgaben gleichmäßig auf beide Kapazitätsanbieter verteilt werden.

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --capacity-providers MyCapacityProvider1 MyCapacityProvider2 \
    --default-capacity-provider-
strategy capacityProvider=MyCapacityProvider1,weight=1 capacityProvider=MyCapacityProvide
```

Ausgabe:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enabled"
            }
        ],
        "capacityProviders": [
            "MyCapacityProvider1",
            "MyCapacityProvider2"
        ],
        "defaultCapacityProviderStrategy": [
            {
                "capacityProvider": "MyCapacityProvider1",
                "weight": 1,
                "base": 0
```

```
            },
            {
                "capacityProvider": "MyCapacityProvider2",
                "weight": 1,
                "base": 0
            }
        ],
        "attachments": [
           {
               "id": "0fb0c8f4-6edd-4de1-9b09-17e470ee1918",
               "type": "asp",
               "status": "PRECREATED",
               "details": [
                   {
                       "name": "capacityProviderName",
                       "value": "MyCapacityProvider1"
                   },
                   {
                       "name": "scalingPlanName",
                       "value": "ECSManagedAutoScalingPlan-a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111"
                   }
               ]
            },
            {
                "id": "ae592060-2382-4663-9476-b015c685593c",
                "type": "asp",
                "status": "PRECREATED",
                "details": [
                    {
                        "name": "capacityProviderName",
                        "value": "MyCapacityProvider2"
                    },
                    {
                        "name": "scalingPlanName",
                        "value": "ECSManagedAutoScalingPlan-a1b2c3d4-5678-90ab-
cdef-EXAMPLE22222"
                    }
                ]
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
```

```
}
```

Weitere Informationen finden Sie unter Cluster-Kapazitätsanbieter im Entwicklerhandbuch für Amazon ECS.

Beispiel 3: So erstellen Sie einen neuen Cluster mit mehreren Tags

Im folgenden Beispiel für `create-cluster` wird ein Cluster mit mehreren Tags erstellt. Weitere Informationen zum Hinzufügen von Tags mithilfe der Kurzsyntax finden Sie unter Verwenden der Kurzsyntax mit der AWS Befehlszeilenschnittstelle im AWS CLI-Benutzerhandbuch.

```
aws ecs create-cluster \
    --cluster-name MyCluster \
    --tags key=key1,value=value1 key=key2,value=value2
```

Ausgabe:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [
            {
                "key": "key1",
                "value": "value1"
            },
            {
                "key": "key2",
                "value": "value2"
            }
        ]
     }
 }
```

Weitere Informationen finden Sie unter Erstellen eines Clusters im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie in der Befehlsreferenz CreateCluster.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <clusterName>\s

                Where:
                   clusterName - The name of the ECS cluster to create.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String
clusterName) {
        try {
            ExecuteCommandConfiguration commandConfiguration =
ExecuteCommandConfiguration.builder()
                    .logging(ExecuteCommandLogging.DEFAULT)
                    .build();

            ClusterConfiguration clusterConfiguration =
ClusterConfiguration.builder()
                    .executeCommandConfiguration(commandConfiguration)
                    .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                    .clusterName(clusterName)
                    .configuration(clusterConfiguration)
                    .build();

            CreateClusterResponse response =
ecsClient.createCluster(clusterRequest);
            return response.cluster().clusterArn();

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet erstellt einen Amazon-ECS-Cluster.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet erstellt einen Amazon-ECS-Cluster.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),
 aws_sdk_ecs::Error> {
    let cluster = client.create_cluster().cluster_name(name).send().await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
```

```
}
```

- Einzelheiten zur API finden Sie CreateClusterin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`CreateService`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateService` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Service mit einer Fargate-Aufgabe

Das folgende Beispiel für `create-service` zeigt, wie Sie einen Service mithilfe einer Fargate-Aufgabe erstellen.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name MyService \
    --task-definition sample-fargate:1 \
    --desired-count 2 \
    --launch-type FARGATE \
    --platform-version LATEST \
    --network-configuration
 'awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPubli
 \
    --tags key=key1,value=value1 key=key2,value=value2 key=key3,value=value3
```

Ausgabe:

```
{
    "service": {
```

```
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/
MyService",
        "serviceName": "MyService",
          "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
sample-fargate:1",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/1234567890123456789",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/sample-fargate:1",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": 1557119253.821,
                "updatedAt": 1557119253.821,
                "launchType": "FARGATE",
                "platformVersion": "1.3.0",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                }
            }
        ],
```

```
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": 1557119253.821,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "schedulingStrategy": "REPLICA",
        "tags": [
            {
                "key": "key1",
                "value": "value1"
            },
            {
                "key": "key2",
                "value": "value2"
            },
            {
                "key": "key3",
                "value": "value3"
            }
        ],
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Service im Entwicklerhandbuch für Amazon ECS.

Beispiel 2: So erstellen Sie einen Service mit dem EC2-Starttyp

Das folgende Beispiel für `create-service` zeigt, wie Sie einen Service mit dem Namen `ecs-simple-service` mithilfe einer Aufgabe erstellen, die den EC2-Starttyp verwendet. Der Service verwendet die `sleep360`-Aufgabendefinition und behält 1 Instanziierung der Aufgabe bei.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name ecs-simple-service \
    --task-definition sleep360:2 \
    --desired-count 1
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/ecs-
simple-service",
        "serviceName": "ecs-simple-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
sleep360:2",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/1234567890123456789",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/sleep360:2",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "createdAt": 1557206498.798,
```

```
                "updatedAt": 1557206498.798,
                "launchType": "EC2"
            }
        ],
        "events": [],
        "createdAt": 1557206498.798,
        "placementConstraints": [],
        "placementStrategy": [],
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Service im Entwicklerhandbuch für Amazon ECS.

Beispiel 3: So erstellen Sie einen Service, der einen externen Bereitstellungs-Controller verwendet

Im folgenden Beispiel für `create-service` wird ein Service erstellt, der einen externen Bereitstellungs-Controller verwendet.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name MyService \
    --deployment-controller type=EXTERNAL \
    --desired-count 1
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/
MyService",
        "serviceName": "MyService",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
```

```
        "pendingCount": 0,
        "launchType": "EC2",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "taskSets": [],
        "deployments": [],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": 1557128207.101,
        "placementConstraints": [],
        "placementStrategy": [],
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "EXTERNAL"
        },
        "enableECSManagedTags": false,
        "propagateTags": "NONE"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Service im Entwicklerhandbuch für Amazon ECS.

Beispiel 4: So erstellen Sie einen neuen Service hinter einem Load Balancer

Das folgende Beispiel für `create-service` zeigt, wie Sie einen Service erstellen, der sich hinter einem Load Balancer befindet. Sie müssen einen Load Balancer in derselben Region wie Ihre Container-Instance konfiguriert haben. In diesem Beispiel werden die Option `--cli-input-json` und eine JSON-Eingabedatei mit dem Namen `ecs-simple-service-elb.json` und dem folgenden Inhalt verwendet.

```
aws ecs create-service \
    --cluster MyCluster \
    --service-name ecs-simple-service-elb \
    --cli-input-json file://ecs-simple-service-elb.json
```

Inhalt von `ecs-simple-service-elb.json`:

```
 {
```

```
    "serviceName": "ecs-simple-service-elb",
    "taskDefinition": "ecs-demo",
    "loadBalancers": [
        {
            "loadBalancerName": "EC2Contai-EcsElast-123456789012",
            "containerName": "simple-demo",
            "containerPort": 80
        }
    ],
    "desiredCount": 10,
    "role": "ecsServiceRole"
}
```

Ausgabe:

```
{
    "service": {
        "status": "ACTIVE",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
ecs-demo:1",
        "pendingCount": 0,
        "loadBalancers": [
            {
                "containerName": "ecs-demo",
                "containerPort": 80,
                "loadBalancerName": "EC2Contai-EcsElast-123456789012"
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/ecsServiceRole",
        "desiredCount": 10,
        "serviceName": "ecs-simple-service-elb",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/ecs-simple-
service-elb",
        "deployments": [
            {
                "status": "PRIMARY",
                "pendingCount": 0,
                "createdAt": 1428100239.123,
                "desiredCount": 10,
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/ecs-demo:1",
                "updatedAt": 1428100239.123,
```

```
                "id": "ecs-svc/1234567890123456789",
                "runningCount": 0
            }
        ],
        "events": [],
        "runningCount": 0
    }
}
```

Weitere Informationen finden Sie unter Verwenden von Load Balancing zur Verteilung des Amazon-ECS-Servicedatenverkehrs im Entwicklerhandbuch für Amazon ECS.

Beispiel 5: So konfigurieren Sie Amazon-EBS-Volumes bei der Serviceerstellung

Das folgende Beispiel für `create-service` zeigt, wie Amazon-EBS-Volumes für jede Aufgabe konfiguriert werden, die vom Service verwaltet wird. Sie müssen über eine Amazon-ECS-Infrastrukturrolle verfügen, die mit der angehängten verwalteten Richtlinie `AmazonECSInfrastructureRolePolicyForVolumes` konfiguriert ist. Sie müssen eine Aufgabendefinition mit demselben Volume-Namen wie in der `create-service`-Anforderung angeben. In diesem Beispiel werden die Option `--cli-input-json` und eine JSON-Eingabedatei mit dem Namen `ecs-simple-service-ebs.json` und dem folgenden Inhalt verwendet.

```
aws ecs create-service \
    --cli-input-json file://ecs-simple-service-ebs.json
```

Inhalt von `ecs-simple-service-ebs.json`:

```
{
    "cluster": "mycluster",
    "taskDefinition": "mytaskdef",
    "serviceName": "ecs-simple-service-ebs",
    "desiredCount": 2,
    "launchType": "FARGATE",
    "networkConfiguration":{
        "awsvpcConfiguration":{
            "assignPublicIp": "ENABLED",
            "securityGroups": ["sg-12344321"],
            "subnets":["subnet-12344321"]
        }
    },
```

```
    "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
                "roleArn":"arn:aws:iam::123456789012:role/ecsInfrastructureRole",
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
   ]
}
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/ecs-
simple-service-ebs",
        "serviceName": "ecs-simple-service-ebs",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "EC2",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:3",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/7851020056849183687",
```

```
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:3",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-01-21T11:32:38.034000-06:00",
                "updatedAt": "2025-01-21T11:32:38.034000-06:00",
                "launchType": "EC2",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "DISABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7851020056849183687
 in progress.",
                "volumeConfigurations": [
                    {
                        "name": "myEBSVolume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
                        }
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": "2025-01-21T11:32:38.034000-06:00",
```

```
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "DISABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:user/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": false,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "DISABLED"
    }
}
```

Weitere Informationen finden Sie unter Verwenden von Amazon-EBS-Volumes mit Amazon ECS im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie CreateServicein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateService {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <clusterName> <serviceName> <securityGroups>
 <subnets> <taskDefinition>

                                Where:
                                  clusterName - The name of the ECS cluster.
                                  serviceName - The name of the ECS service to
 create.
                                  securityGroups - The name of the security
 group.
                                  subnets - The name of the subnet.
                                  taskDefinition - The name of the task
 definition.
                                """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String clusterName = args[0];
                String serviceName = args[1];
                String securityGroups = args[2];
```

```
                String subnets = args[3];
                String taskDefinition = args[4];
                Region region = Region.US_EAST_1;
                EcsClient ecsClient = EcsClient.builder()
                                .region(region)
                                .build();

                String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
                                taskDefinition);
                System.out.println("The ARN of the service is " + serviceArn);
                ecsClient.close();
        }

        public static String createNewService(EcsClient ecsClient,
                        String clusterName,
                        String serviceName,
                        String securityGroups,
                        String subnets,
                        String taskDefinition) {

                try {
                        AwsVpcConfiguration vpcConfiguration =
AwsVpcConfiguration.builder()
                                        .securityGroups(securityGroups)
                                        .subnets(subnets)
                                        .build();

                        NetworkConfiguration configuration =
NetworkConfiguration.builder()
                                        .awsvpcConfiguration(vpcConfiguration)
                                        .build();

                        CreateServiceRequest serviceRequest =
CreateServiceRequest.builder()
                                        .cluster(clusterName)
                                        .networkConfiguration(configuration)
                                        .desiredCount(1)
                                        .launchType(LaunchType.FARGATE)
                                        .serviceName(serviceName)
                                        .taskDefinition(taskDefinition)
                                        .build();
```

```
                        CreateServiceResponse response =
 ecsClient.createService(serviceRequest);
                        return response.service().serviceArn();

                } catch (EcsException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return "";
        }
}
```

- Einzelheiten zur API finden Sie CreateServicein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Beispielbefehl erstellt in Ihrem Standard-Cluster einen Service namens „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Beispiel 2: Dieser Beispielbefehl erstellt einen Service hinter einem Load Balancer in Ihrem Standardcluster mit dem Namen „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Einzelheiten zur API finden Sie unter CreateService AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Beispielbefehl erstellt in Ihrem Standard-Cluster einen Service namens „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Beispiel 2: Dieser Beispielbefehl erstellt einen Service hinter einem Load Balancer in Ihrem Standardcluster mit dem Namen „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- Einzelheiten zur API finden Sie unter CreateService AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteCluster`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCluster` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

So löschen Sie einen leeren Cluster

Im folgenden Beispiel für `delete-cluster` wird der angegebene leere Cluster gelöscht.

```
aws ecs delete-cluster --cluster MyCluster
```

Ausgabe:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/MyCluster",
        "status": "INACTIVE",
        "clusterName": "MyCluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0
        "statistics": [],
        "tags": []
    }
}
```

Weitere Informationen finden Sie unter Löschen eines Clusters im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie DeleteClusterin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht den angegebenen ECS-Cluster. Sie müssen die Registrierung aller Container-Instances aus diesem Cluster aufheben, bevor Sie ihn löschen können.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteCluster AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht den angegebenen ECS-Cluster. Sie müssen die Registrierung aller Container-Instances aus diesem Cluster aufheben, bevor Sie ihn löschen können.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteCluster AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_cluster(
    client: &aws_sdk_ecs::Client,
    name: &str,
) -> Result<(), aws_sdk_ecs::Error> {
```

```
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteClusterin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`DeleteService`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteService` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

So löschen Sie einen Service

Im folgenden Beispiel für `ecs delete-service` wird der angegebene Service aus einem Cluster gelöscht. Sie können den `--force`-Parameter angeben, um den Service auch dann zu löschen, wenn er nicht auf null Aufgaben herunterskaliert wurde.

```
aws ecs delete-service --cluster MyCluster --service MyService1 --force
```

Weitere Informationen finden Sie unter Löschen eines Service im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie DeleteServicein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterName> <serviceArn>\s

                Where:
                  clusterName - The name of the ECS cluster.
                  serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .build();

            ecsClient.deleteService(serviceRequest);
            System.out.println("The Service was successfully deleted");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteServicein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht den Service mit dem Namen „my-http-service“ im Standard-Cluster. Der Service muss eine gewünschte Anzahl und eine laufende Anzahl von 0 aufweisen, bevor Sie ihn löschen können. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigungsaufforderung zu umgehen, fügen Sie den Schalter -Force hinzu.

```
Remove-ECSService -Service my-http-service
```

Beispiel 2: Löscht den Service mit dem Namen „my-http-service“ im genannten Cluster.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Einzelheiten zur API finden Sie unter DeleteService AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht den Service mit dem Namen „my-http-service“ im Standard-Cluster. Der Service muss eine gewünschte Anzahl und eine laufende Anzahl von 0 aufweisen, bevor Sie ihn löschen können. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigungsaufforderung zu umgehen, fügen Sie den Schalter -Force hinzu.

```
Remove-ECSService -Service my-http-service
```

Beispiel 2: Löscht den Service mit dem Namen „my-http-service“ im genannten Cluster.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- Einzelheiten zur API finden Sie unter DeleteService AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeClusters`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeClusters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen Cluster

Im folgenden Beispiel für `describe-clusters` werden Details zum angegebenen Cluster abgerufen.

```
aws ecs describe-clusters \
    --cluster default
```

Ausgabe:

```
{
    "clusters": [
        {
            "status": "ACTIVE",
            "clusterName": "default",
            "registeredContainerInstancesCount": 0,
            "pendingTasksCount": 0,
            "runningTasksCount": 0,
            "activeServicesCount": 1,
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default"
        }
    ],
    "failures": []
}
```

Weitere Informationen finden Sie unter Amazon ECS-optimiertes Linux-AMI im Entwicklerhandbuch für Amazon ECS.

Beispiel 2: So beschreiben Sie einen Cluster mit der attachment-Option

Im folgenden Beispiel für `describe-clusters` wird die Option ATTACHMENTS spezifiziert. Es werden Details über den angegebenen Cluster und eine Liste von Ressourcen abgerufen, die dem Cluster in Form von Anhängen zugeordnet sind. Wenn Sie einen Kapazitätsanbieter mit einem Cluster verwenden, werden die Ressourcen, entweder AutoScaling Pläne oder Skalierungsrichtlinien, als ASP- oder AS_Policy-ATTACHMENTS dargestellt.

```
aws ecs describe-clusters \
    --include ATTACHMENTS \
    --clusters sampleCluster
```

Ausgabe:

```
{
    "clusters": [
        {
```

```
            "clusterArn": "arn:aws:ecs:af-south-1:123456789222:cluster/
sampleCluster",
            "clusterName": "sampleCluster",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [
                "sampleCapacityProvider"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                {
                    "id": "a1b2c3d4-5678-901b-cdef-EXAMPLE22222",
                    "type": "as_policy",
                    "status": "CREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "sampleCapacityProvider"
                        },
                        {
                            "name": "scalingPolicyName",
                            "value": "ECSManagedAutoScalingPolicy-3048e262-
fe39-4eaf-826d-6f975d303188"
                        }
                    ]
                }
            ],
            "attachmentsStatus": "UPDATE_COMPLETE"
        }
    ],
    "failures": []
}
```

Weitere Informationen finden Sie unter Amazon ECS-optimiertes Linux-AMI im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie DescribeClustersin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> \s

                Where:
                  clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
 DescribeClustersRequest.builder()
                    .clusters(clusterArn)
                    .build();

            DescribeClustersResponse response =
 ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " +
 cluster.clusterName());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet beschreibt einen oder mehrere Ihrer ECS-Cluster.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Ausgabe:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter DescribeClusters AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet beschreibt einen oder mehrere Ihrer ECS-Cluster.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Ausgabe:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter DescribeClusters AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(),
 aws_sdk_ecs::Error> {
    let resp = client.list_clusters().send().await?;

    let cluster_arns = resp.cluster_arns();
    println!("Found {} clusters:", cluster_arns.len());

    let clusters = client
        .describe_clusters()
        .set_clusters(Some(cluster_arns.into()))
        .send()
        .await?;

    for cluster in clusters.clusters() {
        println!("  ARN:  {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeClustersin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DescribeServices`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeServices` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

So beschreiben Sie einen Service

Im folgenden Beispiel für `describe-services` werden Details für den `my-http-service`-Service im Standard-Cluster abgerufen.

```
aws ecs describe-services --services my-http-service
```

Ausgabe:

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:1",
            "pendingCount": 0,
            "loadBalancers": [],
            "desiredCount": 10,
            "createdAt": 1466801808.595,
            "serviceName": "my-http-service",
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/default",
            "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/my-http-
service",
            "deployments": [
                {
                    "status": "PRIMARY",
                    "pendingCount": 0,
                    "createdAt": 1466801808.595,
                    "desiredCount": 10,
                    "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:1",
                    "updatedAt": 1428326312.703,
                    "id": "ecs-svc/1234567890123456789",
                    "runningCount": 10
                }
            ],
            "events": [
                {
                    "message": "(service my-http-service) has reached a steady
 state.",
                    "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
                    "createdAt": 1466801812.435
                }
            ],
            "runningCount": 10
```

```
        }
    ],
    "failures": []
}
```

Weitere Informationen finden Sie unter Services im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie DescribeServicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service aus Ihrem Standard-Cluster abrufen können.

```
Get-ECSService -Service my-hhtp-service
```

Beispiel 2: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service abrufen können, der in dem genannten Cluster ausgeführt wird.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service aus Ihrem Standard-Cluster abrufen können.

```
Get-ECSService -Service my-hhtp-service
```

Beispiel 2: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service abrufen können, der in dem genannten Cluster ausgeführt wird.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeTasks`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTasks` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

Beispiel 1: Um eine einzelne Aufgabe zu beschreiben

Im folgenden Beispiel für `describe-tasks` werden die Details einer Aufgabe in einem Cluster abgerufen. Sie können die Aufgabe angeben, indem Sie entweder die ID oder den vollständigen ARN der Aufgabe verwenden. Dieses Beispiel nutzt den vollständigen ARN der Aufgabe.

```
aws ecs describe-tasks \
    --cluster MyCluster \
    --tasks arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58EXAMPLE
```

Ausgabe:

```
{
    "tasks": [
        {
            "attachments": [],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
```

```
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-08-11T12:21:26.681000-04:00",
            "containerInstanceArn": "arn:aws:ecs:us-
east-1:123456789012:container-instance/test/025c7e2c5e054a6790a29fc1fEXAMPLE",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/4d590253bb114126b7afa7b58eea9221/
a992d1cc-ea46-474a-b6e8-24688EXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58EXAMPLE",
                    "name": "simple-app",
                    "image": "httpd:2.4",
                    "runtimeId":
 "91251eed27db90006ad67b1a08187290869f216557717dd5c39b37c94EXAMPLE",
                    "lastStatus": "RUNNING",
                    "networkBindings": [
                        {
                            "bindIP": "0.0.0.0",
                            "containerPort": 80,
                            "hostPort": 80,
                            "protocol": "tcp"
                        }
                    ],
                    "networkInterfaces": [],
                    "healthStatus": "UNKNOWN",
                    "cpu": "10",
                    "memory": "300"
                }
            ],
            "cpu": "10",
            "createdAt": "2021-08-11T12:21:26.681000-04:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:testupdate",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "EC2",
            "memory": "300",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "simple-app"
                    }
```

```
                ],
                "inferenceAcceleratorOverrides": []
            },
            "pullStartedAt": "2021-08-11T12:21:28.234000-04:00",
            "pullStoppedAt": "2021-08-11T12:21:33.793000-04:00",
            "startedAt": "2021-08-11T12:21:34.945000-04:00",
            "startedBy": "ecs-svc/968695068243EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/4d590253bb114126b7afa7b58eea9221",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/console-sample-app-static2:1",
            "version": 2
        }
    ],
    "failures": []
}
```

Weitere Informationen finden Sie unter Amazon-ECS-Aufgabendefinitionen im Entwicklerhandbuch für Amazon ECS.

Beispiel 2: Um mehrere Aufgaben zu beschreiben

Im folgenden Beispiel für `describe-tasks` werden die Details mehrerer Aufgaben in einem Cluster abgerufen. Sie können die Aufgabe angeben, indem Sie entweder die ID oder den vollständigen ARN der Aufgabe verwenden. In diesem Beispiel werden die vollständigen IDs der Aufgaben genutzt.

```
aws ecs describe-tasks \
    --cluster MyCluster \
    --tasks "74de0355a10a4f979ac495c14EXAMPLE" "d789e94343414c25b9f6bd59eEXAMPLE"
```

Ausgabe:

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d51EXAMPLE",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
```

```
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnet-0d0eab1bb3EXAMPLE"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "macAddress",
                            "value": "0e:89:76:28:07:b3"
                        },
                        {
                            "name": "privateDnsName",
                            "value": "ip-10-0-1-184.ec2.internal"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.1.184"
                        }
                    ]
                }
            ],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-12-20T12:13:37.875000-05:00",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/74de0355a10a4f979ac495c14EXAMPLE/
aad3ba00-83b3-4dac-84d4-11f8cEXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
                    "name": "web",
                    "image": "nginx",
                    "runtimeId": "74de0355a10a4f979ac495c14EXAMPLE-265927825",
```

```
                    "lastStatus": "RUNNING",
                    "networkBindings": [],
                    "networkInterfaces": [
                        {
                            "attachmentId": "d9e7735a-16aa-4128-bc7a-
b2d51EXAMPLE",
                            "privateIpv4Address": "10.0.1.184"
                        }
                    ],
                    "healthStatus": "UNKNOWN",
                    "cpu": "99",
                    "memory": "100"
                }
            ],
            "cpu": "256",
            "createdAt": "2021-12-20T12:13:20.226000-05:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:tdsevicetag",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "FARGATE",
            "memory": "512",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "web"
                    }
                ],
                "inferenceAcceleratorOverrides": []
            },
            "platformVersion": "1.4.0",
            "platformFamily": "Linux",
            "pullStartedAt": "2021-12-20T12:13:42.665000-05:00",
            "pullStoppedAt": "2021-12-20T12:13:46.543000-05:00",
            "startedAt": "2021-12-20T12:13:48.086000-05:00",
            "startedBy": "ecs-svc/988401040018EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/74de0355a10a4f979ac495c14EXAMPLE",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/webserver:2",
            "version": 3,
            "ephemeralStorage": {
```

```
            "sizeInGiB": 20
            }
        },
        {
            "attachments": [
                {
                    "id": "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnet-0d0eab1bb3EXAMPLE"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-064c7766daEXAMPLE"
                        },
                        {
                            "name": "macAddress",
                            "value": "0e:76:83:01:17:a9"
                        },
                        {
                            "name": "privateDnsName",
                            "value": "ip-10-0-1-41.ec2.internal"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.1.41"
                        }
                    ]
                }
            ],
            "attributes": [
                {
                    "name": "ecs.cpu-architecture",
                    "value": "x86_64"
                }
            ],
            "availabilityZone": "us-east-1b",
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
            "connectivity": "CONNECTED",
            "connectivityAt": "2021-12-20T12:13:35.243000-05:00",
            "containers": [
```

```
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:123456789012:container/MyCluster/
d789e94343414c25b9f6bd59eEXAMPLE/9afef792-609b-43a5-bb6a-3efdbEXAMPLE",
                    "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/
MyCluster/d789e94343414c25b9f6bd59eEXAMPLE",
                    "name": "web",
                    "image": "nginx",
                    "runtimeId": "d789e94343414c25b9f6bd59eEXAMPLE-265927825",
                    "lastStatus": "RUNNING",
                    "networkBindings": [],
                    "networkInterfaces": [
                        {
                            "attachmentId":
 "214eb5a9-45cd-4bf8-87bc-57fefEXAMPLE",
                            "privateIpv4Address": "10.0.1.41"
                        }
                    ],
                    "healthStatus": "UNKNOWN",
                    "cpu": "99",
                    "memory": "100"
                }
            ],
            "cpu": "256",
            "createdAt": "2021-12-20T12:13:20.226000-05:00",
            "desiredStatus": "RUNNING",
            "enableExecuteCommand": false,
            "group": "service:tdsevicetag",
            "healthStatus": "UNKNOWN",
            "lastStatus": "RUNNING",
            "launchType": "FARGATE",
            "memory": "512",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "web"
                    }
                ],
                "inferenceAcceleratorOverrides": []
            },
            "platformVersion": "1.4.0",
            "platformFamily": "Linux",
            "pullStartedAt": "2021-12-20T12:13:44.611000-05:00",
            "pullStoppedAt": "2021-12-20T12:13:48.251000-05:00",
```

```
            "startedAt": "2021-12-20T12:13:49.326000-05:00",
            "startedBy": "ecs-svc/988401040018EXAMPLE",
            "tags": [],
            "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/MyCluster/
d789e94343414c25b9f6bd59eEXAMPLE",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:123456789012:task-
definition/webserver:2",
            "version": 3,
            "ephemeralStorage": {
                "sizeInGiB": 20
            }
        }
    ],
    "failures": []
}
```

Weitere Informationen finden Sie unter Amazon-ECS-Aufgabendefinitionen im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie DescribeTasksin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> <taskId>\s

                Where:
                  clusterArn - The ARN of an ECS cluster.
                  taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
 taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                    .cluster(clusterArn)
                    .tasks(taskId)
                    .build();

            DescribeTasksResponse response =
 ecsClient.describeTasks(tasksRequest);
```

```
            List<Task> tasks = response.tasks();
            for (Task task : tasks) {
                System.out.println("The task ARN is " +
 task.taskDefinitionArn());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeTasksin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListClusters`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListClusters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Abrufen von ARN-Informationen für Cluster, -Services und -Aufgaben

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
```

```
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all
the clusters in your AWS account. {e.InnerException}");
        }
    }
```

- Einzelheiten zur API finden Sie ListClustersin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Ihre verfügbaren Cluster auf

Im folgenden Beispiel für `list-clusters` werden alle verfügbaren Cluster aufgelistet.

```
aws ecs list-clusters
```

Ausgabe:

```
{
    "clusterArns": [
        "arn:aws:ecs:us-west-2:123456789012:cluster/MyECSCluster1",
        "arn:aws:ecs:us-west-2:123456789012:cluster/AnotherECSCluster"
    ]
}
```

Weitere Informationen finden Sie unter Amazon ECS-optimiertes Linux-AMI im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie ListClustersin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListClustersin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet gibt eine Liste vorhandener ECS-Cluster zurück.

```
Get-ECSClusterList
```

Ausgabe:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Einzelheiten zur API finden Sie unter ListClusters AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet gibt eine Liste vorhandener ECS-Cluster zurück.

```
Get-ECSClusterList
```

Ausgabe:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- Einzelheiten zur API finden Sie unter ListClusters AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListServices`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListServices` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Einen Containerdienst für VM-Instanzen erstellen
- Abrufen von ARN-Informationen für Cluster, -Services und -Aufgaben

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// List service ARNs available.
/// </summary>
/// <param name="clusterARN">The arn of the ECS cluster.</param>
/// <returns>The ARN list of services in given cluster.</returns>
public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
{
    List<string> serviceArns = new List<string>();

    var request = new ListServicesRequest
    {
        Cluster = clusterARN
    };
    // Call the ListServices API operation and get the list of service ARNs
    var serviceList = _ecsClient.Paginators.ListServices(request);

    await foreach (var serviceARN in serviceList.ServiceArns)
    {
        if (serviceARN is null)
            continue;

        serviceArns.Add(serviceARN);
    }

    if (serviceArns.Count == 0)
    {
        _logger.LogWarning($"No services found in cluster {clusterARN} .");
    }

    return serviceArns;
}
```

- Einzelheiten zur API finden Sie ListServicesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die Services in einem Cluster auf

Das folgende Beispiel für `list-services` zeigt, wie Sie die Services auflisten, die in einem Cluster ausgeführt werden.

```
aws ecs list-services --cluster MyCluster
```

Ausgabe:

```
 {
     "serviceArns": [
         "arn:aws:ecs:us-west-2:123456789012:service/MyCluster/MyService"
     ]
}
```

Weitere Informationen finden Sie unter Services im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie ListServicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Services aufgeführt, die in Ihrem Standard-Cluster ausgeführt werden.

```
Get-ECSClusterService
```

Beispiel 2: In diesem Beispiel werden alle Services aufgeführt, die im angegebenen Cluster ausgeführt werden.

```
Get-ECSClusterService -Cluster myCluster
```

- Einzelheiten zur API finden Sie unter ListServices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Services aufgeführt, die in Ihrem Standard-Cluster ausgeführt werden.

```
Get-ECSClusterService
```

Beispiel 2: In diesem Beispiel werden alle Services aufgeführt, die im angegebenen Cluster ausgeführt werden.

```
Get-ECSClusterService -Cluster myCluster
```

- Einzelheiten zur API finden Sie unter ListServices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListTasks`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTasks` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Abrufen von ARN-Informationen für Cluster, -Services und -Aufgaben

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
```

- Einzelheiten zur API finden Sie ListTasksin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So listen Sie die Aufgaben in einem Cluster auf

Im folgenden Beispiel für `list-tasks` werden alle Aufgaben in einem Cluster aufgelistet.

```
aws ecs list-tasks --cluster default
```

Ausgabe:

```
{
    "taskArns": [
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-11111EXAMPLE",
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
    ]
}
```

Beispiel 2: So listen Sie die Aufgaben auf einer bestimmten Container-Instance auf

Im folgenden Beispiel für `list-tasks` werden die Aufgaben auf einer Container-Instance aufgeführt, wobei die UUID der Container-Instance als Filter verwendet wird.

```
aws ecs list-tasks --cluster default --container-instance a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

Ausgabe:

```
{
    "taskArns": [
        "arn:aws:ecs:us-west-2:123456789012:task/a1b2c3d4-5678-90ab-
cdef-44444EXAMPLE"
    ]
}
```

Weitere Informationen finden Sie unter Amazon-ECS-Aufgabendefinitionen im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie ListTasksin der AWS CLI Befehlsreferenz.

Verwendung von **`UpdateClusterSettings`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateClusterSettings` verwendet wird.

CLI

AWS CLI

So ändern Sie die Einstellungen für Ihren Cluster

Das folgende `update-cluster-settings` Beispiel aktiviert CloudWatch Container Insights mit verbesserter Beobachtbarkeit für den `MyCluster` Cluster.

```
aws ecs update-cluster-settings \
    --cluster MyCluster \
    --settings name=containerInsights,value=enhanced
```

Ausgabe:

```
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-esat-1:123456789012:cluster/MyCluster",
        "clusterName": "default",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "enhanced"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Ändern der Kontoeinstellungen im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie UpdateClusterSettingsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet ändert die Einstellungen, die für einen ECS-Cluster verwendet werden.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting
 @{Name="containerInsights"; Value="disabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Einzelheiten zur API finden Sie unter UpdateClusterSettings AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet ändert die Einstellungen, die für einen ECS-Cluster verwendet werden.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting
 @{Name="containerInsights"; Value="disabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
```

```
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/
LAB-ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- Einzelheiten zur API finden Sie unter UpdateClusterSettings AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateService`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateService` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

Beispiel 1: So ändern Sie die in einem Service verwendete Aufgabendefinition

Im folgenden Beispiel für `update-service` wird der Service `my-http-service` auf die Verwendung der Aufgabendefinition `amazon-ecs-sample` aktualisiert.

```
aws ecs update-service \
    --cluster test \
    --service my-http-service \
```

```
    --task-definition amazon-ecs-sample
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/test/my-http-
service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/test",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "1.4.0",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
amazon-ecs-sample:2",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/7419115625193919142",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/amazon-ecs-sample:2",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
```

```
                "failedTasks": 0,
                "createdAt": "2025-02-21T13:26:02.734000-06:00",
                "updatedAt": "2025-02-21T13:26:02.734000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7419115625193919142
 in progress."
            },
            {
                "id": "ecs-svc/1709597507655421668",
                "status": "ACTIVE",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/old-amazon-ecs-sample:4",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-01-24T11:13:07.621000-06:00",
                "updatedAt": "2025-02-02T16:11:30.838000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                             "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
```

```
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1709597507655421668
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "e40b4d1c-80d9-4834-aaf3-6a268e530e17",
                "createdAt": "2025-02-21T10:31:26.037000-06:00",
                "message": "(my-http-service) has reached a steady state."
            },
            {
                "id": "6ac069ad-fc8b-4e49-a35d-b5574a964c8e",
                "createdAt": "2025-02-21T04:31:22.703000-06:00",
                "message": "(my-http-service) has reached a steady state."
            },
            {
                "id": "265f7d37-dfd1-4880-a846-ec486f341919",
                "createdAt": "2025-02-20T22:31:22.514000-06:00",
                "message": "(my-http-service) has reached a steady state."
            }
        ],
        "createdAt": "2024-10-30T17:12:43.218000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321",
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
```

```
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "DISABLED"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren eines Amazon-ECS-Service mithilfe der Konsole im Entwicklerhandbuch für Amazon ECS.

Beispiel 2: So ändern Sie die Anzahl der Aufgaben in einem Service

Im folgenden Beispiel für `update-service` wird die gewünschte Aufgabenanzahl des Service `my-http-service` auf 2 aktualisiert.

```
aws ecs update-service \
    --cluster MyCluster \
    --service my-http-service \
    --desired-count 2
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 1,
        "pendingCount": 0,
        "capacityProviderStrategy": [
            {
                "capacityProvider": "FARGATE",
                "weight": 1,
                "base": 0
            }
        ],
```

```
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:123456789012:task-definition/
MyTaskDefinition",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/1976744184940610707",
                "status": "PRIMARY",
                "taskkDefinition": "arn:aws:ecs:us-east-1:123456789012:task-
definition/MyTaskDefinition",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 1,
                "failedTasks": 0,
                "createdAt": "2024-12-03T16:24:25.225000-05:00",
                "updatedAt": "2024-12-03T16:25:15.837000-05:00",
                "capacityProviderStrategy": [
                    {
                        "capacityProvider": "FARGATE",
                        "weight": 1,
                        "base": 0
                    }
                ],
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-0d0eab1bb38d5ca64",
                            "subnet-0db5010045995c2d5"
                        ],
```

```
                        "securityGroups": [
                            "sg-02556bf85a191f59a"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1976744184940610707
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "f27350b9-4b2a-4e2e-b72e-a4b68380de45",
                "createdAt": "2024-12-30T13:24:07.345000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "e764ec63-f53f-45e3-9af2-d99f922d2957",
                "createdAt": "2024-12-30T12:32:21.600000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "28444756-c2fa-47f8-bd60-93a8e05f3991",
                "createdAt": "2024-12-08T19:26:10.367000-05:00",
                "message": "(service my-http-service) has reached a steady
 state."
            }
        ],
        "createdAt": "2024-12-03T16:24:25.225000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-0d0eab1bb38d5ca64",
                    "subnet-0db5010045995c2d5"
                ],
                "securityGroups": [
                    "sg-02556bf85a191f59a"
```

```
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/Admin",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren eines Amazon-ECS-Service mithilfe der Konsole im Entwicklerhandbuch für Amazon ECS.

Beispiel 3: So konfigurieren Sie Amazon-EBS-Volumes für Anhänge beim Service-Update

Im folgenden Beispiel für `update-service` wird der Service `my-http-service` aktualisiert, sodass er Amazon-EBS-Volumes verwendet. Sie müssen über eine Amazon-ECS-Infrastrukturrolle verfügen, die mit der angehängten verwalteten Richtlinie `AmazonECSInfrastructureRolePolicyForVolumes` konfiguriert ist. Sie müssen eine Aufgabendefinition mit demselben Volume-Namen wie in der `update-service`-Anforderung angeben und `configuredAtLaunch` auf `true` festlegen. In diesem Beispiel werden die Option `--cli-input-json` und eine JSON-Eingabedatei mit dem Namen `ebs.json` verwendet.

```
aws ecs update-service \
    --cli-input-json file://ebs.json
```

Inhalt von `ebs.json`:

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "service": "my-http-service",
   "desiredCount": 2,
```

```
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
                "roleArn":"arn:aws:iam::123456789012:role/ecsInfrastructureRole",
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "iops": 3000,
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
   ]
}
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
```

```
                "rollback": false,
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/2420458347226626275",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:1",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:07:20.519000-06:00",
                "updatedAt": "2025-02-21T15:07:20.519000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321",
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/2420458347226626275
 in progress.",
                "volumeConfigurations": [
                    {
                        "name": "ebs-volume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
```

```
                        }
                    }
                ]
            },
            {
                "id": "ecs-svc/5191625155316533644",
                "status": "ACTIVE",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:2",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-02-21T14:54:48.862000-06:00",
                "updatedAt": "2025-02-21T14:57:22.502000-06:00",
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/5191625155316533644
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "b5823113-c2c5-458e-9649-8c2ed38f23a5",
                "createdAt": "2025-02-21T14:57:22.508000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
```

```
                "id": "b05a48e8-da35-4074-80aa-37ceb3167357",
                "createdAt": "2025-02-21T14:57:22.507000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/5191625155316533644) deployment completed."
            },
            {
                "id": "a10cd55d-4ba6-4cea-a655-5a5d32ada8a0",
                "createdAt": "2025-02-21T14:55:32.833000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 fb9c8df512684aec92f3c57dc3f22361)."
            },
        ],
        "createdAt": "2025-02-21T14:54:48.862000-06:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Weitere Informationen finden Sie unter Verwenden von Amazon-EBS-Volumes mit Amazon ECS im Entwicklerhandbuch für Amazon ECS.

Beispiel 4: So aktualisieren Sie einen Service dahingehend, dass er keine Amazon-EBS-Volumes mehr verwendet

Im folgenden Beispiel für `update-service` wird der Service `my-http-service` aktualisiert, sodass er keine Amazon-EBS-Volumes mehr verwendet. Sie müssen eine Aufgabendefinition angeben, bei der `configuredAtLaunch` auf `false` festgelegt ist.

```
aws ecs update-service \
    --cluster mycluster \
    --task-definition mytaskdef \
    --service my-http-service \
    --desired-count 2 \
    --volume-configurations "[]"
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/mycluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/mycluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 2,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
mytaskdef:3",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
```

```
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/7522791612543716777",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/mytaskdef:3",
                "desiredCount": 0,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:25:38.598000-06:00",
                "updatedAt": "2025-02-21T15:25:38.598000-06:00",
                    "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/7522791612543716777
 in progress."
            },
            {
                "id": "ecs-svc/2420458347226626275",
                "status": "ACTIVE",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/myoldtaskdef:1",
                "desiredCount": 2,
                "pendingCount": 0,
                "runningCount": 2,
                "failedTasks": 0,
                "createdAt": "2025-02-21T15:07:20.519000-06:00",
                "updatedAt": "2025-02-21T15:10:59.955000-06:00",
```

```
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-12344321"
                        ],
                        "securityGroups": [
                            "sg-12344321"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/2420458347226626275
 completed.",
                "volumeConfigurations": [
                    {
                        "name": "ebs-volume",
                        "managedEBSVolume": {
                            "volumeType": "gp3",
                            "sizeInGiB": 100,
                            "iops": 3000,
                            "throughput": 125,
                            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRole",
                            "filesystemType": "ext4"
                        }
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [
            {
                "id": "4f2c3ca1-7800-4048-ba57-bba210ada2ad",
                "createdAt": "2025-02-21T15:10:59.959000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "4b36a593-2d40-4ed6-8be8-b9b699eb6198",
```

```
                "createdAt": "2025-02-21T15:10:59.958000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/2420458347226626275) deployment completed."
            },
            {
                "id": "88380089-14e2-4ef0-8dbb-a33991683371",
                "createdAt": "2025-02-21T15:09:39.055000-06:00",
                "message": "(service my-http-service) has stopped 1 running
 tasks: (task fb9c8df512684aec92f3c57dc3f22361)."
            },
            {
                "id": "97d84243-d52f-4255-89bb-9311391c61f6",
                "createdAt": "2025-02-21T15:08:57.653000-06:00",
                "message": "(service my-http-service) has stopped 1 running
 tasks: (task 33eff090ad2c40539daa837e6503a9bc)."
            },
            {
                "id": "672ece6c-e2d0-4021-b5da-eefb14001687",
                "createdAt": "2025-02-21T15:08:15.631000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 996c02a66ff24f3190a4a8e0c841740f)."
            },
            {
                "id": "a3cf9bea-9be6-4175-ac28-4c68360986eb",
                "createdAt": "2025-02-21T15:07:36.931000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 d5d23c39f89e46cf9a647b9cc6572feb)."
            },
            {
                "id": "b5823113-c2c5-458e-9649-8c2ed38f23a5",
                "createdAt": "2025-02-21T14:57:22.508000-06:00",
                "message": "(service my-http-service) has reached a steady
 state."
            },
            {
                "id": "b05a48e8-da35-4074-80aa-37ceb3167357",
                "createdAt": "2025-02-21T14:57:22.507000-06:00",
                "message": "(service my-http-service) (deployment ecs-
svc/5191625155316533644) deployment completed."
            },
            {
                "id": "a10cd55d-4ba6-4cea-a655-5a5d32ada8a0",
                "createdAt": "2025-02-21T14:55:32.833000-06:00",
```

```
                "message": "(service my-http-service) has started 1 tasks: (task
 fb9c8df512684aec92f3c57dc3f22361)."
            },
            {
                "id": "42da91fa-e26d-42ef-88c3-bb5965c56b2f",
                "createdAt": "2025-02-21T14:55:02.703000-06:00",
                "message": "(service my-http-service) has started 1 tasks: (task
 33eff090ad2c40539daa837e6503a9bc)."
            }
        ],
        "createdAt": "2025-02-21T14:54:48.862000-06:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-12344321"
                ],
                "securityGroups": [
                    "sg-12344321"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/AIDACKCEVSQ6C2EXAMPLE",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Weitere Informationen finden Sie unter Verwenden von Amazon-EBS-Volumes mit Amazon ECS im Entwicklerhandbuch für Amazon ECS.

Beispiel 5: So aktivieren Sie die Neuverteilung von Availability Zones für einen Service

Im folgenden Beispiel für `update-service` wird die Neuverteilung von Availability Zones für den Service `my-http-service` aktiviert.

```
aws ecs update-service \
    --cluster MyCluster \
    --service my-http-service \
    --availability-zone-rebalancing ENABLED
```

Ausgabe:

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:123456789012:service/MyCluster/my-
http-service",
        "serviceName": "my-http-service",
        "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/MyCluster",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 2,
        "runningCount": 1,
        "pendingCount": 0,
        "capacityProviderStrategy": [
            {
                "capacityProvider": "FARGATE",
                "weight": 1,
                "base": 0
            }
        ],
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:123456789012:task-definition/
MyTaskDefinition",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": true,
                "rollback": true
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100,
            "alarms": {
                "alarmNames": [],
                "rollback": false,
```

```
                "enable": false
            }
        },
        "deployments": [
            {
                "id": "ecs-svc/1976744184940610707",
                "status": "PRIMARY",
                "taskkDefinition": "arn:aws:ecs:us-east-1:123456789012:task-
definition/MyTaskDefinition",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 1,
                "failedTasks": 0,
                "createdAt": "2024-12-03T16:24:25.225000-05:00",
                "updatedAt": "2024-12-03T16:25:15.837000-05:00",
                "capacityProviderStrategy": [
                    {
                        "capacityProvider": "FARGATE",
                        "weight": 1,
                        "base": 0
                    }
                ],
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-0d0eab1bb38d5ca64",
                            "subnet-0db5010045995c2d5"
                        ],
                        "securityGroups": [
                            "sg-02556bf85a191f59a"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "COMPLETED",
                "rolloutStateReason": "ECS deployment ecs-svc/1976744184940610707
 completed."
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
```

```
        "createdAt": "2024-12-03T16:24:25.225000-05:00",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-0d0eab1bb38d5ca64",
                    "subnet-0db5010045995c2d5"
                ],
                "securityGroups": [
                    "sg-02556bf85a191f59a"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "healthCheckGracePeriodSeconds": 0,
        "schedulingStrategy": "REPLICA",
        "deploymentController": {
            "type": "ECS"
        },
        "createdBy": "arn:aws:iam::123456789012:role/Admin",
        "enableECSManagedTags": true,
        "propagateTags": "NONE",
        "enableExecuteCommand": false,
        "availabilityZoneRebalancing": "ENABLED"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren eines Amazon-ECS-Service mithilfe der Konsole im Entwicklerhandbuch für Amazon ECS.

- Einzelheiten zur API finden Sie UpdateServicein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class UpdateService {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                   <clusterName> <serviceArn>\s

                Where:
                  clusterName - The cluster name.
                  serviceArn - The service ARN value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
```

```
    }

    public static void updateSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .desiredCount(0)
                    .build();

            ecsClient.updateService(serviceRequest);
            System.out.println("The service was modified");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie UpdateServicein der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Beispielbefehl aktualisiert den Service „my-http-service“, sodass er die Aufgabendefinition „amazon-ecs-sample“ verwendet.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Beispiel 2: Dieser Beispielbefehl aktualisiert die gewünschte Anzahl des Service „my-http-service“ auf 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Einzelheiten zur API finden Sie unter UpdateService AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Beispielbefehl aktualisiert den Service „my-http-service“, sodass er die Aufgabendefinition „amazon-ecs-sample“ verwendet.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Beispiel 2: Dieser Beispielbefehl aktualisiert die gewünschte Anzahl des Service „my-http-service“ auf 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- Einzelheiten zur API finden Sie unter UpdateService AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Amazon ECS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon ECS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon ECS oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Container-Service für Instanzen virtueller Maschinen erstellen
- Rufen Sie ARN-Informationen für Amazon ECS-Cluster, -Services und -Tasks mit einem ab AWS SDK

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur

- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
```

```
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}
```

```
# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
```

```
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
```

```
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
```

```
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
```

```
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
```

```
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
```

```
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
```

```
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
```

```
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi
```

```
    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
    echo "==========================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "==========================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"
```

```
        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
```

```
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
```

```
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
```

```
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
```

```
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
```

```
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
```

```
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
```

```
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
```

```
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
'$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
```

```
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
 security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "========================================="
        echo "CLEANUP COMPLETED"
```

```
        echo "============================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================="
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================="

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
```

```
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"
```

```
    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
```

```
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "==========================================="
echo "STEP 4: SET UP NETWORKING"
```

```
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
```

```
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"
```

```
SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
```

```
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi
```

```
echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetRole
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService
  - Wait

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Einen Dienst erstellen und überwachen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"
```

```
# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
```

```
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
```

```
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
```

```
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi
```

```
    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}
```

```
# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
```

```
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
```

```
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")
```

```
    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
```

```
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
```

```
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
```

```
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
```

```
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}
```

```
# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
```

```
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances

- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Rufen Sie ARN-Informationen für Amazon ECS-Cluster, -Services und -Tasks mit einem ab AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen einer Liste aller Cluster.
- Abrufen von Services für einen Cluster.
- Abrufen von Aufgaben für einen Cluster.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
using Amazon.ECS;
using ECSActions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
```

```
namespace ECSScenario;

public class ECSScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.


    This .NET example performs the following tasks:
        1. List ECS Cluster ARNs.
        2. List services in every cluster
        3. List Task ARNs in every cluster.
    */

    private static ILogger logger = null!;
    private static ECSWrapper _ecsWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
        .Build();

        ILoggerFactory loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ECSScenario>();

        var loggerECSWarpper = LoggerFactory.Create(builder =>
 { builder.AddConsole(); })
          .CreateLogger<ECSWrapper>();

        var amazonECSClient = new AmazonECSClient();
```

```
        _ecsWrapper = new ECSWrapper(amazonECSClient, loggerECSWarpper);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon ECS example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListClusterARNs();
            await ListServiceARNs();
            await ListTaskARNs();

        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// List ECS Cluster ARNs
    /// </summary>
    private static async Task ListClusterARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List Cluster ARNs from ECS.");
        var arns = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var arn in arns)
        {
            Console.WriteLine($"Cluster arn: {arn}");
            Console.WriteLine($"Cluster name: {arn.Split("/").Last()}");
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List services in every cluster
    /// </summary>
    private static async Task ListServiceARNs()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List Service ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting services for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));


            var serviceARNs = await _ecsWrapper.GetServiceARNSAsync(clusterARN);

            foreach (var serviceARN in serviceARNs)
            {
                Console.WriteLine($"Service arn: {serviceARN}");
                Console.WriteLine($"Service name:
{serviceARN.Split("/").Last()}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List tasks in every cluster
    /// </summary>
    private static async Task ListTaskARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. List Task ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting tasks for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));

            var taskARNs = await _ecsWrapper.GetTaskARNsAsync(clusterARN);

            foreach (var taskARN in taskARNs)
            {
```

```
                Console.WriteLine($"Task arn: {taskARN}");
            }
        }
        Console.WriteLine(new string('-', 80));
    }
}
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Amazon-ECS-Aktionen zu verwalten.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.Logging;

namespace ECSActions;

public class ECSWrapper
{
    private readonly AmazonECSClient _ecsClient;
    private readonly ILogger<ECSWrapper> _logger;

    /// <summary>
    /// Constructor for the ECS wrapper.
    /// </summary>
    /// <param name="ecsClient">The injected ECS client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public ECSWrapper(AmazonECSClient ecsClient, ILogger<ECSWrapper> logger)

    {
        _logger = logger;
        _ecsClient = ecsClient;
    }

    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
 account...");
```

```
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all
the clusters in your AWS account. {e.InnerException}");
        }
    }

    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();

        var request = new ListServicesRequest
        {
```

```
            Cluster = clusterARN
        };
        // Call the ListServices API operation and get the list of service ARNs
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }

    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;
```

```
            taskArns.Add(task);
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - ListClusters
  - ListServices
  - ListTasks

Codebeispiele für Amazon EFS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic File System mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Amazon-EFS-Entwicklerhandbuch – Weitere Informationen zu Amazon EFS.
- API-Referenz für Amazon EFS – Details zu allen verfügbaren Amazon-EFS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon EFS AWS SDKs

- Aktionen für Amazon EFS mit AWS SDKs
  - Verwendung von CreateFileSystem mit einer CLI
  - Verwendung von CreateMountTarget mit einer CLI
  - Verwendung von DeleteFileSystem mit einer CLI
  - Verwendung von DeleteMountTarget mit einer CLI
  - Verwendung von DescribeFileSystems mit einer CLI
  - Verwendung von DescribeMountTargets mit einer CLI
  - Verwendung von DescribeTags mit einer CLI

Grundlegende Beispiele für die Verwendung von Amazon EFS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic File System mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon EFS mit AWS SDKs
  - Verwendung von CreateFileSystem mit einer CLI
  - Verwendung von CreateMountTarget mit einer CLI
  - Verwendung von DeleteFileSystem mit einer CLI
  - Verwendung von DeleteMountTarget mit einer CLI
  - Verwendung von DescribeFileSystems mit einer CLI
  - Verwendung von DescribeMountTargets mit einer CLI
  - Verwendung von DescribeTags mit einer CLI

Aktionen für Amazon EFS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon EFS-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Elastic File System.

Beispiele

- Verwendung von CreateFileSystem mit einer CLI
- Verwendung von CreateMountTarget mit einer CLI
- Verwendung von DeleteFileSystem mit einer CLI
- Verwendung von DeleteMountTarget mit einer CLI
- Verwendung von DescribeFileSystems mit einer CLI
- Verwendung von DescribeMountTargets mit einer CLI
- Verwendung von DescribeTags mit einer CLI

Verwendung von **`CreateFileSystem`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFileSystem` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein verschlüsseltes Dateisystem

Im folgenden Beispiel für `create-file-system` wird ein verschlüsseltes Dateisystem erstellt, das den Standard-CMK verwendet. Außerdem wird das Tag `Name=my-file-system` hinzugefügt.

```
aws efs create-file-system \
    --performance-mode generalPurpose \
    --throughput-mode bursting \
    --encrypted \
    --tags Key=Name,Value=my-file-system
```

Ausgabe:

```
{
    "OwnerId": "123456789012",
    "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
    "FileSystemId": "fs-c7a0456e",
    "FileSystemArn": "arn:aws:elasticfilesystem:us-west-2:123456789012:file-
system/fs-48499b4d",
    "CreationTime": 1595286880.0,
    "LifeCycleState": "creating",
    "Name": "my-file-system",
    "NumberOfMountTargets": 0,
```

```
    "SizeInBytes": {
        "Value": 0,
        "ValueInIA": 0,
        "ValueInStandard": 0
    },
    "PerformanceMode": "generalPurpose",
    "Encrypted": true,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-e62c-42e4-
adcf-30d92example",
    "ThroughputMode": "bursting",
    "Tags": [
        {
            "Key": "Name",
            "Value": "my-file-system"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von Amazon-EFS-Dateisystemen im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie CreateFileSystemin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues, leeres Dateisystem. Das Token, das zur Gewährleistung einer idempotenten Erstellung verwendet wird, wird automatisch generiert und kann über das **`CreationToken`**-Element des zurückgegebenen Objekts abgerufen werden.

```
New-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
```

```
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Beispiel 2: Erstellt ein neues, leeres Dateisystem unter Verwendung eines benutzerdefinierten Tokens, um eine idempotente Erstellung sicherzustellen.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Einzelheiten zur API finden Sie unter CreateFileSystem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues, leeres Dateisystem. Das Token, das zur Gewährleistung einer idempotenten Erstellung verwendet wird, wird automatisch generiert und kann über das **`CreationToken`**-Element des zurückgegebenen Objekts abgerufen werden.

```
New-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Beispiel 2: Erstellt ein neues, leeres Dateisystem unter Verwendung eines benutzerdefinierten Tokens, um eine idempotente Erstellung sicherzustellen.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- Einzelheiten zur API finden Sie unter CreateFileSystem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateMountTarget`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateMountTarget` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein Mount-Ziel

Im folgenden Beispiel für `create-mount-target` wird ein Mount-Ziel für das angegebene Dateisystem erstellt.

```
aws efs create-mount-target \
    --file-system-id fs-c7a0456e \
    --subnet-id subnet-02bf4c428bexample \
    --security-groups sg-068f739363example
```

Ausgabe:

```
{
    "OwnerId": "123456789012",
    "MountTargetId": "fsmt-f9a14450",
    "FileSystemId": "fs-c7a0456e",
    "SubnetId": "subnet-02bf4c428bexample",
    "LifeCycleState": "creating",
    "IpAddress": "10.0.1.24",
    "NetworkInterfaceId": "eni-02d542216aexample",
    "AvailabilityZoneId": "use2-az2",
    "AvailabilityZoneName": "us-east-2b",
    "VpcId": "vpc-0123456789abcdef0"
}
```

Weitere Informationen finden Sie unter Erstellen von Mount-Zielen im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie CreateMountTargetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt ein neues Mounting-Ziel für ein Dateisystem. Das angegebene Subnetz wird verwendet, um die Virtual Private Cloud (VPC) zu bestimmen, in der das Mount-Ziel erstellt wird, und die IP-Adresse, die automatisch zugewiesen wird (aus dem Adressbereich des

Subnetzes). Die zugewiesene IP-Adresse kann verwendet werden, um dieses Dateisystem auf einer Amazon-EC2-Instance zu mounten. Da keine Sicherheitsgruppen angegeben wurden, ist die für das Ziel erstellte Netzwerkschnittstelle der Standardsicherheitsgruppe für die VPC des Subnetzes zugeordnet.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Beispiel 2: Erzeugt ein neues Mount-Ziel für das angegebene Dateisystem mit automatisch zugewiesener IP-Adresse. Die für das Mount-Ziel erstellte Netzwerkschnittstelle ist den angegebenen Sicherheitsgruppen zugeordnet (es können bis zu 5 im Format „sg-xxxxxxxx“ angegeben werden).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Beispiel 3: Erstellt ein neues Mount-Ziel für das angegebene Dateisystem mit der angegebenen IP-Adresse.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Einzelheiten zur API finden Sie unter CreateMountTarget AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt ein neues Mounting-Ziel für ein Dateisystem. Das angegebene Subnetz wird verwendet, um die Virtual Private Cloud (VPC) zu bestimmen, in der das Mount-Ziel erstellt wird, und die IP-Adresse, die automatisch zugewiesen wird (aus dem Adressbereich des

Subnetzes). Die zugewiesene IP-Adresse kann verwendet werden, um dieses Dateisystem auf einer Amazon-EC2-Instance zu mounten. Da keine Sicherheitsgruppen angegeben wurden, ist die für das Ziel erstellte Netzwerkschnittstelle der Standardsicherheitsgruppe für die VPC des Subnetzes zugeordnet.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Beispiel 2: Erzeugt ein neues Mount-Ziel für das angegebene Dateisystem mit automatisch zugewiesener IP-Adresse. Die für das Mount-Ziel erstellte Netzwerkschnittstelle ist den angegebenen Sicherheitsgruppen zugeordnet (es können bis zu 5 im Format „sg-xxxxxxxx“ angegeben werden).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Beispiel 3: Erstellt ein neues Mount-Ziel für das angegebene Dateisystem mit der angegebenen IP-Adresse.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- Einzelheiten zur API finden Sie unter CreateMountTarget AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteFileSystem`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFileSystem` verwendet wird.

CLI

AWS CLI

Um ein Dateisystem zu löschen

Im folgenden Beispiel für `delete-file-system` wird das angegebene Dateisystem gelöscht.

```
aws efs delete-file-system \
    --file-system-id fs-c7a0456e
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines Amazon-EFS-Dateisystems im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie DeleteFileSystemin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht das angegebene Dateisystem, das nicht mehr verwendet wird (wenn das Dateisystem Mount-Ziele hat, müssen diese zuerst entfernt werden). Sie werden um eine Bestätigung gebeten, bevor das Cmdlet fortfährt. Um die Bestätigung zu unterdrücken, verwenden Sie den **`-Force`**-Schalter.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteFileSystem AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht das angegebene Dateisystem, das nicht mehr verwendet wird (wenn das Dateisystem Mount-Ziele hat, müssen diese zuerst entfernt werden). Sie werden um eine Bestätigung gebeten, bevor das Cmdlet fortfährt. Um die Bestätigung zu unterdrücken, verwenden Sie den **`-Force`**-Schalter.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteFileSystem AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteMountTarget`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteMountTarget` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Mount-Ziel

Im folgenden Beispiel für `delete-mount-target` wird das angegebene Mount-Ziel gelöscht.

```
aws efs delete-mount-target \
    --mount-target-id fsmt-f9a14450
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von Mount-Zielen im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie DeleteMountTargetin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht das angegebene Mount-Ziel. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Verwenden Sie den **`-Force`**-Schalter, um das Prompt zu unterdrücken. Beachten Sie, dass dieser Vorgang alle Mounts des Dateisystems über das Ziel zwangsweise beendet. Sie sollten daher in Erwägung ziehen, den Mount des Dateisystems aufzuheben, bevor Sie diesen Befehl ausführen, sofern dies möglich ist.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteMountTarget AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht das angegebene Mount-Ziel. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Verwenden Sie den **`-Force`**-Schalter, um das Prompt zu unterdrücken. Beachten Sie, dass dieser Vorgang alle Mounts des Dateisystems über das Ziel zwangsweise beendet. Sie sollten daher in Erwägung ziehen, den Mount des Dateisystems aufzuheben, bevor Sie diesen Befehl ausführen, sofern dies möglich ist.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DeleteMountTarget AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeFileSystems`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFileSystems` verwendet wird.

CLI

AWS CLI

So beschreiben Sie ein Dateisystem

Im folgenden Beispiel für `describe-file-systems` wird das angegebene Dateisystem beschrieben.

```
aws efs describe-file-systems \
    --file-system-id fs-c7a0456e
```

Ausgabe:

```
{
    "FileSystems": [
        {
            "OwnerId": "123456789012",
            "CreationToken": "console-d7f56c5f-e433-41ca-8307-9d9c0example",
            "FileSystemId": "fs-c7a0456e",
            "FileSystemArn": "arn:aws:elasticfilesystem:us-
west-2:123456789012:file-system/fs-48499b4d",
            "CreationTime": 1595286880.0,
            "LifeCycleState": "available",
```

```
            "Name": "my-file-system",
            "NumberOfMountTargets": 3,
            "SizeInBytes": {
                "Value": 6144,
                "Timestamp": 1600991437.0,
                "ValueInIA": 0,
                "ValueInStandard": 6144
            },
            "PerformanceMode": "generalPurpose",
            "Encrypted": true,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/a59b3472-
e62c-42e4-adcf-30d92example",
            "ThroughputMode": "bursting",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "my-file-system"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten von Amazon-EFS-Dateisystemen im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie DescribeFileSystemsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Sammlung aller Dateisysteme zurück, die sich im Besitz des Kontos des Anrufers in der Region befinden.

```
Get-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
```

```
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Beispiel 2: Gibt die Details des angegebenen Dateisystems zurück.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Beispiel 3: Gibt die Details eines Dateisystems unter Verwendung des bei der Erstellung des Dateisystems angegebenen Idempotenz-Erstellungstokens zurück.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Einzelheiten zur API finden Sie unter DescribeFileSystems AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Sammlung aller Dateisysteme zurück, die sich im Besitz des Kontos des Anrufers in der Region befinden.

```
Get-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
```

```
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Beispiel 2: Gibt die Details des angegebenen Dateisystems zurück.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Beispiel 3: Gibt die Details eines Dateisystems unter Verwendung des bei der Erstellung des Dateisystems angegebenen Idempotenz-Erstellungstokens zurück.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- Einzelheiten zur API finden Sie unter DescribeFileSystems AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMountTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMountTargets` verwendet wird.

CLI

AWS CLI

So beschreiben Sie ein Mount-Ziel

Im folgenden Beispiel für `describe-mount-targets` wird das angegebene Mount-Ziel beschrieben.

```
aws efs describe-mount-targets \
    --mount-target-id fsmt-f9a14450
```

Ausgabe:

```
{
    "MountTargets": [
        {
            "OwnerId": "123456789012",
            "MountTargetId": "fsmt-f9a14450",
            "FileSystemId": "fs-c7a0456e",
```

```
            "SubnetId": "subnet-02bf4c428bexample",
            "LifeCycleState": "creating",
            "IpAddress": "10.0.1.24",
            "NetworkInterfaceId": "eni-02d542216aexample",
            "AvailabilityZoneId": "use2-az2",
            "AvailabilityZoneName": "us-east-2b",
            "VpcId": "vpc-0123456789abcdef0"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von Mount-Zielen im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie DescribeMountTargetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Sammlung von Mount-Zielen zurück, die dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeMountTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Sammlung von Mount-Zielen zurück, die dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- Einzelheiten zur API finden Sie unter DescribeMountTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Tags für ein Dateisystem

Im folgenden Beispiel für `describe-tags` werden die Tags das angegebene Dateisystem beschrieben.

```
aws efs describe-tags \
    --file-system-id fs-c7a0456e
```

Ausgabe:

```
{
    "Tags": [
        {
            "Key": "Name",
            "Value": "my-file-system"
        },
        {
            "Key": "Department",
```

```
            "Value": "Business Intelligence"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten von Dateisystem-Tags im Benutzerhandbuch für Amazon Elastic File System.

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Sammlung von Tags zurück, die derzeit dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Sammlung von Tags zurück, die derzeit dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
Key         Value
---         -----
Name        My File System
```

```
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Amazon EKS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Elastic Kubernetes Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon EKS-Benutzerhandbuch – Weitere Informationen zu Amazon EKS.
- API-Referenz für Amazon EKS – Details zu allen verfügbaren Amazon EKS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon EKS AWS SDKs
  - Aktionen für Amazon EKS mit AWS SDKs
    - Verwenden Sie mit CreateClustereinem AWS SDK oder CLI
    - Verwendung von CreateFargateProfile mit einer CLI
    - Verwendung von CreateNodegroup mit einer CLI
    - Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
    - Verwendung von DeleteFargateProfile mit einer CLI
    - Verwendung von DeleteNodegroup mit einer CLI
    - Verwendung von DescribeCluster mit einer CLI

    - Verwendung von DescribeFargateProfile mit einer CLI
    - Verwendung von DescribeNodegroup mit einer CLI
    - Verwendung von DescribeUpdate mit einer CLI
    - Verwendung von ListClusters mit einer CLI
    - Verwendung von ListFargateProfiles mit einer CLI
    - Verwendung von ListNodegroups mit einer CLI
    - Verwendung von ListTagsForResource mit einer CLI
    - Verwendung von ListUpdates mit einer CLI
    - Verwendung von TagResource mit einer CLI
    - Verwendung von UntagResource mit einer CLI
    - Verwendung von UpdateClusterConfig mit einer CLI
    - Verwendung von UpdateClusterVersion mit einer CLI
- Szenarien für die Verwendung von Amazon EKS AWS SDKs
  - Erste Schritte mit verwalteten Kubernetes-Clustern

Grundlegende Beispiele für die Verwendung von Amazon EKS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Elastic Kubernetes Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon EKS mit AWS SDKs
  - Verwenden Sie mit CreateClustereinem AWS SDK oder CLI
  - Verwendung von CreateFargateProfile mit einer CLI
  - Verwendung von CreateNodegroup mit einer CLI
  - Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
  - Verwendung von DeleteFargateProfile mit einer CLI
  - Verwendung von DeleteNodegroup mit einer CLI
  - Verwendung von DescribeCluster mit einer CLI
  - Verwendung von DescribeFargateProfile mit einer CLI
  - Verwendung von DescribeNodegroup mit einer CLI
  - Verwendung von DescribeUpdate mit einer CLI

- Verwendung von ListClusters mit einer CLI
- Verwendung von ListFargateProfiles mit einer CLI
- Verwendung von ListNodegroups mit einer CLI
- Verwendung von ListTagsForResource mit einer CLI
- Verwendung von ListUpdates mit einer CLI
- Verwendung von TagResource mit einer CLI
- Verwendung von UntagResource mit einer CLI
- Verwendung von UpdateClusterConfig mit einer CLI
- Verwendung von UpdateClusterVersion mit einer CLI

Aktionen für Amazon EKS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon EKS-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon EKS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon EKS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Elastic Kubernetes Service.

Beispiele

- Verwenden Sie mit CreateClustereinem AWS SDK oder CLI
- Verwendung von CreateFargateProfile mit einer CLI
- Verwendung von CreateNodegroup mit einer CLI
- Verwenden Sie mit einem DeleteCluster AWS SDK oder CLI
- Verwendung von DeleteFargateProfile mit einer CLI
- Verwendung von DeleteNodegroup mit einer CLI
- Verwendung von DescribeCluster mit einer CLI
- Verwendung von DescribeFargateProfile mit einer CLI
- Verwendung von DescribeNodegroup mit einer CLI
- Verwendung von DescribeUpdate mit einer CLI

- Verwendung von ListClusters mit einer CLI
- Verwendung von ListFargateProfiles mit einer CLI
- Verwendung von ListNodegroups mit einer CLI
- Verwendung von ListTagsForResource mit einer CLI
- Verwendung von ListUpdates mit einer CLI
- Verwendung von TagResource mit einer CLI
- Verwendung von UntagResource mit einer CLI
- Verwendung von UpdateClusterConfig mit einer CLI
- Verwendung von UpdateClusterVersion mit einer CLI

Verwenden Sie mit **`CreateCluster`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

So erstellen Sie einen neuen Cluster

Dieser Beispielbefehl erstellt ein Cluster namens `prod` Ihrer Standardregion.

Befehl:

```
aws eks create-cluster --name prod \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18
```

Ausgabe:

```
{
```

```
    "cluster": {
        "name": "prod",
        "arn": "arn:aws:eks:us-west-2:012345678910:cluster/prod",
        "createdAt": 1527808069.147,
        "version": "1.10",
        "roleArn": "arn:aws:iam::012345678910:role/eks-service-role-
AWSServiceRoleForAmazonEKS-J7ONKE3BQ4PI",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-6782e71e",
                "subnet-e7e761ac"
            ],
            "securityGroupIds": [
                "sg-6979fe18"
            ],
            "vpcId": "vpc-950809ec"
        },
        "status": "CREATING",
        "certificateAuthority": {}
    }
}
```

So erstellen Sie einen neuen Cluster mit aktiviertem privaten Endpunktzugriff und aktivierter Protokollierung

Dieser Beispielbefehl erstellt ein Cluster namens `example` in Ihrer Standardregion, bei dem der öffentliche Endpunktzugriff deaktiviert ist, der private Endpunktzugriff aktiviert ist und alle Protokollierungstypen aktiviert sind.

Befehl:

```
aws eks create-cluster --name example --kubernetes-version 1.12 \
--role-arn arn:aws:iam::012345678910:role/example-cluster-
ServiceRole-1XWBQWYSFRE2Q \
--resources-vpc-
config subnetIds=subnet-0a188dccd2f9a632f,subnet-09290d93da4278664,subnet-0f21dd86e0e9113
 \
--logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Ausgabe:

```
{
```

```
    "cluster": {
        "name": "example",
        "arn": "arn:aws:eks:us-west-2:012345678910:cluster/example",
        "createdAt": 1565804921.901,
        "version": "1.12",
        "roleArn": "arn:aws:iam::012345678910:role/example-cluster-
ServiceRole-1XWBQWYSFRE2Q",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0a188dccd2f9a632f",
                "subnet-09290d93da4278664",
                "subnet-0f21dd86e0e91134a",
                "subnet-0173dead68481a583",
                "subnet-051f70a57ed6fcab6",
                "subnet-01322339c5c7de9b4"
            ],
            "securityGroupIds": [
                "sg-0c5b580845a031c10"
            ],
            "vpcId": "vpc-0f622c01f68d4afec",
            "endpointPublicAccess": false,
            "endpointPrivateAccess": true
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "status": "CREATING",
        "certificateAuthority": {},
        "platformVersion": "eks.3"
    }
}
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer Cluster namens „prod“ erstellt.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Cluster namens „prod“ erstellt.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
```

```
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- Einzelheiten zur API finden Sie unter CreateCluster AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster = client
        .create_cluster()
        .name(name)
        .role_arn(arn)
        .resources_vpc_config(
            VpcConfigRequest::builder()
                .set_subnet_ids(Some(subnet_ids))
                .build(),
        )
        .send()
        .await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateClusterin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`CreateFargateProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFargateProfile` verwendet wird.

CLI

AWS CLI

Beispiel 1: Erstellen eines EKS-Fargate-Profils für einen Selektor mit einem Namespace

Im folgenden Beispiel für `create-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit einem Namespace erstellt.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default"}]'
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/a2c72bca-318e-abe8-8ed1-27c6d4892e9e",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:38:47.368000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default"
            }
        ],
        "status": "CREATING",
```

```
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Erstellen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

Beispiel 2: Erstellen eines EKS-Fargate-Profils für einen Selektor mit einem Namespace und Labels

Im folgenden Beispiel für `create-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit einem Namespace und Labels erstellt.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default", "labels": {"labelname1":
 "labelvalue1"}}]'
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/88c72bc7-e8a4-fa34-44e4-2f1397224bb3",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:33:48.125000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default",
                "labels": {
                    "labelname1": "labelvalue1"
                }
            }
        ],
```

```
        "status": "CREATING",
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Erstellen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

Beispiel 3: Erstellen eines EKS-Fargate-Profils für einen Selektor mit einem Namespace und Labels sowie IDs von Subnetzen, in denen ein Pod gestartet werden soll

Im folgenden Beispiel für `create-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit einem Namespace und Labels sowie IDs von Subnetzen erstellt, in denen ein Pod gestartet werden soll.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default", "labels": {"labelname1":
 "labelvalue1"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]'
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/e8c72bc8-e87b-5eb6-57cb-ed4fe57577e3",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:35:58.640000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default",
```

```
                "labels": {
                    "labelname1": "labelvalue1"
                }
            }
        ],
        "status": "CREATING",
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Erstellen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

Beispiel 4: Erstellen eines EKS-Fargate-Profils für einen Selektor mit mehreren Namespaces und Labels sowie IDs von Subnetzen, in denen ein Pod gestartet werden soll

Im folgenden Beispiel für `create-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit mehreren Namespaces und Labels sowie IDs von Subnetzen erstellt, in denen ein Pod gestartet werden soll.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
    --selectors '[{"namespace": "default1", "labels": {"labelname1":
 "labelvalue1", "labelname2": "labelvalue2"}}, {"namespace": "default2",
 "labels": {"labelname1": "labelvalue1", "labelname2": "labelvalue2"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]' \
    --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-
key-2": "value-2"}'
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/4cc72bbf-b766-8ee6-8d29-e62748feb3cd",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T12:15:55.271000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
```

```
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default1",
                "labels": {
                    "labelname2": "labelvalue2",
                    "labelname1": "labelvalue1"
                }
            },
            {
                "namespace": "default2",
                "labels": {
                    "labelname2": "labelvalue2",
                    "labelname1": "labelvalue1"
                }
            }
        ],
        "status": "CREATING",
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Erstellen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

Beispiel 5: Erstellen eines EKS-Fargate-Profils mit einem Platzhalter-Selektor für Namespaces und Labels sowie IDs von Subnetzen, in denen ein Pod gestartet werden soll

Im folgenden Beispiel für `create-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit mehreren Namespaces und Labels sowie IDs von Subnetzen erstellt, in denen ein Pod gestartet werden soll.

```
aws eks create-fargate-profile \
    --cluster-name my-eks-cluster \
    --pod-execution-role-arn arn:aws:iam::111122223333:role/role-name \
    --fargate-profile-name my-fargate-profile \
```

```
    --selectors '[{"namespace": "prod*", "labels": {"labelname*?": "*value1"}},
 {"namespace": "*dev*", "labels": {"labelname*?": "*value*"}}]' \
    --subnets '["subnet-09d912bb63ef21b9a", "subnet-04ad87f71c6e5ab4d",
 "subnet-0e2907431c9988b72"]' \
    --tags '{"eks-fargate-profile-key-1": "value-1" , "eks-fargate-profile-
key-2": "value-2"}'
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/e8c72bd6-5966-0bfe-b77b-1802893e5a6f",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T13:05:20.550000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "prod*",
                "labels": {
                    "labelname*?": "*value1"
                }
            },
            {
                "namespace": "*dev*",
                "labels": {
                    "labelname*?": "*value*"
                }
            }
        ],
        "status": "CREATING",
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Erstellen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie CreateFargateProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet erstellt ein AWS Fargate-Profil für Ihren Amazon EKS-Cluster. Sie müssen mindestens ein Fargate-Profil in einem Cluster haben, um Pods auf der Fargate-Infrastruktur planen zu können.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName
 TEST -Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae"
 -PodExecutionRoleArn arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole -Selector @{Namespace="default"}
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter CreateFargateProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet erstellt ein AWS Fargate-Profil für Ihren Amazon EKS-Cluster. Sie müssen mindestens ein Fargate-Profil in einem Cluster haben, um Pods auf der Fargate-Infrastruktur planen zu können.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName
 TEST -Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae"
 -PodExecutionRoleArn arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole -Selector @{Namespace="default"}
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter CreateFargateProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateNodegroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateNodegroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Beispiel 1: Erstellen einer verwalteten Knotengruppe für einen Amazon-EKS-Cluster

Im folgenden Beispiel für `create-nodegroup` wird eine verwaltete Knotengruppe für einen Amazon-EKS-Cluster erstellt.

```
aws eks create-nodegroup \
```

```
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=3,desiredSize=1 \
    --region us-east-2
```

Ausgabe:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/bac7550f-b8b8-5fbb-4f3e-7502a931119e",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T13:19:32.260000-04:00",
        "modifiedAt": "2024-04-04T13:19:32.260000-04:00",
        "status": "CREATING",
        "capacityType": "ON_DEMAND",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
            "t3.medium"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72, subnet-04ad87f71c6e5ab4d,
 subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "diskSize": 20,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
```

```
        },
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer verwalteten Knotengruppe im Benutzerhandbuch für Amazon EKS.

Beispiel 2: Erstellen einer verwalteten Knotengruppe für einen Amazon-EKS-Cluster mit benutzerdefinierten Instance-Typen und benutzerdefinierter Datenträgergröße

Im folgenden Beispiel für `create-nodegroup` wird eine verwaltete Knotengruppe für einen Amazon-EKS-Cluster mit benutzerdefinierten Instance-Typen und benutzerdefinierter Datenträgergröße erstellt.

```
aws eks create-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=3,desiredSize=1 \
    --capacity-type ON_DEMAND \
    --instance-types 'm5.large' \
    --disk-size 50 \
    --region us-east-2
```

Ausgabe:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/c0c7551b-e4f9-73d9-992c-a450fdb82322",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T13:46:07.595000-04:00",
        "modifiedAt": "2024-04-04T13:46:07.595000-04:00",
        "status": "CREATING",
        "capacityType": "ON_DEMAND",
```

```
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
            "m5.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
        },
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer verwalteten Knotengruppe im Benutzerhandbuch für Amazon EKS.

Beispiel 3: Erstellen einer verwalteten Knotengruppe für einen Amazon-EKS-Cluster mit Instance-Typen, Datenträgergröße, AMI-Typ, Kapazitätstyp, Update-Konfiguration, Labels, Taints und Tags gemäß Benutzerdefinition

Im folgenden Beispiel für `create-nodegroup` wird eine verwaltete Knotengruppe für einen Amazon-EKS-Cluster mit Instance-Typen, Datenträgergröße, AMI-Typ, Kapazitätstyp, Update-Konfiguration, Labels, Taints und Tags gemäß Benutzerdefinition erstellt.

```
aws eks create-nodegroup  \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup \
    --node-role arn:aws:iam::111122223333:role/role-name \
```

```
    --
subnets "subnet-0e2907431c9988b72" "subnet-04ad87f71c6e5ab4d" "subnet-09d912bb63ef21b9a"
 \
    --scaling-config minSize=1,maxSize=5,desiredSize=4 \
    --instance-types 't3.large' \
    --disk-size 50 \
    --ami-type AL2_x86_64 \
    --capacity-type SPOT \
    --update-config maxUnavailable=2 \
    --labels '{"my-eks-nodegroup-label-1": "value-1" , "my-eks-nodegroup-
label-2": "value-2"}' \
    --taints '{"key": "taint-key-1" , "value": "taint-value-1", "effect":
 "NO_EXECUTE"}' \
    --tags '{"my-eks-nodegroup-key-1": "value-1" , "my-eks-nodegroup-key-2":
 "value-2"}'
```

Ausgabe:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/88c75524-97af-0cb9-a9c5-7c0423ab5314",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-04T14:05:07.940000-04:00",
        "modifiedAt": "2024-04-04T14:05:07.940000-04:00",
        "status": "CREATING",
        "capacityType": "SPOT",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 5,
            "desiredSize": 4
        },
        "instanceTypes": [
            "t3.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
```

```
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {
            "my-eks-nodegroup-label-2": "value-2",
            "my-eks-nodegroup-label-1": "value-1"
        },
        "taints": [
            {
                "key": "taint-key-1",
                "value": "taint-value-1",
                "effect": "NO_EXECUTE"
            }
        ],
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 2
        },
        "tags": {
            "my-eks-nodegroup-key-1": "value-1",
            "my-eks-nodegroup-key-2": "value-2"
        }
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer verwalteten Knotengruppe im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie CreateNodegroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet erstellt eine verwaltete Worker-Kknotengruppe für einen Amazon EKS-Cluster. Sie können nur eine Knotengruppe für Ihren Cluster erstellen, die der aktuellen Kubernetes-Version für den Cluster entspricht. Alle Knotengruppen werden mit der neuesten AMI-Release-Version für die jeweilige Kubernetes-Unterversion des Clusters erstellt.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
```

```
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter CreateNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet erstellt eine verwaltete Worker-Kknotengruppe für einen Amazon EKS-Cluster. Sie können nur eine Knotengruppe für Ihren Cluster erstellen, die der aktuellen Kubernetes-Version für den Cluster entspricht. Alle Knotengruppen werden mit der neuesten AMI-Release-Version für die jeweilige Kubernetes-Unterversion des Clusters erstellt.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter CreateNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteCluster`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Löschen einer Steuerebene für Amazon-EKS-Cluster

Im folgenden Beispiel für `delete-cluster` wird eine Steuerebene eines Amazon-EKS-Clusters gelöscht.

```
aws eks delete-cluster \
    --name my-eks-cluster
```

Ausgabe:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.27",
        "endpoint": "https://DALSJ343KE23J3RN45653DSKJTT647TYD.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "0.0.0.0/0"
            ]
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
```

```
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
DALSJ343KE23J3RN45653DSKJTT647TYD"
            }
        },
        "status": "DELETING",
        "certificateAuthority": {
            "data": "XXX_CA_DATA_XXX"
        },
        "platformVersion": "eks.16",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
```

```
}
```

Weitere Informationen finden Sie unter Erweiterungen für Amazon EKS im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie DeleteClusterin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht die Steuerebene des Amazon-EKS-Clusters.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Einzelheiten zur API finden Sie unter DeleteCluster AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht die Steuerebene des Amazon-EKS-Clusters.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- Einzelheiten zur API finden Sie unter DeleteCluster AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteClusterin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DeleteFargateProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFargateProfile` verwendet wird.

CLI

AWS CLI

Beispiel 1: Erstellen eines EKS-Fargate-Profils für einen Selektor mit einem Namespace

Im folgenden Beispiel für `delete-fargate-profile` wird ein EKS-Fargate-Profil für einen Selektor mit einem Namespace erstellt.

```
aws eks delete-fargate-profile \
    --cluster-name my-eks-cluster \
    --fargate-profile-name my-fargate-profile
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/1ac72bb3-3fc6-2631-f1e1-98bff53bed62",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-03-19T11:48:39.975000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/role-name",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "default",
                "labels": {
                    "foo": "bar"
                }
            }
        ],
        "status": "DELETING",
        "tags": {}
    }
}
```

Weitere Informationen finden Sie unter AWS -Fargate-Profil – Löschen eines Fargate-Profils im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie DeleteFargateProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht ein AWS Fargate-Profil. Wenn Sie ein Fargate-Profil löschen, werden alle Pods, die auf Fargate ausgeführt werden und mit diesem Profil erstellt wurden, ebenfalls gelöscht.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on
 target "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DeleteFargateProfileV4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht ein AWS Fargate-Profil. Wenn Sie ein Fargate-Profil löschen, werden alle Pods, die auf Fargate ausgeführt werden und mit diesem Profil erstellt wurden, ebenfalls gelöscht.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on
 target "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
```

```
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DeleteFargateProfileV5).

Verwendung von **`DeleteNodegroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteNodegroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Beispiel 1: Löschen einer verwalteten Knotengruppe für einen Amazon-EKS-Cluster

Im folgenden Beispiel für `delete-nodegroup` wird eine verwaltete Knotengruppe für einen Amazon-EKS-Cluster gelöscht.

```
aws eks delete-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup
```

Ausgabe:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/1ec75f5f-0e21-dcc0-b46e-f9c442685cd8",
        "clusterName": "my-eks-cluster",
```

```
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-08T13:25:15.033000-04:00",
        "modifiedAt": "2024-04-08T13:25:31.252000-04:00",
        "status": "DELETING",
        "capacityType": "SPOT",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 5,
            "desiredSize": 4
        },
        "instanceTypes": [
            "t3.large"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {
            "my-eks-nodegroup-label-2": "value-2",
            "my-eks-nodegroup-label-1": "value-1"
        },
        "taints": [
            {
                "key": "taint-key-1",
                "value": "taint-value-1",
                "effect": "NO_EXECUTE"
            }
        ],
        "diskSize": 50,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 2
        },
        "tags": {
            "my-eks-nodegroup-key-1": "value-1",
            "my-eks-nodegroup-key-2": "value-2"
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie DeleteNodegroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht eine Amazon-EKS-Knotengruppe für einen Cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter DeleteNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht eine Amazon-EKS-Knotengruppe für einen Cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter DeleteNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeCluster`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Beschreiben eines aktiv ausgeführten EKS-Add-Ons in Ihrem Amazon-EKS-Cluster

Im folgenden Beispiel für `describe-cluster` wird das aktiv ausgeführte EKS-Add-On in Ihrem Amazon-EKS-Cluster beschrieben.

```
aws eks describe-cluster \
    --name my-eks-cluster
```

Ausgabe:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
```

```
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
```

```
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeClusterin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Amazon-EKS-Cluster zurück.

```
Get-EKSCluster -Name "PROD"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
```

```
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Einzelheiten zur API finden Sie unter DescribeCluster AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Amazon-EKS-Cluster zurück.

```
Get-EKSCluster -Name "PROD"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- Einzelheiten zur API finden Sie unter DescribeCluster AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeFargateProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFargateProfile` verwendet wird.

CLI

AWS CLI

Beschreiben eines Fargate-Profils

Im folgenden Beispiel für `describe-fargate-profile` wird ein Fargate-Profil beschrieben.

```
aws eks describe-fargate-profile \
    --cluster-name my-eks-cluster \
    --fargate-profile-name my-fargate-profile
```

Ausgabe:

```
{
    "fargateProfile": {
        "fargateProfileName": "my-fargate-profile",
        "fargateProfileArn": "arn:aws:eks:us-east-2:111122223333:fargateprofile/
my-eks-cluster/my-fargate-profile/96c766ce-43d2-f9c9-954c-647334391198",
        "clusterName": "my-eks-cluster",
        "createdAt": "2024-04-11T10:42:52.486000-04:00",
        "podExecutionRoleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-
cluster-farga-FargatePodExecutionRole-1htfAaJdJUEO",
        "subnets": [
            "subnet-09d912bb63ef21b9a",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-0e2907431c9988b72"
        ],
        "selectors": [
            {
                "namespace": "prod*",
                "labels": {
                    "labelname*?": "*value1"
                }
            },
            {
                "namespace": "*dev*",
                "labels": {
                    "labelname*?": "*value*"
                }
            }
        ],
        "status": "ACTIVE",
```

```
        "tags": {
            "eks-fargate-profile-key-2": "value-2",
            "eks-fargate-profile-key-1": "value-1"
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeFargateProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem AWS Fargate-Profil zurück.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeFargateProfileV4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem AWS Fargate-Profil zurück.

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeFargateProfileV5).

Verwendung von **`DescribeNodegroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeNodegroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Beschreiben einer verwalteten Knotengruppe für einen Amazon-EKS-Cluster

Im folgenden Beispiel für `describe-nodegroup` wird eine verwaltete Knotengruppe für einen Amazon-EKS-Cluster beschrieben.

```
aws eks describe-nodegroup \
    --cluster-name my-eks-cluster \
    --nodegroup-name my-eks-nodegroup
```

Ausgabe:

```
{
    "nodegroup": {
        "nodegroupName": "my-eks-nodegroup",
        "nodegroupArn": "arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-
cluster/my-eks-nodegroup/a8c75f2f-df78-a72f-4063-4b69af3de5b1",
        "clusterName": "my-eks-cluster",
        "version": "1.26",
        "releaseVersion": "1.26.12-20240329",
        "createdAt": "2024-04-08T11:42:10.555000-04:00",
        "modifiedAt": "2024-04-08T11:44:12.402000-04:00",
        "status": "ACTIVE",
        "capacityType": "ON_DEMAND",
        "scalingConfig": {
            "minSize": 1,
            "maxSize": 3,
            "desiredSize": 1
        },
        "instanceTypes": [
            "t3.medium"
        ],
        "subnets": [
            "subnet-0e2907431c9988b72",
            "subnet-04ad87f71c6e5ab4d",
            "subnet-09d912bb63ef21b9a"
        ],
        "amiType": "AL2_x86_64",
        "nodeRole": "arn:aws:iam::111122223333:role/role-name",
        "labels": {},
        "resources": {
            "autoScalingGroups": [
                {
                    "name": "eks-my-eks-nodegroup-a8c75f2f-df78-
a72f-4063-4b69af3de5b1"
                }
            ]
        },
        "diskSize": 20,
        "health": {
            "issues": []
        },
        "updateConfig": {
            "maxUnavailable": 1
        },
```

```
        "tags": {}
    }
}
```

- Einzelheiten zur API finden Sie DescribeNodegroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über eine Amazon-EKS-Knotengruppe zurück.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter DescribeNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über eine Amazon-EKS-Knotengruppe zurück.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- Einzelheiten zur API finden Sie unter DescribeNodegroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeUpdate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeUpdate` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Aktualisierung für einen Cluster

Im folgenden Beispiel für `describe-update` wird eine Aktualisierung für einen Cluster beschrieben.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id 10bddb13-a71b-425a-b0a6-71cd03e59161
```

Ausgabe:

```
{
    "update": {
        "id": "10bddb13-a71b-425a-b0a6-71cd03e59161",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "false"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            }
        ],
        "createdAt": "2024-03-14T10:01:26.297000-04:00",
        "errors": []
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren der Kubernetes-Version eines Amazon-EKS-Clusters im Benutzerhandbuch für Amazon EKS.

Beispiel 2: So beschreiben Sie eine Aktualisierung für einen Cluster

Im folgenden Beispiel für `describe-update` wird eine Aktualisierung für einen Cluster beschrieben.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id e4994991-4c0f-475a-a040-427e6da52966
```

Ausgabe:

```
{
    "update": {
        "id": "e4994991-4c0f-475a-a040-427e6da52966",
        "status": "Successful",
        "type": "AssociateEncryptionConfig",
        "params": [
            {
                "type": "EncryptionConfig",
                "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
            }
        ],
        "createdAt": "2024-03-14T11:01:26.297000-04:00",
        "errors": []
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren der Kubernetes-Version eines Amazon-EKS-Clusters im Benutzerhandbuch für Amazon EKS.

Beispiel 3: So beschreiben Sie eine Aktualisierung für einen Cluster

Im folgenden Beispiel für `describe-update` wird eine Aktualisierung für einen Cluster beschrieben.

```
aws eks describe-update \
    --name my-eks-cluster \
    --update-id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

Ausgabe:

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.29"
            },
            {
```

```
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
        "createdAt": "2024-03-14T12:05:26.297000-04:00",
        "errors": []
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren der Kubernetes-Version eines Amazon-EKS-Clusters im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie DescribeUpdatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über ein Update für Ihren Amazon-EKS-Cluster oder die zugewiesene verwaltete Knotengruppe zurück.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter DescribeUpdate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über ein Update für Ihren Amazon-EKS-Cluster oder die zugewiesene verwaltete Knotengruppe zurück.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter DescribeUpdate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListClusters`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListClusters` verwendet wird.

CLI

AWS CLI

So listen Sie alle installierten Add-Ons im Amazon-EKS-Cluster „my-eks-cluster“ auf`

Im folgenden Beispiel für `list-clusters` werden alle installierten Add-Ons im Amazon-EKS-Cluster „my-eks-cluster“ aufgelistet.

```
aws eks list-clusters
```

Ausgabe:

```
{
    "clusters": [
        "prod",
        "qa",
        "stage",
        "my-eks-cluster"
    ]
}
```

- Einzelheiten zur API finden Sie ListClustersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Cluster in Ihrem AWS-Konto in der angegebenen Region auf.

```
Get-EKSClusterList
```

Ausgabe:

```
PROD
```

- Einzelheiten zur API finden Sie unter ListClusters AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Cluster in Ihrem AWS-Konto in der angegebenen Region auf.

```
Get-EKSClusterList
```

Ausgabe:

```
PROD
```

- Einzelheiten zur API finden Sie unter ListClusters AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListFargateProfiles`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListFargateProfiles` verwendet wird.

CLI

AWS CLI

So listen Sie alle Fargate-Profile im Amazon-EKS-Cluster „my-eks-cluster“ auf`

Im folgenden Beispiel für `list-fargate-profiles` werden alle Fargate-Profile im Amazon-EKS-Cluster „my-eks-cluster“ aufgelistet.

```
aws eks list-fargate-profiles \
    --cluster-name my-eks-cluster
```

Ausgabe:

```
{
    "fargateProfileNames": [
        "my-fargate-profile"
    ]
}
```

- Einzelheiten zur API finden Sie ListFargateProfilesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die AWS Fargate-Profile auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Ausgabe:

```
EKSFargate
EKSFargateProfile
```

- Einzelheiten zur API finden Sie unter ListFargateProfiles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die AWS Fargate-Profile auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Ausgabe:

```
EKSFargate
EKSFargateProfile
```

- Einzelheiten zur API finden Sie unter ListFargateProfiles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListNodegroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListNodegroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwalteten Kubernetes-Clustern

CLI

AWS CLI

Auflisten aller Knotengruppen in einem Amazon-EKS-Cluster

Im folgenden Beispiel für `list-nodegroups` werden alle Knotengruppen in einem Amazon-EKS-Cluster aufgelistet.

```
aws eks list-nodegroups \
    --cluster-name my-eks-cluster
```

Ausgabe:

```
{
    "nodegroups": [
        "my-eks-managed-node-group",
        "my-eks-nodegroup"
    ]
}
```

- Einzelheiten zur API finden Sie ListNodegroupsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Knotengruppen auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSNodegroupList -ClusterName PROD
```

Ausgabe:

```
ProdEKSNodeGroup
```

- Einzelheiten zur API finden Sie unter ListNodegroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Knotengruppen auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSNodegroupList -ClusterName PROD
```

Ausgabe:

```
ProdEKSNodeGroup
```

- Einzelheiten zur API finden Sie unter ListNodegroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListTagsForResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForResource` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Tags für einen Amazon-EKS-Cluster-ARN auf

Im folgenden Beispiel für `list-tags-for-resource` werden alle Tags für einen Amazon-EKS-Cluster-ARN aufgelistet.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster
```

Ausgabe:

```
{
    "tags": {
        "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
        "alpha.eksctl.io/cluster-name": "my-eks-cluster",
        "karpenter.sh/discovery": "my-eks-cluster",
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
        "auto-delete": "no",
        "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
        "EKS-Cluster-Name": "my-eks-cluster",
        "alpha.eksctl.io/cluster-oidc-enabled": "true",
        "aws:cloudformation:logical-id": "ControlPlane",
        "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
        "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
    }
}
```

Beispiel 2: So listen Sie alle Tags für einen Amazon-EKS-Knotengruppen-ARN auf

Im folgenden Beispiel für `list-tags-for-resource` werden alle Tags für einen Amazon-EKS-Knotengruppen-ARN aufgelistet.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c
```

Ausgabe:

```
{
    "tags": {
        "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-nodegroup-my-eks-
managed-node-group",
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-nodegroup-my-eks-managed-node-
group/eaa20310-e219-11ee-b851-0ab9ad8228ff",
```

```
        "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
        "EKS-Cluster-Name": "my-eks-cluster",
        "alpha.eksctl.io/nodegroup-type": "managed",
        "NodeGroup Name 1": "my-eks-managed-node-group",
        "k8s.io/cluster-autoscaler/enabled": "true",
        "nodegroup-role": "worker",
        "alpha.eksctl.io/cluster-name": "my-eks-cluster",
        "alpha.eksctl.io/nodegroup-name": "my-eks-managed-node-group",
        "karpenter.sh/discovery": "my-eks-cluster",
        "NodeGroup Name 2": "AmazonLinux-Linux-Managed-NG-v1-26-v1",
        "auto-delete": "no",
        "k8s.io/cluster-autoscaler/my-eks-cluster": "owned",
        "aws:cloudformation:logical-id": "ManagedNodeGroup",
        "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z"
    }
}
```

Beispiel 3: So listen Sie alle Tags für einen Fargate-Profil-ARN in Amazon EKS auf

Im folgenden Beispiel für `list-tags-for-resource` werden alle Tags für einen Fargate-Profil-ARN in Amazon EKS aufgelistet.

```
aws eks list-tags-for-resource  \
    --resource-arn arn:aws:eks:us-east-2:111122223333:fargateprofile/my-eks-
cluster/my-fargate-profile/d6c76780-e541-0725-c816-36754cab734b
```

Ausgabe:

```
{
    "tags": {
        "eks-fargate-profile-key-2": "value-2",
        "eks-fargate-profile-key-1": "value-1"
    }
}
```

Beispiel 4: Um alle Tags für einen Amazon Add-on EKS-ARN aufzulisten

Das folgende `list-tags-for-resource` Beispiel listet alle Tags für einen Amazon Add-on EKS-ARN auf.

```
aws eks list-tags-for-resource \
```

```
    --resource-arn arn:aws:eks:us-east-2:111122223333:addon/my-eks-cluster/vpc-cni/0ec71efc-98dd-3203-60b0-4b939b2a5e5f
```

Ausgabe:

```
{
    "tags": {
        "eks-addon-key-2": "value-2",
        "eks-addon-key-1": "value-1"
    }
}
```

Beispiel 5: So listen Sie alle Tags für einen OIDC-Identitätsanbieter-ARN in Amazon EKS auf

Im folgenden Beispiel für `list-tags-for-resource` werden alle Tags für einen OIDC-Identitätsanbieter-ARN in Amazon EKS aufgelistet.

```
aws eks list-tags-for-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:identityproviderconfig/my-eks-cluster/oidc/my-identity-provider/8ac76722-78e4-cec1-ed76-d49eea058622
```

Ausgabe:

```
{
    "tags": {
        "my-identity-provider": "test"
    }
}
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die Tags für eine Amazon-EKS-Ressource auf.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Ausgabe:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die Tags für eine Amazon-EKS-Ressource auf.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Ausgabe:

```
Key  Value
---  -----
Name EKSPRODCLUSTER
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListUpdates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListUpdates` verwendet wird.

CLI

AWS CLI

So listen Sie die Aktualisierungen für einen Cluster auf

In diesem Beispielbefehl werden die laufenden Aktualisierungen für einen Cluster namens `example` in Ihrer Standardregion aufgelistet.

Befehl:

```
aws eks list-updates --name example
```

Ausgabe:

```
{
```

```
    "updateIds": [
        "10bddb13-a71b-425a-b0a6-71cd03e59161"
    ]
}
```

- Einzelheiten zur API finden Sie ListUpdatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet listet die Updates auf, die einem Amazon EKS-Cluster oder einer verwalteten Knotengruppe in Ihrer AWS-Konto, in der angegebenen Region, zugeordnet sind.

```
Get-EKSUpdateList -Name "PROD"
```

Ausgabe:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Einzelheiten zur API finden Sie unter ListUpdates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet listet die Updates auf, die einem Amazon EKS-Cluster oder einer verwalteten Knotengruppe in Ihrer AWS-Konto, in der angegebenen Region, zugeordnet sind.

```
Get-EKSUpdateList -Name "PROD"
```

Ausgabe:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- Einzelheiten zur API finden Sie unter ListUpdates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`TagResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie die angegebenen Tags einem Amazon-EKS-Cluster hinzu

Im folgenden Beispiel für `tag-resource` werden die angegebenen Tags einem Amazon-EKS-Cluster hinzugefügt.

```
aws eks tag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \
    --tag 'my-eks-cluster-test-1=test-value-1,my-eks-cluster-dev-1=dev-value-2'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So fügen Sie die angegebenen Tags einer Amazon-EKS-Knotengruppe hinzu

Im folgenden Beispiel für `tag-resource` werden die angegebenen Tags einer Amazon-EKS-Knotengruppe hinzugefügt.

```
aws eks tag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \
    --tag 'my-eks-nodegroup-test-1=test-value-1,my-eks-nodegroup-dev-1=dev-
value-2'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet ordnet die angegebenen Tags einer Ressource mit dem angegebenen resourceArn zu.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -Tag @{Name = "EKSPRODCLUSTER"}
```

- Einzelheiten zur API finden Sie unter TagResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet ordnet die angegebenen Tags einer Ressource mit dem angegebenen resourceArn zu.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -Tag @{Name = "EKSPRODCLUSTER"}
```

- Einzelheiten zur API finden Sie unter TagResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UntagResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResource` verwendet wird.

CLI

AWS CLI

Beispiel 1: So löschen Sie die angegebenen Tags aus einem Amazon-EKS-Cluster

Im folgenden Beispiel für `untag-resource` werden die angegebenen Tags aus einem Amazon-EKS-Cluster gelöscht.

```
aws eks untag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster \
    --tag-keys "my-eks-cluster-test-1" "my-eks-cluster-dev-1"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So löschen Sie die angegebenen Tags aus einer Amazon-EKS-Knotengruppe

Im folgenden Beispiel für `untag-resource` werden die angegebenen Tags aus einer Amazon-EKS-Knotengruppe gelöscht.

```
aws eks untag-resource \
    --resource-arn arn:aws:eks:us-east-2:111122223333:nodegroup/my-eks-cluster/
my-eks-managed-node-group/60c71ed2-2cfb-020f-a5f4-ad32477f198c \
    --tag-keys "my-eks-nodegroup-test-1" "my-eks-nodegroup-dev-1"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet löscht angegebene Tags aus einer EKS-Ressource.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/
PROD" -TagKey "Name"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter UntagResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet löscht angegebene Tags aus einer EKS-Ressource.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/
PROD" -TagKey "Name"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter UntagResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateClusterConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateClusterConfig` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie den Cluster-Endpunktzugriff

In diesem Beispielbefehl wird ein Cluster aktualisiert, um den öffentlichen Endpunktzugriff zu deaktivieren und privaten Endpunktzugriff zu aktivieren.

Befehl:

```
aws eks update-cluster-config --name example \
--resources-vpc-config endpointPublicAccess=false,endpointPrivateAccess=true
```

Ausgabe:

```
{
    "update": {
        "id": "ec883c93-2e9e-407c-a22f-8f6fa6e67d4f",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "false"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            }
        ],
        "createdAt": 1565806986.506,
        "errors": []
    }
}
```

So aktivieren Sie die Protokollierung für einen Cluster

In diesem Beispielbefehl werden alle Protokollierungstypen der Cluster-Steuerebene für einen Cluster namens `example` aktiviert.

Befehl:

```
aws eks update-cluster-config --name example \
--logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Ausgabe:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

- Einzelheiten zur API finden Sie UpdateClusterConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert eine Konfiguration von Amazon EKS Cluster. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter UpdateClusterConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert eine Konfiguration von Amazon EKS Cluster. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- Einzelheiten zur API finden Sie unter UpdateClusterConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateClusterVersion`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateClusterVersion` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie einen Amazon-EKS-Cluster mit dem Namen „my-eks-cluster“ auf die angegebene Kubernetes-Version

Im folgenden Beispiel für `update-cluster-version` wird ein Amazon-EKS-Cluster auf die angegebene Kubernetes-Version aktualisiert.

```
aws eks update-cluster-version \
    --name my-eks-cluster \
    --kubernetes-version 1.27
```

Ausgabe:

```
{
    "update": {
        "id": "e4091a28-ea14-48fd-a8c7-975aeb469e8a",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.27"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.16"
            }
        ],
        "createdAt": "2024-04-12T16:56:01.082000-04:00",
        "errors": []
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren der Kubernetes-Version eines Amazon-EKS-Clusters im Benutzerhandbuch für Amazon EKS.

- Einzelheiten zur API finden Sie UpdateClusterVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Cmdlet aktualisiert einen Amazon-EKS-Cluster auf die angegebene Kubernetes-Version. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Ausgabe:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Einzelheiten zur API finden Sie unter UpdateClusterVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Cmdlet aktualisiert einen Amazon-EKS-Cluster auf die angegebene Kubernetes-Version. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Ausgabe:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- Einzelheiten zur API finden Sie unter UpdateClusterVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Amazon EKS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon EKS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon EKS aufrufen oder mit anderen kombinieren AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit verwalteten Kubernetes-Clustern

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
```

```
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
```

```
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
```

```
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."
```

```
# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
```

```
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
```

```
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
```

```
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi
```

```
# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks

- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Codebeispiele für Elastic Beanstalk mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elastic Beanstalk mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für AWS Elastic Beanstalk – Weitere Informationen zu Elastic Beanstalk.
- API-Referenz für Elastic Beanstalk – Details zu allen verfügbaren Aktionen von Elastic Beanstalk.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Elastic Beanstalk AWS SDKs
  - Aktionen für Elastic Beanstalk mit AWS SDKs
    - Verwenden Sie DescribeApplicationsmit einem AWS SDK oder CLI
    - Verwenden Sie ListAvailableSolutionStacksmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateApplicationmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung von Elastic Beanstalk AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von mit SDKs AWS Elastic Beanstalk verwenden können. AWS

Beispiele

- Aktionen für Elastic Beanstalk mit AWS SDKs
  - Verwenden Sie DescribeApplicationsmit einem AWS SDK oder CLI
  - Verwenden Sie ListAvailableSolutionStacksmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateApplicationmit einem AWS SDK oder CLI

Aktionen für Elastic Beanstalk mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Elastic Beanstalk Beanstalk-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Elastic Beanstalk -API-Referenz.

Beispiele

- Verwenden Sie DescribeApplicationsmit einem AWS SDK oder CLI
- Verwenden Sie ListAvailableSolutionStacksmit einem AWS SDK oder CLI
- Verwenden Sie UpdateApplicationmit einem AWS SDK oder CLI

Verwenden Sie **`DescribeApplications`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeApplications` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine Liste von Anwendungen an

Mit dem folgenden Befehl werden Informationen über Anwendungen in der aktuellen Region abgerufen:

```
aws elasticbeanstalk describe-applications
```

Ausgabe:

```
{
    "Applications": [
        {
            "ApplicationName": "ruby",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-13T21:05:44.376Z",
            "Versions": [
                "Sample Application"
            ],
            "DateCreated": "2015-08-13T21:05:44.376Z"
        },
        {
            "ApplicationName": "pythonsample",
            "Description": "Application created from the EB CLI using \"eb init
\"",
            "Versions": [
                "Sample Application"
            ],
            "DateCreated": "2015-08-13T19:05:43.637Z",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-13T19:05:43.637Z"
        },
        {
            "ApplicationName": "nodejs-example",
            "ConfigurationTemplates": [],
            "DateUpdated": "2015-08-06T17:50:02.486Z",
            "Versions": [
                "add elasticache",
                "First Release"
            ],
            "DateCreated": "2015-08-06T17:50:02.486Z"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeApplicationsin der AWS CLI Befehlsreferenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
```

```
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Error listing environments for application
 #{application_name}: #{e.message}")
  end
end
```

- Einzelheiten zur API finden Sie DescribeApplicationsin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`ListAvailableSolutionStacks`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAvailableSolutionStacks` verwendet wird.

CLI

AWS CLI

So zeigen Sie Lösungs-Stacks an

Mit dem folgenden Befehl werden Lösungs-Stacks für alle derzeit verfügbaren und alle in der Vergangenheit verwendeten Plattformkonfigurationen aufgerufen:

```
aws elasticbeanstalk list-available-solution-stacks
```

Ausgabe (gekürzt):

```
{
    "SolutionStacks": [
        "64bit Amazon Linux 2015.03 v2.0.0 running Node.js",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.6",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.5",
        "64bit Amazon Linux 2015.03 v2.0.0 running PHP 5.4",
        "64bit Amazon Linux 2015.03 v2.0.0 running Python 3.4",
        "64bit Amazon Linux 2015.03 v2.0.0 running Python 2.7",
        "64bit Amazon Linux 2015.03 v2.0.0 running Python",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Puma)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.2 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Puma)",
```

```
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.1 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Puma)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 2.0 (Passenger
 Standalone)",
        "64bit Amazon Linux 2015.03 v2.0.0 running Ruby 1.9.3",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 8 Java 8",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 7",
        "64bit Amazon Linux 2015.03 v2.0.0 running Tomcat 7 Java 6",
        "64bit Windows Server Core 2012 R2 running IIS 8.5",
        "64bit Windows Server 2012 R2 running IIS 8.5",
        "64bit Windows Server 2012 running IIS 8",
        "64bit Windows Server 2008 R2 running IIS 7.5",
        "64bit Amazon Linux 2015.03 v2.0.0 running Docker 1.6.2",
        "64bit Amazon Linux 2015.03 v2.0.0 running Multi-container Docker 1.6.2
 (Generic)",
        "64bit Debian jessie v2.0.0 running GlassFish 4.1 Java 8 (Preconfigured -
 Docker)",
        "64bit Debian jessie v2.0.0 running GlassFish 4.0 Java 7 (Preconfigured -
 Docker)",
        "64bit Debian jessie v2.0.0 running Go 1.4 (Preconfigured - Docker)",
        "64bit Debian jessie v2.0.0 running Go 1.3 (Preconfigured - Docker)",
        "64bit Debian jessie v2.0.0 running Python 3.4 (Preconfigured - Docker)",
    ],
    "SolutionStackDetails": [
        {
            "PermittedFileTypes": [
                "zip"
            ],
            "SolutionStackName": "64bit Amazon Linux 2015.03 v2.0.0 running
 Node.js"
        },
        ...
    ]
}
```

- Einzelheiten zur API finden Sie ListAvailableSolutionStacksin der AWS CLI Befehlsreferenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
  end

  private
```

```
  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- Einzelheiten zur API finden Sie ListAvailableSolutionStacksin der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`UpdateApplication`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateApplication` verwendet wird.

CLI

AWS CLI

So ändern Sie die Beschreibung einer Anwendung

Mit dem folgenden Befehl wird die Beschreibung einer Anwendung namens `my-app` aktualisiert:

```
aws elasticbeanstalk update-application --application-name my-app --
description "my Elastic Beanstalk application"
```

Ausgabe:

```
{
    "Application": {
        "ApplicationName": "my-app",
        "Description": "my Elastic Beanstalk application",
        "Versions": [
            "2fba-stage-150819_234450",
            "bf07-stage-150820_214945",
            "93f8",
            "fd7c-stage-150820_000431",
            "22a0-stage-150819_185942"
```

```
        ],
        "DateCreated": "2015-08-13T19:15:50.449Z",
        "ConfigurationTemplates": [],
        "DateUpdated": "2015-08-20T22:34:56.195Z"
    }
}
```

- Einzelheiten zur API finden Sie UpdateApplicationin der AWS CLI Befehlsreferenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key:
key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end

  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
```

```
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie UpdateApplicationin der AWS SDK für Ruby API-Referenz.

Codebeispiele für Elastic Load Balancing — Version 1 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Elastic Load Balancing — Version 1 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Benutzerhandbuch für Elastic Load Balancing – Version 1 – Weitere Informationen zu Elastic Load Balancing – Version 1.
- API-Referenz für Elastic Load Balancing – Version 1 – Details zu allen verfügbaren Aktionen von Elastic Load Balancing – Version 1.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Elastic Load Balancing — Version 1 mit AWS SDKs
  - Aktionen für Elastic Load Balancing — Version 1 mit AWS SDKs
    - Verwendung von AddTags mit einer CLI
    - Verwendung von ApplySecurityGroupsToLoadBalancer mit einer CLI
    - Verwendung von AttachLoadBalancerToSubnets mit einer CLI
    - Verwendung von ConfigureHealthCheck mit einer CLI
    - Verwendung von CreateAppCookieStickinessPolicy mit einer CLI
    - Verwendung von CreateLbCookieStickinessPolicy mit einer CLI
    - Verwendung von CreateLoadBalancer mit einer CLI
    - Verwendung von CreateLoadBalancerListeners mit einer CLI
    - Verwendung von CreateLoadBalancerPolicy mit einer CLI
    - Verwendung von DeleteLoadBalancer mit einer CLI
    - Verwendung von DeleteLoadBalancerListeners mit einer CLI
    - Verwendung von DeleteLoadBalancerPolicy mit einer CLI
    - Verwendung von DeregisterInstancesFromLoadBalancer mit einer CLI
    - Verwendung von DescribeInstanceHealth mit einer CLI
    - Verwendung von DescribeLoadBalancerAttributes mit einer CLI
    - Verwendung von DescribeLoadBalancerPolicies mit einer CLI
    - Verwendung von DescribeLoadBalancerPolicyTypes mit einer CLI
    - Verwendung von DescribeLoadBalancers mit einer CLI
    - Verwendung von DescribeTags mit einer CLI

- Verwendung von DetachLoadBalancerFromSubnets mit einer CLI
- Verwendung von DisableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von EnableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von RegisterInstancesWithLoadBalancer mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetLoadBalancerListenerSslCertificate mit einer CLI
- Verwendung von SetLoadBalancerPoliciesForBackendServer mit einer CLI
- Verwendung von SetLoadBalancerPoliciesOfListener mit einer CLI

Grundlegende Beispiele für Elastic Load Balancing — Version 1 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Elastic Load Balancing - Version 1 mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Elastic Load Balancing — Version 1 mit AWS SDKs
  - Verwendung von AddTags mit einer CLI
  - Verwendung von ApplySecurityGroupsToLoadBalancer mit einer CLI
  - Verwendung von AttachLoadBalancerToSubnets mit einer CLI
  - Verwendung von ConfigureHealthCheck mit einer CLI
  - Verwendung von CreateAppCookieStickinessPolicy mit einer CLI
  - Verwendung von CreateLbCookieStickinessPolicy mit einer CLI
  - Verwendung von CreateLoadBalancer mit einer CLI
  - Verwendung von CreateLoadBalancerListeners mit einer CLI
  - Verwendung von CreateLoadBalancerPolicy mit einer CLI
  - Verwendung von DeleteLoadBalancer mit einer CLI
  - Verwendung von DeleteLoadBalancerListeners mit einer CLI
  - Verwendung von DeleteLoadBalancerPolicy mit einer CLI
  - Verwendung von DeregisterInstancesFromLoadBalancer mit einer CLI

- Verwendung von DescribeInstanceHealth mit einer CLI
- Verwendung von DescribeLoadBalancerAttributes mit einer CLI
- Verwendung von DescribeLoadBalancerPolicies mit einer CLI
- Verwendung von DescribeLoadBalancerPolicyTypes mit einer CLI
- Verwendung von DescribeLoadBalancers mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DetachLoadBalancerFromSubnets mit einer CLI
- Verwendung von DisableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von EnableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von RegisterInstancesWithLoadBalancer mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetLoadBalancerListenerSslCertificate mit einer CLI
- Verwendung von SetLoadBalancerPoliciesForBackendServer mit einer CLI
- Verwendung von SetLoadBalancerPoliciesOfListener mit einer CLI

Aktionen für Elastic Load Balancing — Version 1 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Elastic Load Balancing — Version 1-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Elastic Load Balancing – Version 1.

Beispiele

- Verwendung von AddTags mit einer CLI
- Verwendung von ApplySecurityGroupsToLoadBalancer mit einer CLI
- Verwendung von AttachLoadBalancerToSubnets mit einer CLI
- Verwendung von ConfigureHealthCheck mit einer CLI
- Verwendung von CreateAppCookieStickinessPolicy mit einer CLI
- Verwendung von CreateLbCookieStickinessPolicy mit einer CLI
- Verwendung von CreateLoadBalancer mit einer CLI

- Verwendung von CreateLoadBalancerListeners mit einer CLI
- Verwendung von CreateLoadBalancerPolicy mit einer CLI
- Verwendung von DeleteLoadBalancer mit einer CLI
- Verwendung von DeleteLoadBalancerListeners mit einer CLI
- Verwendung von DeleteLoadBalancerPolicy mit einer CLI
- Verwendung von DeregisterInstancesFromLoadBalancer mit einer CLI
- Verwendung von DescribeInstanceHealth mit einer CLI
- Verwendung von DescribeLoadBalancerAttributes mit einer CLI
- Verwendung von DescribeLoadBalancerPolicies mit einer CLI
- Verwendung von DescribeLoadBalancerPolicyTypes mit einer CLI
- Verwendung von DescribeLoadBalancers mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DetachLoadBalancerFromSubnets mit einer CLI
- Verwendung von DisableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von EnableAvailabilityZonesForLoadBalancer mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von RegisterInstancesWithLoadBalancer mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetLoadBalancerListenerSslCertificate mit einer CLI
- Verwendung von SetLoadBalancerPoliciesForBackendServer mit einer CLI
- Verwendung von SetLoadBalancerPoliciesOfListener mit einer CLI

Verwendung von **`AddTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddTags` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Load Balancer ein Tag hinzu

In diesem Beispiel werden dem angegebenen Load Balancer Tags hinzugefügt.

Befehl:

```
aws elb add-tags --load-balancer-name my-load-balancer --
tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

- Einzelheiten zur API finden Sie AddTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Hier werden die angegebenen Tags zum spezifizierten Load Balancer hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um ein Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Einzelheiten zur API finden Sie unter AddTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Hier werden die angegebenen Tags zum spezifizierten Load Balancer hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie verwenden, New-Object um ein Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
```

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- Einzelheiten zur API finden Sie unter AddTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **ApplySecurityGroupsToLoadBalancer** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ApplySecurityGroupsToLoadBalancer` verwendet wird.

CLI

AWS CLI

So ordnen Sie einem Load Balancer in einer VPC eine Sicherheitsgruppe zu

In diesem Beispiel wird dem angegebenen Load Balancer in einer VPC eine Sicherheitsgruppe zugeordnet.

Befehl:

```
aws elb apply-security-groups-to-load-balancer --load-balancer-name my-load-balancer --security-groups sg-fc448899
```

Ausgabe:

```
{
  "SecurityGroups": [
      "sg-fc448899"
  ]
}
```

- Einzelheiten zur API finden Sie ApplySecurityGroupsToLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die aktuelle Sicherheitsgruppe für den angegebenen Load Balancer durch die spezifizierte Sicherheitsgruppe ersetzt.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Ausgabe:

```
sg-87654321
```

Beispiel 2: Um die aktuelle Sicherheitsgruppe beizubehalten und eine zusätzliche Sicherheitsgruppe anzugeben, geben Sie sowohl die vorhandene als auch die neue Sicherheitsgruppe an.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Ausgabe:

```
sg-12345678
sg-87654321
```

- Einzelheiten zur API finden Sie unter ApplySecurityGroupsToLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die aktuelle Sicherheitsgruppe für den angegebenen Load Balancer durch die spezifizierte Sicherheitsgruppe ersetzt.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Ausgabe:

```
sg-87654321
```

Beispiel 2: Um die aktuelle Sicherheitsgruppe beizubehalten und eine zusätzliche Sicherheitsgruppe anzugeben, geben Sie sowohl die vorhandene als auch die neue Sicherheitsgruppe an.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Ausgabe:

```
sg-12345678
sg-87654321
```

- Einzelheiten zur API finden Sie unter ApplySecurityGroupsToLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachLoadBalancerToSubnets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachLoadBalancerToSubnets` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Load Balancer Subnetze hinzu

In diesem Beispiel wird das angegebene Subnetz den bereits konfigurierten Subnetzen für den entsprechenden Load Balancer hinzugefügt.

Befehl:

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-load-balancer --
subnets subnet-0ecac448
```

Ausgabe:

```
{
   "Subnets": [
       "subnet-15aaab61",
       "subnet-0ecac448"
   ]
}
```

- Einzelheiten zur API finden Sie AttachLoadBalancerToSubnetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Hier wird das angegebene Subnetz zu den Subnetzen hinzugefügt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe enthält die vollständige Liste der Subnetze.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-12345678
subnet-87654321
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancerToSubnets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Hier wird das angegebene Subnetz zu den Subnetzen hinzugefügt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe enthält die vollständige Liste der Subnetze.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-12345678
subnet-87654321
```

- Einzelheiten zur API finden Sie unter AttachLoadBalancerToSubnets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ConfigureHealthCheck`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ConfigureHealthCheck` verwendet wird.

CLI

AWS CLI

So geben Sie die Einstellungen für die Zustandsprüfungen für EC2-Backend-Instances an

In diesem Beispiel werden die Einstellungen für die Zustandsprüfungen angegeben, die zum Bewerten des Zustands von EC2-Instances im Backend verwenden werden.

Befehl:

```
aws elb configure-health-check --load-balancer-name my-load-balancer --health-check Target=HTTP:80/
png,Interval=30,UnhealthyThreshold=2,HealthyThreshold=2,Timeout=3
```

Ausgabe:

```
{
   "HealthCheck": {
       "HealthyThreshold": 2,
       "Interval": 30,
       "Target": "HTTP:80/png",
       "Timeout": 3,
       "UnhealthyThreshold": 2
   }
}
```

- Einzelheiten zur API finden Sie ConfigureHealthCheckin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Einstellungen für die Zustandsprüfung für den angegebenen Load Balancer konfiguriert.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
```

```
>> -HealthCheck_Timeout 3
```

Ausgabe:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Einzelheiten zur API finden Sie unter ConfigureHealthCheck AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Einstellungen für die Zustandsprüfung für den angegebenen Load Balancer konfiguriert.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Ausgabe:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- Einzelheiten zur API finden Sie unter ConfigureHealthCheck AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateAppCookieStickinessPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAppCookieStickinessPolicy` verwendet wird.

CLI

AWS CLI

So generieren Sie eine Stickiness-Richtlinie für Ihren HTTPS-Load-Balancer

In diesem Beispiel wird eine Stickiness-Richtlinie generiert, die den Gültigkeitsdauerwerten für Sticky Sessions des anwendungsgenerierten Cookies entspricht.

Befehl:

```
aws elb create-app-cookie-stickiness-policy --load-balancer-name my-load-balancer
 --policy-name my-app-cookie-policy --cookie-name my-app-cookie
```

- Einzelheiten zur API finden Sie CreateAppCookieStickinessPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Stickiness-Richtlinie erstellt, die sich an die Lebensdauer des angegebenen, von der Anwendung generierten Cookies für Sticky Sessions hält.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-app-cookie-policy -CookieName my-app-cookie
```

- Einzelheiten zur API finden Sie unter CreateAppCookieStickinessPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Stickiness-Richtlinie erstellt, die sich an die Lebensdauer des angegebenen, von der Anwendung generierten Cookies für Sticky Sessions hält.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-app-cookie-policy -CookieName my-app-cookie
```

- Einzelheiten zur API finden Sie unter CreateAppCookieStickinessPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateLbCookieStickinessPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLbCookieStickinessPolicy` verwendet wird.

CLI

AWS CLI

So generieren Sie eine auf Dauer basierende Stickiness-Richtlinie für Ihren HTTPS-Load-Balancer

In diesem Beispiel wird eine Stickiness-Richtlinie mit Gültigkeitsdauerwerten für Sticky Sessions generiert, die von der festgelegten Ablaufzeit abhängig sind.

Befehl:

```
aws elb create-lb-cookie-stickiness-policy --load-balancer-name my-load-balancer
 --policy-name my-duration-cookie-policy --cookie-expiration-period 60
```

- Einzelheiten zur API finden Sie CreateLbCookieStickinessPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions durch die festgelegte Ablaufzeit (in Sekunden) bestimmt wird).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy -CookieExpirationPeriod 60
```

Beispiel 2: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions von der Lebensdauer des Browsers (User-Agent) abhängt).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy
```

- Einzelheiten zur API finden Sie unter CreateLbCookieStickinessPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions durch die festgelegte Ablaufzeit (in Sekunden) bestimmt wird).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy -CookieExpirationPeriod 60
```

Beispiel 2: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions von der Lebensdauer des Browsers (User-Agent) abhängt).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName
 my-duration-cookie-policy
```

- Einzelheiten zur API finden Sie unter CreateLbCookieStickinessPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLoadBalancer` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen HTTP-Load-Balancer

In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

Befehl:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --subnets subnet-15aaab61 --security-groups sg-a61988c3
```

Ausgabe:

```
{
```

```
    "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"
}
```

In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener erstellt. EC2-Classic

Befehl:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --availability-zones us-west-2a us-west-2b
```

Ausgabe:

```
{
    "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

So erstellen Sie einen HTTPS-Load-Balancer

In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener in einer VPC erstellt.

Befehl:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" "Prot
certificate/my-server-cert" --subnets subnet-15aaab61 --security-groups sg-
a61988c3
```

Ausgabe:

```
{
    "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com"
}
```

In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener erstellt. EC2-Classic

Befehl:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80" "Prot
certificate/my-server-cert" --availability-zones us-west-2a us-west-2b
```

Ausgabe:

```
{
    "DNSName": "my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

So erstellen Sie einen internen Load Balancer

In diesem Beispiel wird ein interner Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

Befehl:

```
aws elb create-load-balancer --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
 --scheme internal --subnets subnet-a85db0df --security-groups sg-a61988c3
```

Ausgabe:

```
{
    "DNSName": "internal-my-load-balancer-123456789.us-west-2.elb.amazonaws.com"
}
```

- Einzelheiten zur API finden Sie CreateLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Beispiel 2: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener erstellt. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone
 us-west-2a` -Listener $httpListener
```

Ausgabe:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Beispiel 3: In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener erstellt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-
west-2a -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Beispiel 2: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener erstellt. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone
 us-west-2a` -Listener $httpListener
```

Ausgabe:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Beispiel 3: In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener erstellt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-
west-2a -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateLoadBalancerListeners`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLoadBalancerListeners` verwendet wird.

CLI

AWS CLI

So erstellen Sie HTTP-Listener für einen Load Balancer

In diesem Beispiel wird mithilfe des HTTP-Protokolls ein Listener für Ihren Load Balancer an Port 80 erstellt.

Befehl:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
```

So erstellen Sie HTTPS-Listener für einen Load Balancer

In diesem Beispiel wird mithilfe des HTTPS-Protokolls ein Listener für Ihren Load Balancer an Port 443 erstellt.

Befehl:

```
aws elb create-load-balancer-listeners --load-balancer-name my-load-balancer --
listeners "Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80"
```

- Einzelheiten zur API finden Sie CreateLoadBalancerListenersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer ein HTTPS-Listener hinzugefügt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerListeners AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer ein HTTPS-Listener hinzugefügt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerListeners AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateLoadBalancerPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLoadBalancerPolicy` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Richtlinie, die das Proxy-Protokoll auf einem Load Balancer aktiviert

In diesem Beispiel wird eine Richtlinie erstellt, die das Proxy-Protokoll auf dem angegebenen Load Balancer aktiviert.

Befehl:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-ProxyProtocol-policy --policy-type-name ProxyProtocolPolicyType --
policy-attributes AttributeName=ProxyProtocol,AttributeValue=true
```

So erstellen Sie eine SSL-Aushandlungsrichtlinie mit der empfohlenen Sicherheitsrichtlinie

In diesem Beispiel wird unter Verwendung der empfohlenen Sicherheitsrichtlinie eine SSL-Aushandlungsrichtlinie für den angegebenen HTTPS-Load-Balancer erstellt.

Befehl:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-
balancer --policy-name my-SSLNegotiation-policy --policy-type-
```

```
name SSLNegotiationPolicyType --policy-attributes AttributeName=Reference-
Security-Policy,AttributeValue=ELBSecurityPolicy-2015-03
```

So erstellen Sie eine SSL-Aushandlungsrichtlinie mit einer benutzerdefinierten Sicherheitsrichtlinie

In diesem Beispiel wird mithilfe einer benutzerdefinierten Sicherheitsrichtlinie eine SSL-Aushandlungsrichtlinie für den HTTPS-Load-Balancer erstellt, indem die Protokolle und Chiffren aktiviert werden.

Befehl:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-
balancer --policy-name my-SSLNegotiation-policy --policy-type-
name SSLNegotiationPolicyType --policy-attributes AttributeName=Protocol-
SSLv3,AttributeValue=true AttributeName=Protocol-
TLSv1.1,AttributeValue=true AttributeName=DHE-RSA-AES256-
SHA256,AttributeValue=true AttributeName=Server-Defined-Cipher-
Order,AttributeValue=true
```

So erstellen Sie eine Richtlinie für öffentliche Schlüssel

In diesem Beispiel wird eine Richtlinie für öffentliche Schlüssel erstellt.

Befehl:

```
aws elb create-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-PublicKey-policy --policy-type-name PublicKeyPolicyType --policy-
attributes AttributeName=PublicKey,AttributeValue=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgK
+dS74kj//c6x7ROtusUaeQCTgIUkayttRDWchuqo1pHC1u
+n5xxXnBBe2ejbb2WRsKIQ5rXEeixsjFpFsojpSQKkzhVGI6mJVZBJDVKSHmswnwLBdofLhzvllpovBPTHe
+o4haAWvDBALJU0pkSI1FecPHcs2hwxf14zHoXy1e2k36A64nXW43wtfx5qcVSIxtCEOjnYRg7RPvybaGfQ
+v6Iaxb/+7J5kEvZhTFQId+bSiJImF1FSUT1W1xwzBZPUbcUkkXDj45vC2s3Z8E
+Lk7a3uZhvsQHLZnrfuWjBWGWvZ/MhZYgEXAMPLE
```

So erstellen Sie eine Authentifizierungsrichtlinie für Backend-Server

In diesem Beispiel wird eine Authentifizierungsrichtlinie für Backend-Server erstellt, die eine Authentifizierung mithilfe einer Richtlinie für öffentliche Schlüssel auf Ihrer Backend-Instance ermöglicht.

Befehl:

```
aws elb create-load-balancer-policy --load-balancer-name my-
load-balancer --policy-name my-authentication-policy --policy-
type-name BackendServerAuthenticationPolicyType --policy-
attributes AttributeName=PublicKeyPolicyName,AttributeValue=my-PublicKey-policy
```

- Einzelheiten zur API finden Sie CreateLoadBalancerPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Proxyprotokollrichtlinie für einen angegebenen Load Balancer erstellt.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -
Property @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Proxyprotokollrichtlinie für einen angegebenen Load Balancer erstellt.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -
Property @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancerPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLoadBalancer` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Load Balancer

In diesem Beispiel wird der angegebene Load Balancer gelöscht.

Befehl:

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteLoadBalancerListeners`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLoadBalancerListeners` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Listener von einem Load Balancer

In diesem Beispiel wird der Listener für den angegebenen Port vom angegebenen Load Balancer gelöscht.

Befehl:

```
aws elb delete-load-balancer-listeners --load-balancer-name my-load-balancer --
load-balancer-ports 80
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerListenersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Listener auf Port 80 für den angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 80
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener
 (DeleteLoadBalancerListeners)" on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerListeners AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Listener auf Port 80 für den angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 80
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ELBLoadBalancerListener
 (DeleteLoadBalancerListeners)" on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerListeners AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteLoadBalancerPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLoadBalancerPolicy` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Richtlinie von einem Load Balancer

In diesem Beispiel wird die angegebene Richtlinie aus dem angegebenen Load Balancer gelöscht. Die Richtlinie darf bei keinem Listener aktiviert sein.

Befehl:

```
aws elb delete-load-balancer-policy --load-balancer-name my-load-balancer --
policy-name my-duration-cookie-policy
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie aus dem angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie aus dem angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancerPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterInstancesFromLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterInstancesFromLoadBalancer` verwendet wird.

CLI

AWS CLI

So melden Sie Instances von einem Load Balancer ab

In diesem Beispiel wird die Registrierung der angegebenen Instance beim angegebenen Load Balancer aufgehoben.

Befehl:

```
aws elb deregister-instances-from-load-balancer --load-balancer-name my-load-balancer --instances i-d6f6fae3
```

Ausgabe:

```
{
    "Instances": [
        {
            "InstanceId": "i-207d9717"
        },
        {
            "InstanceId": "i-afefb49b"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DeregisterInstancesFromLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Einzelheiten zur API finden Sie unter DeregisterInstancesFromLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- Einzelheiten zur API finden Sie unter DeregisterInstancesFromLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstanceHealth`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceHealth` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Zustand der Instances für einen Load Balancer

In diesem Beispiel wird der Zustand der Instances für den angegebenen Load Balancer beschrieben.

Befehl:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer
```

Ausgabe:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-207d9717",
          "ReasonCode": "N/A",
          "State": "InService",
          "Description": "N/A"
      },
      {
          "InstanceId": "i-afefb49b",
          "ReasonCode": "N/A",
          "State": "InService",
          "Description": "N/A"
      }
  ]
}
```

So beschreiben Sie den Zustand einer Instance für einen Load Balancer

In diesem Beispiel wird der Zustand der angegebenen Instance für den angegebenen Load Balancer beschrieben.

Befehl:

```
aws elb describe-instance-health --load-balancer-name my-load-balancer --
instances i-7299c809
```

Nachstehend ist eine Beispielantwort für eine Instance abgebildet, die gerade registriert wird.

Ausgabe:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-7299c809",
          "ReasonCode": "ELB",
          "State": "OutOfService",
          "Description": "Instance registration is still in progress."
    }
  ]
}
```

Nachstehend ist eine Beispielantwort für eine fehlerhafte Instance abgebildet.

Ausgabe:

```
{
  "InstanceStates": [
      {
          "InstanceId": "i-7299c809",
          "ReasonCode": "Instance",
          "State": "OutOfService",
          "Description": "Instance has failed at least the UnhealthyThreshold
 number of health checks consecutively."
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeInstanceHealthin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status der Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Description                    InstanceId                     ReasonCode
         State
-----------                    ----------                     ----------
         -----
N/A                            i-87654321                     N/A
         InService
Instance has failed at lea... i-12345678                     Instance
         OutOfService
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des Status der angegebenen Instance angezeigt.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Ausgabe:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Einzelheiten zur API finden Sie unter DescribeInstanceHealth AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status der Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Description                    InstanceId                     ReasonCode
         State
-----------                    ----------                     ----------
         -----
```

```
N/A                             i-87654321                    N/A
         InService
Instance has failed at lea... i-12345678                    Instance
         OutOfService
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des Status der angegebenen Instance angezeigt.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Ausgabe:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- Einzelheiten zur API finden Sie unter DescribeInstanceHealth AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancerAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancerAttributes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Attribute eines Load Balancers

In diesem Beispiel werden die Attribute des angegebenen Load Balancers beschrieben.

Befehl:

```
aws elb describe-load-balancer-attributes --load-balancer-name my-load-balancer
```

Ausgabe:

```
{
  "LoadBalancerAttributes": {
      "ConnectionDraining": {
          "Enabled": false,
          "Timeout": 300
      },
      "CrossZoneLoadBalancing": {
          "Enabled": true
      },
      "ConnectionSettings": {
          "IdleTimeout": 30
      },
      "AccessLog": {
          "Enabled": false
    }
  }
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancerAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Attribute für den angegebenen Load Balancer beschrieben.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Attribute für den angegebenen Load Balancer beschrieben.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancerPolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancerPolicies` verwendet wird.

CLI

AWS CLI

So beschreiben Sie alle Richtlinien, die einem Load Balancer zugeordnet sind

In diesem Beispiel werden alle Richtlinien beschrieben, die dem angegebenen Load Balancer zugeordnet sind.

Befehl:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer
```

Ausgabe:

```
{
  "PolicyDescriptions": [
    {
      "PolicyAttributeDescriptions": [
```

```
        {
          "AttributeName": "ProxyProtocol",
          "AttributeValue": "true"
        }
      ],
      "PolicyName": "my-ProxyProtocol-policy",
      "PolicyTypeName": "ProxyProtocolPolicyType"
    },
    {
        "PolicyAttributeDescriptions": [
            {
                "AttributeName": "CookieName",
                "AttributeValue": "my-app-cookie"
            }
        ],
        "PolicyName": "my-app-cookie-policy",
        "PolicyTypeName": "AppCookieStickinessPolicyType"
    },
    {
      "PolicyAttributeDescriptions": [
        {
          "AttributeName": "CookieExpirationPeriod",
          "AttributeValue": "60"
        }
      ],
      "PolicyName": "my-duration-cookie-policy",
      "PolicyTypeName": "LBCookieStickinessPolicyType"
    },
    .
    .
    .
  ]
}
```

So beschreiben Sie eine bestimmte Richtlinie, die einem Load Balancer zugeordnet ist

In diesem Beispiel wird die angegebene Richtlinie beschrieben, die dem angegebenen Load Balancer zugeordnet ist.

Befehl:

```
aws elb describe-load-balancer-policies --load-balancer-name my-load-balancer --
policy-name my-authentication-policy
```

Ausgabe:

```
{
  "PolicyDescriptions": [
      {
          "PolicyAttributeDescriptions": [
              {
                  "AttributeName": "PublicKeyPolicyName",
                  "AttributeValue": "my-PublicKey-policy"
              }
          ],
          "PolicyName": "my-authentication-policy",
          "PolicyTypeName": "BackendServerAuthenticationPolicyType"
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancerPoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Richtlinien beschrieben, die dem angegebenen Load Balancer zugeordnet sind.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Beispiel 2: In diesem Beispiel werden die Attribute der angegebenen Richtlinie beschrieben.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Ausgabe:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Beispiel 3: In diesem Beispiel werden die vordefinierten Richtlinien beschrieben, einschließlich der Beispielrichtlinien. Die Namen der Beispielrichtlinien beginnen mit dem Präfix „ELBSample-“.

```
Get-ELBLoadBalancerPolicy
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien beschrieben, die dem angegebenen Load Balancer zugeordnet sind.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Beispiel 2: In diesem Beispiel werden die Attribute der angegebenen Richtlinie beschrieben.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Ausgabe:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Beispiel 3: In diesem Beispiel werden die vordefinierten Richtlinien beschrieben, einschließlich der Beispielrichtlinien. Die Namen der Beispielrichtlinien beginnen mit dem Präfix „ELBSample-“.

```
Get-ELBLoadBalancerPolicy
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
```

```
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancerPolicyTypes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancerPolicyTypes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die von Elastic Load Balancing definierten Richtlinientypen für Load Balancers

In diesem Beispiel werden die Richtlinientypen für Load Balancers beschrieben, mit denen Sie Richtlinienkonfigurationen für Ihren Load Balancer erstellen können.

Befehl:

```
aws elb describe-load-balancer-policy-types
```

Ausgabe:

```
{
  "PolicyTypeDescriptions": [
```

```
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ONE",
                "AttributeName": "ProxyProtocol",
                "AttributeType": "Boolean"
            }
        ],
        "PolicyTypeName": "ProxyProtocolPolicyType",
        "Description": "Policy that controls whether to include the IP address
and port of the originating request for TCP messages. This policy operates on
TCP/SSL listeners only"
    },
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ONE",
                "AttributeName": "PublicKey",
                "AttributeType": "String"
            }
        ],
        "PolicyTypeName": "PublicKeyPolicyType",
        "Description": "Policy containing a list of public keys to
accept when authenticating the back-end server(s). This policy cannot be
applied directly to back-end servers or listeners but must be part of a
BackendServerAuthenticationPolicyType."
    },
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ONE",
                "AttributeName": "CookieName",
                "AttributeType": "String"
            }
        ],
        "PolicyTypeName": "AppCookieStickinessPolicyType",
        "Description": "Stickiness policy with session lifetimes controlled by
the lifetime of the application-generated cookie. This policy can be associated
only with HTTP/HTTPS listeners."
    },
    {
        "PolicyAttributeTypeDescriptions": [
            {
                "Cardinality": "ZERO_OR_ONE",
```

```
                    "AttributeName": "CookieExpirationPeriod",
                    "AttributeType": "Long"
                }
            ],
            "PolicyTypeName": "LBCookieStickinessPolicyType",
            "Description": "Stickiness policy with session lifetimes controlled by
 the browser (user-agent) or a specified expiration period. This policy can be
 associated only with HTTP/HTTPS listeners."
        },
        {
            "PolicyAttributeTypeDescriptions": [
                .
                .
                .
            ],
            "PolicyTypeName": "SSLNegotiationPolicyType",
            "Description": "Listener policy that defines the ciphers and protocols
 that will be accepted by the load balancer. This policy can be associated only
 with HTTPS/SSL listeners."
        },
        {
            "PolicyAttributeTypeDescriptions": [
                {
                    "Cardinality": "ONE_OR_MORE",
                    "AttributeName": "PublicKeyPolicyName",
                    "AttributeType": "PolicyName"
                }
            ],
            "PolicyTypeName": "BackendServerAuthenticationPolicyType",
            "Description": "Policy that controls authentication to back-end
 server(s) and contains one or more policies, such as an instance of a
 PublicKeyPolicyType. This policy can be associated only with back-end servers
 that are using HTTPS/SSL."
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancerPolicyTypesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Richtlinientypen abgerufen, die von Elastic Load Balancing unterstützt werden.

```
Get-ELBLoadBalancerPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 2: In diesem Beispiel wird der angegebene Richtlinientyp beschrieben.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des angegebenen Richtlinientyps angezeigt.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Ausgabe:

```
Policy that controls whether to include the IP address and port of the
 originating request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerPolicyTypes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinientypen abgerufen, die von Elastic Load Balancing unterstützt werden.

```
Get-ELBLoadBalancerPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 2: In diesem Beispiel wird der angegebene Richtlinientyp beschrieben.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des angegebenen Richtlinientyps angezeigt.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Ausgabe:

```
Policy that controls whether to include the IP address and port of the
 originating request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerPolicyTypes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancers`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancers` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Load Balancers

In diesem Beispiel werden alle Ihre Load Balancers beschrieben.

Befehl:

```
aws elb describe-load-balancers
```

So beschreiben Sie einen Ihrer Load Balancers

In diesem Beispiel wird der angegebene Load Balancer beschrieben.

Befehl:

```
aws elb describe-load-balancers --load-balancer-name my-load-balancer
```

Nachstehend finden Sie eine Beispielantwort für einen HTTPS-Load-Balancer in einer VPC.

Ausgabe:

```
{
  "LoadBalancerDescriptions": [
    {
      "Subnets": [
          "subnet-15aaab61"
      ],
      "CanonicalHostedZoneNameID": "Z3DZXE0EXAMPLE",
      "CanonicalHostedZoneName": "my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com",
      "ListenerDescriptions": [
          {
              "Listener": {
                  "InstancePort": 80,
                  "LoadBalancerPort": 80,
                  "Protocol": "HTTP",
                  "InstanceProtocol": "HTTP"
              },
              "PolicyNames": []
          },
          {
              "Listener": {
                  "InstancePort": 443,
                  "SSLCertificateId": "arn:aws:iam::123456789012:server-
certificate/my-server-cert",
                  "LoadBalancerPort": 443,
                  "Protocol": "HTTPS",
                  "InstanceProtocol": "HTTPS"
              },
              "PolicyNames": [
                  "ELBSecurityPolicy-2015-03"
              ]
          }
```

```
        ],
        "HealthCheck": {
            "HealthyThreshold": 2,
            "Interval": 30,
            "Target": "HTTP:80/png",
            "Timeout": 3,
            "UnhealthyThreshold": 2
        },
        "VPCId": "vpc-a01106c2",
        "BackendServerDescriptions": [
            {
                "InstancePort": 80,
                "PolicyNames": [
                    "my-ProxyProtocol-policy"
                ]
            }
        ],
        "Instances": [
            {
                "InstanceId": "i-207d9717"
            },
            {
                "InstanceId": "i-afefb49b"
            }
        ],
        "DNSName": "my-load-balancer-1234567890.us-west-2.elb.amazonaws.com",
        "SecurityGroups": [
            "sg-a61988c3"
        ],
        "Policies": {
            "LBCookieStickinessPolicies": [
                {
                    "PolicyName": "my-duration-cookie-policy",
                    "CookieExpirationPeriod": 60
                }
            ],
            "AppCookieStickinessPolicies": [],
            "OtherPolicies": [
                "my-PublicKey-policy",
                "my-authentication-policy",
                "my-SSLNegotiation-policy",
                "my-ProxyProtocol-policy",
                "ELBSecurityPolicy-2015-03"
            ]
```

```
      },
      "LoadBalancerName": "my-load-balancer",
      "CreatedTime": "2015-03-19T03:24:02.650Z",
      "AvailabilityZones": [
          "us-west-2a"
      ],
      "Scheme": "internet-facing",
      "SourceSecurityGroup": {
          "OwnerAlias": "123456789012",
          "GroupName": "my-elb-sg"
      }
    }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Load Balancer aufgelistet.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Ausgabe:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Beispiel 2: In diesem Beispiel wird der angegebene Load Balancer beschrieben.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
```

```
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      :
 {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Load Balancer in der aktuellen AWS Region.

```
Get-ELBLoadBalancer
```

Beispiel 4: In diesem Beispiel werden alle Ihre Load Balancer in allen verfügbaren AWS-Regionen beschrieben.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Load Balancer aufgelistet.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Ausgabe:

```
LoadBalancerName
```

```
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Beispiel 2: In diesem Beispiel wird der angegebene Load Balancer beschrieben.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-
west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      :
 {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Load Balancer in der aktuellen AWS Region.

```
Get-ELBLoadBalancer
```

Beispiel 4: In diesem Beispiel werden alle Ihre Load Balancer in allen verfügbaren AWS-Regionen beschrieben.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die einem Load Balancer zugeordneten Tags

In diesem Beispiel werden die Tags beschrieben, die dem angegebenen Load Balancer zugewiesen sind.

Befehl:

```
aws elb describe-tags --load-balancer-name my-load-balancer
```

Ausgabe:

```
{
  "TagDescriptions": [
      {
          "Tags": [
              {
                  "Value": "lima",
                  "Key": "project"
              },
              {
                  "Value": "digital-media",
                  "Key": "department"
              }
          ],
          "LoadBalancerName": "my-load-balancer"
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags für die angegebenen Load Balancers aufgelistet.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Ausgabe:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Beispiel 2: In diesem Beispiel werden die Tags für den angegebenen Load Balancers beschrieben.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Ausgabe:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für die angegebenen Load Balancers aufgelistet.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Ausgabe:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Beispiel 2: In diesem Beispiel werden die Tags für den angegebenen Load Balancers beschrieben.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Ausgabe:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachLoadBalancerFromSubnets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachLoadBalancerFromSubnets` verwendet wird.

CLI

AWS CLI

So trennen Sie Load Balancers von Subnetzen

In diesem Beispiel wird der angegebene Load Balancer vom angegebenen Subnetz getrennt.

Befehl:

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-load-balancer
 --subnets subnet-0ecac448
```

Ausgabe:

```
{
   "Subnets": [
```

```
        "subnet-15aaab61"
    ]
}
```

- Einzelheiten zur API finden Sie DetachLoadBalancerFromSubnetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz aus den Subnetzen entfernt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe umfasst die verbleibenden Subnetze.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-87654321
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancerFromSubnets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz aus den Subnetzen entfernt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe umfasst die verbleibenden Subnetze.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-87654321
```

- Einzelheiten zur API finden Sie unter DetachLoadBalancerFromSubnets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DisableAvailabilityZonesForLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DisableAvailabilityZonesForLoadBalancer` verwendet wird.

CLI

AWS CLI

So deaktivieren Sie Availability Zones für einen Load Balancer

Im folgenden Beispiel wird die angegebene Availability Zone aus der Menge der Availability Zones für den angegebenen Load Balancer entfernt.

Befehl:

```
aws elb disable-availability-zones-for-load-balancer --load-balancer-name my-load-balancer --availability-zones us-west-2a
```

Ausgabe:

```
{
    "AvailabilityZones": [
        "us-west-2b"
    ]
}
```

- Einzelheiten zur API finden Sie DisableAvailabilityZonesForLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone aus dem spezifizierten Load Balancer entfernt. Die Ausgabe umfasst die verbleibenden Availability Zones.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2b
```

- Einzelheiten zur API finden Sie unter DisableAvailabilityZonesForLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone aus dem spezifizierten Load Balancer entfernt. Die Ausgabe umfasst die verbleibenden Availability Zones.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2b
```

- Einzelheiten zur API finden Sie unter DisableAvailabilityZonesForLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`EnableAvailabilityZonesForLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableAvailabilityZonesForLoadBalancer` verwendet wird.

CLI

AWS CLI

So aktivieren Sie Availability Zones für einen Load Balancer

In diesem Beispiel wird dem angegebenen Load Balancer die angegebene Availability Zone hinzugefügt.

Befehl:

```
aws elb enable-availability-zones-for-load-balancer --load-balancer-name my-load-
balancer --availability-zones us-west-2b
```

Ausgabe:

```
{
    "AvailabilityZones": [
        "us-west-2a",
        "us-west-2b"
    ]
}
```

- Einzelheiten zur API finden Sie EnableAvailabilityZonesForLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone zum angegebenen Load Balancer hinzugefügt. Die Ausgabe enthält die vollständige Liste der Availability Zones.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2a
us-west-2b
```

- Einzelheiten zur API finden Sie unter EnableAvailabilityZonesForLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone zum angegebenen Load Balancer hinzugefügt. Die Ausgabe enthält die vollständige Liste der Availability Zones.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2a
us-west-2b
```

- Einzelheiten zur API finden Sie unter EnableAvailabilityZonesForLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyLoadBalancerAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyLoadBalancerAttributes` verwendet wird.

CLI

AWS CLI

So ändern Sie die Attribute eines Load Balancers

In diesem Beispiel wird das Attribut `CrossZoneLoadBalancing` des angegebenen Load Balancers geändert.

Befehl:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer --
load-balancer-attributes "{\"CrossZoneLoadBalancing\":{\"Enabled\":true}}"
```

Ausgabe:

```
{
    "LoadBalancerAttributes": {
        "CrossZoneLoadBalancing": {
            "Enabled": true
        }
    },
    "LoadBalancerName": "my-load-balancer"
}
```

In diesem Beispiel wird das Attribut `ConnectionDraining` des angegebenen Load Balancers geändert.

Befehl:

```
aws elb modify-load-balancer-attributes --load-balancer-name my-load-balancer
 --load-balancer-attributes "{\"ConnectionDraining\":{\"Enabled\":true,\"Timeout
\":300}}"
```

Ausgabe:

```
{
    "LoadBalancerAttributes": {
        "ConnectionDraining": {
            "Enabled": true,
            "Timeout": 300
        }
    },
    "LoadBalancerName": "my-load-balancer"
}
```

- Einzelheiten zur API finden Sie ModifyLoadBalancerAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das zonenübergreifende Load Balancing für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Beispiel 2: In diesem Beispiel wird das Connection Draining für den angegebenen Load Balancer deaktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Beispiel 3: In diesem Beispiel wird die Zugriffsprotokollierung für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das zonenübergreifende Load Balancing für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Beispiel 2: In diesem Beispiel wird das Connection Draining für den angegebenen Load Balancer deaktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Beispiel 3: In diesem Beispiel wird die Zugriffsprotokollierung für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterInstancesWithLoadBalancer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterInstancesWithLoadBalancer` verwendet wird.

CLI

AWS CLI

So registrieren Sie Instances bei einem Load Balancer

In diesem Beispiel wird die angegebene Instance beim angegebenen Load Balancer registriert.

Befehl:

```
aws elb register-instances-with-load-balancer --load-balancer-name my-load-
balancer --instances i-d6f6fae3
```

Ausgabe:

```
{
   "Instances": [
       {
           "InstanceId": "i-d6f6fae3"
       },
       {
           "InstanceId": "i-207d9717"
       },
       {
           "InstanceId": "i-afefb49b"
       }
   ]
}
```

- Einzelheiten zur API finden Sie RegisterInstancesWithLoadBalancerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance beim spezifizierten Load Balancer registriert.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
```

- Einzelheiten zur API finden Sie unter RegisterInstancesWithLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance beim spezifizierten Load Balancer registriert.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
```

- Einzelheiten zur API finden Sie unter RegisterInstancesWithLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTags` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einem Load Balancer

In diesem Beispiel wird ein Tag vom angegebenen Load Balancer entfernt.

Befehl:

```
aws elb remove-tags --load-balancer-name my-load-balancer --tags project
```

- Einzelheiten zur API finden Sie RemoveTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Tag aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Bei Powershell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter RemoveTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Bei Powershell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- Einzelheiten zur API finden Sie unter RemoveTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetLoadBalancerListenerSslCertificate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetLoadBalancerListenerSslCertificate` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie das SSL-Zertifikat für einen HTTPS-Load-Balancer

In diesem Beispiel wird das vorhandene SSL-Zertifikat für den angegebenen HTTPS-Load-Balancer ersetzt.

Befehl:

```
aws elb set-load-balancer-listener-ssl-certificate --load-balancer-
name my-load-balancer --load-balancer-port 443 --ssl-certificate-
id arn:aws:iam::123456789012:server-certificate/new-server-cert
```

- Einzelheiten zur API finden Sie SetLoadBalancerListenerSslCertificatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Zertifikat ersetzt, das die SSL-Verbindungen für den angegebenen Listener beendet.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-
cert"
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerListenerSslCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat ersetzt, das die SSL-Verbindungen für den angegebenen Listener beendet.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-
cert"
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerListenerSslCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetLoadBalancerPoliciesForBackendServer`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetLoadBalancerPoliciesForBackendServer` verwendet wird.

CLI

AWS CLI

So ersetzen Sie die Richtlinien, die einem Port für eine Backend-Instance zugeordnet sind

Im folgenden Beispiel werden die Richtlinien ersetzt, die derzeit dem angegebenen Port zugeordnet sind.

Befehl:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names my-ProxyProtocol-policy
```

So entfernen Sie alle Richtlinien, die derzeit einem Port Ihrer Backend-Instance zugeordnet sind

In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Port zugeordnet sind.

Befehl:

```
aws elb set-load-balancer-policies-for-backend-server --load-balancer-name my-load-balancer --instance-port 80 --policy-names []
```

Verwenden Sie den Befehl `describe-load-balancer-policies`, um zu bestätigen, dass die Richtlinien entfernt wurden.

- Einzelheiten zur API finden Sie SetLoadBalancerPoliciesForBackendServerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Port durch die spezifizierte Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Port zugeordnet sind.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -InstancePort 80
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesForBackendServer AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Port durch die spezifizierte Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Port zugeordnet sind.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesForBackendServer AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetLoadBalancerPoliciesOfListener`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetLoadBalancerPoliciesOfListener` verwendet wird.

CLI

AWS CLI

So ersetzen Sie die Richtlinien, die einem Listener zugeordnet sind

In diesem Beispiel werden die Richtlinien ersetzt, die derzeit dem angegebenen Listener zugeordnet sind.

Befehl:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-
balancer --load-balancer-port 443 --policy-names my-SSLNegotiation-policy
```

So entfernen Sie alle Richtlinien, die Ihrem Listener zugeordnet sind

In diesem Beispiel werden alle Richtlinien entfernt, die derzeit dem angegebenen Listener zugeordnet sind.

Befehl:

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name my-load-
balancer --load-balancer-port 443 --policy-names []
```

Verwenden Sie den Befehl `describe-load-balancer-policies`, um zu bestätigen, dass die Richtlinien vom Load Balancer entfernt wurden.

- Einzelheiten zur API finden Sie SetLoadBalancerPoliciesOfListenerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Listener durch die angegebene Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Listener zugeordnet sind.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesOfListener AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Listener durch die angegebene Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Listener zugeordnet sind.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- Einzelheiten zur API finden Sie unter SetLoadBalancerPoliciesOfListener AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Elastic Load Balancing — Version 2 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Elastic Load Balancing — Version 2 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch für Elastic Load Balancing – Version 2 – Weitere Informationen zu Elastic Load Balancing – Version 2.
- API-Referenz für Elastic Load Balancing – Version 2 – Details zu allen verfügbaren Aktionen von Elastic Load Balancing – Version 2.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Elastic Load Balancing — Version 2 mit AWS SDKs
  - Hello Elastic Load Balancing
  - Aktionen für Elastic Load Balancing — Version 2 mit AWS SDKs
    - Verwendung von AddListenerCertificates mit einer CLI
    - Verwendung von AddTags mit einer CLI

- Verwenden Sie mit einem CreateListener AWS SDK oder CLI
- Verwenden Sie mit CreateLoadBalancereinem AWS SDK oder CLI
- Verwendung von CreateRule mit einer CLI
- Verwenden Sie mit einem CreateTargetGroup AWS SDK oder CLI
- Verwendung von DeleteListener mit einer CLI
- Verwenden Sie mit einem DeleteLoadBalancer AWS SDK oder CLI
- Verwendung von DeleteRule mit einer CLI
- Verwenden Sie mit einem DeleteTargetGroup AWS SDK oder CLI
- Verwendung von DeregisterTargets mit einer CLI
- Verwendung von DescribeAccountLimits mit einer CLI
- Verwendung von DescribeListenerCertificates mit einer CLI
- Verwendung von DescribeListeners mit einer CLI
- Verwendung von DescribeLoadBalancerAttributes mit einer CLI
- Verwenden Sie mit einem DescribeLoadBalancers AWS SDK oder CLI
- Verwendung von DescribeRules mit einer CLI
- Verwendung von DescribeSslPolicies mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTargetGroupAttributes mit einer CLI
- Verwenden Sie mit einem DescribeTargetGroups AWS SDK oder CLI
- Verwenden Sie mit einem DescribeTargetHealth AWS SDK oder CLI
- Verwendung von ModifyListener mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von ModifyRule mit einer CLI
- Verwendung von ModifyTargetGroup mit einer CLI
- Verwendung von ModifyTargetGroupAttributes mit einer CLI
- Verwendung von RegisterTargets mit einer CLI
- Verwendung von RemoveListenerCertificates mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetIpAddressType mit einer CLI

- Verwendung von SetRulePriorities mit einer CLI

    - Verwendung von SetSecurityGroups mit einer CLI
    - Verwendung von SetSubnets mit einer CLI
- Szenarien für Elastic Load Balancing — Version 2 mit AWS SDKs
  - Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
  - Erste Schritte mit dem Lastenausgleich

Grundlegende Beispiele für Elastic Load Balancing — Version 2 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Elastic Load Balancing – Version 2 mit AWS SDKs verwendet werden.

Beispiele

- Hello Elastic Load Balancing
- Aktionen für Elastic Load Balancing — Version 2 mit AWS SDKs
  - Verwendung von AddListenerCertificates mit einer CLI
  - Verwendung von AddTags mit einer CLI
  - Verwenden Sie mit einem CreateListener AWS SDK oder CLI
  - Verwenden Sie mit CreateLoadBalancereinem AWS SDK oder CLI
  - Verwendung von CreateRule mit einer CLI
  - Verwenden Sie mit einem CreateTargetGroup AWS SDK oder CLI
  - Verwendung von DeleteListener mit einer CLI
  - Verwenden Sie mit einem DeleteLoadBalancer AWS SDK oder CLI
  - Verwendung von DeleteRule mit einer CLI
  - Verwenden Sie mit einem DeleteTargetGroup AWS SDK oder CLI
  - Verwendung von DeregisterTargets mit einer CLI
  - Verwendung von DescribeAccountLimits mit einer CLI
  - Verwendung von DescribeListenerCertificates mit einer CLI
  - Verwendung von DescribeListeners mit einer CLI
  - Verwendung von DescribeLoadBalancerAttributes mit einer CLI
  - Verwenden Sie mit einem DescribeLoadBalancers AWS SDK oder CLI

- Verwendung von DescribeRules mit einer CLI
- Verwendung von DescribeSslPolicies mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTargetGroupAttributes mit einer CLI
- Verwenden Sie mit einem DescribeTargetGroups AWS SDK oder CLI
- Verwenden Sie mit einem DescribeTargetHealth AWS SDK oder CLI
- Verwendung von ModifyListener mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von ModifyRule mit einer CLI
- Verwendung von ModifyTargetGroup mit einer CLI
- Verwendung von ModifyTargetGroupAttributes mit einer CLI
- Verwendung von RegisterTargets mit einer CLI
- Verwendung von RemoveListenerCertificates mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetIpAddressType mit einer CLI
- Verwendung von SetRulePriorities mit einer CLI
- Verwendung von SetSecurityGroups mit einer CLI
- Verwendung von SetSubnets mit einer CLI

Hello Elastic Load Balancing

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Elastic Load Balancing beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloLoadBalancer {

        public static void main(String[] args) {
                ElasticLoadBalancingV2Client loadBalancingV2Client =
 ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

                DescribeLoadBalancersResponse loadBalancersResponse =
 loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
                List<LoadBalancer> loadBalancerList =
 loadBalancersResponse.loadBalancers();
                for (LoadBalancer lb : loadBalancerList)
                        System.out.println("Load Balancer DNS name = " +
 lb.dnsName());
        }
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
```

```
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_elbv2(elbv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Elastic Load Balancing V2
 client and list
    up to ten of the load balancers for your account.
    This example uses the default settings specified in your shared credentials
```

```
    and config files.

    :param elbv2_client: A Boto3 Elastic Load Balancing V2 client object.
    """
    print("Hello, Elastic Load Balancing! Let's list some of your load
 balancers:")
    load_balancers = elbv2_client.describe_load_balancers(PageSize=10).get(
        "LoadBalancers", []
    )
    if load_balancers:
        for lb in load_balancers:
            print(f"\t{lb['LoadBalancerName']}: {lb['DNSName']}")
    else:
        print("Your account doesn't have any load balancers.")


if __name__ == "__main__":
    hello_elbv2(boto3.client("elbv2"))
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin AWS SDK for Python (Boto3) API Reference.

Aktionen für Elastic Load Balancing — Version 2 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Elastic Load Balancing — Version 2-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Elastic Load Balancing – Version 2 API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Elastic Load Balancing — Version 2 mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Elastic Load Balancing – Version 2.

Beispiele

- Verwendung von AddListenerCertificates mit einer CLI
- Verwendung von AddTags mit einer CLI
- Verwenden Sie mit einem CreateListener AWS SDK oder CLI

- Verwenden Sie mit CreateLoadBalancereinem AWS SDK oder CLI
- Verwendung von CreateRule mit einer CLI
- Verwenden Sie mit einem CreateTargetGroup AWS SDK oder CLI
- Verwendung von DeleteListener mit einer CLI
- Verwenden Sie mit einem DeleteLoadBalancer AWS SDK oder CLI
- Verwendung von DeleteRule mit einer CLI
- Verwenden Sie mit einem DeleteTargetGroup AWS SDK oder CLI
- Verwendung von DeregisterTargets mit einer CLI
- Verwendung von DescribeAccountLimits mit einer CLI
- Verwendung von DescribeListenerCertificates mit einer CLI
- Verwendung von DescribeListeners mit einer CLI
- Verwendung von DescribeLoadBalancerAttributes mit einer CLI
- Verwenden Sie mit einem DescribeLoadBalancers AWS SDK oder CLI
- Verwendung von DescribeRules mit einer CLI
- Verwendung von DescribeSslPolicies mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeTargetGroupAttributes mit einer CLI
- Verwenden Sie mit einem DescribeTargetGroups AWS SDK oder CLI
- Verwenden Sie mit einem DescribeTargetHealth AWS SDK oder CLI
- Verwendung von ModifyListener mit einer CLI
- Verwendung von ModifyLoadBalancerAttributes mit einer CLI
- Verwendung von ModifyRule mit einer CLI
- Verwendung von ModifyTargetGroup mit einer CLI
- Verwendung von ModifyTargetGroupAttributes mit einer CLI
- Verwendung von RegisterTargets mit einer CLI
- Verwendung von RemoveListenerCertificates mit einer CLI
- Verwendung von RemoveTags mit einer CLI
- Verwendung von SetIpAddressType mit einer CLI
- Verwendung von SetRulePriorities mit einer CLI
- Verwendung von SetSecurityGroups mit einer CLI

- Verwendung von SetSubnets mit einer CLI

Verwendung von **`AddListenerCertificates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddListenerCertificates` verwendet wird.

CLI

AWS CLI

So fügen Sie einem sicheren Listener ein Zertifikat hinzu

In diesem Beispiel wird dem angegebenen sicheren Listener ein Zertifikat hinzugefügt.

Befehl:

```
aws elbv2 add-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --
certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705
```

Ausgabe:

```
{
  "Certificates": [
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
          "IsDefault": false
      }
  ]
}
```

- Einzelheiten zur API finden Sie AddListenerCertificatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird dem angegebenen Listener ein zusätzliches Zertifikat hinzugefügt.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
 -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Ausgabe:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-
b1d7-5217feba1d97 False
```

- Einzelheiten zur API finden Sie unter AddListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Listener ein zusätzliches Zertifikat hinzugefügt.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
 -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Ausgabe:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-
b1d7-5217feba1d97 False
```

- Einzelheiten zur API finden Sie unter AddListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AddTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddTags` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Load Balancer Tags hinzu

Im folgenden Beispiel für `add-tags` werden dem angegebenen Load Balancer die Tags `project` und `department` hinzugefügt.

```
aws elbv2 add-tags \
    --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --tags "Key=project,Value=lima" "Key=department,Value=digital-media"
```

- Einzelheiten zur API finden Sie AddTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues Tag zur angegebenen Ressource **`AWS.Tools.ElasticLoadBalancingV2`** hinzugefügt.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Einzelheiten zur API finden Sie unter AddTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Tag zur angegebenen Ressource **`AWS.Tools.ElasticLoadBalancingV2`** hinzugefügt.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- Einzelheiten zur API finden Sie unter AddTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateListener`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateListener` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
```

```
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
```

```
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Einzelheiten zur API finden Sie CreateListenerin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen HTTP-Listener

Im folgenden Beispiel für `create-listener` wird ein HTTP-Listener für den angegebenen Application Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --protocol HTTP \
    --port 80 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Weitere Informationen finden Sie unter Tutorial: Einen Application Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Application Load Balancers.

Beispiel 2: So erstellen Sie einen HTTPS-Listener

Im folgenden Beispiel für `create-listener` wird ein HTTPS-Listener für den angegebenen Application Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet. Sie müssen ein SSL-Zertifikat für einen HTTPS-Listener angeben. Sie können Zertifikate mit AWS Certificate Manager (ACM) erstellen und verwalten. Alternativ können Sie mithilfe von SSL/TLS Tools ein Zertifikat erstellen, das Zertifikat von einer Zertifizierungsstelle (CA) signieren lassen und das Zertifikat in AWS Identity and Access Management (IAM) hochladen.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
```

```
    --protocol HTTPS \
    --port 443 \
    --certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \
    --ssl-policy ELBSecurityPolicy-2016-08 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Weitere Informationen finden Sie unter Hinzufügen eines HTTPS-Listeners im Benutzerhandbuch für Application Load Balancers.

Beispiel 3: So erstellen Sie einen TCP-Listener

Im folgenden Beispiel für `create-listener` wird ein TCP-Listener für den angegebenen Network Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \
    --protocol TCP \
    --port 80 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

Weitere Informationen finden Sie unter Tutorial: Einen Network Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Network Load Balancer.

Beispiel 4: So erstellen Sie einen TLS-Listener

Im folgenden Beispiel für `create-listener` wird ein TLS-Listener für den angegebenen Network Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet. Sie müssen ein SSL-Zertifikat für einen TLS-Listener angeben.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --protocol TLS \
    --port 443 \
    --certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557 \
```

```
    --ssl-policy ELBSecurityPolicy-2016-08 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Weitere Informationen finden Sie unter TLS-Listener für Ihren Network Load Balancer im Benutzerhandbuch für Network Load Balancers.

Beispiel 5: So erstellen Sie einen UDP-Listener

Im folgenden Beispiel für `create-listener` wird ein UDP-Listener für den angegebenen Network Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e \
    --protocol UDP \
    --port 53 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-targets/b6bba954d1361c78
```

Weitere Informationen finden Sie unter Tutorial: Einen Network Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Network Load Balancer.

Beispiel 6: So erstellen Sie einen Listener für das angegebene Gateway und die Weiterleitung

Im folgenden Beispiel für `create-listener` wird ein Listener für den angegebenen Gateway Load Balancer erstellt, der Anfragen an die angegebene Zielgruppe weiterleitet.

```
aws elbv2 create-listener \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/e0f9b3d5c7f7d3d6 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/my-glb-targets/007ca469fae3bb1615
```

Ausgabe:

```
{
    "Listeners": [
        {
```

```
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:listener/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6/
afc127db15f925de",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-agw-lb-example2/e0f9b3d5c7f7d3d6",
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:targetgroup/test-tg-agw-2/007ca469fae3bb1615",
                    "ForwardConfig": {
                        "TargetGroups": [
                            {
                                "TargetGroupArn":
 "arn:aws:elasticloadbalancing:us-east-1:850631746142:targetgroup/test-tg-
agw-2/007ca469fae3bb1615"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte mit Gateway Load Balancers using the AWS CLI im Benutzerhandbuch für Gateway Load Balancers.

- Einzelheiten zur API finden Sie CreateListenerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
```

```
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
           Action action = Action.builder()
```

```
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateListenerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { Listeners } = await client.send(
```

```
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
```

- Einzelheiten zur API finden Sie CreateListenerin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer ALB-Listener mit der Standardaktion „Weiterleiten“ erstellt, um Datenverkehr an die angegebene Zielgruppe zu senden.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -
Protocol "HTTP" -DefaultAction $defaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
```

```
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8001
Protocol        : HTTP
SslPolicy       :
```

- Einzelheiten zur API finden Sie unter CreateListener AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer ALB-Listener mit der Standardaktion „Weiterleiten“ erstellt, um Datenverkehr an die angegebene Zielgruppe zu senden.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -
Protocol "HTTP" -DefaultAction $defaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8001
```

```
Protocol        : HTTP
SslPolicy       :
```

- Einzelheiten zur API finden Sie unter CreateListener AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
 to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
 listener for.
        :param target_group: An existing target group that is added as a listener
 to the
                             load balancer.
```

```
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
```

```
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie CreateListenerin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateLoadBalancer`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLoadBalancer` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
```

```
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
```

```
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- Einzelheiten zur API finden Sie CreateLoadBalancerin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Internet-facing Load Balancer

Im folgenden `create-load-balancer` Beispiel wird ein Internet-facing Application Load Balancer erstellt und die Availability Zones für die angegebenen Subnetze aktiviert.

```
aws elbv2 create-load-balancer \
    --name my-load-balancer \
    --subnets subnet-b7d581c0 subnet-8360a9e7
```

Ausgabe:

```
{
    "LoadBalancers": [
        {
            "Type": "application",
            "Scheme": "internet-facing",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
```

```
                    "ZoneName": "us-west-2a",
                    "SubnetId": "subnet-8360a9e7"
                },
                {
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-b7d581c0"
                }
            ],
            "CreatedTime": "2017-08-25T21:26:12.920Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-5943793c"
            ],
            "LoadBalancerName": "my-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
        }
    ]
}
```

Weitere Informationen finden Sie unter Tutorial: Einen Application Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Application Load Balancers.

Beispiel 2: So erstellen Sie einen internen Load Balancer

Im folgenden Beispiel für `create-load-balancer` wird ein interner Application Load Balancer erstellt. Außerdem werden die Availability Zones für die angegebenen Subnetze aktiviert.

```
aws elbv2 create-load-balancer \
    --name my-internal-load-balancer \
    --scheme internal \
    --subnets subnet-b7d581c0 subnet-8360a9e7
```

Ausgabe:

```
{
    "LoadBalancers": [
        {
```

```
            "Type": "application",
            "Scheme": "internal",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-west-2a",
                    "SubnetId": "subnet-8360a9e7"
                },
                {
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-b7d581c0"
                }
            ],
            "CreatedTime": "2016-03-25T21:29:48.850Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "internal-my-internal-load-balancer-1529930873.us-
west-2.elb.amazonaws.com",
            "SecurityGroups": [
                "sg-5943793c"
            ],
            "LoadBalancerName": "my-internal-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-internal-load-balancer/5b49b8d4303115c2"
        }
    ]
}
```

Weitere Informationen finden Sie unter Tutorial: Einen Application Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Application Load Balancers.

Beispiel 3: So erstellen Sie einen Network Load Balancer

Im folgenden `create-load-balancer` Beispiel wird ein Internet-facing Network Load Balancer erstellt und die Availability Zone für das angegebene Subnetz aktiviert. Dabei wird die angegebene Elastic-IP-Adresse über eine Subnetzzuordnung der Netzwerkschnittstelle zugewiesen, die von den Load-Balancer-Knoten für die Availability Zone verwendet wird.

```
aws elbv2 create-load-balancer \
    --name my-network-load-balancer \
```

```
    --type network \
    --subnet-mappings SubnetId=subnet-b7d581c0,AllocationId=eipalloc-64d5890a
```

Ausgabe:

```
{
    "LoadBalancers": [
        {
            "Type": "network",
            "Scheme": "internet-facing",
            "IpAddressType": "ipv4",
            "VpcId": "vpc-3ac0fb5f",
            "AvailabilityZones": [
                {
                    "LoadBalancerAddresses": [
                        {
                            "IpAddress": "35.161.207.171",
                            "AllocationId": "eipalloc-64d5890a"
                        }
                    ],
                    "ZoneName": "us-west-2b",
                    "SubnetId": "subnet-5264e837"
                }
            ],
            "CreatedTime": "2017-10-15T22:41:25.657Z",
            "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
            "DNSName": "my-network-load-balancer-5d1b75f4f1cee11e.elb.us-
west-2.amazonaws.com",
            "LoadBalancerName": "my-network-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/net/my-network-load-balancer/5d1b75f4f1cee11e"
        }
    ]
}
```

Weitere Informationen finden Sie unter Tutorial: Einen Network Load Balancer mithilfe der AWS CLI erstellen im Benutzerhandbuch für Network Load Balancer.

Beispiel 4: So erstellen Sie einen Gateway Load Balancer

Im folgenden Beispiel für `create-load-balancer` wird ein interner Gateway Load Balancer erstellt. Außerdem werden die Availability Zones für die angegebenen Subnetze aktiviert.

```
aws elbv2 create-load-balancer \
    --name my-gateway-load-balancer \
    --type gateway \
    --subnets subnet-dc83f691 subnet-a62583f9
```

Ausgabe:

```
{
    "LoadBalancers": [
        {
            "Type": "gateway",
            "VpcId": "vpc-838475fe",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-east-1b",
                    "SubnetId": "subnet-a62583f9"
                },
            {
                    "ZoneName": "us-east-1a",
                    "SubnetId": "subnet-dc83f691"
                }
            ],
            "CreatedTime": "2021-07-14T19:33:43.324000+00:00",
            "LoadBalancerName": "my-gateway-load-balancer",
            "State": {
                "Code": "provisioning"
            },
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
east-1:850631746142:loadbalancer/gwy/my-gateway-load-balancer/dfbb5a7d32cdee79"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte mit Gateway Load Balancers using the AWS CLI im Benutzerhandbuch für Gateway Load Balancers.

- Einzelheiten zur API finden Sie CreateLoadBalancerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /*
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();
```

```
            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateLoadBalancerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { LoadBalancers } = await client.send(
  new CreateLoadBalancerCommand({
    Name: NAMES.loadBalancerName,
    Subnets: state.subnets,
  }),
);
state.loadBalancerDns = LoadBalancers[0].DNSName;
state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
await waitUntilLoadBalancerAvailable(
  { client },
  { Names: [NAMES.loadBalancerName] },
);
```

- Einzelheiten zur API finden Sie CreateLoadBalancerin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer, mit dem Internet verbundener Application Load Balancer mit zwei Subnetzen erstellt.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Ausgabe:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer, mit dem Internet verbundener Application Load Balancer mit zwei Subnetzen erstellt.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Ausgabe:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
```

```
Scheme                  : internet-facing
SecurityGroups          : {sg-07c3414abb8811cbd}
State                   : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                    : application
VpcId                   : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter CreateLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
 subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
```

```
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

- Einzelheiten zur API finden Sie CreateLoadBalancerin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`CreateRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRule` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Regel mit einer Pfadbedingung und einer Weiterleitungsaktion

Im folgenden Beispiel für `create-rule` wird eine Regel erstellt, die Anfragen an die angegebene Zielgruppe weiterleitet, wenn die URL das vorgeschriebene Muster enthält.

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --priority 5 \
    --conditions file://conditions-pattern.json
    --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Inhalt von `conditions-pattern.json`:

```
[
    {
        "Field": "path-pattern",
        "PathPatternConfig": {
            "Values": ["/images/*"]
        }
    }
]
```

Beispiel 2: So erstellen Sie eine Regel mit einer Hostbedingung und einer festen Antwort

Im folgenden Beispiel für `create-rule` wird eine Regel erstellt, die eine feste Antwort liefert, wenn der Hostname im Host-Header mit dem angegebenen Hostnamen übereinstimmt.

```
aws elbv2 create-rule \
```

```
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --priority 10 \
    --conditions file://conditions-host.json \
    --actions file://actions-fixed-response.json
```

Inhalt von `conditions-host.json`

```
[
  {
      "Field": "host-header",
      "HostHeaderConfig": {
          "Values": ["*.example.com"]
      }
  }
]
```

Inhalt von `actions-fixed-response.json`

```
[
    {
        "Type": "fixed-response",
        "FixedResponseConfig": {
            "MessageBody": "Hello world",
            "StatusCode": "200",
            "ContentType": "text/plain"
        }
    }
]
```

Beispiel 3: So erstellen Sie eine Regel mit einer Quell-IP-Adressbedingung, einer Authentifizierungsaktion und einer Weiterleitungsaktion

Im folgenden Beispiel für `create-rule` wird eine Regel erstellt, die den Benutzer authentifiziert, wenn die Quell-IP-Adresse mit der angegebenen IP-Adresse übereinstimmt, und die Anfrage an die angegebene Zielgruppe weiterleitet, wenn die Authentifizierung erfolgreich ist.

```
aws elbv2 create-rule \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
```

```
    --priority 20 \
    --conditions file://conditions-source-ip.json \
    --actions file://actions-authenticate.json
```

Inhalt von `conditions-source-ip.json`

```
[
    {
        "Field": "source-ip",
        "SourceIpConfig": {
            "Values": ["192.0.2.0/24", "198.51.100.10/32"]
        }
    }
]
```

Inhalt von `actions-authenticate.json`

```
[
    {
        "Type": "authenticate-oidc",
        "AuthenticateOidcConfig": {
            "Issuer": "https://idp-issuer.com",
            "AuthorizationEndpoint": "https://authorization-endpoint.com",
            "TokenEndpoint": "https://token-endpoint.com",
            "UserInfoEndpoint": "https://user-info-endpoint.com",
            "ClientId": "abcdefghijklmnopqrstuvwxyz123456789",
            "ClientSecret": "12345678901234567890123456789 0",
            "SessionCookieName": "my-cookie",
            "SessionTimeout": 3600,
            "Scope": "email",
            "AuthenticationRequestExtraParams": {
                "display": "page",
                "prompt": "login"
            },
            "OnUnauthenticatedRequest": "deny"
        },
        "Order": 1
    },
    {
        "Type": "forward",
        "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
east-1:880185128111:targetgroup/cli-test/642a97ecb0e0f26b",
        "Order": 2
```

```
    }
]
```

- Einzelheiten zur API finden Sie CreateRuleein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Listener-Regel mit einer Aktion mit fester Antwort erstellt, die auf dem Kunden-Header-Wert für den angegebenen Listener basiert.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -
Action $newRuleAction -Condition $newRuleCondition -Priority 10
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Einzelheiten zur API finden Sie unter CreateRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Listener-Regel mit einer Aktion mit fester Antwort erstellt, die auf dem Kunden-Header-Wert für den angegebenen Listener basiert.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -
Action $newRuleAction -Condition $newRuleCondition -Priority 10
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Einzelheiten zur API finden Sie unter CreateRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateTargetGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTargetGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
```

```
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }
```

- Einzelheiten zur API finden Sie CreateTargetGroupin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Zielgruppe für einen Application Load Balancer

Im folgenden Beispiel für `create-target-group` wird eine Zielgruppe für einen Application Load Balancer erstellt, in der Sie Ziele nach Instance-ID registrieren (der Zieltyp ist `instance`). Diese Zielgruppe verwendet das HTTP-Protokoll, Port 80 und die Standardeinstellungen für die Zustandsprüfung für eine HTTP-Zielgruppe.

```
aws elbv2 create-target-group \
    --name my-targets \
    --protocol HTTP \
    --port 80 \
    --target-type instance \
    --vpc-id vpc-3ac0fb5f
```

Ausgabe:

```
{
```

```
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
            "TargetGroupName": "my-targets",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen einer Zielgruppe im Benutzerhandbuch für Application Load Balancers.

Beispiel 2: So erstellen Sie eine Zielgruppe, um Datenverkehr von einem Application Load Balancer zu einer Lambda-Funktion weiterzuleiten

Im folgenden Beispiel für `create-target-group` wird eine Zielgruppe für einen Application Load Balancer erstellt, in der das Ziel eine Lambda-Funktion ist (der Zieltyp ist `lambda`). Zustandsprüfungen sind für diese Zielgruppe standardmäßig deaktiviert.

```
aws elbv2 create-target-group \
    --name my-lambda-target \
    --target-type lambda
```

Ausgabe:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-lambda-target/a3003e085dbb8ddc",
            "TargetGroupName": "my-lambda-target",
            "HealthCheckEnabled": false,
            "HealthCheckIntervalSeconds": 35,
            "HealthCheckTimeoutSeconds": 30,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "TargetType": "lambda",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Weitere Informationen finden Sie unter Lambda-Funktionen als Ziele im Benutzerhandbuch zu Application Load Balancers.

Beispiel 3: So erstellen Sie eine Zielgruppe für einen Network Load Balancer

Im folgenden Beispiel für `create-target-group` wird eine Zielgruppe für einen Network Load Balancer erstellt, in der Sie Ziele nach IP-Adresse registrieren (der Zieltyp ist `ip`). Diese Zielgruppe verwendet das TCP-Protokoll, Port 80 und die Standardeinstellungen für die Zustandsprüfung für eine TCP-Zielgruppe.

```
aws elbv2 create-target-group \
    --name my-ip-targets \
    --protocol TCP \
    --port 80 \
    --target-type ip \
    --vpc-id vpc-3ac0fb5f
```

Ausgabe:

```
{
    "TargetGroups": [
```

```
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-ip-targets/b6bba954d1361c78",
            "TargetGroupName": "my-ip-targets",
            "Protocol": "TCP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckEnabled": true,
            "HealthCheckProtocol": "TCP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 10,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "TargetType": "ip",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen einer Zielgruppe im Benutzerhandbuch für Network Load Balancers.

Beispiel 4: So erstellen Sie eine Zielgruppe, um Datenverkehr von einem Network Load Balancer zu einem Application Load Balancer weiterzuleiten

Im folgenden Beispiel für `create-target-group` wird eine Zielgruppe für einen Network Load Balancer erstellt, in der Sie einen Application Load Balancer als Ziel registrieren (der Zieltyp ist `alb`).

aws elbv2 create-target-group --name my-alb-target --protocol TCP --port 80 --target-type alb --vpc-id vpc-3ac0fb5f

Ausgabe:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target/a3003e085dbb8ddc",
            "TargetGroupName": "my-alb-target",
            "Protocol": "TCP",
            "Port": 80,
```

```
            "VpcId": "vpc-838475fe",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 6,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200-399"
            },
            "TargetType": "alb",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen einer Zielgruppe mit einem Application Load Balancer im Benutzerhandbuch für Network Load Balancers.

Beispiel 5: So erstellen Sie eine Zielgruppe für einen Gateway Load Balancer

Im folgenden Beispiel für `create-target-group` wird eine Zielgruppe für einen Gateway Load Balancer erstellt, wobei das Ziel eine Instance und das Zielgruppenprotokoll `GENEVE` ist.

```
aws elbv2 create-target-group \
    --name my-glb-targetgroup \
    --protocol GENEVE \
    --port 6081 \
    --target-type instance \
    --vpc-id vpc-838475fe
```

Ausgabe:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-glb-targetgroup/00c3d57eacd6f40b6f",
            "TargetGroupName": "my-glb-targetgroup",
            "Protocol": "GENEVE",
```

```
            "Port": 6081,
            "VpcId": "vpc-838475fe",
            "HealthCheckProtocol": "TCP",
            "HealthCheckPort": "80",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 10,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "TargetType": "instance"
        }
    ]
}
```

Weitere Informationen finden Sie unter Create a target group < https://docs.aws.amazon.com/elasticloadbalancing/latest/gateway/create-target-group.html >`__ im Gateway Load Balancer User Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz CreateTargetGroup.AWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
```

```
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
```

- Einzelheiten zur API finden Sie CreateTargetGroupin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
```

```
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
```

- Einzelheiten zur API finden Sie CreateTargetGroupin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Zielgruppe mit den angegebenen Parametern erstellt.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName            : NewTargetGroup
TargetType                 : instance
UnhealthyThresholdCount    : 5
VpcId                      : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter CreateTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Zielgruppe mit den angegebenen Parametern erstellt.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName            : NewTargetGroup
TargetType                 : instance
UnhealthyThresholdCount    : 5
VpcId                      : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter CreateTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
 specifies how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
 times and
        lower thresholds. In production, you might want to decrease the
 sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
```

```
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie CreateTargetGroupin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DeleteListener`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteListener` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Lastenausgleich

CLI

AWS CLI

So löschen Sie einen Listener

Im folgenden Beispiel für `delete-listener` wird der angegebene Listener gelöscht.

```
aws elbv2 delete-listener \
    --listener-arn arn:aws:elasticloadbalancing:ua-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- Einzelheiten zur API finden Sie DeleteListenerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Listener gelöscht.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Beispiel 2: In diesem Beispiel wird der angegebene Listener aus dem Load Balancer entfernt.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteListener AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Listener gelöscht.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Beispiel 2: In diesem Beispiel wird der angegebene Listener aus dem Load Balancer entfernt.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteListener AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteLoadBalancer`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLoadBalancer` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
```

```
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie einen Load Balancer

Im folgenden Beispiel für `delete-load-balancer` wird der angegebene Load Balancer gelöscht.

```
aws elbv2 delete-load-balancer \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
await client.send(
  new DeleteLoadBalancerCommand({
    LoadBalancerArn: loadBalancer.LoadBalancerArn,
  }),
);
await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
  const lb = await findLoadBalancer(NAMES.loadBalancerName);
  if (lb) {
    throw new Error("Load balancer still exists.");
  }
});
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteLoadBalancer AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
```

```
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DeleteLoadBalancerin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DeleteRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRule` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Regel

Im folgenden Beispiel für `delete-rule` wird die angegebene Regel gelöscht.

```
aws elbv2 delete-rule \
    --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/1291d13826f405c3
```

- Einzelheiten zur API finden Sie DeleteRulein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus dem Listener entfernt

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus dem Listener entfernt

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteTargetGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTargetGroup` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DeleteTargetGroupin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie eine Zielgruppe

Im folgenden Beispiel für `delete-target-group` wird die angegebene Zielgruppe gelöscht.

```
aws elbv2 delete-target-group \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines Load Balancers im Benutzerhandbuch für Application Load Balancer.

- Einzelheiten zur API finden Sie DeleteTargetGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

- Einzelheiten zur API finden Sie DeleteTargetGroupin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
try {
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    client.send(
      new DeleteTargetGroupCommand({
        TargetGroupArn: TargetGroups[0].TargetGroupArn,
      }),
    ),
  );
} catch (e) {
  state.deleteLoadBalancerTargetGroupError = e;
}
```

- Einzelheiten zur API finden Sie DeleteTargetGroupin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe entfernt.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe entfernt.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter DeleteTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
 self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
 target_group_name)

            # Use a custom waiter to wait until the target group is no longer
 available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
```

```
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

- Einzelheiten zur API finden Sie DeleteTargetGroupin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DeregisterTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterTargets` verwendet wird.

CLI

AWS CLI

Beispiel 1: So melden Sie ein Ziel von einer Zielgruppe ab

Im folgenden Beispiel für `deregister-targets` wird die angegebene Instance von der angegebenen Zielgruppe entfernt.

```
aws elbv2 deregister-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
    --targets Id=i-1234567890abcdef0
```

Beispiel 2: So melden Sie ein Ziel ab, das mithilfe von Port-Überschreibungen registriert wurde

Im folgenden Beispiel für `deregister-targets` wird eine Instance aus einer Zielgruppe entfernt, die mithilfe von Port-Überschreibungen registriert wurde.

```
aws elbv2 deregister-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \
    --targets Id=i-1234567890abcdef0,Port=80 Id=i-1234567890abcdef0,Port=766
```

- Einzelheiten zur API finden Sie DeregisterTargetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Registrierung der Instance „i-0672a4c4cdeae3111” aus der angegebenen Zielgruppe aufgehoben.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Einzelheiten zur API finden Sie unter DeregisterTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung der Instance „i-0672a4c4cdeae3111” aus der angegebenen Zielgruppe aufgehoben.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- Einzelheiten zur API finden Sie unter DeregisterTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAccountLimits`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAccountLimits` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Ihre Limits für Elastic Load Balancing

Das folgende `describe-account-limits` Beispiel zeigt die Elastic Load Balancing Balancing-Limits für Ihr AWS Konto in der aktuellen Region.

```
aws elbv2 describe-account-limits
```

Ausgabe:

```
{
    "Limits": [
        {
            "Name": "target-groups",
            "Max": "3000"
        },
        {
            "Name": "targets-per-application-load-balancer",
            "Max": "1000"
        },
        {
            "Name": "listeners-per-application-load-balancer",
            "Max": "50"
        },
        {
            "Name": "rules-per-application-load-balancer",
            "Max": "100"
        },
        {
            "Name": "network-load-balancers",
            "Max": "50"
        },
        {
            "Name": "targets-per-network-load-balancer",
            "Max": "3000"
        },
        {
            "Name": "targets-per-availability-zone-per-network-load-balancer",
            "Max": "500"
        },
        {
            "Name": "listeners-per-network-load-balancer",
            "Max": "50"
        },
        {
            "Name": "condition-values-per-alb-rule",
            "Max": "5"
        },
        {
            "Name": "condition-wildcards-per-alb-rule",
            "Max": "5"
        },
        {
```

```
            "Name": "target-groups-per-application-load-balancer",
            "Max": "100"
        },
        {
            "Name": "target-groups-per-action-on-application-load-balancer",
            "Max": "5"
        },
        {
            "Name": "target-groups-per-action-on-network-load-balancer",
            "Max": "1"
        },
        {
            "Name": "certificates-per-application-load-balancer",
            "Max": "25"
        },
        {
            "Name": "certificates-per-network-load-balancer",
            "Max": "25"
        },
        {
            "Name": "targets-per-target-group",
            "Max": "1000"
        },
        {
            "Name": "target-id-registrations-per-application-load-balancer",
            "Max": "1000"
        },
        {
            "Name": "network-load-balancer-enis-per-vpc",
            "Max": "1200"
        },
        {
            "Name": "application-load-balancers",
            "Max": "50"
        },
        {
            "Name": "gateway-load-balancers",
            "Max": "100"
        },
        {
            "Name": "gateway-load-balancers-per-vpc",
            "Max": "100"
        },
        {
```

```
            "Name": "geneve-target-groups",
            "Max": "100"
        },
        {
            "Name": "targets-per-availability-zone-per-gateway-load-balancer",
            "Max": "300"
        }
    ]
}
```

Weitere Informationen finden Sie unter Quotas in der Allgemeinen AWS -Referenz.

- Einzelheiten zur API finden Sie DescribeAccountLimitsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl listet die ELB2-Kontolimits für eine bestimmte Region auf.

```
Get-ELB2AccountLimit
```

Ausgabe:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl listet die ELB2-Kontolimits für eine bestimmte Region auf.

```
Get-ELB2AccountLimit
```

Ausgabe:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- Einzelheiten zur API finden Sie unter DescribeAccountLimits AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeListenerCertificates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeListenerCertificates` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Zertifikate für einen sicheren Listener

In diesem Beispiel werden die Zertifikate für den angegebenen sicheren Listener beschrieben.

Befehl:

```
aws elbv2 describe-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

Ausgabe:

```
{
  "Certificates": [
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705",
          "IsDefault": false
      },
      {
          "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/3dcb0a41-bd72-4774-9ad9-756919c40557",
          "IsDefault": false
      },
      {
          "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
fe59da96-6f58-4a22-8eed-6d0d50477e1d",
          "IsDefault": true
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeListenerCertificatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Zertifikat für den angegebenen Listener beschrieben.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-
east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c True
```

- Einzelheiten zur API finden Sie unter DescribeListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat für den angegebenen Listener beschrieben.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
CertificateArn
   IsDefault
--------------
   ---------
arn:aws:acm:us-
east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c True
```

- Einzelheiten zur API finden Sie unter DescribeListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeListeners`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeListeners` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen Listener

In diesem Beispiel wird der angegebene Listener beschrieben.

Befehl:

```
aws elbv2 describe-listeners --listener-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2
```

Ausgabe:

```
{
  "Listeners": [
      {
          "Port": 80,
          "Protocol": "HTTP",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2"
      }
  ]
}
```

So beschreiben Sie die Listener für einen Load Balancer

In diesem Beispiel werden die Listener für den angegebenen Load Balancer beschrieben.

Befehl:

```
aws elbv2 describe-listeners --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

Ausgabe:

```
{
```

```
  "Listeners": [
      {
          "Port": 443,
          "Protocol": "HTTPS",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ],
          "SslPolicy": "ELBSecurityPolicy-2015-05",
          "Certificates": [
              {
                  "CertificateArn": "arn:aws:iam::123456789012:server-
certificate/my-server-cert"
              }
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65"
      },
      {
          "Port": 80,
          "Protocol": "HTTP",
          "DefaultActions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ],
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2"
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeListenersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die Listener der angegebenen ALB/NLB.

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Einzelheiten zur API finden Sie unter DescribeListeners AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt Listener der angegebenen. ALB/NLB

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- Einzelheiten zur API finden Sie unter DescribeListeners AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeLoadBalancerAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancerAttributes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Load-Balancer-Attribute

Im folgenden Beispiel für `describe-load-balancer-attributes` werden die Attribute des angegebenen Load Balancers angezeigt.

```
aws elbv2 describe-load-balancer-attributes \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

Die folgende Beispielausgabe zeigt die Attribute für einen Application Load Balancer.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "access_logs.s3.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.bucket"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "60",
            "Key": "idle_timeout.timeout_seconds"
        },
        {
            "Value": "false",
            "Key": "deletion_protection.enabled"
        },
        {
            "Value": "true",
            "Key": "routing.http2.enabled"
        }
    ]
}
```

Die folgende Beispielausgabe beinhaltet die Attribute für einen Network Load Balancer.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "access_logs.s3.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.bucket"
        },
        {
            "Value": "",
```

```
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "false",
            "Key": "deletion_protection.enabled"
        },
        {
            "Value": "false",
            "Key": "load_balancing.cross_zone.enabled"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancerAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl beschreibt die Attribute eines bestimmten Load Balancers.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Ausgabe:

```
Key                              Value
---                              -----
access_logs.s3.enabled           false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled      false
access_logs.s3.bucket
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl beschreibt die Attribute eines bestimmten Load Balancers.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Ausgabe:

```
Key                              Value
---                              -----
access_logs.s3.enabled           false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled      false
access_logs.s3.bucket
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeLoadBalancers`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeLoadBalancers` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
```

```
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie einen Load Balancer

In diesem Beispiel wird der angegebene Load Balancer beschrieben.

Befehl:

```
aws elbv2 describe-load-balancers --load-balancer-
arns arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-
load-balancer/50dc6c495c0c9188
```

Ausgabe:

```
{
```

```
  "LoadBalancers": [
      {
          "Type": "application",
          "Scheme": "internet-facing",
          "IpAddressType": "ipv4",
          "VpcId": "vpc-3ac0fb5f",
          "AvailabilityZones": [
              {
                  "ZoneName": "us-west-2a",
                  "SubnetId": "subnet-8360a9e7"
              },
              {
                  "ZoneName": "us-west-2b",
                  "SubnetId": "subnet-b7d581c0"
              }
          ],
          "CreatedTime": "2016-03-25T21:26:12.920Z",
          "CanonicalHostedZoneId": "Z2P70J7EXAMPLE",
          "DNSName": "my-load-balancer-424835706.us-west-2.elb.amazonaws.com",
          "SecurityGroups": [
              "sg-5943793c"
          ],
          "LoadBalancerName": "my-load-balancer",
          "State": {
              "Code": "active"
          },
          "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
      }
  ]
}
```

So beschreiben Sie alle Load Balancers

In diesem Beispiel werden alle Ihre Load Balancers beschrieben.

Befehl:

```
aws elbv2 describe-load-balancers
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Load Balancer für die angegebene Region angezeigt.

```
Get-ELB2LoadBalancer
```

Ausgabe:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Load Balancer für die angegebene Region angezeigt.

```
Get-ELB2LoadBalancer
```

Ausgabe:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
```

```
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- Einzelheiten zur API finden Sie unter DescribeLoadBalancers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
```

```
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

- Einzelheiten zur API finden Sie DescribeLoadBalancersin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DescribeRules`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRules` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Regel

Im folgenden Beispiel für `describe-rules` werden Details für die angegebene Regel gezeigt.

```
aws elbv2 describe-rules \
    --rule-arns arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-
rule/app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee
```

Beispiel 2: So beschreiben Sie die Regeln für einen Listener

Im folgenden Beispiel für `describe-rules` werden Details zu den Regeln für den angegebenen Listener gezeigt. Die Ausgabe enthält die Standardregel und alle anderen Regeln, die Sie hinzugefügt haben.

```
aws elbv2 describe-rules \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2
```

- Einzelheiten zur API finden Sie DescribeRulesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Listener-Regeln für den angegebenen Listener-ARN beschrieben.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
```

```
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Einzelheiten zur API finden Sie unter DescribeRules AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Listener-Regeln für den angegebenen Listener-ARN beschrieben.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- Einzelheiten zur API finden Sie unter DescribeRules AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeSslPolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSslPolicies` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie die für die SSL-Aushandlung verwendeten Richtlinien nach Load-Balancer-Typ auf

Im folgenden Beispiel für `describe-ssl-policies` werden die Namen der Richtlinien angezeigt, die Sie für die SSL-Aushandlung mit einem Application Load Balancer verwenden können. Im Beispiel wird der Parameter `--query` verwendet, um nur die Namen der Richtlinien anzuzeigen.

```
aws elbv2 describe-ssl-policies \
    --load-balancer-type application \
    --query SslPolicies[*].Name
```

Ausgabe:

```
[
    "ELBSecurityPolicy-2016-08",
    "ELBSecurityPolicy-TLS13-1-2-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",
    "ELBSecurityPolicy-TLS13-1-1-2021-06",
    "ELBSecurityPolicy-TLS13-1-0-2021-06",
    "ELBSecurityPolicy-TLS13-1-3-2021-06",
    "ELBSecurityPolicy-TLS-1-2-2017-01",
    "ELBSecurityPolicy-TLS-1-1-2017-01",
    "ELBSecurityPolicy-TLS-1-2-Ext-2018-06",
    "ELBSecurityPolicy-FS-2018-06",
    "ELBSecurityPolicy-2015-05",
    "ELBSecurityPolicy-TLS-1-0-2015-04",
    "ELBSecurityPolicy-FS-1-2-Res-2019-08",
    "ELBSecurityPolicy-FS-1-1-2019-08",
    "ELBSecurityPolicy-FS-1-2-2019-08",
    "ELBSecurityPolicy-FS-1-2-Res-2020-10"
]
```

Beispiel 2: So listen Sie die Richtlinien auf, die ein bestimmtes Protokoll unterstützen

Im folgenden Beispiel für `describe-ssl-policies` werden die Namen der Richtlinien angezeigt, die das TLS-1.3-Protokoll unterstützen. Im Beispiel wird der Parameter `--query` verwendet, um nur die Namen der Richtlinien anzuzeigen.

```
aws elbv2 describe-ssl-policies \
    --load-balancer-type application \
    --query SslPolicies[?contains(SslProtocols,'TLSv1.3')].Name
```

Ausgabe:

```
[
    "ELBSecurityPolicy-TLS13-1-2-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Res-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext1-2021-06",
    "ELBSecurityPolicy-TLS13-1-2-Ext2-2021-06",
    "ELBSecurityPolicy-TLS13-1-1-2021-06",
    "ELBSecurityPolicy-TLS13-1-0-2021-06",
    "ELBSecurityPolicy-TLS13-1-3-2021-06"
]
```

Beispiel 3: So zeigen Sie die Chiffren für eine Richtlinie an

Im folgenden Beispiel für `describe-ssl-policies` werden die Namen der Chiffren der angegebenen Richtlinie angezeigt. Im Beispiel wird der Parameter `--query` verwendet, um nur die Namen der Chiffren anzuzeigen. Die erste Chiffre in der Liste hat Priorität 1. Die übrigen Chiffren sind in der Reihenfolge ihrer Priorität geordnet.

```
aws elbv2 describe-ssl-policies \
    --names ELBSecurityPolicy-TLS13-1-2-2021-06 \
    --query SslPolicies[*].Ciphers[*].Name
```

Ausgabe:

```
[
    "TLS_AES_128_GCM_SHA256",
    "TLS_AES_256_GCM_SHA384",
    "TLS_CHACHA20_POLY1305_SHA256",
    "ECDHE-ECDSA-AES128-GCM-SHA256",
```

```
    "ECDHE-RSA-AES128-GCM-SHA256",
    "ECDHE-ECDSA-AES128-SHA256",
    "ECDHE-RSA-AES128-SHA256",
    "ECDHE-ECDSA-AES256-GCM-SHA384",
    "ECDHE-RSA-AES256-GCM-SHA384",
    "ECDHE-ECDSA-AES256-SHA384",
    "ECDHE-RSA-AES256-SHA384"
]
```

Weitere Informationen finden Sie unter Sicherheitsrichtlinien im Benutzerhandbuch für Application Load Balancers.

- Einzelheiten zur API finden Sie DescribeSslPoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet alle verfügbaren Listener-Richtlinien für ElasticLoadBalancingV2 auf.

```
Get-ELB2SSLPolicy
```

Ausgabe:

```
Ciphers
                                        Name
 SslProtocols
-------
                                        ----
 ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06          {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Einzelheiten zur API finden Sie unter DescribeSslPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet alle verfügbaren Listener-Richtlinien für auf. ElasticLoadBalancingV2

```
Get-ELB2SSLPolicy
```

Ausgabe:

```
Ciphers
                                        Name
 SslProtocols
-------
                                        ----
 ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06          {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05             {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- Einzelheiten zur API finden Sie unter DescribeSslPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die einem Load Balancer zugeordneten Tags

In diesem Beispiel werden die Tags beschrieben, die dem angegebenen Load Balancer zugewiesen sind.

Befehl:

```
aws elbv2 describe-tags --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188
```

Ausgabe:

```
{
  "TagDescriptions": [
      {
          "ResourceArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
          "Tags": [
              {
                  "Value": "lima",
                  "Key": "project"
              },
              {
                  "Value": "digital-media",
                  "Key": "department"
              }
          ]
      }
  ]
}
```

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags für die angegebene Ressource aufgelistet.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
ResourceArn
              Tags
-----------
              ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für die angegebene Ressource aufgelistet.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
ResourceArn
              Tags
-----------
              ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTargetGroupAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTargetGroupAttributes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Zielgruppenattribute

Im folgenden Beispiel für `describe-target-group-attributes` werden die Attribute der angegebenen Zielgruppe angezeigt.

```
aws elbv2 describe-target-group-attributes \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Die Ausgabe enthält die Attribute, wenn das Protokoll HTTP oder HTTPS ist und der Zieltyp `instance` oder `ip` ist.

```
{
    "Attributes": [
```

```
        {
            "Value": "false",
            "Key": "stickiness.enabled"
        },
        {
            "Value": "300",
            "Key": "deregistration_delay.timeout_seconds"
        },
        {
            "Value": "lb_cookie",
            "Key": "stickiness.type"
        },
        {
            "Value": "86400",
            "Key": "stickiness.lb_cookie.duration_seconds"
        },
        {
            "Value": "0",
            "Key": "slow_start.duration_seconds"
        }
    ]
}
```

Die folgende Ausgabe enthält die Attribute, wenn das Protokoll HTTP oder HTTPS ist und der Zieltyp `lambda` ist.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "lambda.multi_value_headers.enabled"
        }
    ]
}
```

Die folgende Ausgabe enthält die Attribute, wenn das Protokoll TCP, TLS, UDP oder TCP_UDP ist.

```
{
    "Attributes": [
        {
            "Value": "false",
            "Key": "proxy_protocol_v2.enabled"
```

```
        },
        {
            "Value": "300",
            "Key": "deregistration_delay.timeout_seconds"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeTargetGroupAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Attribute der angegebenen Zielgruppe beschrieben.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  300
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroupAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Attribute der angegebenen Zielgruppe beschrieben.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
Key                                     Value
---                                     -----
stickiness.enabled                      false
deregistration_delay.timeout_seconds    300
stickiness.type                         lb_cookie
stickiness.lb_cookie.duration_seconds   86400
slow_start.duration_seconds             0
load_balancing.algorithm.type           round_robin
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroupAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeTargetGroups`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTargetGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Zielgruppe

Im folgenden Beispiel für `describe-target-groups` werden Details für die angegebene Zielgruppe gezeigt.

```
aws elbv2 describe-target-groups \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Ausgabe:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
```

```
            "TargetGroupName": "my-targets",
            "Protocol": "HTTP",
            "Port": 80,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckPort": "traffic-port",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "HealthCheckPath": "/",
            "Matcher": {
                "HttpCode": "200"
            },
            "LoadBalancerArns": [
                "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
            ],
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Beispiel 2: So beschreiben Sie alle Zielgruppen für einen Load Balancer

Im folgenden Beispiel für `describe-target-groups` werden Details zu allen Zielgruppen für den angegebenen Load Balancer gezeigt. Im Beispiel wird der Parameter `--query` verwendet, um nur die Namen der Zielgruppen anzuzeigen.

```
aws elbv2 describe-target-groups \
    --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --query TargetGroups[*].TargetGroupName
```

Ausgabe:

```
[
    "my-instance-targets",
    "my-ip-targets",
    "my-lambda-target"
```

```
]
```

Weitere Informationen finden Sie unter Zielgruppen im Application Load Balancers Guide.

- Einzelheiten zur API finden Sie DescribeTargetGroupsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { TargetGroups } = await client.send(
  new DescribeTargetGroupsCommand({
    Names: [NAMES.loadBalancerTargetGroupName],
  }),
);
```

- Einzelheiten zur API finden Sie DescribeTargetGroupsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe beschrieben.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
```

```
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe beschrieben.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
```

```
TargetGroupName             : test-tg
TargetType                  : instance
UnhealthyThresholdCount     : 2
VpcId                       : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter DescribeTargetGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeTargetHealth`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTargetHealth` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit dem Lastenausgleich

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }
```

- Einzelheiten zur API finden Sie DescribeTargetHealthin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie den Zustand der Ziele für eine Zielgruppe

Im folgenden Beispiel für `describe-target-health` werden Zustandsdetails für die Ziele der angegebenen Zielgruppe gezeigt. Diese Ziele sind fehlerfrei.

```
aws elbv2 describe-target-health \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Ausgabe:

```
{
```

```
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-ceddcd4d",
                "Port": 80
            },
            "TargetHealth": {
                "State": "healthy"
            }
        },
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

Beispiel 2: So beschreiben Sie den Zustand eines Ziels

Im folgenden Beispiel für `describe-target-health` werden Details zum Zustand des angegebenen Ziels gezeigt. Das Ziel ist fehlerfrei.

```
aws elbv2 describe-target-health \
    --targets Id=i-0f76fade,Port=80 \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

Ausgabe:

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
```

```
            },
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

Die folgende Beispielausgabe bezieht sich auf ein Ziel, dessen Zielgruppe nicht in einer Aktion für einen Listener angegeben ist. Dieses Ziel kann keinen Datenverkehr vom Load Balancer empfangen.

```
{
    "TargetHealthDescriptions": [
    {
        "HealthCheckPort": "80",
        "Target": {
            "Id": "i-0f76fade",
            "Port": 80
        },
            "TargetHealth": {
                "State": "unused",
                "Reason": "Target.NotInUse",
                "Description": "Target group is not configured to receive traffic
 from the load balancer"
            }
        }
    ]
}
```

Die folgende Beispielausgabe bezieht sich auf ein Ziel, dessen Zielgruppe gerade in einer Aktion für einen Listener angegeben wurde. Das Ziel wird noch registriert.

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
```

```
                "State": "initial",
                "Reason": "Elb.RegistrationInProgress",
                "Description": "Target registration is in progress"
            }
        }
    ]
}
```

Nachstehend sehen Sie eine Beispielausgabe für ein fehlerhaftes Ziel.

```
{
    "TargetHealthDescriptions": [
        {
            "HealthCheckPort": "80",
            "Target": {
                "Id": "i-0f76fade",
                "Port": 80
            },
            "TargetHealth": {
                "State": "unhealthy",
                "Reason": "Target.Timeout",
                "Description": "Connection to target timed out"
            }
        }
    ]
}
```

Die folgende Beispielausgabe bezieht sich auf ein Ziel, bei dem es sich um eine Lambda-Funktion handelt und für das Zustandsprüfungen deaktiviert sind.

```
{
    "TargetHealthDescriptions": [
        {
            "Target": {
                "Id": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
                "AvailabilityZone": "all",
            },
            "TargetHealth": {
                "State": "unavailable",
                "Reason": "Target.HealthCheckDisabled",
                "Description": "Health checks are not enabled for this target"
            }
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeTargetHealthin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Checks the health of the instances in the target group.
   public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
       DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
               .names(targetGroupName)
               .build();

       DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

       DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

.targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
               .build();

       DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
       return healthResponse.targetHealthDescriptions();
   }
```

- Einzelheiten zur API finden Sie DescribeTargetHealthin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const { TargetHealthDescriptions } = await client.send(
  new DescribeTargetHealthCommand({
    TargetGroupArn: TargetGroups[0].TargetGroupArn,
  }),
);
```

- Einzelheiten zur API finden Sie DescribeTargetHealthin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Zustand der in der angegebenen Zielgruppe vorhandenen Ziele zurückgegeben.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckPort Target
 TargetHealth
--------------- ------
 ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Einzelheiten zur API finden Sie unter DescribeTargetHealth AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Zustand der in der angegebenen Zielgruppe vorhandenen Ziele zurückgegeben.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckPort Target
 TargetHealth
--------------- ------
 ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- Einzelheiten zur API finden Sie unter DescribeTargetHealth AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client
```

```
    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
                    "Ensure the load balancer exists, is in the correct AWS
region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

- Einzelheiten zur API finden Sie DescribeTargetHealthin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`ModifyListener`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyListener` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie die Standardaktion in eine Weiterleitungsaktion

Im folgenden Beispiel für `modify-listener` wird die Standardaktion in eine `forward`-Aktion für den angegebenen Listener geändert.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --default-
actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f
```

Ausgabe:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTP",
            "Port": 80,
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-new-targets/2453ed029918f21f"
                }
            ]
        }
    ]
}
```

Beispiel 2: So ändern Sie die Standardaktion in eine Umleitungsaktion

Im folgenden Beispiel für `modify-listener` wird die Standardaktion in eine `redirect`-Aktion für den angegebenen Listener geändert.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 \
    --default-actions Type=redirect,
 RedirectConfig='{Protocol=HTTPS,StatusCode=HTTP_302}'
```

Ausgabe:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTP",
            "Port": 80,
            "DefaultActions": [
                {
                    "Type": "redirect",
                    "RedirectConfig": {
                        "Protocol": "HTTPS",
                        "Port": "#{port}",
                        "Host": "#{host}",
                        "Path": "/#{path}",
                        "Query": "#{query}",
                        "StatusCode": "HTTP_302",
                    }
                }
            ]
        }
    ]
}
```

Beispiel 3: So ändern Sie das Serverzertifikat

Im folgenden Beispiel für `modify-listener` wird das Serverzertifikat für den angegebenen HTTPS-Listener geändert.

```
aws elbv2 modify-listener \
    --listener-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65 \
    --certificates CertificateArn=arn:aws:iam::123456789012:server-certificate/
my-new-server-cert
```

Ausgabe:

```
{
    "Listeners": [
        {
            "ListenerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188",
            "Protocol": "HTTPS",
            "Port": 443,
            "DefaultActions": [
                {
                    "Type": "forward",
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067"
                }
            ],
            "SslPolicy": "ELBSecurityPolicy-2015-05",
            "Certificates": [
                {
                    "CertificateArn": "arn:aws:iam::123456789012:server-
certificate/my-new-server-cert"
                }
            ],
        }
    ]
}
```

Weitere Informationen finden Sie unter Listener-Regeln im Benutzerhandbuch für Application Load Balancers.

- Einzelheiten zur API finden Sie ModifyListenerin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel ändert die Standardaktion des angegebenen Listeners in eine feststehende Antwort.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Einzelheiten zur API finden Sie unter ModifyListener AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel ändert die Standardaktion des angegebenen Listeners in eine feststehende Antwort.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
```

```
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/
app/testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- Einzelheiten zur API finden Sie unter ModifyListener AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyLoadBalancerAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyLoadBalancerAttributes` verwendet wird.

CLI

AWS CLI

So aktivieren Sie den Löschschutz

In diesem Beispiel wird der Löschschutz für den angegebenen Load Balancer aktiviert.

Befehl:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=deletion_protection.enabled,Value=true
```

Ausgabe:

```
{
  "Attributes": [
      {
          "Value": "true",
          "Key": "deletion_protection.enabled"
      },
      {
          "Value": "false",
          "Key": "access_logs.s3.enabled"
      },
      {
          "Value": "60",
          "Key": "idle_timeout.timeout_seconds"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.prefix"
      },
      {
          "Value": "",
          "Key": "access_logs.s3.bucket"
      }
  ]
}
```

So ändern Sie das Leerlauf-Timeout

In diesem Beispiel wird der Timeoutwert für die Leerlaufzeit für den angegebenen Load Balancer geändert.

Befehl:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --attributes Key=idle_timeout.timeout_seconds,Value=30
```

Ausgabe:

```
{
  "Attributes": [
      {
```

```
            "Value": "30",
            "Key": "idle_timeout.timeout_seconds"
        },
        {
            "Value": "false",
            "Key": "access_logs.s3.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "true",
            "Key": "deletion_protection.enabled"
        },
        {
            "Value": "",
            "Key": "access_logs.s3.bucket"
        }
    ]
}
```

So aktivieren Sie Zugriffsprotokolle

In diesem Beispiel werden Zugriffsprotokolle für den angegebenen Load Balancer aktiviert. Dazu muss sich der S3-Bucket in derselben Region wie der Load Balancer befinden und über eine Richtlinie verfügen, die Zugriff auf den Elastic-Load-Balancing-Service gewährt.

Befehl:

```
aws elbv2 modify-load-balancer-attributes --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --
attributes Key=access_logs.s3.enabled,Value=true Key=access_logs.s3.bucket,Value=my-
loadbalancer-logs Key=access_logs.s3.prefix,Value=myapp
```

Ausgabe:

```
{
  "Attributes": [
      {
          "Value": "true",
          "Key": "access_logs.s3.enabled"
```

```
        },
        {
            "Value": "my-load-balancer-logs",
            "Key": "access_logs.s3.bucket"
        },
        {
            "Value": "myapp",
            "Key": "access_logs.s3.prefix"
        },
        {
            "Value": "60",
            "Key": "idle_timeout.timeout_seconds"
        },
        {
            "Value": "false",
            "Key": "deletion_protection.enabled"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ModifyLoadBalancerAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Attribute des angegebenen Load Balancers geändert.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key
 = 'deletion_protection.enabled'; Value = 'true'}
```

Ausgabe:

```
Key                                             Value
---                                             -----
deletion_protection.enabled                     true
access_logs.s3.enabled                          false
access_logs.s3.bucket
access_logs.s3.prefix
```

```
idle_timeout.timeout_seconds                    60
routing.http2.enabled                           true
routing.http.drop_invalid_header_fields.enabled false
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Attribute des angegebenen Load Balancers geändert.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key
 = 'deletion_protection.enabled'; Value = 'true'}
```

Ausgabe:

```
Key                                             Value
---                                             -----
deletion_protection.enabled                     true
access_logs.s3.enabled                          false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                    60
routing.http2.enabled                           true
routing.http.drop_invalid_header_fields.enabled false
```

- Einzelheiten zur API finden Sie unter ModifyLoadBalancerAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyRule`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyRule` verwendet wird.

CLI

AWS CLI

So ändern Sie eine Regel

Im folgenden Beispiel für `modify-rule` werden die Aktionen und Bedingungen für die angegebene Regel aktualisiert.

```
aws elbv2 modify-rule \
  --actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
  --conditions Field=path-pattern,Values='/images/*'
  --rule-arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener-rule/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2/9683b2d02a6cabee
```

Ausgabe:

```
{
    "Rules": [
        {
            "Priority": "10",
            "Conditions": [
                {
                    "Field": "path-pattern",
                    "Values": [
                        "/images/*"
                    ]
                }
            ],
            "RuleArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/9683b2d02a6cabee",
            "IsDefault": false,
            "Actions": [
                {
                    "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                    "Type": "forward"
                }
            ]
        }
    ]
}
```

- Einzelheiten zur API finden Sie ModifyRulein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die angegebenen Listener-Regelkonfigurationen geändert.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Einzelheiten zur API finden Sie unter ModifyRule AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die angegebenen Listener-Regelkonfigurationen geändert.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}
```

```
Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- Einzelheiten zur API finden Sie unter ModifyRule AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyTargetGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyTargetGroup` verwendet wird.

CLI

AWS CLI

So ändern Sie die Konfiguration der Zustandsprüfungen für eine Zielgruppe

Im folgenden Beispiel für `modify-target-group` wird die Konfiguration der Zustandsprüfungen geändert, die zur Bewertung des Zustands der Ziele für die angegebene Zielgruppe verwendet werden. Aufgrund der Art und Weise, wie die CLI Kommas analysiert, müssen Sie dabei den Bereich für die Option `--matcher` mit einfachen statt mit doppelten Anführungszeichen umgeben.

```
aws elbv2 modify-target-group \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f \
    --health-check-protocol HTTPS \
    --health-check-port 443 \
    --matcher HttpCode='200,299'
```

Ausgabe:

```
{
    "TargetGroups": [
        {
            "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-https-targets/2453ed029918f21f",
            "TargetGroupName": "my-https-targets",
            "Protocol": "HTTPS",
            "Port": 443,
            "VpcId": "vpc-3ac0fb5f",
            "HealthCheckProtocol": "HTTPS",
            "HealthCheckPort": "443",
            "HealthCheckEnabled": true,
            "HealthCheckIntervalSeconds": 30,
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 5,
            "UnhealthyThresholdCount": 2,
            "Matcher": {
                "HttpCode": "200,299"
            },
            "LoadBalancerArns": [
                "arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188"
            ],
            "TargetType": "instance",
            "ProtocolVersion": "HTTP1",
            "IpAddressType": "ipv4"
        }
    ]
}
```

Weitere Informationen finden Sie unter Zielgruppen im Application Load Balancers Guide.

- Einzelheiten zur API finden Sie ModifyTargetGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Eigenschaften der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -
HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter ModifyTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Eigenschaften der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -
HealthCheckIntervalSecond 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
```

```
HealthCheckProtocol         : HTTP
HealthCheckTimeoutSeconds   : 5
HealthyThresholdCount       : 5
LoadBalancerArns            : {}
Matcher                     : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                        : 80
Protocol                    : HTTP
TargetGroupArn              : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName             : test-tg
TargetType                  : instance
UnhealthyThresholdCount     : 2
VpcId                       : vpc-2cfd7000
```

- Einzelheiten zur API finden Sie unter ModifyTargetGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyTargetGroupAttributes`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyTargetGroupAttributes` verwendet wird.

CLI

AWS CLI

So ändern Sie das Zeitlimit für die Verzögerung bei der Abmeldung

In diesem Beispiel wird das Zeitlimit für die Verzögerung bei der Abmeldung auf den angegebenen Wert für die angegebene Zielgruppe festgelegt.

Befehl:

```
aws elbv2 modify-target-group-attributes --target-group-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067 --
attributes Key=deregistration_delay.timeout_seconds,Value=600
```

Ausgabe:

```
{
  "Attributes": [
```

```
        {
            "Value": "false",
            "Key": "stickiness.enabled"
        },
        {
            "Value": "600",
            "Key": "deregistration_delay.timeout_seconds"
        },
        {
            "Value": "lb_cookie",
            "Key": "stickiness.type"
        },
        {
            "Value": "86400",
            "Key": "stickiness.lb_cookie.duration_seconds"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ModifyTargetGroupAttributesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Attribut deregistration_delay der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Ausgabe:

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  600
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
```

```
load_balancing.algorithm.type         round_robin
```

- Einzelheiten zur API finden Sie unter ModifyTargetGroupAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut deregistration_delay der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Ausgabe:

```
Key                                   Value
---                                   -----
stickiness.enabled                    false
deregistration_delay.timeout_seconds  600
stickiness.type                       lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds           0
load_balancing.algorithm.type         round_robin
```

- Einzelheiten zur API finden Sie unter ModifyTargetGroupAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterTargets` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Lastenausgleich

CLI

AWS CLI

Beispiel 1: So registrieren Sie Ziele anhand der Instance-ID bei einer Zielgruppe

Im folgenden Beispiel für `register-targets` werden die angegebenen Instances bei einer Zielgruppe registriert. Die Zielgruppe muss den Zieltyp `instance` aufweisen.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067 \
    --targets Id=i-1234567890abcdef0 Id=i-0abcdef1234567890
```

Beispiel 2: So registrieren Sie Ziele mithilfe von Port-Überschreibungen bei einer Zielgruppe

Im folgenden Beispiel für `register-targets` wird die angegebene Instance über mehrere Ports bei einer Zielgruppe registriert. Dadurch können Sie Container auf derselben Instance wie Ziele in der Zielgruppe registrieren.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-internal-targets/3bb63f11dfb0faf9 \
    --targets Id=i-0598c7d356eba48d7,Port=80 Id=i-0598c7d356eba48d7,Port=766
```

Beispiel 3: So registrieren Sie Ziele anhand der IP-Adresse bei einer Zielgruppe

Im folgenden Beispiel für `register-targets` werden die angegebenen IP-Adressen bei einer Zielgruppe registriert. Die Zielgruppe muss den Zieltyp `ip` aufweisen.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \
    --targets Id=10.0.1.15 Id=10.0.1.23
```

Beispiel 4: So registrieren Sie eine Lambda-Funktion als Ziel

Im folgenden Beispiel für `register-targets` werden die angegebenen IP-Adressen bei einer Zielgruppe registriert. Die Zielgruppe muss den Zieltyp `lambda` aufweisen. Sie müssen Elastic Load Balancing Berechtigungen zum Aufrufen der Lambda-Funktion gewähren.

```
aws elbv2 register-targets \
    --target-group-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-tcp-ip-targets/8518e899d173178f \
    --targets Id=arn:aws:lambda:us-west-2:123456789012:function:my-function
```

- Einzelheiten zur API finden Sie RegisterTargetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Instance „i-0672a4c4cdeae3111” bei der angegebenen Zielgruppe registriert.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id
 = 'i-0672a4c4cdeae3111'}
```

- Einzelheiten zur API finden Sie unter RegisterTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Instance „i-0672a4c4cdeae3111” bei der angegebenen Zielgruppe registriert.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id
 = 'i-0672a4c4cdeae3111'}
```

- Einzelheiten zur API finden Sie unter RegisterTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveListenerCertificates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveListenerCertificates` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein Zertifikat von einem sicheren Listener

In diesem Beispiel wird ein Zertifikat vom angegebenen sicheren Listener entfernt.

Befehl:

```
aws elbv2 remove-listener-certificates --listener-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/
app/my-load-balancer/50dc6c495c0c9188/f2f7dc8efc522ab2 --
certificates CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/5cc54884-f4a3-4072-80be-05b9ba72f705
```

- Einzelheiten zur API finden Sie RemoveListenerCertificatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Zertifikat aus der spezifizierten Zielgruppe entfernt.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -
ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter RemoveListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Zertifikat aus der spezifizierten Zielgruppe entfernt.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -
```

```
ListenerArn 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter RemoveListenerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTags` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einem Load Balancer

Im folgenden Beispiel für `remove-tags` werden die Tags `project` und `department` vom angegebenen Load Balancer entfernt.

```
aws elbv2 remove-tags \
    --resource-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 \
    --tag-keys project department
```

- Einzelheiten zur API finden Sie RemoveTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Tag für den angegebenen Schlüssel entfernt.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter RemoveTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Tag für den angegebenen Schlüssel entfernt.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- Einzelheiten zur API finden Sie unter RemoveTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetIpAddressType`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetIpAddressType` verwendet wird.

CLI

AWS CLI

So stellen Sie den Adresstyp eines Load Balancers ein

In diesem Beispiel wird der Adresstyp des angegebenen Load Balancers auf `dualstack` eingestellt. Die Subnetze des Load Balancers müssen über zugeordnete IPv6-CIDR-Blöcke verfügen.

Befehl:

```
aws elbv2 set-ip-address-type --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --ip-address-type dualstack
```

Ausgabe:

```
{
    "IpAddressType": "dualstack"
}
```

- Einzelheiten zur API finden Sie SetIpAddressTypein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der IP-Adresstyp des Load Balancers von 'IPv4' auf '' geändert. DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Ausgabe:

```
Value
-----
dualstack
```

- Einzelheiten zur API finden Sie unter SetIpAddressType AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der IP-Adresstyp des Load Balancers von 'IPv4' auf '' geändert. DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Ausgabe:

```
Value
-----
dualstack
```

- Einzelheiten zur API finden Sie unter SetIpAddressType AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetRulePriorities`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetRulePriorities` verwendet wird.

CLI

AWS CLI

So stellen Sie die Regelpriorität ein

In diesem Beispiel wird die Priorität der angegebenen Regel eingestellt.

Befehl:

```
aws elbv2 set-rule-priorities --rule-
priorities RuleArn=arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/1291d13826f405c3,Priority=5
```

Ausgabe:

```
{
  "Rules": [
      {
          "Priority": "5",
          "Conditions": [
              {
                  "Field": "path-pattern",
                  "Values": [
                      "/img/*"
                  ]
              }
          ],
          "RuleArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:listener-rule/app/my-load-balancer/50dc6c495c0c9188/
f2f7dc8efc522ab2/1291d13826f405c3",
          "IsDefault": false,
          "Actions": [
              {
                  "TargetGroupArn": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
                  "Type": "forward"
              }
          ]
      }
  ]
}
```

- Einzelheiten zur API finden Sie SetRulePrioritiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Priorität der angegebenen Listener-Regel geändert.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
```

```
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Einzelheiten zur API finden Sie unter SetRulePriorities AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Priorität der angegebenen Listener-Regel geändert.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/
app/test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- Einzelheiten zur API finden Sie unter SetRulePriorities AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetSecurityGroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetSecurityGroups` verwendet wird.

CLI

AWS CLI

So ordnen Sie einem Load Balancer eine Sicherheitsgruppe zu

In diesem Beispiel wird dem angegebenen Load Balancer die genannte Sicherheitsgruppe zugeordnet.

Befehl:

```
aws elbv2 set-security-groups --load-balancer-
arn arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/my-load-
balancer/50dc6c495c0c9188 --security-groups sg-5943793c
```

Ausgabe:

```
{
  "SecurityGroupIds": [
      "sg-5943793c"
  ]
}
```

- Einzelheiten zur API finden Sie SetSecurityGroupsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer die Sicherheitsgruppe „sg-07c3414abb8811cbd” hinzugefügt.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Ausgabe:

```
sg-07c3414abb8811cbd
```

- Einzelheiten zur API finden Sie unter SetSecurityGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer die Sicherheitsgruppe „sg-07c3414abb8811cbd” hinzugefügt.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Ausgabe:

```
sg-07c3414abb8811cbd
```

- Einzelheiten zur API finden Sie unter SetSecurityGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SetSubnets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SetSubnets` verwendet wird.

CLI

AWS CLI

So aktivieren Sie Availability Zones für einen Load Balancer

In diesem Beispiel wird die Availability Zone für das entsprechende Subnetz des angegebenen Load Balancers aktiviert.

Befehl:

```
aws elbv2 set-subnets --load-balancer-arn arn:aws:elasticloadbalancing:us-
west-2:123456789012:loadbalancer/app/my-load-balancer/50dc6c495c0c9188 --
subnets subnet-8360a9e7 subnet-b7d581c0
```

Ausgabe:

```
{
  "AvailabilityZones": [
      {
          "SubnetId": "subnet-8360a9e7",
          "ZoneName": "us-west-2a"
      },
      {
          "SubnetId": "subnet-b7d581c0",
          "ZoneName": "us-west-2b"
      }
  ]
}
```

- Einzelheiten zur API finden Sie SetSubnetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Subnetze des angegebenen Load Balancers geändert.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Ausgabe:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Einzelheiten zur API finden Sie unter SetSubnets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Subnetze des angegebenen Load Balancers geändert.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Ausgabe:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- Einzelheiten zur API finden Sie unter SetSubnets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für Elastic Load Balancing — Version 2 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Elastic Load Balancing — Version 2 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Elastic Load Balancing – Version 2 oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
- Erste Schritte mit dem Lastenausgleich

Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
  calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
```

```
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);
```

```
            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =
```

```
 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
```

```
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
```

```
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
```

```
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
```

```
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
```

```
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();
```

```
            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();
```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
```

```
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
```

```
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
```

```
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
      type: "confirm",
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
```

```
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
```

```
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
```

```
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
```

```
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
```

```
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
```

```
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
```

```
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
```

```
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
```

```
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
```

```
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
```

```
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
```

```
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
```

```
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
```

```
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
```

```
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
```

```
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
```

```
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
```

```
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
```

```
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
```

```
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
```

```
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
```

```
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
```

```
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
```

```
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
```

```
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
```

```
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
```

```
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
```

```
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```

```
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
```

```
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
```

```
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
```

```
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.

- AttachLoadBalancerTargetGroups
- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")
```

```
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
```

```
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
```

```
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
```

```
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
```

```
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
```

```
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.
```

```
        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
```

```
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
```

```
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
```

```
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
```

```
            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
```

```
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
```

```
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
```

```
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
```

```
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
```

```
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")
```

```
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
```

```
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
```

```
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
 {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
```

```
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
```

```
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
 found. "
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
```

```
        Initializes the RecommendationService class with the necessary
parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
```

```
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
```

```
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Erste Schritte mit dem Lastenausgleich

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Application Load Balancers
- Erstellen einer Zielgruppe
- Erstellen eines Listeners
- Überprüfen Ihrer Konfiguration

- Fügen Sie einen HTTPS-Listener hinzu (optional)
- Fügen Sie pfadbasiertes Routing hinzu (optional)
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
```

```
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null ||
 true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN"
 2>/dev/null || true
```

```
        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/
dev/null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/
null && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30
 seconds... (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID"
 >&2
            echo "You may need to delete it manually using: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
```

```
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=elastic-load-
balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict
 to specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo
 "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
```

```
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo
 "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-
name,Values=running" \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null
 || echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
```

```
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-
gs \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
```

```
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r
 CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup

- DeleteTargetGroup
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetHealth
- DescribeVpcs
- Hilfe
- RegisterTargets
- Wait

Codebeispiele für die ElastiCache Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon ElastiCache mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- ElastiCache Benutzerhandbuch — Weitere Informationen zu ElastiCache.
- ElastiCache API-Referenz — Details zu allen verfügbaren ElastiCache Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung ElastiCache AWS SDKs
  - Erste Schritte mit In-Memory-Caching

Szenarien für die Verwendung ElastiCache AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien ElastiCache mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen

können, indem Sie mehrere Funktionen innerhalb ElastiCache oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit In-Memory-Caching

Erste Schritte mit In-Memory-Caching

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine Sicherheitsgruppe für den ElastiCache Zugriff ein
- Erstellen Sie einen serverlosen Valkey-Cache
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")
```

```
if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379"
 == *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
```

```
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380"
 == *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from
 any IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs
 2>&1); then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""
```

```
while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' |
 awk -F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
```

```
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
```

```
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to
 complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
```

```
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Codebeispiele für die MediaStore Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elemental MediaStore mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- MediaStore Benutzerhandbuch — Weitere Informationen zu MediaStore.
- MediaStore API-Referenz — Details zu allen verfügbaren MediaStore Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung MediaStore AWS SDKs
  - Aktionen zur Verwendung MediaStore AWS SDKs
    - Verwenden Sie CreateContainermit einem AWS SDK oder CLI
    - Verwenden Sie DeleteContainermit einem AWS SDK oder CLI
    - Verwenden Sie DeleteObjectmit einem AWS SDK
    - Verwenden Sie DescribeContainermit einem AWS SDK oder CLI
    - Verwenden Sie GetObjectmit einem AWS SDK oder CLI
    - Verwenden Sie ListContainersmit einem AWS SDK oder CLI
    - Verwenden Sie PutObjectmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung MediaStore AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Elemental MediaStore mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung MediaStore AWS SDKs
  - Verwenden Sie CreateContainermit einem AWS SDK oder CLI
  - Verwenden Sie DeleteContainermit einem AWS SDK oder CLI
  - Verwenden Sie DeleteObjectmit einem AWS SDK
  - Verwenden Sie DescribeContainermit einem AWS SDK oder CLI
  - Verwenden Sie GetObjectmit einem AWS SDK oder CLI
  - Verwenden Sie ListContainersmit einem AWS SDK oder CLI
  - Verwenden Sie PutObjectmit einem AWS SDK oder CLI

Aktionen zur Verwendung MediaStore AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne MediaStore Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Elemental MediaStore -API-Referenz.

Beispiele

- Verwenden Sie CreateContainermit einem AWS SDK oder CLI
- Verwenden Sie DeleteContainermit einem AWS SDK oder CLI
- Verwenden Sie DeleteObjectmit einem AWS SDK
- Verwenden Sie DescribeContainermit einem AWS SDK oder CLI
- Verwenden Sie GetObjectmit einem AWS SDK oder CLI
- Verwenden Sie ListContainersmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectmit einem AWS SDK oder CLI

Verwenden Sie **`CreateContainer`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateContainer` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Container

Im folgenden Beispiel für `create-container` wird ein neuer leerer Container erstellt.

```
aws mediastore create-container --container-name ExampleContainer
```

Ausgabe:

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines Containers im AWS Elemental MediaStore User Guide.

- Einzelheiten zur API finden Sie CreateContainerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateContainerin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteContainer`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteContainer` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Container

Im folgenden Beispiel für `delete-container` wird der angegebene Container gelöscht. Sie können einen Container nur löschen, wenn er keine Objekte enthält.

```
aws mediastore delete-container \
    --container-name=ExampleLiveDemo
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines Containers im AWS Elemental MediaStore User Guide.

- Einzelheiten zur API finden Sie DeleteContainerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
 String containerName) {
        try {
            CreateContainerRequest containerRequest =
 CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
 mediaStoreClient.createContainer(containerRequest);
```

```
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
 containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
 containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteContainerin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteObject`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
```

```
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item
 to delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
```

```
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData,
 String completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeContainer`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeContainer` verwendet wird.

CLI

AWS CLI

So zeigen Sie die Details für einen Container an

Im folgenden Beispiel für `describe-container` werden die Details des angegebenen Containers angezeigt.

```
aws mediastore describe-container \
    --container-name ExampleContainer
```

Ausgabe:

```
{
    "Container": {
        "CreationTime": 1563558086,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

Weitere Informationen finden Sie unter Anzeigen der Details für einen Container im AWS Elemental MediaStore User Guide.

- Einzelheiten zur API finden Sie unter DescribeContainer AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }
```

```
    public static String checkContainer(MediaStoreClient mediaStoreClient, String
 containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
 DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
 mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
 containerResponse.container().name());
            System.out.println("The container ARN is " +
 containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie DescribeContainerin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetObject`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObject` verwendet wird.

CLI

AWS CLI

So laden Sie ein Objekt herunter

Im folgenden Beispiel für `get-object` wird ein Objekt in den angegebenen Endpunkt heruntergeladen.

```
aws mediastore-data get-object \
```

```
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path=/folder_name/README.md README.md
```

Ausgabe:

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

So laden Sie einen Teil eines Objekts herunter

Im folgenden Beispiel für `get-object` wird ein Teil eines Objekts in den angegebenen Endpunkt heruntergeladen.

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path /folder_name/README.md \
    --range="bytes=0-100" README2.md
```

Ausgabe:

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

Weitere Informationen finden Sie unter Objekt herunterladen im AWS MediaStore Elemental-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetObjectin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>
```

```
               Where:
                  completePath - The path of the object in the container (for
example, Videos5/sampleVideo.mp4).
                  containerName - The name of the container.
                  savePath - The path on the local drive where the file is
saved, including the file name (for example, C:/AWS/myvid.mp4).
               """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       String completePath = args[0];
       String containerName = args[1];
       String savePath = args[2];

       Region region = Region.US_EAST_1;
       URI uri = new URI(getEndpoint(containerName));
       MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
               .endpointOverride(uri)
               .region(region)
               .build();

       getMediaObject(mediaStoreData, completePath, savePath);
       mediaStoreData.close();
   }

   public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
completePath, String savePath) {

       try {
           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .path(completePath)
                   .build();

           // Write out the data to a file.
           ResponseInputStream<GetObjectResponse> data =
mediaStoreData.getObject(objectRequest);
           byte[] buffer = new byte[data.available()];
           data.read(buffer);

           File targetFile = new File(savePath);
```

```
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListContainers`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListContainers` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine Liste der Container an

Im folgenden Beispiel für `list-containers` wird eine Liste aller Container angezeigt, die dem Konto zugeordnet sind.

```
aws mediastore list-containers
```

Ausgabe:

```
{
    "Containers": [
        {
            "CreationTime": 1505317931,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

Weitere Informationen finden Sie unter Eine Liste von Containern anzeigen im AWS Elemental MediaStore User Guide.

- Einzelheiten zur API finden Sie unter ListContainers AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
```

```
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListContainersin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **PutObject**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObject` verwendet wird.

CLI

AWS CLI

So laden Sie ein Objekt hoch

Im folgenden Beispiel für `put-object` wird ein Objekt in den angegebenen Container hochgeladen. Sie können einen Ordnerpfad angeben, unter dem das Objekt im Container gespeichert wird. Wenn der Ordner bereits existiert, MediaStore speichert AWS Elemental das Objekt im Ordner. Wenn der Ordner nicht vorhanden, legt der Service ihn an und speichert das Objekt in dem Ordner.

```
aws mediastore-data put-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --body README.md \
    --path /folder_name/README.md \
    --cache-control "max-age=6, public" \
    --content-type binary/octet-stream
```

Ausgabe:

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

Weitere Informationen finden Sie unter Hochladen eines Objekts im AWS Elemental-Benutzerhandbuch. MediaStore

- Einzelheiten zur API finden Sie PutObjectin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file
 location to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
```

```
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon EMR mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon EMR mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-EMR-Managementhandbuch – Weitere Informationen zu Amazon EMR.
- API-Referenz für Amazon EMR – Details zu allen verfügbaren Amazon-EMR-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon EMR AWS SDKs
  - Aktionen für Amazon EMR unter Verwendung AWS SDKs
    - Verwenden Sie AddJobFlowStepsmit einem AWS SDK
    - Verwenden Sie DescribeClustermit einem AWS SDK oder CLI
    - Verwenden Sie DescribeStepmit einem AWS SDK oder CLI
    - Verwenden Sie ListStepsmit einem AWS SDK oder CLI
    - Verwenden Sie RunJobFlowmit einem AWS SDK
    - Verwenden Sie TerminateJobFlowsmit einem AWS SDK
- Szenarien für die Verwendung von Amazon EMR AWS SDKs
  - Erstellen Sie einen kurzlebigen Amazon EMR-Cluster und führen Sie einen Schritt mit einem AWS SDK
  - Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen
  - Führen Sie ein Shell-Skript aus, um Bibliotheken auf Amazon EMR-Instances mit einem zu installieren AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon EMR AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon EMR mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon EMR unter Verwendung AWS SDKs

- Verwenden Sie AddJobFlowStepsmit einem AWS SDK
- Verwenden Sie DescribeClustermit einem AWS SDK oder CLI
- Verwenden Sie DescribeStepmit einem AWS SDK oder CLI
- Verwenden Sie ListStepsmit einem AWS SDK oder CLI
- Verwenden Sie RunJobFlowmit einem AWS SDK
- Verwenden Sie TerminateJobFlowsmit einem AWS SDK

Aktionen für Amazon EMR unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon EMR-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-EMR-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon EMR AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon-EMR-API-Referenz.

Beispiele

- Verwenden Sie AddJobFlowStepsmit einem AWS SDK
- Verwenden Sie DescribeClustermit einem AWS SDK oder CLI
- Verwenden Sie DescribeStepmit einem AWS SDK oder CLI
- Verwenden Sie ListStepsmit einem AWS SDK oder CLI
- Verwenden Sie RunJobFlowmit einem AWS SDK
- Verwenden Sie TerminateJobFlowsmit einem AWS SDK

Verwenden Sie **`AddJobFlowSteps`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `AddJobFlowSteps` verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie einen Spark-Schritt hinzu, der vom Cluster ausgeführt wird, sobald er hinzugefügt wurde.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):
    """
    Adds a job step to the specified cluster. This example adds a Spark
    step, which is run by the cluster as soon as it is added.

    :param cluster_id: The ID of the cluster.
    :param name: The name of the step.
    :param script_uri: The URI where the Python script is stored.
    :param script_args: Arguments to pass to the Python script.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly added step.
    """
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id,
            Steps=[
                {
                    "Name": name,
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            script_uri,
                            *script_args,
                        ],
                    },
```

```
                }
            ],
        )
        step_id = response["StepIds"][0]
        logger.info("Started step with ID %s", step_id)
    except ClientError:
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)
        raise
    else:
        return step_id
```

Führen Sie einen Befehl des Amazon-EMR-Dateisystems (EMRFS) als Auftragsschritt auf einem Cluster aus. Dies kann verwendet werden, um EMRFS-Befehle auf einem Cluster zu automatisieren, anstatt Befehle manuell über eine SSH-Verbindung auszuführen.

```
import boto3
from botocore.exceptions import ClientError


def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
    """
    Add an EMRFS command as a job flow step to an existing cluster.

    :param command: The EMRFS command to run.
    :param bucket_url: The URL of a bucket that contains tracking metadata.
    :param cluster_id: The ID of the cluster to update.
    :param emr_client: The Boto3 Amazon EMR client object.
    :return: The ID of the added job flow step. Status can be tracked by calling
             the emr_client.describe_step() function.
    """
    job_flow_step = {
        "Name": "Example EMRFS Command Step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
            "Jar": "command-runner.jar",
            "Args": ["/usr/bin/emrfs", command, bucket_url],
        },
    }

    try:
```

```
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id, Steps=[job_flow_step]
        )
        step_id = response["StepIds"][0]
        print(f"Added step {step_id} to cluster {cluster_id}.")
    except ClientError:
        print(f"Couldn't add a step to cluster {cluster_id}.")
        raise
    else:
        return step_id


def usage_demo():
    emr_client = boto3.client("emr")
    # Assumes the first waiting cluster has EMRFS enabled and has created
 metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=["WAITING"])["Clusters"][0]
    add_emrfs_step(
        "sync", "s3://elasticmapreduce/samples/cloudfront", cluster["Id"],
 emr_client
    )


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie AddJobFlowStepsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Build args list for Spark submit
        DATA lt_args TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( 'spark-submit' ) TO lt_args.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( '--deploy-mode' ) TO lt_args.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( 'cluster' ) TO lt_args.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_script_uri ) TO lt_args.
        APPEND LINES OF it_script_args TO lt_args.

        " Create step configuration
        DATA(lo_hadoop_jar_step) = NEW /aws1/cl_emrhadoopjarstepcfg(
          iv_jar = 'command-runner.jar'
          it_args = lt_args
        ).

        DATA(lo_step_config) = NEW /aws1/cl_emrstepconfig(
          iv_name = iv_name
          iv_actiononfailure = 'CONTINUE'
          io_hadoopjarstep = lo_hadoop_jar_step
        ).

        DATA lt_steps TYPE /aws1/cl_emrstepconfig=>tt_stepconfiglist.
        APPEND lo_step_config TO lt_steps.

        DATA(lo_result) = lo_emr->addjobflowsteps(
          iv_jobflowid = iv_cluster_id
          it_steps = lt_steps
        ).

        " Get first step ID
        DATA(lt_step_ids) = lo_result->get_stepids( ).
        READ TABLE lt_step_ids INDEX 1 INTO DATA(lo_step_id_obj).
        IF sy-subrc = 0.
          ov_step_id = lo_step_id_obj->get_value( ).
          MESSAGE |Step added with ID { ov_step_id }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AddJobFlowStepsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeCluster`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Big-Data-Verarbeitungsclustern

CLI

AWS CLI

Befehl:

```
aws emr describe-cluster --cluster-id j-XXXXXXXX
```

Ausgabe:

```
For release-label based uniform instance groups cluster:

        {
            "Cluster": {
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1436475075.199,
                        "CreationDateTime": 1436474656.563,
                    },
                    "State": "WAITING",
                    "StateChangeReason": {
                        "Message": "Waiting for steps to run"
                    }
                },
                "Ec2InstanceAttributes": {
                    "ServiceAccessSecurityGroup": "sg-xxxxxxxx",
                    "EmrManagedMasterSecurityGroup": "sg-xxxxxxxx",
                    "IamInstanceProfile": "EMR_EC2_DefaultRole",
                    "Ec2KeyName": "myKey",
                    "Ec2AvailabilityZone": "us-east-1c",
```

```
                "EmrManagedSlaveSecurityGroup": "sg-yyyyyyyyy"
            },
            "Name": "My Cluster",
            "ServiceRole": "EMR_DefaultRole",
            "Tags": [],
            "TerminationProtected": true,
            "UnhealthyNodeReplacement": true,
            "ReleaseLabel": "emr-4.0.0",
            "NormalizedInstanceHours": 96,
            "InstanceGroups": [
                {
                    "RequestedInstanceCount": 2,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1436475074.245,
                            "CreationDateTime": 1436474656.564,
                            "EndDateTime": 1436638158.387
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": "",
                        }
                    },
                    "Name": "CORE",
                    "InstanceGroupType": "CORE",
                    "Id": "ig-YYYYYYY",
                    "Configurations": [],
                    "InstanceType": "m3.large",
                    "Market": "ON_DEMAND",
                    "RunningInstanceCount": 2
                },
                {
                    "RequestedInstanceCount": 1,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1436475074.245,
                            "CreationDateTime": 1436474656.564,
                            "EndDateTime": 1436638158.387
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": "",
                        }
                    },
```

```
                        "Name": "MASTER",
                        "InstanceGroupType": "MASTER",
                        "Id": "ig-XXXXXXXXX",
                        "Configurations": [],
                        "InstanceType": "m3.large",
                        "Market": "ON_DEMAND",
                        "RunningInstanceCount": 1
                    }
                ],
                "Applications": [
                    {
                        "Name": "Hadoop"
                    }
                ],
                "VisibleToAllUsers": true,
                "BootstrapActions": [],
                "MasterPublicDnsName":
 "ec2-54-147-144-78.compute-1.amazonaws.com",
                "AutoTerminate": false,
                "Id": "j-XXXXXXXX",
                "Configurations": [
                    {
                        "Properties": {
                            "fs.s3.consistent.retryPeriodSeconds": "20",
                            "fs.s3.enableServerSideEncryption": "true",
                            "fs.s3.consistent": "false",
                            "fs.s3.consistent.retryCount": "2"
                        },
                        "Classification": "emrfs-site"
                    }
                ]
            }
        }


For release-label based instance fleet cluster:
{
    "Cluster": {
        "Status": {
            "Timeline": {
                "ReadyDateTime": 1487897289.705,
                "CreationDateTime": 1487896933.942
            },
            "State": "WAITING",
```

```
            "StateChangeReason": {
                "Message": "Waiting for steps to run"
            }
        },
        "Ec2InstanceAttributes": {
            "EmrManagedMasterSecurityGroup": "sg-xxxxx",
            "RequestedEc2AvailabilityZones": [],
            "RequestedEc2SubnetIds": [],
            "IamInstanceProfile": "EMR_EC2_DefaultRole",
            "Ec2AvailabilityZone": "us-east-1a",
            "EmrManagedSlaveSecurityGroup": "sg-xxxxx"
        },
        "Name": "My Cluster",
        "ServiceRole": "EMR_DefaultRole",
        "Tags": [],
        "TerminationProtected": false,
        "UnhealthyNodeReplacement": false,
        "ReleaseLabel": "emr-5.2.0",
        "NormalizedInstanceHours": 472,
        "InstanceCollectionType": "INSTANCE_FLEET",
        "InstanceFleets": [
            {
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1487897212.74,
                        "CreationDateTime": 1487896933.948
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "ProvisionedSpotCapacity": 1,
                "Name": "MASTER",
                "InstanceFleetType": "MASTER",
                "LaunchSpecifications": {
                    "SpotSpecification": {
                        "TimeoutDurationMinutes": 60,
                        "TimeoutAction": "TERMINATE_CLUSTER"
                    }
                },
                "TargetSpotCapacity": 1,
                "ProvisionedOnDemandCapacity": 0,
                "InstanceTypeSpecifications": [
```

```
                    {
                        "BidPrice": "0.5",
                        "InstanceType": "m3.xlarge",
                        "WeightedCapacity": 1
                    }
                ],
                "Id": "if-xxxxxxx",
                "TargetOnDemandCapacity": 0
            }
        ],
        "Applications": [
            {
                "Version": "2.7.3",
                "Name": "Hadoop"
            }
        ],
        "ScaleDownBehavior": "TERMINATE_AT_INSTANCE_HOUR",
        "VisibleToAllUsers": true,
        "BootstrapActions": [],
        "MasterPublicDnsName": "ec2-xxx-xx-xxx-xx.compute-1.amazonaws.com",
        "AutoTerminate": false,
        "Id": "j-xxxxx",
        "Configurations": []
    }
}

For ami based uniform instance group cluster:

    {
        "Cluster": {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1399400564.432,
                    "CreationDateTime": 1399400268.62
                },
                "State": "WAITING",
                "StateChangeReason": {
                    "Message": "Waiting for steps to run"
                }
            },
            "Ec2InstanceAttributes": {
                "IamInstanceProfile": "EMR_EC2_DefaultRole",
                "Ec2AvailabilityZone": "us-east-1c"
            },
```

```
            "Name": "My Cluster",
            "Tags": [],
            "TerminationProtected": true,
            "UnhealthyNodeReplacement": true,
            "RunningAmiVersion": "2.5.4",
            "InstanceGroups": [
                {
                    "RequestedInstanceCount": 1,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1399400558.848,
                            "CreationDateTime": 1399400268.621
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Name": "Master instance group",
                    "InstanceGroupType": "MASTER",
                    "InstanceType": "m1.small",
                    "Id": "ig-ABCD",
                    "Market": "ON_DEMAND",
                    "RunningInstanceCount": 1
                },
                {
                    "RequestedInstanceCount": 2,
                    "Status": {
                        "Timeline": {
                            "ReadyDateTime": 1399400564.439,
                            "CreationDateTime": 1399400268.621
                        },
                        "State": "RUNNING",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Name": "Core instance group",
                    "InstanceGroupType": "CORE",
                    "InstanceType": "m1.small",
                    "Id": "ig-DEF",
                    "Market": "ON_DEMAND",
                    "RunningInstanceCount": 2
                }
```

```
            ],
            "Applications": [
                {
                    "Version": "1.0.3",
                    "Name": "hadoop"
                }
            ],
            "BootstrapActions": [],
            "VisibleToAllUsers": false,
            "RequestedAmiVersion": "2.4.2",
            "LogUri": "s3://myLogUri/",
            "AutoTerminate": false,
            "Id": "j-XXXXXXXX"
        }
    }
```

- Einzelheiten zur API finden Sie DescribeClusterin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response["Cluster"]
        logger.info("Got data for cluster %s.", cluster["Name"])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
```

```
        raise
    else:
        return cluster
```

- Einzelheiten zur API finden Sie DescribeClusterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_emr->describecluster(
          iv_clusterid = iv_cluster_id
        ).
        DATA(lo_cluster) = oo_result->get_cluster( ).
        DATA(lv_cluster_name) = lo_cluster->get_name( ).
        MESSAGE |Retrieved cluster information for { lv_cluster_name }| TYPE 'I'.
      CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
        lv_error = lo_invalid_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeClusterin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeStep`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStep` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Big-Data-Verarbeitungsclustern

CLI

AWS CLI

Mit dem folgenden Befehl wird ein Schritt mit der Schritt-ID `s-3LZC0QUT43AM` in einem Cluster mit der Cluster-ID `j-3SD91U2E1L2QX` beschrieben:

```
aws emr describe-step --cluster-id j-3SD91U2E1L2QX --step-id s-3LZC0QUT43AM
```

Ausgabe:

```
{
    "Step": {
        "Status": {
            "Timeline": {
                "EndDateTime": 1433200470.481,
                "CreationDateTime": 1433199926.597,
                "StartDateTime": 1433200404.959
            },
            "State": "COMPLETED",
            "StateChangeReason": {}
        },
        "Config": {
            "Args": [
                "s3://us-west-2.elasticmapreduce/libs/hive/hive-script",
                "--base-path",
                "s3://us-west-2.elasticmapreduce/libs/hive/",
                "--install-hive",
                "--hive-versions",
                "0.13.1"
            ],
            "Jar": "s3://us-west-2.elasticmapreduce/libs/script-runner/script-
runner.jar",
```

```
            "Properties": {}
        },
        "Id": "s-3LZC0QUT43AM",
        "ActionOnFailure": "TERMINATE_CLUSTER",
        "Name": "Setup hive"
    }
}
```

- Einzelheiten zur API finden Sie DescribeStepin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current
 state of
    the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response["Step"]
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- Einzelheiten zur API finden Sie DescribeStepin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_emr->describestep(
      iv_clusterid = iv_cluster_id
      iv_stepid = iv_step_id
    ).
    DATA(lo_step) = oo_result->get_step( ).
    DATA(lv_step_name) = lo_step->get_name( ).
    MESSAGE |Retrieved step information for { lv_step_name }| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeStepin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListSteps`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListSteps` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl werden alle Schritte in einem Cluster mit der Cluster-ID `j-3SD91U2E1L2QX` aufgelistet:

```
aws emr list-steps --cluster-id j-3SD91U2E1L2QX
```

- Einzelheiten zur API finden Sie ListStepsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps
 are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """
    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response["Steps"]
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- Einzelheiten zur API finden Sie ListStepsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_emr->liststeps(
      iv_clusterid = iv_cluster_id
    ).
    DATA(lt_steps) = oo_result->get_steps( ).
    DATA(lv_step_count) = lines( lt_steps ).
    MESSAGE |Retrieved { lv_step_count } steps for cluster| TYPE 'I'.
  CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
    lv_error = lo_invalid_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListStepsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RunJobFlow`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `RunJobFlow` verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def run_job_flow(
    name,
    log_uri,
    keep_alive,
    applications,
    job_flow_role,
    service_role,
    security_groups,
    steps,
    emr_client,
):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
components.html.

    :param name: The name of the cluster.
    :param log_uri: The URI where logs are stored. This can be an Amazon S3
 bucket URL,
                    such as 's3://my-log-bucket'.
    :param keep_alive: When True, the cluster is put into a Waiting state after
 all
                       steps are run. When False, the cluster terminates itself
 when
                       the step queue is empty.
    :param applications: The applications to install on each instance in the
 cluster,
```

```
                        such as Hive or Spark.
   :param job_flow_role: The IAM role assumed by the cluster.
   :param service_role: The IAM role assumed by the service.
   :param security_groups: The security groups to assign to the cluster
instances.
                           Amazon EMR adds all needed rules to these groups, so
                           they can be empty if you require only the default
rules.
   :param steps: The job flow steps to add to the cluster. These are run in
order
                when the cluster is ready.
   :param emr_client: The Boto3 EMR client object.
   :return: The ID of the newly created cluster.
   """
   try:
       response = emr_client.run_job_flow(
           Name=name,
           LogUri=log_uri,
           ReleaseLabel="emr-5.30.1",
           Instances={
               "MasterInstanceType": "m5.xlarge",
               "SlaveInstanceType": "m5.xlarge",
               "InstanceCount": 3,
               "KeepJobFlowAliveWhenNoSteps": keep_alive,
               "EmrManagedMasterSecurityGroup": security_groups["manager"].id,
               "EmrManagedSlaveSecurityGroup": security_groups["worker"].id,
           },
           Steps=[
               {
                   "Name": step["name"],
                   "ActionOnFailure": "CONTINUE",
                   "HadoopJarStep": {
                       "Jar": "command-runner.jar",
                       "Args": [
                           "spark-submit",
                           "--deploy-mode",
                           "cluster",
                           step["script_uri"],
                           *step["script_args"],
                       ],
                   },
               }
               for step in steps
           ],
```

```
            Applications=[{"Name": app} for app in applications],
            JobFlowRole=job_flow_role.name,
            ServiceRole=service_role.name,
            EbsRootVolumeSize=10,
            VisibleToAllUsers=True,
        )
        cluster_id = response["JobFlowId"]
        logger.info("Created cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't create cluster.")
        raise
    else:
        return cluster_id
```

- Einzelheiten zur API finden Sie RunJobFlowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create instances configuration
        DATA(lo_instances) = NEW /aws1/cl_emrjobflowinstsconfig(
          iv_masterinstancetype = 'm5.xlarge'
          iv_slaveinstancetype = 'm5.xlarge'
          iv_instancecount = 3
          iv_keepjobflowalivewhennos00 = iv_keep_alive
          iv_emrmanagedmastersecgroup = iv_primary_sec_grp
          iv_emrmanagedslavesecgroup = iv_secondary_sec_grp
        ).

        DATA(lo_result) = lo_emr->runjobflow(
          iv_name = iv_name
```

```
          iv_loguri = iv_log_uri
          iv_releaselabel = 'emr-5.30.1'
          io_instances = lo_instances
          it_steps = it_steps
          it_applications = it_applications
          iv_jobflowrole = iv_job_flow_role
          iv_servicerole = iv_service_role
          iv_ebsrootvolumesize = 10
          iv_visibletoallusers = abap_true
        ).

        ov_cluster_id = lo_result->get_jobflowid( ).
        MESSAGE 'EMR cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrclientexc INTO DATA(lo_client_error).
        lv_error = lo_client_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RunJobFlowin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`TerminateJobFlows`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `TerminateJobFlows` verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def terminate_cluster(cluster_id, emr_client):
    """
```

```
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is
lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- Einzelheiten zur API finden Sie TerminateJobFlowsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA lt_cluster_ids TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
        APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_cluster_id ) TO
lt_cluster_ids.

        lo_emr->terminatejobflows(
          it_jobflowids = lt_cluster_ids
        ).
        MESSAGE 'EMR cluster terminated successfully.' TYPE 'I'.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
```

```
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie TerminateJobFlowsin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon EMR AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon EMR mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon EMR oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie einen kurzlebigen Amazon EMR-Cluster und führen Sie einen Schritt mit einem AWS SDK
- Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen
- Führen Sie ein Shell-Skript aus, um Bibliotheken auf Amazon EMR-Instances mit einem zu installieren AWS SDK

Erstellen Sie einen kurzlebigen Amazon EMR-Cluster und führen Sie einen Schritt mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie einen kurzlebigen Amazon EMR-Cluster erstellen, der einen Schritt ausführt und nach Abschluss des Schritts automatisch beendet wird.

Python

SDK für Python (Boto3)

Erstellen Sie mithilfe von Apache Spark einen kurzlebigen Amazon EMR-Cluster, der den Wert von Pi schätzt, um eine große Anzahl von Berechnungen zu parallelisieren. Der Job schreibt

Ausgaben in Amazon EMR-Protokolle und in einen Amazon Simple Storage Service (Amazon S3)-Bucket. Der Cluster beendet sich selbst, nachdem der Job abgeschlossen wurde.

- Erstellen Sie einen Amazon-S3-Bucket und uploaden Sie ein Job-Skript.
- Erstellen Sie AWS Identity and Access Management (IAM-) Rollen.
- Erstellen Sie Amazon Elastic Compute Cloud (Amazon EC2)-Sicherheitsgruppen.
- Erstellen Sie einen kurzlebigen Cluster und führen Sie einen einzelnen Arbeitsschritt aus.

Dieses Beispiel lässt sich am besten auf GitHub ansehen. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon EMR

Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial
```

```
set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
```

```
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
```

```
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse
```

```
from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
```

```
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
```

```
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi
```

```
# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi
```

```
echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."
```

```
# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
```

```
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - SSH
  - TerminateClusters
  - Wait

Führen Sie ein Shell-Skript aus, um Bibliotheken auf Amazon EMR-Instances mit einem zu installieren AWS SDK

Das folgende Codebeispiel zeigt, wie Sie AWS Systems Manager ein Shell-Skript auf Amazon EMR-Instances ausführen, das zusätzliche Bibliotheken installiert. Auf diese Weise können Sie die Instance-Verwaltung automatisieren, anstatt Befehle manuell über eine SSH-Verbindung auszuführen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import argparse
import time
import boto3


def install_libraries_on_core_nodes(cluster_id, script_path, emr_client,
 ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object
 URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=["CORE"]
    )["Instances"]
    core_instance_ids = [node["Ec2InstanceId"] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh",
    ]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
            InstanceIds=core_instance_ids,
```

```
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": [command]},
            TimeoutSeconds=3600,
        )["Command"]["CommandId"]
        while True:
            # Verify the previous step succeeded before running the next step.
            cmd_result = ssm_client.list_commands(CommandId=command_id)
["Commands"][0]
            if cmd_result["StatusDetails"] == "Success":
                print(f"Command succeeded.")
                break
            elif cmd_result["StatusDetails"] in ["Pending", "InProgress"]:
                print(f"Command status is {cmd_result['StatusDetails']},
 waiting...")
                time.sleep(10)
            else:
                print(f"Command status is {cmd_result['StatusDetails']},
 quitting.")
                raise RuntimeError(
                    f"Command {command} failed to run. "
                    f"Details: {cmd_result['StatusDetails']}"
                )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("cluster_id", help="The ID of the cluster.")
    parser.add_argument("script_path", help="The path to the script in Amazon
 S3.")
    args = parser.parse_args()

    emr_client = boto3.client("emr")
    ssm_client = boto3.client("ssm")

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ListInstancesin AWS SDK for Python (Boto3) API Reference.

Codebeispiele für AWS Entity Resolution verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Entity Resolution mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS Entity Resolution Benutzerhandbuch — Weitere Informationen zu AWS Entity Resolution.
- AWS Entity Resolution API-Referenz — Details zu allen verfügbaren AWS Entity Resolution Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Entity Resolution verwenden AWS SDKs
  - Hallo AWS Entity Resolution
  - Lernen Sie die Grundlagen von AWS Entity Resolution mit einem AWS SDK
  - Aktionen für AWS Entity Resolution verwenden AWS SDKs
    - Verwenden Sie CreateMatchingWorkflowmit einem AWS SDK
    - Verwenden Sie CreateSchemaMappingmit einem AWS SDK
    - Verwenden Sie DeleteMatchingWorkflowmit einem AWS SDK
    - Verwenden Sie DeleteSchemaMappingmit einem AWS SDK
    - Verwenden Sie GetMatchingJobmit einem AWS SDK
    - Verwenden Sie GetSchemaMappingmit einem AWS SDK
    - Verwenden Sie ListSchemaMappingsmit einem AWS SDK
    - Verwenden Sie StartMatchingJobmit einem AWS SDK
    - Verwenden Sie TagResourcemit einem AWS SDK

Grundlegende Beispiele für AWS Entity Resolution verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Entity Resolution mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS Entity Resolution
- Lernen Sie die Grundlagen von AWS Entity Resolution mit einem AWS SDK
- Aktionen für AWS Entity Resolution verwenden AWS SDKs
  - Verwenden Sie CreateMatchingWorkflowmit einem AWS SDK
  - Verwenden Sie CreateSchemaMappingmit einem AWS SDK
  - Verwenden Sie DeleteMatchingWorkflowmit einem AWS SDK
  - Verwenden Sie DeleteSchemaMappingmit einem AWS SDK
  - Verwenden Sie GetMatchingJobmit einem AWS SDK
  - Verwenden Sie GetSchemaMappingmit einem AWS SDK
  - Verwenden Sie ListSchemaMappingsmit einem AWS SDK
  - Verwenden Sie StartMatchingJobmit einem AWS SDK
  - Verwenden Sie TagResourcemit einem AWS SDK

Hallo AWS Entity Resolution

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS Entity Resolution beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloEntityResoultion {

    private static final Logger logger =
 LoggerFactory.getLogger(HelloEntityResoultion.class);

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;
    public static void main(String[] args) {
        listMatchingWorkflows();
    }

    public static EntityResolutionAsyncClient getResolutionAsyncClient() {
        if (entityResolutionAsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
 call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
```

```
                .retryStrategy(RetryMode.STANDARD)
                .build();

            entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return entityResolutionAsyncClient;
    }

    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved
 workflows.
     * It uses an asynchronous approach with a paginator and waits for the
 operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Einzelheiten zur API finden Sie ListMatchingWorkflowsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  EntityResolutionClient,
  ListMatchingWorkflowsCommand,
} from "@aws-sdk/client-entityresolution";

export const main = async () => {
  const region = "eu-west-1";
  const erClient = new EntityResolutionClient({ region: region });
  try {
    const command = new ListMatchingWorkflowsCommand({});
    const response = await erClient.send(command);
    const workflowSummaries = response.workflowSummaries;
    for (const workflowSummary of workflowSummaries) {
      console.log(`Attribute name: ${workflowSummaries[0].workflowName} `);
    }
    if (workflowSummaries.length === 0) {
      console.log("No matching workflows found.");
    }
  } catch (error) {
    console.error(
      `An error occurred in listing the workflow summaries: ${error.message} \n
 Exiting program.`,
    );
    return;
  }
};
```

- Einzelheiten zur API finden Sie ListMatchingWorkflowsin der AWS SDK für JavaScript API-Referenz.

Lernen Sie die Grundlagen von AWS Entity Resolution mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine Schemazuordnung.
- Erstellen Sie einen AWS Entity Resolution Workflow.
- Starten Sie den passenden Auftrag für den Workflow.
- Rufen Sie Details für den passenden Auftrag ab.
- Rufen Sie die Schemazuordnung ab.
- Listen Sie alle Schemazuordnungen auf.
- Markieren Sie die Schemazuordnungsressource.
- Löschen Sie die AWS Entity Resolution Assets.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Entity Resolution Funktionen demonstriert werden.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
```

```
   private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
"EntityResolutionRoleArn";
   private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
   private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
   private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
   private static String glueBucketName;
   private static String workflowName = "workflow-" + UUID.randomUUID();

   private static String jsonSchemaMappingName = "jsonschema-" +
UUID.randomUUID();
   private static String jsonSchemaMappingArn = null;
   private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
   private static String roleARN;
   private static String csvGlueTableArn;
   private static String jsonGlueTableArn;
   private static Scanner scanner = new Scanner(System.in);

   private static EntityResActions actions = new EntityResActions();

   public static void main(String[] args) throws InterruptedException {

       logger.info("Welcome to the AWS Entity Resolution Scenario.");
       logger.info("""
           AWS Entity Resolution is a fully-managed machine learning service
provided by
           Amazon Web Services (AWS) that helps organizations extract, link,
and
           organize information from multiple data sources. It leverages
natural
           language processing and deep learning models to identify and resolve
           entities, such as people, places, organizations, and products,
           across structured and unstructured data.

           With Entity Resolution, customers can build robust data integration
           pipelines to combine and reconcile data from multiple systems,
databases,
           and documents. The service can handle ambiguous, incomplete, or
conflicting
           information, and provide a unified view of entities and their
relationships.
           This can be particularly valuable in applications such as customer
360,
           fraud detection, supply chain management, and knowledge management,
where
```

```
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java
2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application,
the next step uploads
            a CloudFormation template whose resulting stack creates the following
resources:
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
```

```
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
            """;
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
 ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
 found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
 bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
            consolidated, accurate view for improved data quality and decision-
making.

            In this example, the schema mapping lines up with the fields in the
 JSON and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " +
 jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
 ce.getMessage(), ce);
            }
```

```
            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " +
csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
```

```
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
                logger.error("Workflow conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
```

```
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other
API methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
```

```
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be
tagged. For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("8. View the results of the AWS Entity Resolution
Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running
state.
            In order to view the results, you need to wait for the workflow that
we started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can
perform
            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName,
jobId).join();
                logger.info("Number of input records: {}",
metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
                logger.info("The following represents the output data generated
by the Entity Resolution workflow based on the JSON and CSV input data. The
output data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match
even though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.
```

```
                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the
workflow? (y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.

            You can delete the workflow manually in the AWS Management Console at
a later time.

            If you already waited for the workflow to complete in the previous
step,
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId,
workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
```

```
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}",
ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this
scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
                return;
            }

        } else {
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }
```

```
    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;

            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
"SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
```

```
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: "
 + e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

Eine Wrapper-Klasse für AWS Entity Resolution SDK-Methoden.

```
public class EntityResActions {

    private static final String PREFIX = "eroutput/";
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResActions.class);

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;

    private static S3AsyncClient s3AsyncClient;

    public static EntityResolutionAsyncClient getResolutionAsyncClient() {
        if (entityResolutionAsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
```

```
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return entityResolutionAsyncClient;
    }

    public static S3AsyncClient getS3AsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
```

```
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping
is deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema
mapping: " + schemaName, exception);
                }
            });
    }

    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping
to the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });
```

```
        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has
been deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
```

```
    * Creates a schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to create
    * @return a {@link CompletableFuture} that represents the asynchronous
creation of the schema mapping
    */
   public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
       List<SchemaInputAttribute> schemaAttributes = null;
       if (schemaName.startsWith("json")) {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute
           );
       } else {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).
           );
       }

       CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .mappedInputFields(schemaAttributes)
           .build();

       return getResolutionAsyncClient().createSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("[{}] schema mapping Created Successfully!",
schemaName);
               } else {
                   if (exception == null) {
```

```
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema
mapping already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema
mapping: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse>
getSchemaMappingAsync(String schemaName) {
        GetSchemaMappingRequest mappingRequest =
GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema
mapping was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job
information is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could
not be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: "
+ exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the
matching job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already
running. Resolve conflicts before starting a new job.", cause);
```

```
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }
```

```
                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: "
+ exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName           the name of the workflow to be created
     * @param outputBucket           the S3 bucket path where the workflow
output will be stored
     * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to
be used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of
the created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();
```

```
        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
```

```
                        throw new CompletionException("Invalid request: Please
check input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The resource to tag was
not found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: "
+ exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id
was not found.", cause);
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
     * Uploads data to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the data to
     * @param jsonData   the JSON data to be uploaded
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous
operation of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

    public void uploadInputData(String bucketName, String jsonData, String
csvData) {
```

```
        // Upload JSON data.
        String jsonKey = "jsonData/data.json";
        PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(jsonKey)
            .contentType("application/json")
            .build();

        CompletableFuture<PutObjectResponse> jsonUploadResponse
= getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

        // Upload CSV data.
        String csvKey = "csvData/data.csv";
        PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(csvKey)
            .contentType("text/csv")
            .build();
        CompletableFuture<PutObjectResponse> csvUploadResponse
= getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

        CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    // Wrap an AWS exception.
                    throw new CompletionException("Failed to upload files", ex);
                }
            }).join();

    }

    /**
     * Finds the latest file in the S3 bucket that starts with "run-" in any
depth of subfolders
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
```

```
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with
"run-" in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of
subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }

            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
```

```
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String
bucketName, String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: "
+ exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from
S3.");
                }
            });
    }

    /**
     * Parses CSV content from a String into a list of records.
     */
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }
```

```
    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ?
row[columnIndex].length() : 0)
                .max(Integer::compareTo)
                .orElse(0));
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for
better readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3
===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
    }

    private static void printRow(String[] row, int[] columnWidths, boolean
isHeader) {
        String border = IntStream.range(0, columnWidths.length)
```

```
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows
  - ListSchemaMappings
  - StartMatchingJob
  - TagResource

JavaScript

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS Entity Resolution Funktionen demonstriert werden.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
} from "@aws-sdk/client-cloudformation";
import {
  EntityResolutionClient,
  CreateSchemaMappingCommand,
  CreateMatchingWorkflowCommand,
  GetMatchingJobCommand,
  StartMatchingJobCommand,
  GetSchemaMappingCommand,
  ListSchemaMappingsCommand,
  TagResourceCommand,
  DeleteMatchingWorkflowCommand,
  DeleteSchemaMappingCommand,
  ConflictException,
  ValidationException,
} from "@aws-sdk/client-entityresolution";
```

```
import {
  DeleteObjectsCommand,
  DeleteBucketCommand,
  PutObjectCommand,
  S3Client,
  ListObjectsCommand,
} from "@aws-sdk/client-s3";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { readFile } from "node:fs/promises";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = `${data.inputs.entityResolutionStack}`;

/*The inputs for this example can be edited in the ../input.json.*/
import data from "../inputs.json" with { type: "json" };
const skipWhenErrors = (state) => state.errors.length > 0;
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  verbose: "false",
  skipWhen: skipWhenErrors,
});

const region = "eu-west-1";

const entityResolutionClient = new EntityResolutionClient({ region: region });
const cloudFormationClient = new CloudFormationClient({ region: region });
const s3Client = new S3Client({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "AWS Entity Resolution is a fully-managed machine learning service provided by
 " +
    "Amazon Web Services (AWS) that helps organizations extract, link, and " +
```

```
    "organize information from multiple data sources. It leverages natural " +
    "language processing and deep learning models to identify and resolve " +
    "entities, such as people, places, organizations, and products, " +
    "across structured and unstructured data.\n" +
    "\n" +
    "With Entity Resolution, customers can build robust data integration " +
    "pipelines to combine and reconcile data from multiple systems, databases, "
 +
    "and documents. The service can handle ambiguous, incomplete, or conflicting
 " +
    "information, and provide a unified view of entities and their relationships.
 " +
    "This can be particularly valuable in applications such as customer 360, " +
    "fraud detection, supply chain management, and knowledge management, where "
 +
    "accurate entity identification is crucial.\n" +
    "\n" +
    "The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x " +
    "provides a set of methods to programmatically interact with the AWS Entity "
 +
    "Resolution service. This allows developers to automate the entity
 extraction, " +
    "linking, and deduplication process as part of their data processing
 workflows. " +
    "With Entity Resolution, organizations can unlock the value of their data, "
 +
    "improve decision-making, and enhance customer experiences by having a
 reliable, " +
    "comprehensive view of their key entities.",

  { header: true },
);
const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "To prepare the AWS resources needed for this scenario application, the next
 step uploads " +
    "a CloudFormation template whose resulting stack creates the following
 resources:\n" +
    "- An AWS Glue Data Catalog table \n" +
    "- An AWS IAM role \n" +
    "- An AWS S3 bucket \n" +
    "- An AWS Entity Resolution Schema \n" +
    "It can take a couple minutes for the Stack to finish creating the
 resources.",
```

```
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/entity-resolution-basics/entity-
resolution-basics-template.yml`,
        "utf8",
      );
      await cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );

      state.entityResolutionRole = stack.Stacks[0].Outputs[1];
      state.jsonGlueTable = stack.Stacks[0].Outputs[2];
      state.CSVGlueTable = stack.Stacks[0].Outputs[3];
      state.glueDataBucket = stack.Stacks[0].Outputs[0];
      state.stackName = stack.StackName;
      console.log(state.glueDataBucket);
      console.log(
        `The  ARN of the EntityResolution Role is
 ${state.entityResolutionRole.OutputValue}`,
      );
      console.log(
        `The ARN of the Json Glue Table is ${state.jsonGlueTable.OutputValue}`,
```

```
      );
      console.log(
        `The ARN of the CSV Glue Table is ${state.CSVGlueTable.OutputValue}`,
      );
      console.log(
        `The name of the Glue Data Bucket is
 ${state.glueDataBucket.OutputValue}\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the following JSON in ../data.json to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );
      const bucketName = state.glueDataBucket.OutputValue;

      const putObjectParams = {
        Bucket: bucketName,
        Key: "jsonData/data.json",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.json`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.json file data uploaded to the
 ${state.glueDataBucket.OutputValue} S3 bucket.\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the CSV data in ../data.csv to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );

      const bucketName = state.glueDataBucket.OutputValue;
```

```
      const putObjectParams = {
        Bucket: bucketName,
        Key: "csvData/data.csv",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.csv`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.csv file data uploaded to the
 ${state.glueDataBucket.OutputValue} S3 bucket.`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateSchemaMapping = new ScenarioOutput(
  "displayCreateSchemaMapping",
  "1. Create Schema Mapping" +
    "Entity Resolution schema mapping aligns and integrates data from " +
    "multiple sources by identifying and matching corresponding entities " +
    "like customers or products. It unifies schemas, resolves conflicts, " +
    "and uses machine learning to link related entities, enabling a " +
    "consolidated, accurate view for improved data quality and decision-making."
 +
    "\n" +
    "In this example, the schema mapping lines up with the fields in the JSON and
 CSV objects. That is, " +
    " it contains these fields: id, name, and email. ",
);

const sdkCreateSchemaMapping = new ScenarioAction(
  "sdkCreateSchemaMapping",
  async (/** @type {State} */ state) => {
    const createSchemaMappingParamsJson = {
      schemaName: `${data.inputs.schemaNameJson}`,
      mappedInputFields: [
        {
          fieldName: "id",
```

```
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
      ],
    };
    const createSchemaMappingParamsCSV = {
      schemaName: `${data.inputs.schemaNameCSV}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
        {
          fieldName: "phone",
          type: "PROVIDER_ID",
          subType: "STRING",
        },
      ],
    };
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsJson,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameJson = response.schemaName;
      state.schemaArn = response.schemaArn;
      state.idOutputAttribute = response.mappedInputFields[0].fieldName;
      state.nameOutputAttribute = response.mappedInputFields[1].fieldName;
      state.emailOutputAttribute = response.mappedInputFields[2].fieldName;
```

```
      console.log("The JSON schema mapping name is ", state.schemaNameJson);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema mapping already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsCSV,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameCSV = response.schemaName;
      state.phoneOutputAttribute = response.mappedInputFields[3].fieldName;
      console.log("The CSV schema mapping name is ", state.schemaNameCSV);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateMatchingWorkflow = new ScenarioOutput(
  "displayCreateMatchingWorkflow",
  "2. Create an AWS Entity Resolution Workflow. " +
    "An Entity Resolution matching workflow identifies and links records " +
    "across datasets that represent the same real-world entity, such as " +
    "customers or products. Using techniques like schema mapping, " +
    "data profiling, and machine learning algorithms, " +
    "it evaluates attributes like names or emails to detect duplicates " +
    "or relationships, even with variations or inconsistencies. " +
    "The workflow outputs consolidated, de-duplicated data." +
    "\n" +
    "We will use the machine learning-based matching technique.",
);
```

```
const sdkCreateMatchingWorkflow = new ScenarioAction(
  "sdkCreateMatchingWorkflow",
  async (/** @type {State} */ state) => {
    const createMatchingWorkflowParams = {
      roleArn: `${state.entityResolutionRole.OutputValue}`,
      workflowName: `${data.inputs.workflowName}`,
      description: "Created by using the AWS SDK for JavaScript (v3).",
      inputSourceConfig: [
        {
          inputSourceARN: `${state.jsonGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameJson}`,
          applyNormalization: false,
        },
        {
          inputSourceARN: `${state.CSVGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameCSV}`,
          applyNormalization: false,
        },
      ],
      outputSourceConfig: [
        {
          outputS3Path: `s3://${state.glueDataBucket.OutputValue}/eroutput`,
          output: [
            {
              name: state.idOutputAttribute,
            },
            {
              name: state.nameOutputAttribute,
            },
            {
              name: state.emailOutputAttribute,
            },
            {
              name: state.phoneOutputAttribute,
            },
          ],
          applyNormalization: false,
        },
      ],
      resolutionTechniques: { resolutionType: "ML_MATCHING" },
    };
    try {
      const command = new CreateMatchingWorkflowCommand(
        createMatchingWorkflowParams,
```

```
      );
      const response = await entityResolutionClient.send(command);
      state.workflowArn = response.workflowArn;
      console.log(
        `Workflow created successfully.\n The workflow ARN is:
 ${response.workflowArn}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
      if (caught instanceof ValidationException) {
        console.error(
          `There was a validation exception: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);
const displayMatchingJobOfWorkflow = new ScenarioOutput(
  "displayMatchingJobOfWorkflow",
  "3. Start the matching job of the workflow",
);

const sdkMatchingJobOfWorkflow = new ScenarioAction(
  "sdk",
  async (/** @type {State} */ state) => {
    const matchingJobOfWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    try {
      const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
      const response = await entityResolutionClient.send(command);
      state.jobID = response.jobId;
      console.log(`Job ID: ${state.jobID} \n
The matching job was successfully started.`);
    } catch (caught) {
      if (caught instanceof ConflictException) {
```

```
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);

const displayGetDetailsforJob = new ScenarioOutput(
  "displayGetDetailsforJob",
  `4. While the matching job is running, let's look at other API methods. First,
 let's get details for the job `,
);

const sdkGetDetailsforJob = new ScenarioAction(
  "sdkGetDetailsforJob",
  async (/** @type {State} */ state) => {
    const getDetailsforJobParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${state.jobID}`,
    };
    try {
      const command = new GetMatchingJobCommand(getDetailsforJobParams);
      const response = await entityResolutionClient.send(command);
      state.Status = response.status;
      state.response = response;
      console.log(`Job status: ${state.Status} `);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetSchemaMappingJson = new ScenarioOutput(
  "displayGetSchemaMappingJson",
  "5. Get the schema mapping for the JSON data.",
);

const sdkGetSchemaMappingJson = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
```

```
    const getSchemaMappingJsonParams = {
      schemaName: `${data.inputs.schemaNameJson}`,
    };
    try {
      const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
      const response = await entityResolutionClient.send(command);
      console.log("Schema·mapping·ARN·is:·", response.schemaArn);
      const resultMappings = response.mappedInputFields;
      const noOfResultMappings = resultMappings.length;
      for (let i = 0; i < noOfResultMappings; i++) {
        console.log(
          `Attribute name: ${resultMappings[i].fieldName} `,
          `Attribute type: ${resultMappings[i].type}`,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayListSchemaMappings = new ScenarioOutput(
  "displayListSchemaMappings",
  "6. List Schema Mappings.",
);

const sdkListSchemaMappings = new ScenarioAction(
  "sdkListSchemaMappings",
  async (/** @type {State} */ state) => {
    try {
      const command = new ListSchemaMappingsCommand({});
      const response = await entityResolutionClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
```

```
);

const displayTagTheJsonSchema = new ScenarioOutput(
  "display",
  "7. Tag the resource. \n" +
    "Tags can help you organize and categorize your Entity Resolution resources.
 " +
    "You can also use them to scope user permissions by granting a user
 permission " +
    "to access or change only resources with certain tag values. " +
    "In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged. For
 this example, " +
    "the SchemaMapping is tagged.",
);

const sdkTagTheJsonSchema = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const tagResourceCommandParams = {
      resourceArn: state.schemaArn,
      tags: {
        tag1: "tag1Value",
        tag2: "tag2Value",
      },
    };
    try {
      const command = new TagResourceCommand(tagResourceCommandParams);
      const response = await entityResolutionClient.send(command);
      console.log("Successfully tagged the resource.");
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetJobInfo = new ScenarioOutput(
  "displayGetJobInfo",
  "8. View the results of the AWS Entity Resolution Workflow.\n " +
    "Please perform this task manually in the AWS Management Console. ",
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
```

```
  "9. Delete the resources \n" +
    "You cannot delete a workflow that is in a running state. So this will take
 ~30 minutes.\n" +
    "If you don't want to delete the resources, simply exit this application.",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    console.log(
      "You selected to delete the resources. This will take about 30 minutes.",
    );
    await wait(1800);
    const bucketName = state.glueDataBucket.OutputValue;
    try {
      const emptyBucket = async ({ bucketName }) => {
        const listObjectsCommand = new ListObjectsCommand({
          Bucket: bucketName,
        });
        const { Contents } = await s3Client.send(listObjectsCommand);
        const keys = Contents.map((c) => c.Key);

        const deleteObjectsCommand = new DeleteObjectsCommand({
          Bucket: bucketName,
          Delete: { Objects: keys.map((key) => ({ Key: key })) },
        });
        await s3Client.send(deleteObjectsCommand);
        console.log(`Bucket ${bucketName} emptied successfully.\n`);
      };
      await emptyBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteBucket = async ({ bucketName }) => {
        const command = new DeleteBucketCommand({ Bucket: bucketName });
        await s3Client.send(command);
        console.log(`Bucket ${bucketName} deleted successfully.\n`);
      };
      await deleteBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
```

```
      console.log(
        "Now we will delete the CloudFormation stack, which deletes the resources
that were created at the beginning of the scenario.",
      );
      const deleteStackParams = { StackName: `${state.stackName}` };
      const command = new DeleteStackCommand(deleteStackParams);
      const response = await cloudFormationClient.send(command);
      console.log("CloudFormation stack deleted successfully.");
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteWorkflowParams = {
        workflowName: `${data.inputs.workflowName}`,
      };
      const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
      const response = await entityResolutionClient.send(command);
      console.log("Workflow deleted successfully!");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `Job associated with workflow ${data.inputs.workflowName} is still
running, so can't be deleted.
          Neither can schemas ${data.inputs.schemaNameJson} and
${data.inputs.schemaNameCSV} associated with it. Please confirm this workflow is
finished in the AWS Management Console, then delete it manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteJSONschemaMapping = {
        schemaName: `${data.inputs.schemaNameJson}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteJSONschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully. ");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameJson} can't be deleted because it
is associated with workflow
```

```
          ${data.inputs.workflowName}, which is still running. Please confirm
 this workflow is finished in the AWS Management Console, then delete it
 manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteCSVschemaMapping = {
        schemaName: `${data.inputs.schemaNameCSV}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteCSVschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully.");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameCSV} can't be deleted because it is
 associated with workflow ${data.inputs.workflowName}, which is still running.
 Please confirm this workflow is finished in the AWS Management Console, then
 delete it manually.`,
        );
        throw caught;
      }
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.confirmDeleteResources === "",
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you "
 +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming
 languages, have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Entity Resolution Basics Scenario", [
```

```
  greet,
  pressEnter,
  displayBuildCloudFormationStack,
  sdkBuildCloudFormationStack,
  pressEnter,
  displayCreateSchemaMapping,
  sdkCreateSchemaMapping,
  pressEnter,
  displayCreateMatchingWorkflow,
  sdkCreateMatchingWorkflow,
  pressEnter,
  displayMatchingJobOfWorkflow,
  sdkMatchingJobOfWorkflow,
  pressEnter,
  displayGetDetailsforJob,
  sdkGetDetailsforJob,
  pressEnter,
  displayGetSchemaMappingJson,
  sdkGetSchemaMappingJson,
  pressEnter,
  displayListSchemaMappings,
  sdkListSchemaMappings,
  pressEnter,
  displayTagTheJsonSchema,
  sdkTagTheJsonSchema,
  pressEnter,
  displayGetJobInfo,
  pressEnter,
  displayDeleteResources,
  pressEnter,
  sdkDeleteResources,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
```

```
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows
  - ListSchemaMappings
  - StartMatchingJob
  - TagResource

Aktionen für AWS Entity Resolution verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Entity Resolution Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Entity Resolution -API-Referenz.

Beispiele

- Verwenden Sie CreateMatchingWorkflowmit einem AWS SDK
- Verwenden Sie CreateSchemaMappingmit einem AWS SDK
- Verwenden Sie DeleteMatchingWorkflowmit einem AWS SDK

- Verwenden Sie DeleteSchemaMappingmit einem AWS SDK
- Verwenden Sie GetMatchingJobmit einem AWS SDK
- Verwenden Sie GetSchemaMappingmit einem AWS SDK
- Verwenden Sie ListSchemaMappingsmit einem AWS SDK
- Verwenden Sie StartMatchingJobmit einem AWS SDK
- Verwenden Sie TagResourcemit einem AWS SDK

Verwenden Sie **`CreateMatchingWorkflow`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateMatchingWorkflow` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                 the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName            the name of the workflow to be created
     * @param outputBucket            the S3 bucket path where the workflow
output will be stored
     * @param jsonGlueTableArn        the ARN of the Glue Data Catalog table to
be used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
```

```
     * @return a CompletableFuture that, when completed, will return the ARN of
the created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
```

```
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please
check input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- Einzelheiten zur API finden Sie CreateMatchingWorkflowin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createMatchingWorkflowParams = {
    roleArn: `${data.inputs.roleArn}`,
    workflowName: `${data.inputs.workflowName}`,
    description: "Created by using the AWS SDK for JavaScript (v3).",
    inputSourceConfig: [
      {
        inputSourceARN: `${data.inputs.JSONinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameJson}`,
        applyNormalization: false,
      },
      {
        inputSourceARN: `${data.inputs.CSVinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameCSV}`,
        applyNormalization: false,
      },
    ],
    outputSourceConfig: [
```

```
      {
        outputS3Path: `s3://${data.inputs.myBucketName}/eroutput`,
        output: [
          {
            name: "id",
          },
          {
            name: "name",
          },
          {
            name: "email",
          },
          {
            name: "phone",
          },
        ],
        applyNormalization: false,
      },
    ],
    resolutionTechniques: { resolutionType: "ML_MATCHING" },
  };
  try {
    const command = new CreateMatchingWorkflowCommand(
      createMatchingWorkflowParams,
    );
    const response = await erClient.send(command);

    console.log(
      `Workflow created successfully.\n The workflow ARN is:
 ${response.workflowArn}`,
    );
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie CreateMatchingWorkflowin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateSchemaMapping`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSchemaMapping` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to create
    * @return a {@link CompletableFuture} that represents the asynchronous
creation of the schema mapping
    */
   public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
       List<SchemaInputAttribute> schemaAttributes = null;
       if (schemaName.startsWith("json")) {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute
           );
       } else {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.U
```

```
SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeTy

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttribute

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema
mapping already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema
mapping: " + exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateSchemaMappingin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createSchemaMappingParamsJson = {
    schemaName: `${data.inputs.schemaNameJson}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
    ],
  };
```

```
  const createSchemaMappingParamsCSV = {
    schemaName: `${data.inputs.schemaNameCSV}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
      {
        fieldName: "phone",
        type: "PROVIDER_ID",
        subType: "STRING",
      },
    ],
  };
  try {
    const command = new CreateSchemaMappingCommand(
      createSchemaMappingParamsJson,
    );
    const response = await erClient.send(command);
    console.log("The JSON schema mapping name is ", response.schemaName);
  } catch (error) {
    console.log("error ", error.message);
  }
};
```

- Einzelheiten zur API finden Sie CreateSchemaMappingin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DeleteMatchingWorkflow`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteMatchingWorkflow` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously deletes a workflow with the specified name.
    *
    * @param workflowName the name of the workflow to be deleted
    * @return a {@link CompletableFuture} that completes when the workflow has
been deleted
    * @throws RuntimeException if the deletion of the workflow fails
    */
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
 not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
 exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie DeleteMatchingWorkflowin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
```

```
  try {
    const deleteWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
    const response = await erClient.send(command);
    console.log("Workflow deleted successfully!", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Einzelheiten zur API finden Sie DeleteMatchingWorkflowin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DeleteSchemaMapping`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteSchemaMapping` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
```

```
    * @return a {@link CompletableFuture} that completes when the schema mapping
is deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete schema
mapping: " + schemaName, exception);
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteSchemaMappingin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const deleteSchemaMapping = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new DeleteSchemaMappingCommand(deleteSchemaMapping);
    const response = await erClient.send(command);
    console.log("Schema mapping deleted successfully. ", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Einzelheiten zur API finden Sie DeleteSchemaMappingin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetMatchingJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetMatchingJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
    *
    * @param jobId        the ID of the job to retrieve
    * @param workflowName the name of the workflow associated with the job
    * @return a {@link CompletableFuture} that completes when the job
information is available or an exception occurs
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
```

```
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
 while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could
 not be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
 message.
                    throw new CompletionException("Error fetching matching job: "
 + exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie GetMatchingJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetMatchingJobCommand,
  EntityResolutionClient,
```

```
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const getJobInfoParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new GetMatchingJobCommand(getJobInfoParams);
      const response = await erClient.send(command);
      console.log(`Job status: ${response.status}`);
    } catch (error) {
      console.log("error ", error.message);
    }
  }
};
```

- Einzelheiten zur API finden Sie GetMatchingJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetSchemaMapping`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetSchemaMapping` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse>
getSchemaMappingAsync(String schemaName) {
       GetSchemaMappingRequest mappingRequest =
GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   response.mappedInputFields().forEach(attribute ->
                       logger.info("Attribute Name: " + attribute.fieldName() +
                           ", Attribute Type: " + attribute.type().toString()));
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested schema
mapping was not found.", cause);
```

```
                    }

                    // Wrap other exceptions in a CompletionException with the
 message.
                    throw new CompletionException("Failed to get schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie GetSchemaMappingin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const getSchemaMappingJsonParams = {
    schemaName: `${data.inputs.schemaNameJson}`,
```

```
  };
  try {
    const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
    const response = await erClient.send(command);
    console.log(response);
    console.log(
      `Schema mapping for the JSON data:\n ${response.mappedInputFields[0]}`,
    );
    console.log("Schema mapping ARN is: ", response.schemaArn);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie GetSchemaMappingin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`ListSchemaMappings`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListSchemaMappings` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping
to the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
```

- Einzelheiten zur API finden Sie ListSchemaMappingsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.
```

```
import { fileURLToPath } from "node:url";

import {
  ListSchemaMappingsCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const listSchemaMappingsParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new ListSchemaMappingsCommand(listSchemaMappingsParams);
      const response = await erClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  }
```

- Einzelheiten zur API finden Sie ListSchemaMappingsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`StartMatchingJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartMatchingJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the
matching job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already
running. Resolve conflicts before starting a new job.", cause);
```

```
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
 exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- Einzelheiten zur API finden Sie StartMatchingJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
import {
  StartMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const matchingJobOfWorkflowParams = {
    workflowName: `${data.inputs.workflowName}`,
  };
```

```
  try {
    const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
    const response = await erClient.send(command);
    console.log(`Job ID: ${response.jobID} \n
The matching job was successfully started.`);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie StartMatchingJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`TagResource`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
```

```
   public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
       Map<String, String> tags = new HashMap<>();
       tags.put("tag1", "tag1Value");
       tags.put("tag2", "tag2Value");

       TagResourceRequest request = TagResourceRequest.builder()
           .resourceArn(schemaMappingARN)
           .tags(tags)
           .build();

       return getResolutionAsyncClient().tagResource(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully tagged the resource, log the success message.
                   logger.info("Successfully tagged the resource.");
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource to tag was
not found.", cause);
                   }
                   throw new CompletionException("Failed to tag the resource: "
+ exception.getMessage(), exception);
               }
           });
   }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  TagResourceCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const tagResourceCommandParams = {
    resourceArn: `${data.inputs.schemaArn}`,
    tags: {
      tag1: "tag1Value",
      tag2: "tag2Value",
    },
  };
  try {
    const command = new TagResourceCommand(tagResourceCommandParams);
    const response = await erClient.send(command);
    console.log("Successfully tagged the resource.");
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK für JavaScript API-Referenz.

Codebeispiele für die Verwendung von OpenSearch Diensten AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon OpenSearch Service mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- OpenSearch Service-Benutzerhandbuch — Weitere Informationen zum OpenSearch Service.
- OpenSearch Service-API-Referenz — Details zu allen verfügbaren OpenSearch Serviceaktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von OpenSearch Diensten AWS SDKs
  - Hallo Service OpenSearch
  - Erfahren Sie mehr über die Kernoperationen von Amazon OpenSearch Service mit einem AWS SDK
  - Aktionen für OpenSearch Dienste, die AWS SDKs
    - Verwenden Sie AddTagsmit einem AWS SDK
    - Verwenden Sie ChangeProgressmit einem AWS SDK

- Verwenden Sie CreateDomainmit einem AWS SDK
- Verwenden Sie DeleteDomainmit einem AWS SDK
- Verwenden Sie DescribeDomainmit einem AWS SDK
- Verwenden Sie ListDomainNamesmit einem AWS SDK
- Verwenden Sie ListTagsmit einem AWS SDK
- Verwenden Sie UpdateDomainConfigmit einem AWS SDK

- Szenarien für die Verwendung von OpenSearch Diensten AWS SDKs
  - Erste Schritte mit Such- und Analysemaschinen

Grundlegende Beispiele für die Verwendung von OpenSearch Diensten AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon OpenSearch Service mit AWS SDKs verwenden können.

Beispiele

- Hallo Service OpenSearch
- Erfahren Sie mehr über die Kernoperationen von Amazon OpenSearch Service mit einem AWS SDK
- Aktionen für OpenSearch Dienste, die AWS SDKs
  - Verwenden Sie AddTagsmit einem AWS SDK
  - Verwenden Sie ChangeProgressmit einem AWS SDK
  - Verwenden Sie CreateDomainmit einem AWS SDK
  - Verwenden Sie DeleteDomainmit einem AWS SDK
  - Verwenden Sie DescribeDomainmit einem AWS SDK
  - Verwenden Sie ListDomainNamesmit einem AWS SDK
  - Verwenden Sie ListTagsmit einem AWS SDK
  - Verwenden Sie UpdateDomainConfigmit einem AWS SDK

Hallo Service OpenSearch

Das folgende Codebeispiel zeigt, wie Sie mit OpenSearch Service beginnen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
```

```
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- Einzelheiten zur API finden Sie ListVersionsin der AWS SDK for Java 2.x API-Referenz.

Erfahren Sie mehr über die Kernoperationen von Amazon OpenSearch Service mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Dienstdomäne.
- Stellt detaillierte Informationen zu einer bestimmten OpenSearch Dienstdomäne bereit.
- Listet alle OpenSearch Dienstdomänen auf, die dem Konto gehören.
- Wartet, bis der Änderungsstatus der OpenSearch Dienstdomäne den Status „Abgeschlossen“ erreicht hat.
- Ändert die Konfiguration einer vorhandenen OpenSearch Dienstdomäne.
- Fügt der OpenSearch Service-Domain ein Tag hinzu.
- Listet die mit einer OpenSearch Dienstdomäne verknüpften Tags auf.
- Entfernt Tags aus einer OpenSearch Dienstdomäne.
- Löscht die OpenSearch Dienstdomäne.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, in dem die OpenSearch Servicefunktionen demonstriert werden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.

            Use the Amazon OpenSearch Service API to create, configure, and
 manage OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are
 focused on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
```

```
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/
latest/clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
 engine,
            which is an open-source search and analytics engine derived from
 Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources
 and storage that hosts
```

```
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check
on the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
```

```
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is
creating a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling
the number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource
was not found: Error message: {}, Error code {}",
resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
```

```
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change,
allowing you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of
instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
```

```
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: "
+ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
            associate with an OpenSearch Service domain. You can use these tags
to track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: "
+ cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " +
rt.getMessage(), rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
```

```
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we
created in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
 }
```

Eine Wrapper-Klasse für OpenSearch Service SDK-Methoden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import
 software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import
 software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
```

```
import
 software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
 created domain
```

```
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
```

```
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has
been deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse>
deleteSpecificDomainAsync(String domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }

    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the
domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
```

```
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list
of {@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain
names on success
            });
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();
```

```
        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress
for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response =
getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
```

```
                                throw new RuntimeException("Failed to check
domain progress", ex);
                            }
                            return resp;
                        }).join();

                    String state =
response.changeProgressStatus().statusAsString();  // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds =
(System.currentTimeMillis() - startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " +
state + " | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
```

```
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }


    /**
     * Asynchronously lists the tags associated with the specified Amazon
Resource Name (ARN).
     * @param arn the Amazon Resource Name (ARN) of the resource for which to
list the tags
     * @return a {@link CompletableFuture} that, when completed, will contain a
list of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();
```

```
        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

Aktionen für OpenSearch Dienste, die AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne OpenSearch Serviceaktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die OpenSearch Service-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von OpenSearch Diensten AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon OpenSearch Service API-Referenz.

Beispiele

- Verwenden Sie AddTagsmit einem AWS SDK
- Verwenden Sie ChangeProgressmit einem AWS SDK
- Verwenden Sie CreateDomainmit einem AWS SDK
- Verwenden Sie DeleteDomainmit einem AWS SDK
- Verwenden Sie DescribeDomainmit einem AWS SDK
- Verwenden Sie ListDomainNamesmit einem AWS SDK
- Verwenden Sie ListTagsmit einem AWS SDK
- Verwenden Sie UpdateDomainConfigmit einem AWS SDK

Verwenden Sie **`AddTags`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`AddTags`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
```

```
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Einzelheiten zur API finden Sie AddTagsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ChangeProgress`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ChangeProgress`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
    * @param domainName the name of the OpenSearch domain to check the progress
for
    * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
    */
   public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
       DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
           .domainName(domainName)
           .build();

       return CompletableFuture.runAsync(() -> {
           boolean isCompleted = false;
           long startTime = System.currentTimeMillis();

           while (!isCompleted) {
               try {
                   // Handle the async client call using `join` to block
synchronously for the result
                   DescribeDomainChangeProgressResponse response =
getAsyncClient()
                       .describeDomainChangeProgress(request)
                       .handle((resp, ex) -> {
                           if (ex != null) {
                               throw new RuntimeException("Failed to check
domain progress", ex);
                           }
                           return resp;
                       }).join();
```

```
                    String state =
response.changeProgressStatus().statusAsString();  // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds =
(System.currentTimeMillis() - startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " +
state + " | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- Einzelheiten zur API finden Sie ChangeProgressin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateDomain`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDomain` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Such- und Analysemaschinen
- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
```

```
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
 createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
 createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
 throwable);
                });
    }
```

- Einzelheiten zur API finden Sie CreateDomainin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }

    val ebsOptionsOb =
```

```
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- Einzelheiten zur API finden Sie CreateDomainin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteDomain`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDomain` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Such- und Analysemaschinen
- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a specific domain asynchronously.
    * @param domainName the name of the domain to be deleted
    * @return a {@link CompletableFuture} that completes when the domain has
been deleted
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteDomainResponse>
deleteSpecificDomainAsync(String domainName) {
       DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
           .domainName(domainName)
           .build();

       // Delete domain asynchronously
       return getAsyncClient().deleteDomain(domainRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteDomainin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteDomainin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeDomain`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeDomain`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Such- und Analysemaschinen
- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Describes the specified domain asynchronously.
    *
    * @param domainName the name of the domain to describe
    * @return a {@link CompletableFuture} that completes with the ARN of the
domain
    * @throws RuntimeException if the domain description fails
    */
   public CompletableFuture<String> describeDomainAsync(String domainName) {
       DescribeDomainRequest request = DescribeDomainRequest.builder()
           .domainName(domainName)
           .build();

       return getAsyncClient().describeDomain(request)
           .handle((response, exception) -> {  // Handle both response and
exception
               if (exception != null) {
                   throw new RuntimeException("Failed to describe domain",
exception);
               }
               DomainStatus domainStatus = response.domainStatus();
               String endpoint = domainStatus.endpoint();
               String arn = domainStatus.arn();
               String engineVersion = domainStatus.engineVersion();
               logger.info("Domain endpoint is: " + endpoint);
               logger.info("ARN: " + arn);
               System.out.println("Engine version: " + engineVersion);

               return arn;  // Return ARN when successful
           });
   }
```

- Einzelheiten zur API finden Sie DescribeDomainin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListDomainNames`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListDomainNames` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list
of {@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain
names on success
```

```
            });
    }
```

- Einzelheiten zur API finden Sie ListDomainNamesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListDomainNamesin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListTags`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListTags`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String
domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();

       return getAsyncClient().addTags(addTagsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    throw new RuntimeException("Failed to add tags to the domain:
" + domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- Einzelheiten zur API finden Sie ListTagsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UpdateDomainConfig`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `UpdateDomainConfig` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erfahren Sie mehr über die Kernabläufe von OpenSearch Service

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
```

```
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
 UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
 configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- Einzelheiten zur API finden Sie UpdateDomainConfigin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
```

```
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient
 ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- Einzelheiten zur API finden Sie UpdateDomainConfigin der API-Referenz zum AWS SDK für Kotlin.

Szenarien für die Verwendung von OpenSearch Diensten AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in OpenSearch Service mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb von OpenSearch Service oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Such- und Analysemaschinen

Erste Schritte mit Such- und Analysemaschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Dienstdomäne
- Laden Sie Daten auf Ihre Domain hoch
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
```

```
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to
 delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/
null 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
```

```
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
```

```
# This policy restricts access to specific actions and should be further
 restricted to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for
 completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack
 creation failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
```

```
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo
 "No"; fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel",
 "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
 "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
 "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
 Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955,
 "actor": ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal",
 "Backus, Jim", "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen,
```

```
 Corey", "Birch, Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh,
 Nelson", "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira",
 "McMahon, David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }
```

```
    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous
 (authentication not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential
 backoff..."
```

```
            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT
+1))/$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:"
 | cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
 (HTTP $HEALTH_HTTP_CODE)"
```

```
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
 minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding
 with data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
 '"result"'; then
            echo "# Single document uploaded successfully! (HTTP
 $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
 $UPLOAD_HTTP_CODE)"
        fi
```

```
        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
 '"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
 $BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
```

```
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-
name "$DOMAIN_NAME" --query 'DomainStatus.{AdvancedSecurityOptions:
 AdvancedSecurityOptions, AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
```

```
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user
 works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
```

```
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."
```

```
# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/
_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

Codebeispiele für die EventBridge Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon EventBridge mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- EventBridge Benutzerhandbuch — Weitere Informationen zu EventBridge.
- EventBridge API-Referenz — Details zu allen verfügbaren EventBridge Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung EventBridge AWS SDKs
  - Hallo EventBridge
  - Lernen Sie die Grundlagen von EventBridge mit einem AWS SDK
  - Aktionen zur EventBridge Verwendung AWS SDKs
    - Verwenden Sie DeleteRulemit einem AWS SDK oder CLI
    - Verwenden Sie DescribeRulemit einem AWS SDK oder CLI
    - Verwenden Sie DisableRulemit einem AWS SDK oder CLI
    - Verwenden Sie EnableRulemit einem AWS SDK oder CLI
    - Verwenden Sie ListRuleNamesByTargetmit einem AWS SDK oder CLI
    - Verwenden Sie ListRulesmit einem AWS SDK oder CLI
    - Verwenden Sie ListTargetsByRulemit einem AWS SDK oder CLI
    - Verwenden Sie PutEventsmit einem AWS SDK oder CLI
    - Verwenden Sie PutRulemit einem AWS SDK oder CLI
    - Verwenden Sie PutTargetsmit einem AWS SDK oder CLI
    - Verwenden Sie RemoveTargetsmit einem AWS SDK oder CLI
- Szenarien für die Verwendung EventBridge AWS SDKs
  - Erstellen und lösen Sie eine Regel in Amazon EventBridge mit einem aus AWS SDK
  - Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK
  - Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Grundlegende Beispiele für die Verwendung EventBridge AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon EventBridge mit AWS SDKs verwenden können.

Beispiele

- Hallo EventBridge
- Lernen Sie die Grundlagen von EventBridge mit einem AWS SDK
- Aktionen zur EventBridge Verwendung AWS SDKs
  - Verwenden Sie DeleteRulemit einem AWS SDK oder CLI
  - Verwenden Sie DescribeRulemit einem AWS SDK oder CLI
  - Verwenden Sie DisableRulemit einem AWS SDK oder CLI
  - Verwenden Sie EnableRulemit einem AWS SDK oder CLI
  - Verwenden Sie ListRuleNamesByTargetmit einem AWS SDK oder CLI
  - Verwenden Sie ListRulesmit einem AWS SDK oder CLI
  - Verwenden Sie ListTargetsByRulemit einem AWS SDK oder CLI
  - Verwenden Sie PutEventsmit einem AWS SDK oder CLI
  - Verwenden Sie PutRulemit einem AWS SDK oder CLI
  - Verwenden Sie PutTargetsmit einem AWS SDK oder CLI
  - Verwenden Sie RemoveTargetsmit einem AWS SDK oder CLI

Hallo EventBridge

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von EventBridge beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;

public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();

        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie ListEventBusesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
            for (EventBus bus : buses) {
```

```
                System.out.println("The name of the event bus is: " +
 bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListEventBusesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use
 { eventBrClient ->
```

```
        val response: ListEventBusesResponse =
 eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListEventBusesin der API-Referenz zum AWS SDK für Kotlin.

Lernen Sie die Grundlagen von EventBridge mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine Regel und fügen Sie ihr ein Ziel hinzu.
- Aktivieren und deaktivieren Sie Regeln.
- Listen Sie Regeln und Ziele auf und aktualisieren Sie sie.
- Senden Sie Ereignisse und bereinigen Sie dann die Ressourcen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
```

```
    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();

            var email = await SubscribeToSnsTopic(topicArn);

            await AddSnsTarget(topicArn);

            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);
```

```
            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));
```

```
        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                "\"Effect\": \"Allow\"," +
                                "\"Principal\": {" +
                                $"\"Service\": \"events.amazonaws.com\"" +
                                "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                                "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
                RoleName = roleName
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with
 EventBridge events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events
 enabled.");
```

```
        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge
events enabled.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger
a subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
```

```
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as
an EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic
for email subscriptions.");

        var topicName = _configuration["topicName"];

        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";
```

```
        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
    {
        Console.WriteLine(new string('-', 80));


        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
```

```
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });

        Console.WriteLine($"Use the link in the email you received to confirm
your subscription, then press Enter to continue.");

        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
        return email;
    }

    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when
an Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
```

```
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email
when the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName,
topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await
_eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);
```

```
        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will
trigger a subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");

        var eventRuleName = _configuration["eventRuleName"];
        var targets = await
_eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id}
Input: {t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
```

```
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await
_eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
        if (GetYesNoResponse($"\tDelete all targets and event rule
{eventRuleName}? (y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }

        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic
{topicName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
```

```
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
 ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess",
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Erstellen Sie eine Klasse, die EventBridge Operationen umschließt.

```
/// <summary>
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
 ILogger<EventBridgeWrapper> logger)

    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
 eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
```

```
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
 uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
 null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
```

```
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
 _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
```

```
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
+
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
```

```
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();
```

```
        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.
\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
 event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
```

```
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }

    /// <summary>
```

```
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }
```

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete an event rule by name.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException,
 IOException {
```

```
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                    topicName - The name of the Amazon Simple Notification
Service (Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();
```

```
        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM)
role to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the
user subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to
the specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email
when an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
```

```
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
 the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to
 the S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a
 custom pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn,
 eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has
successfully completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient,
SnsClient snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String
bucketName, String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();
```

```
        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from
role " + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
```

```
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
```

```
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }

    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();
```

```
        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
 ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
 topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();
```

```
        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
```

```
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
```

```
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
```

```
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " +
rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
```

```
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
```

```
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String
bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
```

```
            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
```

```
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets
 the user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon
 SNS) topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON =
        "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"
```

```
    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use
 with Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
    println("3. Create a rule that triggers when an object is uploaded to Amazon
 S3.")
    delay(10000)
    addEventRule(roleArn, bucketName, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("4. List rules on the event bus.")
    listRules()
```

```
    println(DASHES)

    println(DASHES)
    println("5. Create a new SNS topic for testing and let the user subscribe to
 the topic.")
    val topicArn = createSnsTopic(topicName)
    println(DASHES)

    println(DASHES)
    println("6. Add a target to the rule that sends an email to the specified
 topic.")
    println("Enter your email to subscribe to the Amazon SNS topic:")
    val email = sc.nextLine()
    subEmail(topicArn, email)
    println("Use the link in the email you received to confirm your subscription.
 Then press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("7. Create an EventBridge event that sends an email when an Amazon S3
 object is created.")
    addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName,
 bucketName)
    println(DASHES)

    println(DASHES)
    println("8. List targets.")
    listTargets(eventRuleName)
    println(DASHES)

    println(DASHES)
    println(" 9. List the rules for the same target.")
    listTargetRules(topicArn)
    println(DASHES)

    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
```

```
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
    println("17. Send an event to trigger the rule. This will trigger a
 subscription email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
```

```
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully
 completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
```

```
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
```

```
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets
```

```
        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
```

```
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putRule(request)
```

```
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }
```

```
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()
```

```
    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "EventBridgePublishTopic",
                "Effect": "Allow",
                "Principal": {
                    "Service": "events.amazonaws.com"
                },
                "Resource": "*",
                "Action": "sns:Publish"
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
    }
}

suspend fun listRules() {
    val rulesRequest =
```

```
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
```

```
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}
```

```
suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
        return response.role?.arn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Aktionen zur EventBridge Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne EventBridge Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die EventBridge API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung EventBridge AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon EventBridge API-Referenz.

Beispiele

- Verwenden Sie DeleteRulemit einem AWS SDK oder CLI
- Verwenden Sie DescribeRulemit einem AWS SDK oder CLI
- Verwenden Sie DisableRulemit einem AWS SDK oder CLI
- Verwenden Sie EnableRulemit einem AWS SDK oder CLI
- Verwenden Sie ListRuleNamesByTargetmit einem AWS SDK oder CLI
- Verwenden Sie ListRulesmit einem AWS SDK oder CLI
- Verwenden Sie ListTargetsByRulemit einem AWS SDK oder CLI

- Verwenden Sie PutEventsmit einem AWS SDK oder CLI
- Verwenden Sie PutRulemit einem AWS SDK oder CLI
- Verwenden Sie PutTargetsmit einem AWS SDK oder CLI
- Verwenden Sie RemoveTargetsmit einem AWS SDK oder CLI

Verwenden Sie **`DeleteRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Regel anhand ihres Namens.

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteRulein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine CloudWatch Ereignisregel zu löschen

In diesem Beispiel wird die Regel mit dem Namen EC2InstanceStateChanges gelöscht:

```
aws events delete-rule --name "EC2InstanceStateChanges"
```

- Einzelheiten zur API finden Sie DeleteRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- Einzelheiten zur API finden Sie DeleteRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- Einzelheiten zur API finden Sie DeleteRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie den Status einer Regel anhand der Regelbeschreibung ab.

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- Einzelheiten zur API finden Sie DescribeRulein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So zeigen Sie Informationen zu einer CloudWatch Ereignisregel an

In diesem Beispiel werden Informationen zu der Regel mit dem Namen angezeigt DailyLambdaFunction:

```
aws events describe-rule --name "DailyLambdaFunction"
```

- Einzelheiten zur API finden Sie DescribeRuleunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response =
eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- Einzelheiten zur API finden Sie DescribeRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DisableRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Deaktivieren Sie eine Regel anhand ihres Regelnamens.

```
/// <summary>
/// Disable a particular rule on an event bus.
/// </summary
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DisableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
        new DisableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DisableRulein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine CloudWatch Ereignisregel zu deaktivieren

In diesem Beispiel wird die genannte DailyLambdaFunction Regel deaktiviert. Die Regel wird nicht gelöscht:

```
aws events disable-rule --name "DailyLambdaFunction"
```

- Einzelheiten zur API finden Sie DisableRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Deaktivieren Sie eine Regel anhand ihres Regelnamens.

```
   public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
       try {
           if (!isEnabled) {
               System.out.println("Disabling the rule: " + eventRuleName);
               DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();

               eventBrClient.disableRule(ruleRequest);
           } else {
               System.out.println("Enabling the rule: " + eventRuleName);
               EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                       .name(eventRuleName)
                       .build();
               eventBrClient.enableRule(ruleRequest);
           }

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DisableRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Einzelheiten zur API finden Sie DisableRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`EnableRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
/// <summary>
/// Enable a particular rule on an event bus.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableRuleByName(string ruleName)
{
    var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
        new EnableRuleRequest()
        {
            Name = ruleName
        });
    return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie EnableRulein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine CloudWatch Ereignisregel zu aktivieren

In diesem Beispiel wird die genannte Regel aktiviert DailyLambdaFunction, die zuvor deaktiviert wurde:

```
aws events enable-rule --name "DailyLambdaFunction"
```

- Einzelheiten zur API finden Sie EnableRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
```

```
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie EnableRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
```

```
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- Einzelheiten zur API finden Sie EnableRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListRuleNamesByTarget`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListRuleNamesByTarget` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regelnamen mithilfe des Ziels auf.

```
    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
```

```
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await
_amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Einzelheiten zur API finden Sie ListRuleNamesByTargetin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So zeigen Sie alle Regeln mit einem bestimmten Ziel an

In diesem Beispiel werden alle Regeln angezeigt, deren Ziel die Lambda-Funktion MyFunctionName "" ist:

```
aws events list-rule-names-by-target --target-arn "arn:aws:lambda:us-
east-1:123456789012:function:MyFunctionName"
```

- Einzelheiten zur API finden Sie unter ListRuleNamesByTarget AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regelnamen mithilfe des Ziels auf.

```
    public static void listTargetRules(EventBridgeClient eventBrClient, String
 topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
 ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();

        ListRuleNamesByTargetResponse response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }
```

- Einzelheiten zur API finden Sie ListRuleNamesByTargetin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response =
 eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
```

```
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListRuleNamesByTargetin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListRules`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListRules` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regeln für einen Event Bus auf.

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
```

```
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- Einzelheiten zur API finden Sie ListRulesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Liste aller CloudWatch Event-Regeln anzuzeigen

In diesem Beispiel werden alle CloudWatch Event-Regeln in der Region angezeigt:

```
aws events list-rules
```

Um eine Liste von CloudWatch Event-Regeln anzuzeigen, die mit einer bestimmten Zeichenfolge beginnen.

In diesem Beispiel werden alle CloudWatch Event-Regeln in der Region angezeigt, deren Name mit „Täglich“ beginnt:

```
aws events list-rules --name-prefix "Daily"
```

- Einzelheiten zur API finden Sie ListRulesunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " +
rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListRulesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListRulesin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListTargetsByRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTargetsByRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Ziele für eine Regel mithilfe des Regelnamens auf.

```
/// <summary>
/// List all of the targets matching a rule by name.
/// </summary>
/// <param name="ruleName">The name of the rule.</param>
/// <returns>The list of targets.</returns>
public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
{
    var results = new List<Target>();
    var request = new ListTargetsByRuleRequest()
    {
        Rule = ruleName
    };
    ListTargetsByRuleResponse response;
    do
    {
        response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
        results.AddRange(response.Targets);
        request.NextToken = response.NextToken;

    } while (response.NextToken is not null);

    return results;
}
```

- Einzelheiten zur API finden Sie ListTargetsByRulein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um alle Ziele für eine CloudWatch Ereignisregel anzuzeigen

In diesem Beispiel werden alle Ziele der Regel mit dem Namen angezeigt DailyLambdaFunction:

```
aws events list-targets-by-rule --rule  "DailyLambdaFunction"
```

- Einzelheiten zur API finden Sie ListTargetsByRuleunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Ziele für eine Regel mithilfe des Regelnamens auf.

```
   public static void listTargets(EventBridgeClient eventBrClient, String
ruleName) {
       ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
               .rule(ruleName)
               .build();

       ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
       List<Target> targetsList = res.targets();
       for (Target target: targetsList) {
           System.out.println("Target ARN: "+target.arn());
       }
   }
```

- Einzelheiten zur API finden Sie ListTargetsByRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTargetsByRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutEvents`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutEvents` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen

- Erstellen und Auslösen einer Regel

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie ein Ereignis, das einem benutzerdefinierten Muster für eine Regel entspricht.

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });
```

```
        return response.FailedEntryCount == 0;
    }
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Senden Sie das Ereignis.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
```

```
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um ein benutzerdefiniertes Ereignis an CloudWatch Events zu senden

In diesem Beispiel wird ein benutzerdefiniertes Ereignis an CloudWatch Events gesendet. Das Ereignis ist in der Datei putevents.json enthalten:

```
aws events put-events --entries file://putevents.json
```

Die Datei putevents.json hat folgenden Inhalt:

```
[
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value1\", \"key2\": \"value2\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
    "DetailType": "myDetailType"
  },
  {
    "Source": "com.mycompany.myapp",
    "Detail": "{ \"key1\": \"value3\", \"key2\": \"value4\" }",
    "Resources": [
      "resource1",
      "resource2"
    ],
```

```
    "DetailType": "myDetailType"
   }
]
```

- Einzelheiten zur API finden Sie PutEventsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
               .detail(json)
               .detailType("ExampleType")
               .build();

       PutEventsRequest eventsRequest = PutEventsRequest.builder()
               .entries(entry)
               .build();

       eventBrClient.putEvents(eventsRequest);
   }
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
          Resources: resources,
          Source: source,
        },
      ],
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};
```

```
ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }
```

```
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putEvents(eventsRequest)
    }
}
```

- Einzelheiten zur API finden Sie PutEventsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRule` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen und Auslösen einer Regel
- Senden Sie Ereignisbenachrichtigungen an EventBridge

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Bucket von Amazon Simple Storage Service hinzugefügt wird.

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created
 in a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
```

```
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]"
 +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

Erstellen Sie eine Regel, die ein benutzerdefiniertes Muster verwendet.

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";
```

```
        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie die Regel.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
        request.SetRoleArn(role_arn);
```

```
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um Regeln für CloudWatch Ereignisse zu erstellen

Im folgenden Beispiel wird eine Regel erstellt, die jeden Tag um 09:00 Uhr (UTC) ausgelöst wird. Wenn Sie put-targets verwenden, um eine Lambda-Funktion als Ziel dieser Regel hinzuzufügen, können Sie die Lambda-Funktion jeden Tag zur angegebenen Zeit ausführen:

```
aws events put-rule --name "DailyLambdaFunction" --schedule-expression "cron(0 9
 * * ? *)"
```

Im folgenden Beispiel wird eine Regel erstellt, die ausgelöst wird, wenn eine EC2-Instance in der Region den Status ändert:

```
aws events put-rule --name "EC2InstanceStateChanges" --event-pattern "{\"source
\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
 --role-arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

Im folgenden Beispiel wird eine Regel erstellt, die ausgelöst wird, wenn eine EC2-Instance in der Region gestoppt oder beendet wird:

```
aws events put-rule --name "EC2InstanceStateChangeStopOrTerminate" --event-
pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change
 Notification\"],\"detail\":{\"state\":[\"stopped\",\"terminated\"]}}" --role-
arn "arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

- Einzelheiten zur API finden Sie PutRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by
the Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Amazon-Simple-Storage-Service-Bucket hinzugefügt wird.

```
    // Create a new event rule that triggers when an Amazon S3 object is created
in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String
roleArn, String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";

export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
```

```
  // }
  return response;
};
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the
 Kotlin API"
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use
 { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Amazon-Simple-Storage-Service-Bucket hinzugefügt wird.

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
```

```
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- Einzelheiten zur API finden Sie PutRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutTargets`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutTargets` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Senden Sie Ereignisbenachrichtigungen an EventBridge

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Amazon-SNS-Thema als Ziel für eine Regel hinzu.

```
    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string
targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
```

```
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the
default event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string
ruleName, string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
```

```
                    InputTemplate = "\"Notification: an object was uploaded to
 bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
 {e.ErrorCode}");
            });
        }
        return targetID;
    }
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
```

```
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Fügen Sie das Ziel hinzu.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
        }
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie Ziele für CloudWatch Event-Regeln hinzu

Im folgenden Beispiel wird eine Lambda-Funktion als Ziel einer Regel hinzugefügt:

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:MyFunctionName"
```

Im folgenden Beispiel wird ein Amazon-Kinesis-Stream als Ziel festgelegt, sodass Ereignisse, die von dieser Regel erfasst werden, an den Stream weitergeleitet werden:

```
aws events put-targets --rule EC2InstanceStateChanges --targets
 "Id"="1","Arn"="arn:aws:kinesis:us-east-1:123456789012:stream/
MyStream","RoleArn"="arn:aws:iam::123456789012:role/MyRoleForThisRule"
```

Im folgenden Beispiel werden zwei Amazon-Kinesis-Streams als Ziele für eine Regel festgelegt:

```
aws events put-targets --rule DailyLambdaFunction --targets
 "Id"="Target1","Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream1","RoleArn"="arn:aws:iam::379642911888:role/ MyRoleToAccessLambda"
  "Id"="Target2"," Arn"="arn:aws:kinesis:us-east-1:379642911888:stream/
MyStream2","RoleArn"="arn:aws:iam::379642911888:role/MyRoleToAccessLambda"
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Amazon-SNS-Thema als Ziel für eine Regel hinzu.

```
    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
 ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
```

```
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon
SNS target " + topicName + " for bucket "
                + bucketName + ".");
    }
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();
```

```
            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
```

```
    }),
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
```

```
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3
 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
```

```
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target
 $topicName for bucket $bucketName.")
    }
}
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
```

```
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- Einzelheiten zur API finden Sie PutTargetsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`RemoveTargets`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTargets` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Entfernen Sie alle Ziele für eine Regel mithilfe des Regelnamens.

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
```

```
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage},
code {e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie RemoveTargetsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So entfernen Sie ein Ziel für ein Ereignis

In diesem Beispiel wird der Amazon Kinesis Kinesis-Stream mit dem Namen MyStream 1 aus dem Ziel der Regel DailyLambdaFunction entfernt. Bei DailyLambdaFunction seiner Erstellung wurde dieser Stream als Ziel mit der ID Target1 festgelegt:

```
aws events remove-targets --rule "DailyLambdaFunction" --ids "Target1"
```

- Einzelheiten zur API finden Sie RemoveTargetsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Entfernen Sie alle Ziele für eine Regel mithilfe des Regelnamens.

```
   public static void deleteTargetsFromRule(EventBridgeClient eventBrClient,
String eventRuleName) {
       // First, get all targets that will be deleted.
       ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
               .rule(eventRuleName)
               .build();

       ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
       List<Target> allTargets = response.targets();

       // Get all targets and delete them.
       for (Target myTarget : allTargets) {
           RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                   .rule(eventRuleName)
                   .ids(myTarget.id())
                   .build();

           eventBrClient.removeTargets(removeTargetsRequest);
           System.out.println("Successfully removed the target");
       }
   }
```

- Einzelheiten zur API finden Sie RemoveTargetsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie RemoveTargetsin der API-Referenz zum AWS SDK für Kotlin.

Szenarien für die Verwendung EventBridge AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien EventBridge mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb EventBridge oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen und lösen Sie eine Regel in Amazon EventBridge mit einem aus AWS SDK
- Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Erstellen und lösen Sie eine Regel in Amazon EventBridge mit einem aus AWS SDK

Das folgende Codebeispiel zeigt, wie eine Regel in Amazon erstellt und ausgelöst wird EventBridge.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Funktionen in der richtigen Reihenfolge auf.

```
require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
```

```
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'
```

Überprüfen Sie, ob das angegebene Amazon Simple Notification Service (Amazon SNS)-Thema unter den für diese Funktion bereitgestellten Themen vorhanden ist.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Überprüfen Sie, ob das angegebene Thema unter den für den Aufrufer in Amazon SNS verfügbaren Themen vorhanden ist.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
```

```
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Erstellen Sie ein Thema in Amazon SNS und abonnieren Sie dann eine E-Mail-Adresse, um Benachrichtigungen zu diesem Thema zu erhalten.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
```

```
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Prüfen Sie, ob die angegebene AWS Identity and Access Management (IAM-) Rolle unter den für diese Funktion bereitgestellten Rollen existiert.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
```

```
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  false
end
```

Überprüfen Sie, ob die angegebene Rolle unter den für den Aufrufer in IAM verfügbaren Rollen vorhanden ist.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

      if role_found?(response.roles, role_arn)
        puts 'Role found.'
        return true
      end
    end
  end
  puts 'Role not found.'
  false
```

```
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end
```

Erstellen Sie eine Rolle in IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
```

```
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
  'Error'
end
```

Überprüft, ob die angegebene EventBridge Regel unter den für diese Funktion bereitgestellten Regeln existiert.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
```

```
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  false
end
```

Überprüft, ob die angegebene Regel zu den Regeln gehört, die dem Anrufer zur Verfügung stehen. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

      if rule_found?(response.rules, rule_name)
        puts 'Rule found.'
        return true
      end
    end
```

```
  end
  puts 'Rule not found.'
  false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  false
end
```

Erstellen Sie eine Regel in EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
```

```
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
     put_targets_response.failed_entry_count.positive?
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
```

```
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end
```

Prüfen Sie, ob die angegebene Protokollgruppe zu den Protokollgruppen gehört, die dem Anrufer in Amazon CloudWatch Logs zur Verfügung stehen.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
```

```
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

Erstellen Sie eine Protokollgruppe in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  false
end
```

Schreiben Sie ein Ereignis in einen Protokollstream in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
```

```
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '49542672486831074009579604567656788214806863282469607 3EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

Starten Sie eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance neu und fügen Sie Informationen über die zugehörige Aktivität zu einem Protokollstream in CloudWatch Logs hinzu.

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''
```

```
  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )

  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end
```

Zeigt Informationen zur Aktivität für eine Regel in an EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
```

```
    unit: 'Count'
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Zeigt Protokollinformationen für alle Protokolldatenströme in einer Protokollgruppe „ CloudWatch Protokolle“ an.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
```

```
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end
```

Zeigt dem Anrufer eine Erinnerung an, alle zugehörigen AWS Ressourcen, die er nicht mehr benötigt, manuell zu bereinigen.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log
 group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
```

```
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - PutEvents
  - PutRule

Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie ein Bucket aktiviert wird, um S3-Ereignisbenachrichtigungen an ein Amazon SNS-Thema EventBridge und eine Amazon SQS SQS-Warteschlange zu senden und Benachrichtigungen an diese weiterzuleiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
```

```
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Codebeispiele für die Verwendung von EventBridge Scheduler AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon EventBridge Scheduler mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- EventBridge Scheduler-Benutzerhandbuch — Weitere Informationen zu EventBridge Scheduler.
- EventBridge Scheduler-API-Referenz — Details zu allen verfügbaren EventBridge Scheduler-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von EventBridge Scheduler AWS SDKs
  - Hallo Scheduler EventBridge
  - Aktionen für Scheduler unter Verwendung EventBridge AWS SDKs
    - Verwenden Sie CreateSchedulemit einem AWS SDK
    - Verwenden Sie CreateScheduleGroupmit einem AWS SDK
    - Verwenden Sie DeleteSchedulemit einem AWS SDK
    - Verwenden Sie DeleteScheduleGroupmit einem AWS SDK
- Szenarien für die Verwendung von EventBridge Scheduler AWS SDKs
  - Ein vollständiges EventBridge Scheduler-Szenario mit geplanten Ereignissen unter Verwendung eines AWS SDK

Grundlegende Beispiele für die Verwendung von EventBridge Scheduler AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon EventBridge Scheduler mit AWS SDKs verwenden können.

Beispiele

- Hallo Scheduler EventBridge
- Aktionen für Scheduler unter Verwendung EventBridge AWS SDKs
  - Verwenden Sie CreateSchedulemit einem AWS SDK
  - Verwenden Sie CreateScheduleGroupmit einem AWS SDK
  - Verwenden Sie DeleteSchedulemit einem AWS SDK
  - Verwenden Sie DeleteScheduleGroupmit einem AWS SDK

Hallo Scheduler EventBridge

Die folgenden Codebeispiele zeigen, wie Sie mit EventBridge Scheduler beginnen können.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static class HelloScheduler
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the EventBridge Scheduler service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
```

```
                services.AddAWSService<IAmazonScheduler>()
            ).Build();

        // Now the client is available for injection.
        var schedulerClient =
 host.Services.GetRequiredService<IAmazonScheduler>();

        // You can use await and any of the async methods to get a response, or a
 paginator to list schedules or groups.
        var results = new List<ScheduleSummary>();
        var paginateSchedules = schedulerClient.Paginators.ListSchedules(
            new ListSchedulesRequest());
        Console.WriteLine(
            $"Hello AWS Scheduler! Let's list schedules in your account.");
        // Get the entire list using the paginator.
        await foreach (var schedule in paginateSchedules.Schedules)
        {
            results.Add(schedule);
        }
        Console.WriteLine($"\tTotal of {results.Count} schedule(s) available.");
        results.ForEach(s => Console.WriteLine($"\tSchedule: {s.Name}"));
    }
}
```

- Einzelheiten zur API finden Sie ListSchedulesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import
 software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient =
 SchedulerAsyncClient.create();

        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
        });

        // Wait for the asynchronous operation to complete.
        future.join();
```

```
        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n",
 results.size());
    }
}
```

- Einzelheiten zur API finden Sie ListSchedulesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_scheduler(scheduler_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EventBridge Scheduler
    client and list the schedules in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param scheduler_client: A Boto3 Amazon EventBridge Scheduler Client object.
 This object wraps
                             the low-level Amazon EventBridge Scheduler service
 API.
    """
    print("Hello, Amazon EventBridge Scheduler! Let's list some of your
 schedules:\n")
    paginator = scheduler_client.get_paginator("list_schedules")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    schedule_names: [str] = []
    for page in page_iterator:
```

```
        for schedule in page["Schedules"]:
            schedule_names.append(schedule["Name"])

    print(f"{len(schedule_names)} schedule(s) retrieved.")
    for schedule_name in schedule_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    hello_scheduler(boto3.client("scheduler"))
```

- Einzelheiten zur API finden Sie ListSchedulesin AWS SDK for Python (Boto3) API Reference.

Aktionen für Scheduler unter Verwendung EventBridge AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne EventBridge Scheduler-Aktionen mit SDKs ausgeführt werden. AWS Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die EventBridge Scheduler-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von EventBridge Scheduler AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon EventBridge Scheduler API-Referenz.

Beispiele

- Verwenden Sie CreateSchedulemit einem AWS SDK
- Verwenden Sie CreateScheduleGroupmit einem AWS SDK
- Verwenden Sie DeleteSchedulemit einem AWS SDK
- Verwenden Sie DeleteScheduleGroupmit einem AWS SDK

Verwenden Sie **`CreateSchedule`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSchedule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Geplante Ereignisse

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines
when the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which
the schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the
schedule after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible
time window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
```

```
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a
conflict. {ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}'
" +
                              $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CreateScheduleein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
```

```
     * @return true if the schedule was successfully created, false otherwise
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
```

```
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group
 {}, The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while
 creating the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception
 occurred while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: "
 + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateScheduleein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
```

```
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
client.

        :return: An instance of SchedulerWrapper initialized with the default
EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM
role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
```

```
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input":
 input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
 timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]
["Message"]
                )
            raise
```

- Einzelheiten zur API finden Sie CreateScheduleein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Constants for time calculations
    DATA lv_start_date TYPE /aws1/scdstartdate.
    DATA lv_end_date TYPE /aws1/scdenddate.
    DATA lv_start_timestamp TYPE timestamp.
    DATA lv_end_timestamp TYPE timestamp.
    DATA lv_hours_to_run TYPE i VALUE 1.

    " Get current timestamp
    GET TIME STAMP FIELD lv_start_timestamp.

    " Add 1 hour to the current timestamp using CL_ABAP_TSTMP
    lv_end_timestamp = cl_abap_tstmp=>add(
      tstmp = lv_start_timestamp
      secs = lv_hours_to_run * 3600 ).

    " Convert timestamps to decimal format for AWS API
    lv_start_date = lv_start_timestamp.
    lv_end_date = lv_end_timestamp.

    " Prepare flexible time window configuration
    DATA lo_flexible_time_window TYPE REF TO /aws1/cl_scdflexibletimewindow.
    IF iv_use_flexible_time_win = abap_true.
      " iv_use_flexible_time_win = ABAP_TRUE
      " Example: Set MaximumWindowInMinutes to 10 for flexible window
      lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
        iv_mode = 'FLEXIBLE'
        iv_maximumwindowinminutes = 10 ).
    ELSE.
      lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
        iv_mode = 'OFF' ).
    ENDIF.
```

```
        " Prepare target configuration
        " Example iv_target_arn = 'arn:aws:sqs:us-east-1:123456789012:my-queue'
        " Example iv_role_arn = 'arn:aws:iam::123456789012:role/SchedulerRole'
        " Example iv_input = '{"message": "Hello from EventBridge Scheduler"}'
        DATA(lo_target) = NEW /aws1/cl_scdtarget(
          iv_arn = iv_target_arn
          iv_rolearn = iv_role_arn
          iv_input = iv_input ).

        " Set action after completion if needed
        DATA lv_action_after_completion TYPE /aws1/scdactionaftercompletion.
        IF iv_delete_after_completion = abap_true.
          " iv_delete_after_completion = ABAP_TRUE
          lv_action_after_completion = 'DELETE'.
        ELSE.
          lv_action_after_completion = 'NONE'.
        ENDIF.

        " Create the schedule
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_expression = 'rate(15 minutes)'
        " Example iv_schedule_group_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedule(
          iv_name = iv_name
          iv_scheduleexpression = iv_schedule_expression
          iv_groupname = iv_schedule_group_name
          io_target = lo_target
          io_flexibletimewindow = lo_flexible_time_window
          iv_startdate = lv_start_date
          iv_enddate = lv_end_date
          iv_actionaftercompletion = lv_action_after_completion ).

        ov_schedule_arn = lo_result->get_schedulearn( ).
        MESSAGE 'Schedule created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateSchedulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateScheduleGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateScheduleGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Geplante Ereignisse

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await
_amazonScheduler.CreateScheduleGroupAsync(request);
```

```
            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CreateScheduleGroupin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
```

```
    * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse>
createScheduleGroup(String name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();

       logger.info("Initiating createScheduleGroup call for group: {}", name);
       CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
       futureResponse.whenComplete((response, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                   // Rethrow the ConflictException
                   throw (ConflictException) ex.getCause();
               } else {
                   throw new CompletionException("Failed to create schedule
group: " + name, ex);
               }
           } else if (response == null) {
               throw new RuntimeException("Failed to create schedule group:
response was null");
           } else {
               logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
           }
       });

       return futureResponse;
   }
```

- Einzelheiten zur API finden Sie CreateScheduleGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
```

```
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie CreateScheduleGroupin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedulegroup(
          iv_name = iv_name ).

        ov_schedule_group_arn = lo_result->get_schedulegrouparn( ).
        MESSAGE 'Schedule group created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule group: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
```

```
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateScheduleGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteSchedule`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteSchedule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Geplante Ereignisse

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
```

```
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name
'{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteSchedulein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a schedule with the specified name and group name.
     *
```

```
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the
schedule, except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while
deleting schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.",
ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

- Einzelheiten zur API finden Sie DeleteScheduleein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource
 was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]
["Message"]
                )
                raise
```

- Einzelheiten zur API finden Sie DeleteSchedulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_group_name = 'my-schedule-group'
        lo_scd->deleteschedule(
          iv_name = iv_name
          iv_groupname = iv_schedule_group_name ).
        MESSAGE 'Schedule deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteSchedulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteScheduleGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteScheduleGroup` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
{ex.Message}");
            return false;
        }
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteScheduleGroupin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes the specified schedule group.
    *
    * @param name the name of the schedule group to delete
    * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
    * @throws CompletionException if an error occurs while deleting the schedule
group
    */
   public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
       DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
           .name(name)
           .build();

       return getAsyncClient().deleteScheduleGroup(request)
           .thenRun(() -> {
               logger.info("Successfully deleted schedule group {}", name);
           })
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   if (ex instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource was not
found: " + ex.getMessage(), ex);
                   } else {
```

```
                        throw new CompletionException("Error deleting schedule
 group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- Einzelheiten zur API finden Sie DeleteScheduleGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.
```

```
        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie DeleteScheduleGroupin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        lo_scd->deleteschedulegroup(
          iv_name = iv_name ).
        MESSAGE 'Schedule group deleted successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule group not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteScheduleGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von EventBridge Scheduler AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in EventBridge Scheduler mit SDKs implementieren. AWS Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen im EventBridge Scheduler aufrufen oder mit anderen kombinieren. AWS-Services Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Ein vollständiges EventBridge Scheduler-Szenario mit geplanten Ereignissen unter Verwendung eines AWS SDK

Ein vollständiges EventBridge Scheduler-Szenario mit geplanten Ereignissen unter Verwendung eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Stellen Sie einen CloudFormation Stack mit den erforderlichen Ressourcen bereit.
- Erstellen Sie eine EventBridge Scheduler-Zeitplangruppe.
- Erstellen Sie einen einmaligen EventBridge Scheduler-Zeitplan mit einem flexiblen Zeitfenster.

- Erstellen Sie einen sich wiederholenden EventBridge Scheduler-Zeitplan mit einer bestimmten Rate.
- Löschen Sie den EventBridge Scheduler, den Zeitplan und die Zeitplangruppe.
- Bereinigen Sie die Ressourcen und löschen Sie den Stack.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using SchedulerActions;
using Exception = System.Exception;

namespace SchedulerScenario;

public class SchedulerWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the Amazon
 EventBridge Scheduler workflow:

    1. Prepare the Application:
```

```
      - Prompt the user for an email address to use for the subscription for the
 SNS topic subscription.
      - Prompt the user for a name for the Cloud Formation stack.
      - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
      - Store the outputs of the stack into variables for use in the scenario.
      - Create a schedule group for all schedules.

   2. Create one-time Schedule:
      - Create a one-time schedule to send an initial event.
      - Use a Flexible Time Window and set the schedule to delete after
 completion.
      - Wait for the user to receive the event email from SNS.

   3. Create a time-based schedule:
      - Prompt the user for how many X times per Y hours a recurring event
 should be scheduled.
      - Create the scheduled event for X times per hour for Y hours.
      - Wait for the user to receive the event email from SNS.
      - Delete the schedule when the user is finished.

   4. Clean up:
      - Prompt the user for y/n answer if they want to destroy the stack and
 clean up all resources.
      - Delete the schedule group.
      - Destroy the Cloud Formation stack and wait until the stack has been
 removed.
   */

   public static ILogger<SchedulerWorkflow> _logger = null!;
   public static SchedulerWrapper _schedulerWrapper = null!;
   public static IAmazonCloudFormation _amazonCloudFormation = null!;

   private static string _roleArn = null!;
   private static string _snsTopicArn = null!;

   public static bool _interactive = true;
   private static string _stackName = "default-scheduler-scenario-stack-name";
   private static string _scheduleGroupName = "scenario-schedules-group";
   private static string _stackResourcePath = "../../../../../../scenarios/
features/eventbridge_scheduler/resources/cfn_template.yaml";

   public static async Task Main(string[] args)
   {
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<SchedulerWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<SchedulerWorkflow>();

            _schedulerWrapper =
host.Services.GetRequiredService<SchedulerWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge Scheduler
Scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await CreateOneTimeSchedule();
                Console.WriteLine(new string('-', 80));

                Console.WriteLine(new string('-', 80));
                await CreateRecurringSchedule();
```

```
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario,
initiating cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("Amazon EventBridge Scheduler scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        try
        {
            // Prompt the user for an email address to use for the subscription.
            Console.WriteLine("\nThis example creates resources in a
CloudFormation stack, including an SNS topic" +
                          "\nthat will be subscribed to the EventBridge Scheduler
events. " +
                          "\n\nYou will need to confirm the subscription in order
to receive event emails. ");

            var emailAddress = PromptUserForEmail();

            // Prompt the user for a name for the CloudFormation stack
            _stackName = PromptUserForStackName();

            // Deploy the CloudFormation stack
            var deploySuccess = await DeployCloudFormationStack(_stackName,
emailAddress);
```

```
            if (deploySuccess)
            {
                // Create a schedule group for all schedules
                await
_schedulerWrapper.CreateScheduleGroupAsync(_scheduleGroupName);

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }
        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="email">The email to use for the subscription.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
string email)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = { Capability.CAPABILITY_NAMED_IAM }
            };

            // If an email is provided, set the parameter.
            if (!string.IsNullOrWhiteSpace(email))
            {
                request.Parameters = new List<Parameter>()
                {
```

```
                    new() { ParameterKey = "email", ParameterValue = email }
                };
            }

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                // Wait for the stack to be in CREATE_COMPLETE state
                bool stackCreated = await
WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    // Retrieve the output values
                    var success = await GetStackOutputs(response.StackId);
                    return success;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName, email);
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="client">The CloudFormation client.</param>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds.

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation complete.");
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                        describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation failed.");
                    return false;
                }
```

```
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Retrieves the output values from the CloudFormation stack.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    private static async Task<bool> GetStackOutputs(string stackId)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest { StackName =
stackId };

            var describeStacksResponse =
                await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                var stack = describeStacksResponse.Stacks[0];
                _roleArn = GetStackOutputValue(stack, "RoleARN");
                _snsTopicArn = GetStackOutputValue(stack, "SNStopicARN");
                return true;
            }
            else
            {
                _logger.LogError($"No stack found for stack outputs: {stackId}");
                return false;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                ex, $"Failed to retrieve CloudFormation stack outputs:
 {stackId}");
            return false;
        }
    }

    /// <summary>
    /// Get an output value by key from a CloudFormation stack.
    /// </summary>
    /// <param name="stack">The CloudFormation stack.</param>
    /// <param name="outputKey">The key of the output.</param>
    /// <returns>The value as a string.</returns>
    private static string GetStackOutputValue(Stack stack, string outputKey)
    {
        var output = stack.Outputs.First(o => o.OutputKey == outputKey);
        var outputValue = output.OutputValue;
        Console.WriteLine($"Stack output {outputKey}: {outputValue}");
        return outputValue;
    }

    /// <summary>
    /// Creates a one-time schedule to send an initial event.
    /// </summary>
    /// <returns>True if the one-time schedule was created successfully.</
returns>
    public static async Task<bool> CreateOneTimeSchedule()
    {
        var scheduleName =
            PromptUserForResourceName("Enter a name for the one-time schedule:");

        Console.WriteLine($"Creating a one-time schedule named '{scheduleName}' "
 +
                          $"\nto send an initial event in 1 minute with a
 flexible time window...");
        try
        {
            // Create a one-time schedule with a flexible time
            // window set to delete after completion.
            // You may also set a timezone instead of using UTC.
            var scheduledTime = DateTime.UtcNow.AddMinutes(1).ToString("s");

            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"at({scheduledTime})",
```

```
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"One time scheduled event test from schedule {scheduleName}.",
                true,
                useFlexibleTimeWindow: true);

            Console.WriteLine($"Subscription email will receive an email from
 this event.");
            Console.WriteLine($"You must confirm your subscription to receive
 event emails.");

            Console.WriteLine($"One-time schedule '{scheduleName}' created
 successfully.");
            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
 found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the one-time
 schedule '{scheduleName}'.");
            return false;
        }
    }

    /// <summary>
    /// Create a recurring schedule to send events at a specified rate in
 minutes.
    /// </summary>
    /// <returns>True if the recurring schedule was created successfully.</
returns>
    public static async Task<bool> CreateRecurringSchedule()
    {
        Console.WriteLine("Creating a recurring schedule to send events for one
 hour...");

        try
        {
            // Prompt the user for a schedule name.
```

```
            var scheduleName =
                PromptUserForResourceName("Enter a name for the recurring
schedule: ");

            // Prompt the user for the schedule rate (in minutes).
            var scheduleRateInMinutes =
                PromptUserForInteger("Enter the desired schedule rate (in
minutes): ");

            // Create the recurring schedule.
            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"rate({scheduleRateInMinutes} minutes)",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"Recurrent event test from schedule {scheduleName}.");

            Console.WriteLine($"Subscription email will receive an email from
this event.");
            Console.WriteLine($"You must confirm your subscription to receive
event emails.");

            // Delete the schedule when the user is finished.
            if (!_interactive || GetYesNoResponse($"Are you ready to delete the
'{scheduleName}' schedule? (y/n)"))
            {
                await _schedulerWrapper.DeleteScheduleAsync(scheduleName,
_scheduleGroupName);
            }

            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while creating the recurring
schedule.");
            return false;
```

```
        }
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    /// <returns>True if the cleanup was successful.</returns>
    public static async Task<bool> Cleanup()
    {
        // Prompt the user to confirm cleanup.
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete all resources created by this scenario? (y/n)
");
        if (cleanup)
        {
            try
            {
                // Delete the schedule group.
                var groupDeleteSuccess = await
_schedulerWrapper.DeleteScheduleGroupAsync(_scheduleGroupName);

                // Destroy the CloudFormation stack and wait for it to be
removed.
                var stackDeleteSuccess = await
DeleteCloudFormationStack(_stackName, false);

                return groupDeleteSuccess && stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex,
                    "An error occurred while cleaning up the resources.");
                return false;
            }
        }
        _logger.LogInformation("EventBridge Scheduler scenario is complete.");
        return true;
    }

    /// <summary>
    /// Delete the resources in the stack and wait for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
```

```
    /// <returns>True if successful.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
 bool forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{_stackName}' is being deleted.
 This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(_stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{_stackName}' has been
 deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
 '{_stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Wait for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
 forceDelete)
    {
        int retryCount = 0;
```

```
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
describeStacksResponse.Stacks[0].StackStatus == StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus
== StackStatus.DELETE_FAILED)
                {
                    // Try one time to force delete.
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                // Stack does not exist, so it has been successfully deleted.
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }
```

```
    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompt the user for a valid email address.
    /// </summary>
    /// <returns>The valid email address.</returns>
    private static string PromptUserForEmail()
    {
        if (_interactive)
        {
            Console.WriteLine("Enter an email address to use for event
 subscriptions: ");

            string email = Console.ReadLine()!;

            if (!IsValidEmail(email))
            {
                Console.WriteLine("Invalid email address. Please try again.");
                return PromptUserForEmail();
            }
            return email;
        }
        // Used when running without user prompts.
        return "";
    }

    /// <summary>
    /// Prompt the user for a non-empty stack name.
    /// </summary>
    /// <returns>The valid stack name</returns>
```

```
    private static string PromptUserForStackName()
    {
        Console.WriteLine("Enter a name for the AWS Cloud Formation Stack: ");
        if (_interactive)
        {
            string stackName = Console.ReadLine()!;
            var regex = "[a-zA-Z][-a-zA-Z0-9]|arn:[-a-zA-Z0-9:/._+]";
            if (!Regex.IsMatch(stackName, regex))
            {
                Console.WriteLine(
                    $"Invalid stack name. Please use a name that matches the
 pattern {regex}.");
                return PromptUserForStackName();
            }

            return stackName;
        }
        // Used when running without user prompts.
        return _stackName;
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForResourceName(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
            string resourceName = Console.ReadLine()!;
            var regex = "[0-9a-zA-Z-_.]+";
            if (!Regex.IsMatch(resourceName, regex))
            {
                Console.WriteLine($"Invalid resource name. Please use a name that
 matches the pattern {regex}.");
                return PromptUserForResourceName(prompt);
            }
            return resourceName!;
        }
        // Used when running without user prompts.
        return "resource-" + Guid.NewGuid();
    }
```

```
    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static int PromptUserForInteger(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
            string stringResponse = Console.ReadLine()!;
            if (string.IsNullOrWhiteSpace(stringResponse) ||
                !Int32.TryParse(stringResponse, out var intResponse))
            {
                Console.WriteLine($"Invalid integer. ");
                return PromptUserForInteger(prompt);
            }
            return intResponse!;
        }
        // Used when running without user prompts.
        return 1;
    }

    /// <summary>
    /// Use System Mail to check for a valid email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email address.</returns>
    private static bool IsValidEmail(string email)
    {
        try
        {
            var mailAddress = new System.Net.Mail.MailAddress(email);
            return mailAddress.Address == email;
        }
        catch
        {
            // Invalid emails will cause an exception, return false.
            return false;
        }
    }
}
```

Wrapper für Service-Vorgänge.

```
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.Logging;

namespace SchedulerActions;

/// <summary>
/// Wrapper class for Amazon EventBridge Scheduler operations.
/// </summary>
public class SchedulerWrapper
{
    private readonly IAmazonScheduler _amazonScheduler;
    private readonly ILogger<SchedulerWrapper> _logger;

    /// <summary>
    /// Constructor for the SchedulerWrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EventBridge Scheduler
 client.</param>
    /// <param name="logger">The injected logger.</param>
    public SchedulerWrapper(IAmazonScheduler amazonScheduler,
 ILogger<SchedulerWrapper> logger)
    {
        _amazonScheduler = amazonScheduler;
        _logger = logger;
    }

    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines
 when the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which
 the schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the
 schedule after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible
 time window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
```

```
    /// <returns>True if the schedule was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);
```

```
            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}'
" +
                             $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await
_amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
```

```
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name
'{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
```

```
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
```

```
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean
 up all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
 EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
```

```
    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
            Amazon EventBridge Scheduler is a fully managed service that helps
you schedule and execute
            a wide range of tasks and events in the cloud. It's designed to
simplify the process of
            scheduling and managing recurring or one-time events, making it
easier for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its
ability to schedule events
            based on a variety of triggers, including time-based schedules,
custom event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or
to trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
```

```
                    A one-time schedule in Amazon EventBridge Scheduler is an
event trigger that allows
                    you to schedule a one-time event to run at a specific date
and time. This is useful for
                    executing a specific task or workflow at a predetermined
time, without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules
for your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources,
enabling you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
```

```
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part
of that group are deleted.");
        waitForInputToContinue();
        try {

eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
```

```
        }

        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
     * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
     * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
     * in order to receive event emails.
     *
     * @return true if the application preparation was successful, false
otherwise
     */
    public static boolean prepareApplication() {
        logger.info("""
            This example creates resources in a CloudFormation stack, including
an SNS topic
            that will be subscribed to the EventBridge Scheduler events.
            You will need to confirm the subscription in order to receive event
emails.
             """);

        String emailAddress = promptUserForEmail();
        logger.info("You entered {}", emailAddress);

        logger.info("Do you want to use a custom Stack name (y/n) ?");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String newStackName = scanner.nextLine();
            logger.info("You entered {} for the new stack name", newStackName);
            waitForInputToContinue();
            STACK_NAME = newStackName;
        }

        logger.info("Get the roleArn and snsTopicArn values using a
Cloudformation template.");
        waitForInputToContinue();
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
```

```
        roleArn = stackOutputs.get("RoleARN");
        snsTopicArn = stackOutputs.get("SNStopicARN");

        logger.info("The roleARN is {}", roleArn);
        logger.info("The snsTopicArn is {}", snsTopicArn);

        try {
            eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
            logger.info("createScheduleGroupAsync completed successfully.");

        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the
program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
```

```
     * If the provided email address is invalid, the method will prompt the user
to try again.
     *
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
            return promptUserForEmail();
        }
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false}
otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code
false} otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the
one-time schedule:");
```

```
        logger.info("Creating a one-time schedule named {} to send an initial
 event in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) +
 ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
 indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one
 hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
 recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired
 schedule rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + "
 minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
```

```
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
 resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches
 the pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
```

```
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code
 false} otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

Wrapper für Service-Vorgänge.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import
 software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import
 software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import
 software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
```

```
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
 call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            schedulerClient = SchedulerAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return schedulerClient;
    }


    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous
operation of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse>
createScheduleGroup(String name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule
group: " + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

```
   /**
    * Creates a new schedule for a target task.
    *
    * @param name                  the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
           .arn(targetArn)
           .roleArn(roleArn)
           .input(input)
           .build();

       FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
           .mode(useFlexibleTimeWindow
               ? FlexibleTimeWindowMode.FLEXIBLE
               : FlexibleTimeWindowMode.OFF)
           .maximumWindowInMinutes(useFlexibleTimeWindow
```

```
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group
{}, The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while
creating the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception
occurred while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: "
+ ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
```

```
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not
found: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the
schedule, except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
```

```
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
 getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while
 deleting schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.",
 ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SchedulerScenario:
    """
    A scenario that demonstrates how to use Boto3 to schedule and receive events
 using
    the Amazon EventBridge Scheduler.
    """

    def __init__(
        self,
        scheduler_wrapper: SchedulerWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.eventbridge_scheduler = scheduler_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack: ServiceResource = None
        self.schedule_group_name = None
        self.sns_topic_arn = None
        self.role_arn = None

    def run(self) -> None:
        """
        Runs the scenario.
        """

        print(DASHES)
        print("Welcome to the Amazon EventBridge Scheduler Workflow.")
        print(DASHES)

        print(DASHES)
        self.prepare_application()
        print(DASHES)

        print(DASHES)
        self.create_one_time_schedule()
        print(DASHES)

        print(DASHES)
        self.create_recurring_schedule()
        print(DASHES)

        print(DASHES)
        if q.ask(
```

```
            "Do you want to delete all resources created by this workflow? (y/n)
",
            q.is_yesno,
        ):
            self.cleanup()
        print(DASHES)

        print("Amazon EventBridge Scheduler workflow completed.")

    def prepare_application(self) -> None:
        """
        Prepares the application by prompting the user setup information,
deploying a CloudFormation stack and
        creating a schedule group.
        """
        print("Preparing the application...")
        print(
            "\nThis example creates resources in a CloudFormation stack,
including an SNS topic"
            + "\nthat will be subscribed to the EventBridge Scheduler events. "
            + "\n\nYou will need to confirm the subscription in order to receive
event emails. "
        )

        email_address = q.ask("Enter an email address to use for event
subscriptions: ")
        stack_name = q.ask("Enter a name for the AWS Cloud Formation Stack: ")

        template_file = SchedulerScenario.get_template_as_string()

        parameters = [{"ParameterKey": "email", "ParameterValue": email_address}]

        self.stack = self.deploy_cloudformation_stack(
            stack_name, template_file, parameters
        )
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RoleARN":
                self.role_arn = output.get("OutputValue")
            elif output.get("OutputKey") == "SNStopicARN":
                self.sns_topic_arn = output.get("OutputValue")

        if not self.sns_topic_arn or not self.role_arn:
            error_string = f"""
```

```
            Failed to retrieve required outputs from CloudFormation stack.
            'sns_topic_arn'={self.sns_topic_arn}, 'role_arn'={self.role_arn}
            """
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"Stack output RoleARN: {self.role_arn}")
        print(f"Stack output SNStopicARN: a")
        schedule_group_name = "scenario-schedules-group"
        schedule_group_arn = self.eventbridge_scheduler.create_schedule_group(
            schedule_group_name
        )
        print(
            f"Successfully created schedule group '{self.schedule_group_name}':
{schedule_group_arn}."
        )
        self.schedule_group_name = schedule_group_name
        print("Application preparation complete.")

    def create_one_time_schedule(self) -> None:
        """
        Creates a one-time schedule to send an initial event.
        """
        schedule_name = q.ask("Enter a name for the one-time schedule:")

        scheduled_time = datetime.now(timezone.utc) + timedelta(minutes=1)
        formatted_scheduled_time = scheduled_time.strftime("%Y-%m-%dT%H:%M:%S")

        print(
            f"Creating a one-time schedule named '{schedule_name}' "
            + f"\nto send an initial event in 1 minute with a flexible time
window..."
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"at({formatted_scheduled_time})",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"One time scheduled event test from schedule {schedule_name}.",
            delete_after_completion=True,
            use_flexible_time_window=True,
        )
```

```
        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")
        print(f"One-time schedule '{schedule_name}' created successfully.")

    def create_recurring_schedule(self) -> None:
        """
        Create a recurring schedule to send events at a specified rate in
minutes.
        """

        print("Creating a recurring schedule to send events for one hour...")
        schedule_name = q.ask("Enter a name for the recurring schedule: ")
        schedule_rate_in_minutes = q.ask(
            "Enter the desired schedule rate (in minutes): ", q.is_int
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"rate({schedule_rate_in_minutes} minutes)",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"Recurrent event test from schedule {schedule_name}.",
        )

        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")

        if q.ask(
            f"Are you ready to delete the '{schedule_name}' schedule? (y/n)",
q.is_yesno
        ):
            self.eventbridge_scheduler.delete_schedule(
                schedule_name, self.schedule_group_name
            )
```

```
    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str, parameters: [dict[str, str]]
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script
and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :param parameters: The parameters for the CloudFormation stack.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=parameters,
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take
a few minutes."
```

```
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.schedule_group_name:
            schedule_group_name = self.schedule_group_name
            self.schedule_group_name = None
            self.eventbridge_scheduler.delete_schedule_group(schedule_group_name)
            print(f"Successfully deleted schedule group
 '{schedule_group_name}'.")

        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)
        print("Stack deleted, demo complete.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory, "cfn_template.yaml")
        file = open(template_file_path, "r")
        return file.read()


if __name__ == "__main__":
    demo: SchedulerScenario = None
    try:
        scheduler_wrapper = SchedulerWrapper.from_client()
        cloud_formation_resource = resource("cloudformation")
        demo = SchedulerScenario(scheduler_wrapper, cloud_formation_resource)
        demo.run()
```

```
    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup()
```

SchedulerWrapper Klasse, die Amazon EventBridge Scheduler-Aktionen umschließt.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
```

```
        :param schedule_expression: The expression that defines when the schedule
runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM
role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input":
input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
```

```
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]
["Message"]
                )
            raise


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource
 was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]
["Message"]
                )
                raise


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.
```

```
        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Codebeispiele für AWS FIS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Fault Injection Service mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS FIS Benutzerhandbuch — Weitere Informationen zu AWS FIS.
- AWS FIS API-Referenz — Details zu allen verfügbaren AWS FIS Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für AWS FIS verwenden AWS SDKs
  - Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

Szenarien für AWS FIS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS FIS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS FIS oder in Kombination mit anderen AWS-Services

aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Experimentvorlage
- Führen Sie das Experiment durch
- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial
```

```
#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
```

```
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
```

```
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
```

```
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
```

```
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
```

```
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"
```

```
    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
```

```
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
```

```
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done
```

```
if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
```

```
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
```

```
   echo "Deleting experiment template $TEMPLATE_ID..."
   DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
"$TEMPLATE_ID")
   check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

   # Delete CloudWatch alarm
   echo "Deleting CloudWatch alarm $ALARM_NAME..."
   DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
"$ALARM_NAME")
   check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

   # Terminate EC2 instance
   echo "Terminating EC2 instance $INSTANCE_ID..."
   TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
   check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
   echo "Waiting for instance to terminate..."
   aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

   # Clean up IAM resources
   echo "Removing role from instance profile..."
   REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
     --instance-profile-name "$INSTANCE_PROFILE_NAME" \
     --role-name "$EC2_ROLE_NAME")
   check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

   echo "Deleting instance profile..."
   DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
     --instance-profile-name "$INSTANCE_PROFILE_NAME")
   check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

   echo "Deleting FIS role policy..."
   DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
     --role-name "$FIS_ROLE_NAME" \
     --policy-name "$FIS_POLICY_NAME")
   check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

   echo "Detaching policy from EC2 role..."
   DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
     --role-name "$EC2_ROLE_NAME" \
     --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
   check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

   echo "Deleting FIS role..."
   DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
```

```
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages

- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Codebeispiele für Forecast mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Forecast mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Forecast-Benutzerhandbuch – Weitere Informationen zu Forecast.
- API-Referenz für Forecast – Details zu allen verfügbaren Forecast-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Forecast AWS SDKs
  - Aktionen für Forecast unter Verwendung AWS SDKs
    - Verwenden Sie CreateDatasetmit einem AWS SDK
    - Verwenden Sie CreateForecastmit einem AWS SDK
    - Verwenden Sie DeleteDatasetmit einem AWS SDK
    - Verwenden Sie DeleteForecastmit einem AWS SDK
    - Verwenden Sie DescribeForecastmit einem AWS SDK
    - Verwenden Sie ListDatasetGroupsmit einem AWS SDK
    - Verwenden Sie ListForecastsmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Forecast AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Forecast mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Forecast unter Verwendung AWS SDKs
  - Verwenden Sie CreateDatasetmit einem AWS SDK
  - Verwenden Sie CreateForecastmit einem AWS SDK
  - Verwenden Sie DeleteDatasetmit einem AWS SDK
  - Verwenden Sie DeleteForecastmit einem AWS SDK
  - Verwenden Sie DescribeForecastmit einem AWS SDK
  - Verwenden Sie ListDatasetGroupsmit einem AWS SDK
  - Verwenden Sie ListForecastsmit einem AWS SDK

Aktionen für Forecast unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Forecast-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Forecast.

Beispiele

- Verwenden Sie CreateDatasetmit einem AWS SDK
- Verwenden Sie CreateForecastmit einem AWS SDK
- Verwenden Sie DeleteDatasetmit einem AWS SDK
- Verwenden Sie DeleteForecastmit einem AWS SDK
- Verwenden Sie DescribeForecastmit einem AWS SDK
- Verwenden Sie ListDatasetGroupsmit einem AWS SDK
- Verwenden Sie ListForecastsmit einem AWS SDK

Verwenden Sie **`CreateDataset`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateDataset`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name>\s

                Where:
                    name - The name of the data set.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
    }

    public static String createForecastDataSet(ForecastClient forecast, String
 name) {
        try {
            Schema schema = Schema.builder()
                    .attributes(getSchema())
                    .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                    .datasetName(name)
                    .domain("CUSTOM")
                    .datasetType("RELATED_TIME_SERIES")
                    .dataFrequency("D")
                    .schema(schema)
                    .build();
```

```
            CreateDatasetResponse response =
 forecast.createDataset(datasetRequest);
            return response.datasetArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {

        List<SchemaAttribute> schemaList = new ArrayList<>();
        SchemaAttribute att1 = SchemaAttribute.builder()
                .attributeName("item_id")
                .attributeType("string")
                .build();

        SchemaAttribute att2 = SchemaAttribute.builder()
                .attributeName("timestamp")
                .attributeType("timestamp")
                .build();

        SchemaAttribute att3 = SchemaAttribute.builder()
                .attributeName("target_value")
                .attributeType("float")
                .build();

        // Push the SchemaAttribute objects to the List.
        schemaList.add(att1);
        schemaList.add(att2);
        schemaList.add(att3);
        return schemaList;
    }
}
```

- Einzelheiten zur API finden Sie CreateDatasetin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateForecast`** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `CreateForecast`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name> <predictorArn>\s

                Where:
                    name - The name of the forecast.\s
                    predictorArn - The arn of the predictor to use.\s

                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
 String predictorArn) {
        try {
            CreateForecastRequest forecastRequest =
 CreateForecastRequest.builder()
                    .forecastName(name)
                    .predictorArn(predictorArn)
                    .build();

            CreateForecastResponse response =
 forecast.createForecast(forecastRequest);
            return response.forecastArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateForecastin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteDataset`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteDataset`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
```

```
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteDatasetin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteForecast`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteForecast`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
```

```
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteForecastin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeForecast`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeForecast`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <forecastarn>\s

                Where:
                    forecastarn - The arn of the forecast (for example,
 "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                    .forecastArn(forecastarn)
                    .build();

            DescribeForecastResponse response =
 forecast.describeForecast(request);
            System.out.println("The name of the forecast is " +
 response.forecastName());

        } catch (ForecastException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeForecastin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListDatasetGroups`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListDatasetGroups`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                    .maxResults(10)
                    .build();

            ListDatasetGroupsResponse response =
 forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
 myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListDatasetGroupsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListForecasts`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListForecasts`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listAllForeCasts(forecast);
        forecast.close();
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
```

```
                    .maxResults(10)
                    .build();

            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
 forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListForecastsin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon Glacier mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Glacier mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon Glacier Developer Guide — Weitere Informationen zu Amazon Glacier.
- Amazon Glacier API-Referenz — Details zu allen verfügbaren Amazon Glacier-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Glacier AWS SDKs
  - Hallo Amazon Glacier
  - Aktionen für Amazon Glacier mit AWS SDKs
    - Verwenden Sie AddTagsToVaultmit einem AWS SDK oder CLI
    - Verwenden Sie CreateVaultmit einem AWS SDK oder CLI
    - Verwenden Sie mit DeleteArchiveeinem AWS SDK oder CLI
    - Verwenden Sie mit DeleteVaulteinem AWS SDK oder CLI
    - Verwenden Sie mit DeleteVaultNotificationseinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeJobeinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeVaulteinem AWS SDK oder CLI
    - Verwenden Sie GetJobOutputmit einem AWS SDK oder CLI
    - Verwenden Sie mit GetVaultNotificationseinem AWS SDK oder CLI
    - Verwenden Sie mit InitiateJobeinem AWS SDK oder CLI
    - Verwenden Sie mit ListJobseinem AWS SDK oder CLI
    - Verwenden Sie mit ListTagsForVaulteinem AWS SDK oder CLI
    - Verwenden Sie ListVaultsmit einem AWS SDK oder CLI
    - Verwenden Sie mit SetVaultNotificationseinem AWS SDK oder CLI
    - Verwenden Sie mit UploadArchiveeinem AWS SDK oder CLI
    - Verwenden Sie mit UploadMultipartParteinem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Glacier AWS SDKs
  - Archivieren Sie eine Datei in Amazon Glacier, erhalten Sie Benachrichtigungen und initiieren Sie einen Job mit einem AWS SDK
  - Rufen Sie Amazon Glacier-Archivinhalte ab und löschen Sie das Archiv mit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Glacier AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Glacier mit AWS SDKs verwenden können.

Beispiele

- Hallo Amazon Glacier
- Aktionen für Amazon Glacier mit AWS SDKs
  - Verwenden Sie AddTagsToVaultmit einem AWS SDK oder CLI
  - Verwenden Sie CreateVaultmit einem AWS SDK oder CLI
  - Verwenden Sie mit DeleteArchiveeinem AWS SDK oder CLI
  - Verwenden Sie mit DeleteVaulteinem AWS SDK oder CLI
  - Verwenden Sie mit DeleteVaultNotificationseinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeJobeinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeVaulteinem AWS SDK oder CLI
  - Verwenden Sie GetJobOutputmit einem AWS SDK oder CLI
  - Verwenden Sie mit GetVaultNotificationseinem AWS SDK oder CLI
  - Verwenden Sie mit InitiateJobeinem AWS SDK oder CLI
  - Verwenden Sie mit ListJobseinem AWS SDK oder CLI
  - Verwenden Sie mit ListTagsForVaulteinem AWS SDK oder CLI
  - Verwenden Sie ListVaultsmit einem AWS SDK oder CLI
  - Verwenden Sie mit SetVaultNotificationseinem AWS SDK oder CLI
  - Verwenden Sie mit UploadArchiveeinem AWS SDK oder CLI
  - Verwenden Sie mit UploadMultipartParteinem AWS SDK oder CLI

Hallo Amazon Glacier

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon Glacier.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListVaultsin der AWS SDK für .NET API-Referenz.

Aktionen für Amazon Glacier mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Glacier-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Glacier-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Glacier AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Glacier API-Referenz.

Beispiele

- Verwenden Sie AddTagsToVaultmit einem AWS SDK oder CLI
- Verwenden Sie CreateVaultmit einem AWS SDK oder CLI
- Verwenden Sie mit DeleteArchiveeinem AWS SDK oder CLI
- Verwenden Sie mit DeleteVaulteinem AWS SDK oder CLI
- Verwenden Sie mit DeleteVaultNotificationseinem AWS SDK oder CLI
- Verwenden Sie mit DescribeJobeinem AWS SDK oder CLI
- Verwenden Sie mit DescribeVaulteinem AWS SDK oder CLI
- Verwenden Sie GetJobOutputmit einem AWS SDK oder CLI
- Verwenden Sie mit GetVaultNotificationseinem AWS SDK oder CLI
- Verwenden Sie mit InitiateJobeinem AWS SDK oder CLI
- Verwenden Sie mit ListJobseinem AWS SDK oder CLI
- Verwenden Sie mit ListTagsForVaulteinem AWS SDK oder CLI
- Verwenden Sie ListVaultsmit einem AWS SDK oder CLI
- Verwenden Sie mit SetVaultNotificationseinem AWS SDK oder CLI
- Verwenden Sie mit UploadArchiveeinem AWS SDK oder CLI
- Verwenden Sie mit UploadMultipartParteinem AWS SDK oder CLI

Verwenden Sie **`AddTagsToVault`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AddTagsToVault` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Add tags to the items in an Amazon S3 Glacier vault.
/// </summary>
```

```
    /// <param name="vaultName">The name of the vault to add tags to.</param>
    /// <param name="key">The name of the object to tag.</param>
    /// <param name="value">The tag value to add.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddTagsToVaultAsync(string vaultName, string key,
 string value)
    {
        var request = new AddTagsToVaultRequest
        {
            Tags = new Dictionary<string, string>
                {
                    { key, value },
                },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.AddTagsToVaultAsync(request);
        return response.HttpStatusCode == HttpStatusCode.NoContent;
    }
```

- Einzelheiten zur API finden Sie AddTagsToVaultin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl fügt zwei Tags zu einem Tresor mit dem Namen `my-vault` hinzu:

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie AddTagsToVaultin der AWS CLI Befehlsreferenz.

Verwenden Sie **`CreateVault`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVault` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Create an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to create.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateVaultAsync(string vaultName)
{
    var request = new CreateVaultRequest
    {
        // Setting the AccountId to "-" means that
        // the account associated with the current
        // account will be used.
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.CreateVaultAsync(request);

    Console.WriteLine($"Created {vaultName} at: {response.Location}");

    return response.HttpStatusCode == HttpStatusCode.Created;
}
```

- Einzelheiten zur API finden Sie CreateVaultin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt einen neuen Tresor mit dem Namen `my-vault`:

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie CreateVaultin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateVaultin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Erstellen des Tresors.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateVaultin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateVaultin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt einen neuen Tresor für das Benutzerkonto. Da für den AccountId Parameter - kein Wert angegeben wurde, verwenden die Cmdlets den Standardwert „-“, der das aktuelle Konto angibt.

```
New-GLCVault -VaultName myvault
```

Ausgabe:

```
/01234567812/vaults/myvault
```

- Einzelheiten zur API finden Sie unter CreateVault AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt einen neuen Tresor für das Benutzerkonto. Da für den AccountId Parameter - kein Wert angegeben wurde, verwenden die Cmdlets den Standardwert „-“, der das aktuelle Konto angibt.

```
New-GLCVault -VaultName myvault
```

Ausgabe:

```
/01234567812/vaults/myvault
```

- Einzelheiten zur API finden Sie unter CreateVault AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource
```

```
    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

- Einzelheiten zur API finden Sie CreateVaultin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteArchive`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteArchive` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Abrufen von Archivinhalten und Löschen des Archivs

CLI

AWS CLI

So löschen Sie ein Archiv aus einem Tresor

Im folgenden Beispiel für `delete-archive` wird das angegebene Archiv aus `example_vault` entfernt.

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
```

```
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteArchivein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
```

```
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String
 vaultName, String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteArchivein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
 archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- Einzelheiten zur API finden Sie DeleteArchivein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteVault`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVault` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Abrufen von Archivinhalten und Löschen des Archivs

CLI

AWS CLI

Der folgende Befehl löscht einen Tresor mit dem Namen `my-vault`:

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

Dieser Befehl erzeugt keine Ausgabe. Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie DeleteVaultin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteVaultin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

- Einzelheiten zur API finden Sie DeleteVaultin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteVaultNotifications`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVaultNotifications` verwendet wird.

CLI

AWS CLI

So entfernen Sie die SNS-Benachrichtigungen für einen Tresor

Im folgenden Beispiel für `delete-vault-notifications` werden Benachrichtigungen, die von Amazon Simple Notification Service (Amazon SNS) gesendet wurden, für den angegebenen Tresor entfernt.

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteVaultNotificationsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
```

```
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- Einzelheiten zur API finden Sie DeleteVaultNotificationsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeJob`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeJob` verwendet wird.

CLI

AWS CLI

Der folgende Befehl ruft Informationen über einen Auftrag zum Abrufen von Inventar in einem Tresor mit dem Namen `my-vault` ab:

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW
```

Ausgabe:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
    "Completed": false,
    "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "Action": "InventoryRetrieval",
    "CreationDate": "2015-07-17T20:23:41.616Z",
    "StatusCode": "InProgress"
}
```

Die Auftrags-ID finden Sie in der Ausgabe von `aws glacier initiate-job` und `aws glacier list-jobs`. Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie DescribeJobin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details des angegebenen Auftrags zurück. Wenn der Job erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Jobs (ein Archiv oder eine Inventarliste) in das lokale Dateisystem abzurufen.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Ausgabe:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
```

```
JobDescription                :
JobId                         : op1x...JSbthM
JobOutputPath                 :
OutputLocation                :
RetrievalByteRange            : 0-38034479
SelectParameters              :
SHA256TreeHash                : 79f3ea754c02f58...dc57bf4395b
SNSTopic                      :
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Einzelheiten zur API finden Sie unter DescribeJob AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details des angegebenen Auftrags zurück. Wenn der Job erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Jobs (ein Archiv oder eine Inventarliste) in das lokale Dateisystem abzurufen.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Ausgabe:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
```

```
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- Einzelheiten zur API finden Sie unter DescribeJob AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
```

```
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
            return job.status_code
```

- Einzelheiten zur API finden Sie DescribeJobin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeVault`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeVault` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to describe.</param>
    /// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
    public async Task<string> DescribeVaultAsync(string vaultName)
    {
        var request = new DescribeVaultRequest
        {
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.DescribeVaultAsync(request);
```

```
        // Display the information about the vault.
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber
of Archives: {response.NumberOfArchives}\tSize (in bytes):
{response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

- Einzelheiten zur API finden Sie DescribeVaultin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl ruft Daten über einen Tresor mit dem Namen `my-vault` ab:

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie DescribeVaultin der AWS CLI Befehlsreferenz.

Verwenden Sie **`GetJobOutput`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetJobOutput` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Abrufen von Archivinhalten und Löschen des Archivs

CLI

AWS CLI

Der folgende Befehl speichert die Ausgabe eines Tresor-Inventarauftrags in einer Datei im aktuellen Verzeichnis mit dem Namen `output.json`:

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

Die `job-id` ist in der Ausgabe von `aws glacier list-jobs` verfügbar. Beachten Sie, dass der Name der Ausgabedatei ein Positionsargument ist, dem kein Optionsname vorangestellt ist. Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

Ausgabe:

```
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/json"
}
```

`output.json`:

```
{"VaultARN":"arn:aws:glacier:us-west-2:0123456789012:vaults/
my-vault","InventoryDate":"2015-04-07T00:26:18Z","ArchiveList":
[{"ArchiveId":"kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw","ArchiveDescription":"multipart
 upload
 test","CreationDate":"2015-04-06T22:24:34Z","Size":3145728,"SHA256TreeHash":"9628195fcdb
```

- Einzelheiten zur API finden Sie GetJobOutputin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Lädt den Archivinhalt herunter, dessen Abruf im angegebenen Auftrag geplant war, und speichert den Inhalt in einer Datei auf der Festplatte. Der Download überprüft für Sie die Prüfsumme, sofern eine verfügbar ist. Auf Wunsch kann die gesamte Antwort einschließlich der Prüfsumme durch Angabe von **`-Select '*'`** zurückgegeben werden.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- Einzelheiten zur API finden Sie unter GetJobOutput AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Lädt den Archivinhalt herunter, dessen Abruf im angegebenen Auftrag geplant war, und speichert den Inhalt in einer Datei auf der Festplatte. Der Download überprüft für Sie die Prüfsumme, sofern eine verfügbar ist. Auf Wunsch kann die gesamte Antwort einschließlich der Prüfsumme durch Angabe von **`-Select '*'`** zurückgegeben werden.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- Einzelheiten zur API finden Sie unter GetJobOutput AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

- Einzelheiten zur API finden Sie GetJobOutputin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetVaultNotifications`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetVaultNotifications` verwendet wird.

CLI

AWS CLI

Der folgende Befehl ruft eine Beschreibung der Benachrichtigungskonfiguration für einen Tresor mit dem Namen `my-vault` ab:

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

Ausgabe:

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

Wenn keine Benachrichtigungen für den Tresor konfiguriert wurden, wird ein Fehler zurückgegeben. Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie GetVaultNotificationsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.",
vault.name)
            raise
        else:
            return notification
```

- Einzelheiten zur API finden Sie GetVaultNotificationsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`InitiateJob`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `InitiateJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Archiv aus einem Tresor ab. In diesem Beispiel wird die ArchiveTransferManager Klasse verwendet. Einzelheiten zur API finden Sie unter ArchiveTransferManager.

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</
param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string
 vaultName, string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then
 polling SQS queue for the archive to be available.");
```

```
            Console.WriteLine("When the archive is available, downloading will
 begin.");
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
 options);

            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
    }
```

- Einzelheiten zur API finden Sie InitiateJobunter AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird ein Auftrag zum Abrufen eines Bestands des Tresors `my-vault` eingeleitet:

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
 '{"Type": "inventory-retrieval"}'
```

Ausgabe:

```
{
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

Mit dem folgenden Befehl wird ein Auftrag zum Abrufen eines Archivs des Tresors `my-vault` eingeleitet:

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

`job-archive-retrieval.json` ist eine JSON-Datei im lokalen Ordner, die den Auftragstyp, die Archiv-ID und einige optionale Parameter angibt:

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
  "Description": "Retrieve archive on 2015-07-17",
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-topic"
}
```

Archiv-IDs sind in der Ausgabe von `aws glacier upload-archive` und `aws glacier get-job-output` verfügbar.

Ausgabe:

```
{
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
```

```
}
```

Details zum Format der Auftragsparameter finden Sie unter „Einleiten eines Auftrags“ in der API-Referenz zu Amazon Glacier.

- Einzelheiten zur API finden Sie InitiateJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Tresorinventur ab.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName,
 String accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
```

```
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
response.location());
            return response.jobId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }
```

```
            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();

            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier
 vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

- Einzelheiten zur API finden Sie InitiateJobin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Startet einen Auftrag zum Abrufen eines Archivs aus dem angegebenen Tresor, dessen Eigentümer der Benutzer ist. Der Status des Jobs kann mit dem Get-GLCJob Cmdlet überprüft werden. Wenn der Job erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Archivs in das lokale Dateisystem abzurufen.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Ausgabe:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- Einzelheiten zur API finden Sie unter InitiateJob AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Startet einen Auftrag zum Abrufen eines Archivs aus dem angegebenen Tresor, dessen Eigentümer der Benutzer ist. Der Status des Jobs kann mit dem Get-GLCJob Cmdlet überprüft werden. Wenn der Job erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Archivs in das lokale Dateisystem abzurufen.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Ausgabe:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- Einzelheiten zur API finden Sie unter InitiateJob AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Tresorinventur ab.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the
 contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
        When the job completes, you can get the inventory by calling
 get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
            raise
        else:
            return job
```

Rufen Sie ein Archiv aus einem Tresor ab.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
```

```
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3-5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job
```

- Einzelheiten zur API finden Sie InitiateJobin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListJobs`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListJobs` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen
- Abrufen von Archivinhalten und Löschen des Archivs

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List Amazon S3 Glacier jobs.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list jobs for.</param>
    /// <returns>A list of Amazon S3 Glacier jobs.</returns>
    public async Task<List<GlacierJobDescription>> ListJobsAsync(string
vaultName)
    {
        var request = new ListJobsRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the current account.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListJobsAsync(request);

        return response.JobList;
    }
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl listet laufende und kürzlich abgeschlossene Aufträge für einen Tresor mit dem Namen `my-vault` auf:

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

Ausgabe:

```
{
    "JobList": [
        {
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "RetrievalByteRange": "0-3145727",
            "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
            "Completed": false,
            "SHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "JobId": "l7IL5-EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
            "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
            "JobDescription": "Retrieve archive on 2015-07-17",
            "ArchiveSizeInBytes": 3145728,
            "Action": "ArchiveRetrieval",
            "ArchiveSHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "CreationDate": "2015-07-17T21:16:13.840Z",
            "StatusCode": "InProgress"
        },
        {
            "InventoryRetrievalParameters": {
                "Format": "JSON"
            },
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "Completed": false,
            "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
```

```
            "Action": "InventoryRetrieval",
            "CreationDate": "2015-07-17T20:23:41.616Z",
            "StatusCode": ""InProgress""
        }
    ]
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie ListJobsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
```

```
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list
```

- Einzelheiten zur API finden Sie ListJobsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListTagsForVault`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForVault` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
 vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

- Einzelheiten zur API finden Sie ListTagsForVaultin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl listet die Tags auf, die auf einen Tresor mit dem Namen `my-vault` angewendet wurden:

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

Ausgabe:

```
{
    "Tags": {
        "date": "july2015",
```

```
        "id": "1234"
    }
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie ListTagsForVaultin der AWS CLI Befehlsreferenz.

Verwenden Sie **`ListVaults`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListVaults` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the Amazon S3 Glacier vaults associated with the current account.
    /// </summary>
    /// <returns>A list containing information about each vault.</returns>
    public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
    {
        var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });
        var vaultList = new List<DescribeVaultOutput>();

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
```

```
            vaultList.Add(vault);
        }

        return vaultList;
    }
```

- Einzelheiten zur API finden Sie ListVaultsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl listet die Tresore im Standardkonto und der Standardregion auf:

```
aws glacier list-vaults --account-id -
```

Ausgabe:

```
{
    "VaultList": [
        {
            "SizeInBytes": 3178496,
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "LastInventoryDate": "2015-04-07T00:26:19.028Z",
            "VaultName": "my-vault",
            "NumberOfArchives": 1,
            "CreationDate": "2015-04-06T21:23:45.708Z"
        }
    ]
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie ListVaultsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
```

```
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListVaultsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- Einzelheiten zur API finden Sie ListVaultsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`SetVaultNotifications`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetVaultNotifications` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

CLI

AWS CLI

Der folgende Befehl konfiguriert SNS-Benachrichtigungen für einen Tresor mit dem Namen `my-vault`:

```
aws glacier set-vault-notifications --account-id - --vault-name my-vault --vault-notification-config file://notificationconfig.json
```

`notificationconfig.json` ist eine JSON-Datei im aktuellen Ordner, die ein SNS-Thema und die zu veröffentlichenden Ereignisse angibt:

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

- Einzelheiten zur API finden Sie SetVaultNotificationsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
```

```
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
vault.name
            )
            raise
        else:
            return notification
```

- Einzelheiten zur API finden Sie SetVaultNotificationsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UploadArchive`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UploadArchive` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Upload an object to an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
   /// the archive to.</param>
   /// <param name="archiveFilePath">The file path of the archive to upload to
the vault.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
string archiveFilePath)
   {
       try
       {
           var manager = new ArchiveTransferManager(_glacierService);

           // Upload an archive.
           var response = await manager.UploadAsync(vaultName, "upload archive
test", archiveFilePath);
           return response.ArchiveId;
```

```
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return string.Empty;
        }
    }
```

- Einzelheiten zur API finden Sie UploadArchivein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl lädt ein Archiv im aktuellen Ordner mit dem Namen `archive.zip` in einen Tresor mit dem Namen `my-vault` hoch:

```
aws glacier upload-archive --account-id - --vault-name my-vault --
body archive.zip
```

Ausgabe:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

Um ein hochgeladenes Archiv abzurufen, initiieren Sie einen Abrufauftrag mit dem Befehl aws glacier initiate-job.

- Einzelheiten zur API finden Sie UploadArchivein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s
```

```
                Where:
                   strPath - The path to the archive to upload (for example, C:\
\AWS\\test.pdf).
                   vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest,
 path);
            return res.archiveId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
```

```
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s",
inputFile, ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws
IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
```

```
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
```

```
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
```

```
}
```

- Einzelheiten zur API finden Sie UploadArchivein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Hochladen des Archivs.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
```

```
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UploadArchivein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function (err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UploadArchivein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Lädt eine einzelne Datei in den angegebenen Tresor hoch und gibt die Archiv-ID und die berechnete Prüfsumme zurück.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Beispiel 2: Lädt den Inhalt einer Ordnerhierarchie in den angegebenen Tresor im Benutzerkonto hoch. Für jede hochgeladene Datei gibt das Cmdlet den Dateinamen, die entsprechende Archiv-ID und die berechnete Prüfsumme des Archivs aus.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Einzelheiten zur API finden Sie unter UploadArchive AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Lädt eine einzelne Datei in den angegebenen Tresor hoch und gibt die Archiv-ID und die berechnete Prüfsumme zurück.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Beispiel 2: Lädt den Inhalt einer Ordnerhierarchie in den angegebenen Tresor im Benutzerkonto hoch. Für jede hochgeladene Datei gibt das Cmdlet den Dateinamen, die entsprechende Archiv-ID und die berechnete Prüfsumme des Archivs aus.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- Einzelheiten zur API finden Sie unter UploadArchive AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

- Einzelheiten zur API finden Sie UploadArchivein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UploadMultipartPart`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UploadMultipartPart` verwendet wird.

CLI

AWS CLI

Der folgende Befehl lädt den ersten, 1 MiB (1024 x 1024 Byte) umfassenden Teil eines Archivs hoch:

```
aws glacier upload-multipart-part --body part1 --range 'bytes
 0-1048575/*' --account-id - --vault-name my-vault --upload-
id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

Amazon Glacier benötigt bei der Durchführung von Operationen ein Konto-ID-Argument, Sie können jedoch einen Bindestrich verwenden, um das verwendete Konto anzugeben.

Der Parameter „body“ nimmt einen Pfad zu einer Teildatei im lokalen Dateisystem an. Der Parameter „range“ nimmt einen HTTP-Inhaltsbereich an, der die Byte angibt, die der Teil im fertigen Archiv belegt. Die Upload-ID wird vom Befehl `aws glacier initiate-multipart-upload` zurückgegeben und kann auch mithilfe von `aws glacier list-multipart-uploads` abgerufen werden.

Weitere Informationen zu mehrteiligen Uploads auf Amazon Glacier mithilfe der AWS CLI finden Sie unter Verwenden von Amazon Glacier im AWS CLI-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UploadMultipartPartin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines mehrteiligen Uploads mit 1-Megabyte-Blöcken eines Buffer-Objekts.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
```

```
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
      };

    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };
```

```
      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UploadMultipartPartin der AWS SDK für JavaScript API-Referenz.

Szenarien für die Verwendung von Amazon Glacier AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Glacier mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon Glacier aufrufen oder mit anderen kombinieren AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Archivieren Sie eine Datei in Amazon Glacier, erhalten Sie Benachrichtigungen und initiieren Sie einen Job mit einem AWS SDK
- Rufen Sie Amazon Glacier-Archivinhalte ab und löschen Sie das Archiv mit einem AWS SDK

Archivieren Sie eine Datei in Amazon Glacier, erhalten Sie Benachrichtigungen und initiieren Sie einen Job mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon Glacier-Tresor.
- Konfigurieren des Tresors, um Benachrichtigungen in einem Amazon-SNS-Thema zu veröffentlichen.
- Hochladen einer Archivdatei in den Tresor
- Initiieren eines Auftrags zum Abrufen eines Archivs

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon Glacier-Operationen umschließt.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource
```

```
    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
```

```
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3-5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
```

```
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
```

```
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
 vault.name
            )
            raise
        else:
            return notification
```

Rufen Sie Funktionen in der Wrapper-Klasse auf, um einen Tresor zu erstellen und eine Datei hochzuladen. Konfigurieren Sie dann den Tresor so, dass er Benachrichtigungen veröffentlicht, und initiieren Sie einen Auftrag zum Abrufen des Archivs.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification
 of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
```

```
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS
topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateVault
  - InitiateJob

- ListJobs
- ListVaults
- SetVaultNotifications
- UploadArchive

Rufen Sie Amazon Glacier-Archivinhalte ab und löschen Sie das Archiv mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Jobs für einen Amazon Glacier-Tresor auflisten und den Auftragsstatus abrufen.
- Abrufen der Ausgabe eines abgeschlossenen Auftrags zum Archivabruf
- Löschen eines Archivs
- Löschen eines Tresors

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon Glacier-Operationen umschließt.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    @staticmethod
```

```
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise


    @staticmethod
    def delete_vault(vault):
```

```
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

Rufen Sie Funktionen der Wrapper-Klasse auf, um Archivinhalte aus einem abgeschlossenen Auftrag abzurufen, und löschen Sie dann das Archiv.

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this
 script "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
```

```
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again
later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

Codebeispiele für AWS Glue verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Glue mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS Glue Entwicklerhandbuch — Weitere Informationen zu AWS Glue.
- AWS Glue API-Referenz — Details zu allen verfügbaren AWS Glue Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Glue verwenden AWS SDKs
  - Hallo AWS Glue
  - Lernen Sie die Grundlagen von AWS Glue mit einem AWS SDK
  - Aktionen für AWS Glue verwenden AWS SDKs
    - Verwenden Sie mit CreateCrawlereinem AWS SDK
    - Verwenden Sie CreateJobmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteCrawlermit einem AWS SDK
    - Verwenden Sie DeleteDatabasemit einem AWS SDK
    - Verwenden Sie DeleteJobmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteTablemit einem AWS SDK
    - Verwenden Sie GetCrawlermit einem AWS SDK

    - Verwenden Sie GetDatabasemit einem AWS SDK
    - Verwenden Sie GetDatabasesmit einem AWS SDK oder CLI
    - Verwenden Sie GetJobmit einem AWS SDK oder CLI
    - Verwenden Sie GetJobRunmit einem AWS SDK oder CLI
    - Verwenden Sie GetJobRunsmit einem AWS SDK oder CLI
    - Verwenden Sie GetTablesmit einem AWS SDK oder CLI
    - Verwenden Sie ListJobsmit einem AWS SDK
    - Verwenden Sie StartCrawlermit einem AWS SDK oder CLI
    - Verwenden Sie StartJobRunmit einem AWS SDK oder CLI
- Szenarien für AWS Glue verwenden AWS SDKs
  - Erste Schritte mit dem Datenkatalog

Grundlegende Beispiele für AWS Glue verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Glue mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS Glue
- Lernen Sie die Grundlagen von AWS Glue mit einem AWS SDK
- Aktionen für AWS Glue verwenden AWS SDKs
  - Verwenden Sie mit CreateCrawlereinem AWS SDK
  - Verwenden Sie CreateJobmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteCrawlermit einem AWS SDK
  - Verwenden Sie DeleteDatabasemit einem AWS SDK
  - Verwenden Sie DeleteJobmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTablemit einem AWS SDK
  - Verwenden Sie GetCrawlermit einem AWS SDK
  - Verwenden Sie GetDatabasemit einem AWS SDK
  - Verwenden Sie GetDatabasesmit einem AWS SDK oder CLI
  - Verwenden Sie GetJobmit einem AWS SDK oder CLI
  - Verwenden Sie GetJobRunmit einem AWS SDK oder CLI

- Verwenden Sie GetJobRunsmit einem AWS SDK oder CLI
- Verwenden Sie GetTablesmit einem AWS SDK oder CLI
- Verwenden Sie ListJobsmit einem AWS SDK
- Verwenden Sie StartCrawlermit einem AWS SDK oder CLI
- Verwenden Sie StartJobRunmit einem AWS SDK oder CLI

Hallo AWS Glue

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS Glue beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
```

```
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListJobsunter AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError
```

```
def hello_glue():
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for
 Python (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")

        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- Einzelheiten zur API finden Sie ListJobsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
let mut list_jobs = glue.list_jobs().into_paginator().send();
while let Some(list_jobs_output) = list_jobs.next().await {
    match list_jobs_output {
        Ok(list_jobs) => {
            let names = list_jobs.job_names();
            info!(?names, "Found these jobs")
        }
        Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
    }
}
```

- Einzelheiten zur API finden Sie ListJobsin der API-Referenz zum AWS SDK für Rust.

Lernen Sie die Grundlagen von AWS Glue mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3 S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

.NET

SDK für .NET

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die AWS Glue Funktionen umschließt, die im Szenario verwendet werden.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
 executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
 (Amazon S3)
```

```
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
```

```
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
```

```
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
```

```
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }


    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }
```

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
 { JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
```

```
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
```

```
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
```

```
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
```

```
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;
```

```
        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
```

```
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on
{database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();
```

```
        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState ==
"STOPPED" ||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState ==
"TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);
```

```
        jobRuns.ForEach(jobRun =>
        {

 Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}


namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");
```

```
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the
URL to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
```

```
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String
 &bucketName,
                                                      const Aws::String &roleName,
                                                      const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
```

```
                                        clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
```

```
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
```

```
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                            << ". After " << iterations
                            << " seconds elapsed."
                            << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                            << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }
```

```
    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 6. Get tables.
    Aws::String tableName;
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
```

```
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);
```

```
        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);
```

```
                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                          jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
```

```
                                << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome =
s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
                continuationToken =
outcome.GetResult().GetNextContinuationToken();
            }
            else {
```

```
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }

        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
 first ") +
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; +
+i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
```

```
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
```

```
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
```

```
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
 jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
```

```
                                const Aws::Client::ClientConfiguration
&clientConfig) {
   const Aws::Glue::GlueClient client(clientConfig);
   bool result = true;

   // 13. Delete a job.
   if (!job.empty()) {
       Aws::Glue::Model::DeleteJobRequest request;
       request.SetJobName(job);

       Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the job." << std::endl;
       }
       else {
           std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }

   // 14. Delete a database.
   if (!database.empty()) {
       Aws::Glue::Model::DeleteDatabaseRequest request;
       request.SetName(database);

       Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
               request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the database." << std::endl;
       }
       else {
           std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }

   // 15. Delete a crawler.
```

```
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
```

```
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
```

```
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
                                    object.GetKey()));
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." <<
std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
```

```
                            << std::endl;
            }

            continuationToken =
 listObjectsOutcome.GetResult().GetNextContinuationToken();
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." <<
 std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
```

```
    }
    else {
        std::cerr << "Error retrieving object. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) throws InterruptedException {
       final String usage = """

           Usage:
               <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
<scriptLocation> <locationUri> <bucketNameSc>\s

           Where:
               iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
               s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, s3://<bucket name>/read).
               cron - A cron expression used to specify the schedule  (i.e.,
cron(15 12 * * ? *).
               dbName - The database name.\s
               crawlerName - The name of the crawler.\s
               jobName - The name you assign to this job definition.
               scriptLocation - The Amazon S3 path to a script that runs a job.
               locationUri - The location of the database (you can find this
file in resources folder).
               bucketNameSc - The Amazon S3 bucket name used when creating a job
               """;

       if (args.length != 9) {
           System.out.println(usage);
           return;
       }
       Scanner scanner = new Scanner(System.in);
       String iam = args[0];
       String s3Path = args[1];
       String cron = args[2];
       String dbName = args[3];
       String crawlerName = args[4];
       String jobName = args[5];
       String scriptLocation = args[6];
       String locationUri = args[7];
       String bucketNameSc = args[8];

       Region region = Region.US_EAST_1;
       GlueClient glueClient = GlueClient.builder()
           .region(region)
```

```
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL)
service provided by Amazon
            Web Services (AWS). It is designed to simplify the process of
building, running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known
as the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName,
crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
```

```
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
    public static void createDatabase(GlueClient glueClient, String dbName,
 String locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
                .build();

            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                .databaseInput(input)
                .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            throw e;
```

```
        }
    }


    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the
data source
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();
```

```
            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue
service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
```

```
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact
with the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from
the AWS Glue service
     */
    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                .name(databaseName)
                .build();

            GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                .withLocale(Locale.US)
                .withZone(ZoneId.systemDefault());
```

```
            formatter.format(createDate);
            System.out.println("The create date of the database is " +
createDate);

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Retrieves the names of the tables in the specified Glue database.
     *
     * @param glueClient the Glue client to use for the operation
     * @param dbName     the name of the Glue database to retrieve the table
names from
     * @return the name of the first table retrieved, or an empty string if no
tables were found
     */
    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                .databaseName(dbName)
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }
```

```
    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
```

```
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data
catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue
service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
```

```
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;
```

```
                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
```

```
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und starten Sie einen Crawler, der einen öffentlichen Amazon Simple Storage Service (Amazon S3) -Bucket crawlt und eine Metadaten-Datenbank generiert, die die gefundenen CSV-formatted Daten beschreibt.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });
```

```
  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};
```

```
/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Listet Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog

```
const getDatabase = (name) => {
  const client = new GlueClient({});
```

```
  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

/**
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Erstellen und führen Sie einen Job aus, der CSV-Daten aus dem Amazon S3 S3-Quell-Bucket extrahiert, sie durch Entfernen und Umbenennen von Feldern transformiert und die JSON-formatted Ausgabe in einen anderen Amazon S3 S3-Bucket lädt.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
```

```
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
    case "ERROR":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
    case "SUCCEEDED":
      return;
    default:
      break;
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
```

```
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }}
 context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Listet Informationen über Auftragsausführungen auf und zeigt einige der transformierten Daten an.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput>} getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunsCommandOutput} getJobRuns
 */

/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};
```

```
/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

Löscht alle Ressourcen, die von der Demo erstellt wurden.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
```

```
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
   * @type {{ selectedJobNames: string[] }}
   */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
```

```
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
```

```
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
```

```
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

const cleanUpCrawlerStep = async (context) => {
  log("Deleting crawler.");

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log("Crawler is already deleted.");
    } else {
```

```
      throw err;
    }
  }

  return { ...context };
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service
 (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
```

```
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}
```

```
suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
```

```
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}
```

```
suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
```

```
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }
```

```
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;
```

```
        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
```

```
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
```

```
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
```

```
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path):
 Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
```

```
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

```
    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
```

```
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob

- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die die im Szenario verwendeten AWS Glue Funktionen umschließt.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
```

```
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return response["Database"]


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
```

```
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
```

```
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
```

```
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.
```

```
        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
 and use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM)
 role
                                  that AWS Glue can assume to gain access to the
```

```
                                resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output
data
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an
animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the
AWS Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise
```

```
    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a
command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by
the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during
job
                           runs.
        :param job_name: The name to give the job definition that is created
during the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source}
and "
```

```
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
```

```
        print(
            f"When you run the job, it extracts data from {data_source},
transforms it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only
a few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the
first {lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
```

```
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list
of runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
```

```
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
            f"Let's clean up. During this example we created job definition
 '{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these
 tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
```

```
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must
 grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
```

```
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket
 "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Erstellen Sie ein ETL-Skript, das AWS Glue zum Extrahieren, Transformieren und Laden von Daten während Jobausführungen verwendet wird.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
```

```
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
```

```
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die die im Szenario verwendeten AWS Glue Funktionen umschließt.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
```

```
 # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
 # @param db_name [String] The name of the database where the crawler stores its
metadata.
 # @param db_prefix [String] The prefix to be added to the names of tables that
the crawler creates.
 # @param s3_target [String] The S3 path that the crawler will crawl.
 # @return [void]
 def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
   @glue_client.create_crawler(
     name: name,
     role: role_arn,
     database_name: db_name,
     targets: {
       s3_targets: [
         {
           path: s3_target
         }
       ]
     }
   )
 rescue Aws::Glue::Errors::GlueException => e
   @logger.error("Glue could not create crawler: \n#{e.message}")
   raise
 end

 # Starts a crawler with the specified name.
 #
 # @param name [String] The name of the crawler to start.
 # @return [void]
 def start_crawler(name)
   @glue_client.start_crawler(name: name)
 rescue Aws::Glue::Errors::ServiceError => e
   @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
   raise
 end

 # Deletes a crawler with the specified name.
 #
 # @param name [String] The name of the crawler to delete.
 # @return [void]
 def delete_crawler(name)
   @glue_client.delete_crawler(name: name)
 rescue Aws::Glue::Errors::ServiceError => e
   @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
```

```
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
```

```
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
```

```
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
```

```
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end
```

```
  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
 db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
 t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
```

```
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]
['name'], @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end

def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'

  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-
east-1'), iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

Erstellen Sie ein ETL-Skript, das AWS Glue zum Extrahieren, Transformieren und Laden von Daten während Jobausführungen verwendet wird.

```
import sys
from awsglue.transforms import *
```

```
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
```

```
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun

- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und starten Sie einen Crawler, der einen öffentlichen Amazon Simple Storage Service (Amazon S3) -Bucket crawlt und eine Metadaten-Datenbank generiert, die die gefundenen CSV-formatted Daten beschreibt.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
```

```
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler =
 glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

Listet Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
 database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        let tables = tables.table_list();
```

Erstellen und führen Sie einen Job aus, der CSV-Daten aus dem Amazon S3 S3-Quell-Bucket extrahiert, sie durch Entfernen und Umbenennen von Feldern transformiert und die JSON-formatted Ausgabe in einen anderen Amazon S3 S3-Bucket lädt.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
```

```
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

Löscht alle Ressourcen, die von der Demo erstellt wurden.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.

- CreateCrawler
- CreateJob
- DeleteCrawler
- DeleteDatabase
- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription
```

```
let package = Package(
    name: "glue-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "glue-scenario",
            dependencies: [
                .product(name: "AWSGlue", package: "aws-sdk-swift"),
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to demonstrate
// creating and using crawlers and jobs using AWS Glue.
//
// 0. Upload the Python job script to Amazon S3 so it can be used when
```

```
//    calling `startJobRun()` later.
// 1. Create a crawler, pass it the IAM role and the URL of the public Amazon
//    S3 bucket that contains the source data:
//    s3://crawler-public-us-east-1/flight/2016/csv.
// 2. Start the crawler. This takes time, so after starting it, use a loop
//    that calls `getCrawler()` until the state is "READY".
// 3. Get the database created by the crawler, and the tables in the
//    database. Display them to the user.
// 4. Create a job. Pass it the IAM role and the URL to a Python ETL script
//    previously uploaded to the user's S3 bucket.
// 5. Start a job run, passing the following custom arguments. These are
//    expected by the ETL script, so must exactly match.
//    * `--input_database: <name of the database created by the crawler>`
//    * `--input_table: <name of the table created by the crawler>`
//    * `--output_bucket_url: <URL to the scaffold bucket created for the
//      user>`
// 6. Loop and get the job run until it returns one of the following states:
//    "SUCCEEDED", "STOPPED", "FAILED", or "TIMEOUT".
// 7. Output data is stored in a group of files in the user's S3 bucket.
//    Either direct the user to their location or download a file and display
//    the results inline.
// 8. List the jobs for the user's account.
// 9. Get job run details for a job run.
// 10. Delete the demo job.
// 11. Delete the database and tables created by the example.
// 12. Delete the crawler created by the example.

import ArgumentParser
import AWSS3
import Foundation
import Smithy

import AWSClientRuntime
import AWSGlue

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS IAM role to use for AWS Glue calls.")
    var role: String

    @Option(help: "The Amazon S3 bucket to use for this example.")
    var bucket: String

    @Option(help: "The Amazon S3 URL of the data to crawl.")
    var s3url: String = "s3://crawler-public-us-east-1/flight/2016/csv"
```

```
    @Option(help: "The Python script to run as a job with AWS Glue.")
    var script: String = "./flight_etl_job_script.py"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(help: "A prefix string to use when naming tables.")
    var tablePrefix = "swift-glue-basics-table"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "glue-scenario",
        abstract: """
        Demonstrates various features of AWS Glue.
        """,
        discussion: """
        An example showing how to use AWS Glue to create, run, and monitor
        crawlers and jobs.
        """
    )

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
```

```
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Upload a file to an Amazon S3 bucket.
    ///
    /// - Parameters:
    ///   - s3Client: The S3 client to use when uploading the file.
    ///   - path: The local path of the source file to upload.
    ///   - toBucket: The name of the S3 bucket into which to upload the file.
    ///   - key: The key (name) to give the file in the S3 bucket.
    ///
    /// - Returns: `true` if the file is uploaded successfully, otherwise
`false`.
    func uploadFile(s3Client: S3Client, path: String, toBucket: String, key:
String) async -> Bool {
        do {
            let fileData: Data = try Data(contentsOf: URL(fileURLWithPath: path))
            let dataStream = ByteStream.data(fileData)
            _ = try await s3Client.putObject(
                input: PutObjectInput(
                    body: dataStream,
                    bucket: toBucket,
                    key: key
                )
            )
        } catch {
            print("*** An unexpected error occurred uploading the script to the
Amazon S3 bucket \"\(bucket)\".")
            return false
        }

        return true
    }

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
```

```
   ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
   ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
   ///   - databaseName: The name of an AWS Glue database to operate on.
   ///
   /// - Returns: `true` if the crawler is created successfully, otherwise
`false`.
   func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
String,
                      s3Path: String, cronSchedule: String, databaseName:
String) async -> Bool {
       let s3Target = GlueClientTypes.S3Target(path: s3url)
       let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

       do {
           _ = try await glueClient.createCrawler(
               input: CreateCrawlerInput(
                   databaseName: databaseName,
                   description: "Created by the AWS SDK for Swift Scenario
Example for AWS Glue.",
                   name: crawlerName,
                   role: iamRole,
                   schedule: cronSchedule,
                   tablePrefix: tablePrefix,
                   targets: targetList
               )
           )
       } catch _ as AlreadyExistsException {
           print("*** A crawler named \"\(crawlerName)\" already exists.")
           return false
       } catch _ as OperationTimeoutException {
           print("*** The attempt to create the AWS Glue crawler timed out.")
           return false
       } catch {
           print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
           return false
       }

       return true
   }

   /// Delete an AWS Glue crawler.
   ///
   /// - Parameters:
```

```
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
`false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
```

```
   func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
       do {
           let output = try await glueClient.getCrawler(
               input: GetCrawlerInput(name: name)
           )

           // If the crawler or its state is `nil`, report that the crawler
           // is stopping. This may not be what you want for your
           // application but it works for this one!

           guard let crawler = output.crawler else {
               return GlueClientTypes.CrawlerState.stopping
           }
           guard let state = crawler.state else {
               return GlueClientTypes.CrawlerState.stopping
           }
           return state
       } catch {
           return GlueClientTypes.CrawlerState.stopping
       }
   }

   /// Wait until the specified crawler is ready to run.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - name: The name of the crawler to wait for.
   ///
   /// - Returns: `true` if the crawler is ready, `false` if the client is
   ///   stopping (and will therefore never be ready).
   func waitUntilCrawlerReady(glueClient: GlueClient, name: String) async ->
Bool {
       while true {
           let state = await getCrawlerState(glueClient: glueClient, name: name)

           if state == .ready {
               return true
           } else if state == .stopping {
               return false
           }

           // Wait four seconds before trying again.
```

```
            do {
                try await Task.sleep(for: .seconds(4))
            } catch {
                print("*** Error pausing the task.")
            }
        }
    }

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

```
/// Return a list of the AWS Glue jobs listed on the user's account.
///
/// - Parameters:
///   - glueClient: The AWS Glue client to use.
///   - maxJobs: The maximum number of jobs to return (default: 100).
///
/// - Returns: An array of strings listing the names of all available AWS
///   Glue jobs.
func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
    var jobList: [String] = []
    var nextToken: String?

    repeat {
        do {
            let output = try await glueClient.listJobs(
                input: ListJobsInput(
                    maxResults: maxJobs,
                    nextToken: nextToken
                )
            )

            guard let jobs = output.jobNames else {
                return jobList
            }

            jobList = jobList + jobs
            nextToken = output.nextToken
        } catch {
            return jobList
        }
    } while (nextToken != nil)

    return jobList
}

/// Delete an AWS Glue job.
///
/// - Parameters:
///   - glueClient: The AWS Glue client to use.
///   - jobName: The name of the job to delete.
///
/// - Returns: `true` if the job is successfully deleted, otherwise `false`.
func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
    do {
```

```
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }

    /// Create an AWS Glue database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name to give the new database.
    ///   - location: The URL of the source data to use with AWS Glue.
    ///
    /// - Returns: `true` if the database is created successfully, otherwise
`false`.
    func createDatabase(glueClient: GlueClient, name databaseName: String,
location: String) async -> Bool {
        let databaseInput = GlueClientTypes.DatabaseInput(
            description: "Created by the AWS SDK for Swift Glue basic scenario
example.",
            locationUri: location,
            name: databaseName
        )

        do {
            _ = try await glueClient.createDatabase(
                input: CreateDatabaseInput(
                    databaseInput: databaseInput
                )
            )
        } catch {
            return false
        }

        return true
    }

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
```

```
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
-> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
```

```
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.
```

```
            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
```

```
   ///   - glueClient: The AWS Glue client to use.
   ///   - jobName: The name of the job to run.
   ///   - databaseName: The name of the AWS Glue database to run the job
against.
   ///   - tableName: The name of the table in the database to run the job
against.
   ///   - outputURL: The AWS S3 URI of the bucket location into which to
   ///     write the resulting output.
   ///
   /// - Returns: `true` if the job run is started successfully, otherwise
`false`.
   func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                     tableName: String, outputURL: String) async -> String? {
       do {
           let output = try await glueClient.startJobRun(
               input: StartJobRunInput(
                   arguments: [
                       "--input_database": databaseName,
                       "--input_table": tableName,
                       "--output_bucket_url": outputURL
                   ],
                   jobName: jobName,
                   numberOfWorkers: 10,
                   workerType: .g1x
               )
           )

           guard let id = output.jobRunId else {
               return nil
           }

           return id
       } catch {
           return nil
       }
   }

   /// Return a list of the job runs for the specified job.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - jobName: The name of the job for which to return its job runs.
   ///   - maxResults: The maximum number of job runs to return (default:
```

```
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults:
 Int? = nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
```

```
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // A name to give the Python script upon upload to the Amazon S3
        // bucket.
        let scriptName = "jobscript.py"

        // Schedule string in `cron` format, as described here:
        // https://docs.aws.amazon.com/glue/latest/dg/monitor-data-warehouse-
schedule.html
        let cron = "cron(15 12 * * ? *)"

        let glueConfig = try await GlueClient.GlueClientConfiguration(region:
 awsRegion)
        let glueClient = GlueClient(config: glueConfig)

        let s3Config = try await S3Client.S3ClientConfiguration(region:
 awsRegion)
        let s3Client = S3Client(config: s3Config)

        // Create random names for things that need them.

        let crawlerName = tempName(prefix: "swift-glue-basics-crawler")
        let databaseName = tempName(prefix: "swift-glue-basics-db")

        // Create a name for the AWS Glue job.

        let jobName = tempName(prefix: "scenario-job")

        // The URL of the Python script on S3.

        let scriptURL = "s3://\(bucket)/\(scriptName)"

        print("Welcome to the AWS SDK for Swift basic scenario for AWS Glue!")
```

```
        //=====================================================================
        // 0. Upload the Python script to the target bucket so it's available
        //    for use by the Amazon Glue service.
        //=====================================================================

        print("Uploading the Python script: \(script) as key \(scriptName)")
        print("Destination bucket: \(bucket)")
        if !(await uploadFile(s3Client: s3Client, path: script, toBucket: bucket,
key: scriptName)) {
            return
        }

        //=====================================================================
        // 1. Create the database and crawler using the randomized names
        //    generated previously.
        //=====================================================================

        print("Creating database \"\(databaseName)\"...")
        if !(await createDatabase(glueClient: glueClient, name: databaseName,
location: s3url)) {
            print("*** Unable to create the database.")
            return
        }

        print("Creating crawler \"\(crawlerName)\"...")
        if !(await createCrawler(glueClient: glueClient, crawlerName:
crawlerName,
                                 iamRole: role, s3Path: s3url, cronSchedule:
cron,
                                 databaseName: databaseName)) {
            return
        }

        //=====================================================================
        // 2. Start the crawler, then wait for it to be ready.
        //=====================================================================

        print("Starting the crawler and waiting until it's ready...")
        if !(await startCrawler(glueClient: glueClient, name: crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }
```

```
        if !(await waitUntilCrawlerReady(glueClient: glueClient, name:
crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
        }

        //=====================================================================
        // 3. Get the database and table created by the crawler.
        //=====================================================================

        print("Getting the crawler's database...")
        let database = await getDatabase(glueClient: glueClient, name:
databaseName)

        guard let database else {
            print("*** Unable to get the database.")
            return
        }
        print("Database URI: \(database.locationUri ?? "<unknown>")")

        let tableList = await getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)

        print("Found \(tableList.count) table(s):")
        for table in tableList {
            print("  \(table.name ?? "<unnamed>")")
        }

        if tableList.count != 1 {
            print("*** Incorrect number of tables found. There should only be
one.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        guard let tableName = tableList[0].name else {
            print("*** Table is unnamed.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }
```

```
        //=====================================================================
        // 4. Create a job.
        //=====================================================================

        print("Creating a job...")
        if !(await createJob(glueClient: glueClient, name: jobName, role: role,
                             scriptLocation: scriptURL)) {
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 5. Start a job run.
        //=====================================================================

        print("Starting the job...")

        // Construct the Amazon S3 URL for the job run's output. This is in
        // the bucket specified on the command line, with a folder name that's
        // unique for this job run.

        let timeStamp = Date().timeIntervalSince1970
        let jobPath = "\(jobName)-\(Int(timeStamp))"
        let outputURL = "s3://\(bucket)/\(jobPath)"

        // Start the job run.

        let jobRunID = await startJobRun(glueClient: glueClient, name: jobName,
                                         databaseName: databaseName,
                                         tableName: tableName,
                                         outputURL: outputURL)

        guard let jobRunID else {
            print("*** Job run ID is invalid.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
```

```
        // 6. Wait for the job run to indicate that the run is complete.
        //=====================================================================

        print("Waiting for job run to end...")

        var jobRunFinished = false
        var jobRunState: GlueClientTypes.JobRunState

        repeat {
            let jobRun = await getJobRun(glueClient: glueClient, name: jobName,
 id: jobRunID)
            guard let jobRun else {
                print("*** Unable to get the job run.")
                _ = await deleteJob(glueClient: glueClient, name: jobName)
                _ = await deleteDatabase(glueClient: glueClient, name:
 databaseName, withTables: true)
                _ = await deleteCrawler(glueClient: glueClient, name:
 crawlerName)
                return
            }
            jobRunState = jobRun.jobRunState ?? .failed

            //
=====================================================================
            // 7. Output where to find the data if the job run was successful.
            //    If the job run failed for any reason, output an appropriate
            //    error message.
            //
=====================================================================

            switch jobRunState {
                case .succeeded:
                    print("Job run succeeded. JSON files are in the Amazon S3
 path:")
                    print("    \(outputURL)")
                    jobRunFinished = true
                case .stopped:
                    jobRunFinished = true
                case .error:
                    print("*** Error: Job run ended in an error.
 \(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .failed:
```

```
                    print("*** Error: Job run failed. \(jobRun.errorMessage ??
 "")")
                    jobRunFinished = true
                case .timeout:
                    print("*** Warning: Job run timed out.")
                    jobRunFinished = true
                default:
                    do {
                        try await Task.sleep(for: .milliseconds(250))
                    } catch {
                        print("*** Error pausing the task.")
                    }
            }
        } while jobRunFinished != true

        //=====================================================================
        // 7.5. List the job runs for this job, showing each job run's ID and
        // its execution time.
        //=====================================================================

        print("Getting all job runs for the job \(jobName):")
        let jobRuns = await getJobRuns(glueClient: glueClient, name: jobName)

        if jobRuns.count == 0 {
            print("    <no job runs found>")
        } else {
            print("Found \(jobRuns.count) job runs... listing execution times:")
            for jobRun in jobRuns {
                print("    \(jobRun.id ?? "<unnamed>"): \(jobRun.executionTime)
 seconds")
            }
        }

        //=====================================================================
        // 8. List the jobs for the user's account.
        //=====================================================================

        print("\nThe account has the following jobs:")
        let jobs = await listJobs(glueClient: glueClient)

        if jobs.count == 0 {
            print("    <no jobs found>")
        } else {
            for job in jobs {
```

```
                print("    \(job)")
            }
        }

        //=====================================================================
        // 9. Get the job run details for a job run.
        //=====================================================================

        print("Information about the job run:")
        let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)

        guard let jobRun else {
            print("*** Unable to retrieve the job run.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        let startDate = jobRun.startedOn ?? Date(timeIntervalSince1970: 0)
        let endDate = jobRun.completedOn ?? Date(timeIntervalSince1970: 0)
        let dateFormatter: DateFormatter = DateFormatter()
        dateFormatter.dateStyle = .long
        dateFormatter.timeStyle = .long

        print("    Started at: \(dateFormatter.string(from: startDate))")
        print("  Completed at: \(dateFormatter.string(from: endDate))")

        //=====================================================================
        // 10. Delete the job.
        //=====================================================================

        print("\nDeleting the job...")
        _ = await deleteJob(glueClient: glueClient, name: jobName)

        //=====================================================================
        // 11. Delete the database and tables created by this example.
        //=====================================================================

        print("Deleting the database...")
        _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
```

```
        //=====================================================================
        // 12. Delete the crawler.
        //=====================================================================

        print("Deleting the crawler...")
        if !(await deleteCrawler(glueClient: glueClient, name: crawlerName)) {
            return
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases

- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen für AWS Glue verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Glue Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS Glue API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS Glue verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Glue -API-Referenz.

Beispiele

- Verwenden Sie mit CreateCrawlereinem AWS SDK
- Verwenden Sie CreateJobmit einem AWS SDK oder CLI
- Verwenden Sie DeleteCrawlermit einem AWS SDK
- Verwenden Sie DeleteDatabasemit einem AWS SDK
- Verwenden Sie DeleteJobmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTablemit einem AWS SDK
- Verwenden Sie GetCrawlermit einem AWS SDK
- Verwenden Sie GetDatabasemit einem AWS SDK
- Verwenden Sie GetDatabasesmit einem AWS SDK oder CLI
- Verwenden Sie GetJobmit einem AWS SDK oder CLI
- Verwenden Sie GetJobRunmit einem AWS SDK oder CLI

- Verwenden Sie GetJobRunsmit einem AWS SDK oder CLI
- Verwenden Sie GetTablesmit einem AWS SDK oder CLI
- Verwenden Sie ListJobsmit einem AWS SDK
- Verwenden Sie StartCrawlermit einem AWS SDK oder CLI
- Verwenden Sie StartJobRunmit einem AWS SDK oder CLI

Verwenden Sie mit **`CreateCrawler`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateCrawler` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
```

```
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::S3Target s3Target;
       s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
       Aws::Glue::Model::CrawlerTargets crawlerTargets;
       crawlerTargets.AddS3Targets(s3Target);

       Aws::Glue::Model::CreateCrawlerRequest request;
       request.SetTargets(crawlerTargets);
       request.SetName(CRAWLER_NAME);
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
       request.SetRole(roleArn);

       Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully created the crawler." << std::endl;
       }
       else {
           std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
           return false;
```

```
        }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the
data source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler
will store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                        String iam,
                                        String s3Path,
                                        String cron,
                                        String dbName,
                                        String crawlerName) {

       try {
           S3Target s3Target = S3Target.builder()
               .path(s3Path)
               .build();

           List<S3Target> targetList = new ArrayList<>();
```

```
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
```

```
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }
```

```
    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Einzelheiten zur API finden Sie CreateCrawlerin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
 $databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path):
 Result
    {
```

```
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
```

```
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateCrawlerin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
```

```
        }
      ]
    }
  )
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not create crawler: \n#{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    let create_crawler = glue
        .create_crawler()
        .name(self.crawler())
        .database_name(self.database())
        .role(self.iam_role.expose_secret())
        .targets(
            CrawlerTargets::builder()
                .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                .build(),
        )
        .send()
        .await;

    match create_crawler {
        Err(err) => {
            let glue_err: aws_sdk_glue::Error = err.into();
            match glue_err {
                aws_sdk_glue::Error::AlreadyExistsException(_) => {
                    info!("Using existing crawler");
                    Ok(())
```

```
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- Einzelheiten zur API finden Sie CreateCrawlerin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_database_name = 'my-database'
        " iv_table_prefix = 'test_'
        " iv_s3_target = 's3://example-bucket/data/'

        DATA(lt_s3_targets) = VALUE /aws1/cl_glus3target=>tt_s3targetlist(
          ( NEW /aws1/cl_glus3target( iv_path = iv_s3_target ) ) ).

        DATA(lo_targets) = NEW /aws1/cl_glucrawlertargets(
          it_s3targets = lt_s3_targets ).

        lo_glu->createcrawler(
          iv_name = iv_crawler_name
          iv_role = iv_role_arn
          iv_databasename = iv_database_name
          iv_tableprefix = iv_table_prefix
          io_targets = lo_targets ).
        MESSAGE 'Crawler created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Crawler already exists.' TYPE 'E'.
```

```
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateCrawlerin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise
 `false`.
```

```
   func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole:
String,
                      s3Path: String, cronSchedule: String, databaseName:
String) async -> Bool {
       let s3Target = GlueClientTypes.S3Target(path: s3url)
       let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

       do {
           _ = try await glueClient.createCrawler(
               input: CreateCrawlerInput(
                   databaseName: databaseName,
                   description: "Created by the AWS SDK for Swift Scenario
Example for AWS Glue.",
                   name: crawlerName,
                   role: iamRole,
                   schedule: cronSchedule,
                   tablePrefix: tablePrefix,
                   targets: targetList
               )
           )
       } catch _ as AlreadyExistsException {
           print("*** A crawler named \"\(crawlerName)\" already exists.")
           return false
       } catch _ as OperationTimeoutException {
           print("*** The attempt to create the AWS Glue crawler timed out.")
           return false
       } catch {
           print("*** An unexpected error occurred creating the AWS Glue
crawler: \(error.localizedDescription)")
           return false
       }

       return true
   }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Create an AWS Glue job.
   /// </summary>
   /// <param name="jobName">The name of the job.</param>
   /// <param name="roleName">The name of the IAM role to be assumed by
   /// the job.</param>
   /// <param name="description">A description of the job.</param>
   /// <param name="scriptUrl">The URL to the script.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
   {
       var command = new JobCommand
       {
           PythonVersion = "3",
           Name = "glueetl",
           ScriptLocation = scriptUrl,
       };

       var arguments = new Dictionary<string, string>
       {
           { "--input_database", dbName },
           { "--input_table", tableName },
           { "--output_bucket_url", bucketUrl }
       };
```

```
        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);
```

```
        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Einen Auftrag zur Datentransformation erstellen

Im folgenden Beispiel für `create-job` wird ein Streaming-Job erstellt, der ein in S3 gespeichertes Skript ausführt.

```
aws glue create-job \
    --name my-testing-job \
    --role AWSGlueServiceRoleDefault \
    --command '{ \
        "Name": "gluestreaming", \
        "ScriptLocation": "s3://amzn-s3-demo-bucket/folder/" \
    }' \
    --region us-east-1 \
    --output json \
```

```
    --default-arguments '{ \
        "--job-language":"scala", \
        "--class":"GlueApp" \
    }' \
    --profile my-profile \
    --endpoint https://glue.us-east-1.amazonaws.com
```

Inhalt von `test_script.scala`:

```
import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
    def main(sysArgs: Array[String]) {
        val spark: SparkContext = new SparkContext()
        val glueContext: GlueContext = new GlueContext(spark)
        // @params: [JOB_NAME]
        val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME").toArray)
        Job.init(args("JOB_NAME"), glueContext, args.asJava)
        // @type: DataSource
        // @args: [database = "tempdb", table_name = "s3-source",
 transformation_ctx = "datasource0"]
        // @return: datasource0
        // @inputs: []
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
 tableName = "s3-source", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
 ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
 "status", "string")], transformation_ctx = "applymapping1"]
        // @return: applymapping1
        // @inputs: [frame = datasource0]
```

```
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
 "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
 "int"), ("status", "string", "status", "string")), caseSensitive = false,
 transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
 transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
 "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name =
 "my-s3-sink", transformation_ctx = "resolvechoice3"]
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
 Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
 Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
        // @args: [database = "tempdb", table_name = "my-s3-sink",
 transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
 tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
 "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}
```

Ausgabe:

```
{
    "Name": "my-testing-job"
}
```

Weitere Informationen finden Sie unter Authoring Jobs in AWS Glue im AWS Glue Developer Guide.

- Einzelheiten zur API finden Sie CreateJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new AWS Glue job.
    *
    * @param glueClient     the AWS Glue client to use for the operation
    * @param jobName        the name of the job to create
    * @param iam            the IAM role to associate with the job
    * @param scriptLocation the location of the script to be used by the job
    * @throws GlueException if there is an error creating the job
    */
   public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
       try {
           JobCommand command = JobCommand.builder()
               .pythonVersion("3")
               .name("glueetl")
               .scriptLocation(scriptLocation)
               .build();

           CreateJobRequest jobRequest = CreateJobRequest.builder()
               .description("A Job created by using the AWS SDK for Java V2")
               .glueVersion("2.0")
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .name(jobName)
               .role(iam)
               .command(command)
               .build();

           glueClient.createJob(jobRequest);
           System.out.println(jobName + " was successfully created.");
```

```
        } catch (GlueException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel erstellt einen neuen Job in AWS Glue. Der Wert des Befehlsnamens ist immer**`glueetl`**. AWS Glue unterstützt das Ausführen von Jobskripten, die in Python oder Scala geschrieben wurden. In diesem Beispiel ist das Job-Skript (MyTestGlueJob.py) in Python geschrieben. Python-Parameter werden in der **`$DefArgs`** Variablen angegeben und dann an den PowerShell Befehl im **`DefaultArguments`** Parameter übergeben, der eine Hashtabelle akzeptiert. Die Parameter in der **`$JobParams`** Variablen stammen aus der CreateJob API, die im Thema Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) der AWS Glue-API-Referenz dokumentiert ist.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
```

```
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Einzelheiten zur API finden Sie unter CreateJob AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel erstellt einen neuen Job in AWS Glue. Der Wert des Befehlsnamens ist immer**`glueetl`**. AWS Glue unterstützt das Ausführen von Jobskripten, die in Python oder Scala geschrieben wurden. In diesem Beispiel ist das Job-Skript (MyTestGlueJob.py) in Python geschrieben. Python-Parameter werden in der **`$DefArgs`** Variablen angegeben und dann an den PowerShell Befehl im **`DefaultArguments`** Parameter übergeben, der eine Hashtabelle akzeptiert. Die Parameter in der **`$JobParams`** Variablen stammen aus der CreateJob API, die im Thema Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) der AWS Glue-API-Referenz dokumentiert ist.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
```

```
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Einzelheiten zur API finden Sie unter CreateJob AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
 that can
```

```
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateJobin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
```

```
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;
```

- Einzelheiten zur API finden Sie CreateJobin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_description = 'ETL job for data transformation'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_script_location = 's3://example-bucket/scripts/my-script.py'

        DATA(lo_command) = NEW /aws1/cl_glujobcommand(
          iv_name = 'glueetl'
          iv_scriptlocation = iv_script_location
          iv_pythonversion = '3' ).

        lo_glu->createjob(
          iv_name = iv_job_name
          iv_description = iv_description
          iv_role = iv_role_arn
          io_command = lo_command
          iv_glueversion = '3.0' ).
        MESSAGE 'Job created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Job already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateJobin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
```

```
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

- Einzelheiten zur API finden Sie CreateJobin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteCrawler`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteCrawler` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
```

```
        }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client object
    * @param crawlerName the name of the crawler to be deleted
    * @throws GlueException if an error occurs during the deletion process
    */
   public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
       try {
           DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           glueClient.deleteCrawler(deleteCrawlerRequest);
           System.out.println(crawlerName + " was deleted");

       } catch (GlueException e) {
           throw e;
       }
   }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
```

```
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteCrawlerin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        lo_glu->deletecrawler( iv_name = iv_crawler_name ).
        MESSAGE 'Crawler deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_glucrawlerrunningex.
        MESSAGE 'Crawler is currently running.' TYPE 'E'.
```

```
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluschdrtransingex.
        MESSAGE 'Scheduler is transitioning.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
```

```
        return true
    }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteDatabase`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDatabase` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem Datenkatalog

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact
with the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
```

```
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteDatabasein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Einzelheiten zur API finden Sie DeleteDatabasein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_database_name = 'my-database'
        lo_glu->deletedatabase( iv_name = iv_database_name ).
        MESSAGE 'Database deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
```

```
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDatabasein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
```

```
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient:
 glueClient, databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.

            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
```

```
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
```

```
            std::cerr << "Error deleting the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Einen Auftrag löschen

Das folgende Beispiel für `delete-job` löscht einen Auftrag, der nicht mehr benötigt wird.

```
aws glue delete-job \
    --job-name my-testing-job
```

Ausgabe:

```
{
    "JobName": "my-testing-job"
}
```

Weitere Informationen finden Sie unter Arbeiten mit Jobs auf der AWS Glue-Konsole im AWS Glue-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });
```

```
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteJobin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Einzelheiten zur API finden Sie DeleteJobin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        lo_glu->deletejob( iv_jobname = iv_job_name ).
        MESSAGE 'Job deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
```

```
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteJobin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }
```

- Einzelheiten zur API finden Sie DeleteJobin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteTable` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem Datenkatalog

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für .NET API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
```

```
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
```

```
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteTablein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
```

```
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        " iv_database_name = 'my-database'
        " iv_table_name = 'my-table'
        lo_glu->deletetable(
          iv_databasename = iv_database_name
          iv_name = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Table or database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
```

```
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetCrawler`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetCrawler` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information about an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Crawler object describing the crawler.</returns>
public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new GetCrawlerRequest
    {
        Name = crawlerName,
    };
```

```
        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
```

```
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                            crawlerState)
                    << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to
be in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue
service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();
```

```
            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response =
 glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Einzelheiten zur API finden Sie GetCrawlerin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Waiting for crawler";
        do {
```

```
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.
```

```
        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler
```

- Einzelheiten zur API finden Sie GetCrawlerin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Einzelheiten zur API finden Sie GetCrawlerin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        oo_result = lo_glu->getcrawler( iv_name = iv_crawler_name ).
        DATA(lo_crawler) = oo_result->get_crawler( ).
        MESSAGE 'Crawler information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'I'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetCrawlerin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
```

```
import AWSGlue

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }
```

- Einzelheiten zur API finden Sie GetCrawlerin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetDatabase`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetDatabase` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem Datenkatalog

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetDatabaseRequest request;
       request.SetName(CRAWLER_DATABASE_NAME);

       Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

       if (outcome.IsSuccess()) {
           const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

           std::cout << "Successfully retrieve the database\n" <<
                     database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
       }
       else {
           std::cerr << "Error getting the database.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact
with the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from
the AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " +
createDate);

       } catch (GlueException e) {
           throw e;
```

```
        }
    }
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Einzelheiten zur API finden Sie GetDatabasein der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- Einzelheiten zur API finden Sie GetDatabasein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- Einzelheiten zur API finden Sie GetDatabasein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_database_name = 'my-database'
```

```
        oo_result = lo_glu->getdatabase( iv_name = iv_database_name ).
        DATA(lo_database) = oo_result->get_database( ).
        MESSAGE 'Database information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetDatabasein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
```

```
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie GetDatabasein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetDatabases`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDatabases` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Um die Definitionen einiger oder aller Datenbanken im AWS Glue-Datenkatalog aufzulisten

Das folgende Beispiel für `get-databases` gibt Informationen über die Datenbanken im Datenkatalog zurück.

```
aws glue get-databases
```

Ausgabe:

```
{
    "DatabaseList": [
        {
```

```
            "Name": "default",
            "Description": "Default Hive database",
            "LocationUri": "file:/spark-warehouse",
            "CreateTime": 1602084052.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "flights-db",
            "CreateTime": 1587072847.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "legislators",
            "CreateTime": 1601415625.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
```

```
            "CatalogId": "111122223333"
        },
        {
            "Name": "tempdb",
            "CreateTime": 1601498566.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        }
    ]
}
```

Weitere Informationen finden Sie unter Definieren einer Datenbank in Ihrem Datenkatalog im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie GetDatabasesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetDatabasesin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Informationen zu einem Auftrag abrufen

Das folgende Beispiel für `get-job` ruft Informationen zu einem Auftrag ab.

```
aws glue get-job \
    --job-name my-testing-job
```

Ausgabe:

```
{
    "Job": {
        "Name": "my-testing-job",
        "Role": "Glue_DefaultRole",
        "CreatedOn": 1602805698.167,
        "LastModifiedOn": 1602805698.167,
        "ExecutionProperty": {
            "MaxConcurrentRuns": 1
        },
        "Command": {
            "Name": "gluestreaming",
            "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
            "PythonVersion": "2"
        },
        "DefaultArguments": {
```

```
            "--class": "GlueApp",
            "--job-language": "scala"
        },
        "MaxRetries": 0,
        "AllocatedCapacity": 10,
        "MaxCapacity": 10.0,
        "GlueVersion": "1.0"
    }
}
```

Weitere Informationen finden Sie unter Aufträge im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie GetJobin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetJobRun`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetJobRun` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Informationen zu einer Auftragsausführung abrufen

Das folgende Beispiel für `get-job-run` ruft Informationen zu einer Auftragsausführung ab.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

Ausgabe:

```
{
    "JobRun": {
        "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
        "Attempt": 0,
        "JobName": "Combine legistators data",
        "StartedOn": 1602873931.255,
        "LastModifiedOn": 1602874075.985,
        "CompletedOn": 1602874075.985,
        "JobRunState": "SUCCEEDED",
        "Arguments": {
            "--enable-continuous-cloudwatch-log": "true",
            "--enable-metrics": "",
            "--enable-spark-ui": "true",
            "--job-bookmark-option": "job-bookmark-enable",
            "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
        },
        "PredecessorRuns": [],
        "AllocatedCapacity": 10,
        "ExecutionTime": 117,
        "Timeout": 2880,
        "MaxCapacity": 10.0,
        "WorkerType": "G.1X",
        "NumberOfWorkers": 10,
        "LogGroupName": "/aws-glue/jobs",
        "GlueVersion": "2.0"
    }
}
```

Weitere Informationen finden Sie unter Auftragsausführungen im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie GetJobRunin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;
```

```
        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
 ['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
 Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- Einzelheiten zur API finden Sie GetJobRunin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
```

```
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };

        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- Einzelheiten zur API finden Sie GetJobRunin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_run_id = 'jr_abcd1234567890abcdef1234567890abcdef12345678'
        oo_result = lo_glu->getjobrun(
          iv_jobname = iv_job_name
          iv_runid = iv_run_id ).
```

```
        DATA(lo_job_run) = oo_result->get_jobrun( ).
        MESSAGE 'Job run information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Job or job run does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetJobRunin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
```

```
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String)
 async -> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie GetJobRunin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetJobRuns`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetJobRuns` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
```

```
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Informationen über alle Ausführungen eines Auftrags abrufen

Das folgende Beispiel für `get-job-runs` ruft Informationen zu allen Ausführungen eines Auftrags ab.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

Ausgabe:

```
{
    "JobRuns": [
        {
            "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
            "Attempt": 0,
            "JobName": "my-testing-job",
            "StartedOn": 1602873931.255,
            "LastModifiedOn": 1602874075.985,
            "CompletedOn": 1602874075.985,
            "JobRunState": "SUCCEEDED",
            "Arguments": {
                "--enable-continuous-cloudwatch-log": "true",
                "--enable-metrics": "",
                "--enable-spark-ui": "true",
                "--job-bookmark-option": "job-bookmark-enable",
                "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
            },
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 117,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
            "Attempt": 2,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
```

```
            "JobName": "my-testing-job",
            "StartedOn": 1602811168.496,
            "LastModifiedOn": 1602811282.39,
            "CompletedOn": 1602811282.39,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 021AAB703DB20A2D;
                S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuGOqfByajkfL/Tlqt5JBGdEGpigAqzdMDM/U=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 110,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "Attempt": 1,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
            "JobName": "my-testing-job",
            "StartedOn": 1602811020.518,
            "LastModifiedOn": 1602811138.364,
            "CompletedOn": 1602811138.364,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                 Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                 Request ID: 2671D37856AE7ABB;
                 S3 Extended Request ID: RLJCJw20brV
+PpC6GpORahyF2fp9flB5SSb2bTGPnUSPVizLXRl1PN3QZldb+v1o9qRVktNYbW8=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 113,
            "Timeout": 2880,
```

```
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        }
    ]
}
```

Weitere Informationen finden Sie unter Auftragsausführungen im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie GetJobRunsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the
job runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
```

```
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
 Result
    {
        $arguments = ['JobName' => $jobName];
```

```
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- Einzelheiten zur API finden Sie GetJobRunsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
```

```
def get_job_runs(job_name)
  response = @glue_client.get_job_runs(job_name: job_name)
  response.job_runs
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get job runs: \n#{e.message}")
end
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_job_name = 'my-etl-job'
    oo_result = lo_glu->getjobruns( iv_jobname = iv_job_name ).
    DATA(lt_job_runs) = oo_result->get_jobruns( ).
    MESSAGE 'Job runs retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Job does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetJobRunsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults:
 Int? = nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
```

```
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }
```

- Einzelheiten zur API finden Sie GetJobRunsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetTables`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetTables` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
```

```
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetTablesRequest request;
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       std::vector<Aws::Glue::Model::Table> all_tables;
       Aws::String nextToken; // Used for pagination.
       do {
           Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

           if (outcome.IsSuccess()) {
               const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
```

```
                all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Die Definitionen einiger oder aller Tabellen in der angegebenen Datenbank auflisten

Das folgende Beispiel für `get-tables` gibt Informationen zu den Tabellen in der angegebenen Datenbank zurück.

```
aws glue get-tables --database-name 'tempdb'
```

Ausgabe:

```
{
    "TableList": [
        {
            "Name": "my-s3-sink",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730539.0,
            "UpdateTime": 1602730539.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/test-s3-output/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
```

```
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "s3-source",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730658.0,
            "UpdateTime": 1602730658.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "test-kinesis-input",
            "DatabaseName": "tempdb",
            "CreateTime": 1601507001.0,
            "UpdateTime": 1601507001.0,
            "Retention": 0,
            "StorageDescriptor": {
```

```
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "my-testing-stream",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "Parameters": {
                    "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
                    "streamName": "my-testing-stream",
                    "typeOfData": "kinesis"
                },
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        }
    ]
}
```

Weitere Informationen finden Sie unter Definieren von Tabellen im AWS Glue-Datenkatalog im AWS Glue-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetTablesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table
names from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
           List<Table> tables = response.tableList();
           if (tables.isEmpty()) {
               System.out.println("No tables were returned");
           } else {
               for (Table table : tables) {
                   myTableName = table.name();
                   System.out.println("Table name is: " + myTableName);
               }
           }

       } catch (GlueException e) {
           throw e;
       }
       return myTableName;
```

```
    }
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $databaseName = "doc-example-database-$uniqid";
```

```
        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- Einzelheiten zur API finden Sie GetTablesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
```

```
def get_tables(db_name)
  response = @glue_client.get_tables(database_name: db_name)
  response.table_list
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    let tables = glue
        .get_tables()
        .database_name(self.database())
        .send()
        .await
        .map_err(GlueMvpError::from_glue_sdk)?;

    let tables = tables.table_list();
```

- Einzelheiten zur API finden Sie GetTablesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_database_name = 'my-database'
    oo_result = lo_glu->gettables( iv_databasename = iv_database_name ).
    DATA(lt_tables) = oo_result->get_tablelist( ).
    MESSAGE 'Tables retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Database does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetTablesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async
 -> [GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }
```

- Einzelheiten zur API finden Sie GetTablesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListJobs`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListJobs` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::ListJobsRequest listJobsRequest;

       Aws::String nextToken;
       std::vector<Aws::String> allJobNames;

       do {
           if (!nextToken.empty()) {
               listJobsRequest.SetNextToken(nextToken);
           }
           Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                   listJobsRequest);

           if (listRunsOutcome.IsSuccess()) {
               const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
               allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
               nextToken = listRunsOutcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error listing jobs. "
                         << listRunsOutcome.GetError().GetMessage()
                         << std::endl;
```

```
            }
        } while (!nextToken.empty());
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobs = $glueService->listJobs();
```

```
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
 Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- Einzelheiten zur API finden Sie ListJobsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
```

```
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- Einzelheiten zur API finden Sie ListJobsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_glu->listjobs( ).
    DATA(lt_job_names) = oo_result->get_jobnames( ).
    MESSAGE 'Job list retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'No jobs found.' TYPE 'I'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
    }
```

- Einzelheiten zur API finden Sie ListJobsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`StartCrawler`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartCrawler` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Start an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StartCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
               std::cout << "Successfully started crawler." << std::endl;
           }

           std::cout << "This may take a while to run." << std::endl;

           Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
           int iterations = 0;
           while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
               std::this_thread::sleep_for(std::chrono::seconds(1));
               ++iterations;
               if ((iterations % 10) == 0) { // Log status every 10 seconds.
```

```
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                            << ". After " << iterations
                            << " seconds elapsed."
                            << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                            << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Einen Crawler starten

Das folgende Beispiel für `start-crawler` startet einen Crawler.

```
aws glue start-crawler --name my-crawler
```

Ausgabe:

```
None
```

Weitere Informationen finden Sie unter Definieren von Crawlern im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie StartCrawlerin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
```

```
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Einzelheiten zur API finden Sie StartCrawlerin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);
```

```
    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie StartCrawlerin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
```

```
    raise
  end
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- Einzelheiten zur API finden Sie StartCrawlerin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        lo_glu->startcrawler( iv_name = iv_crawler_name ).
        MESSAGE 'Crawler started successfully.' TYPE 'I'.
      CATCH /aws1/cx_glucrawlerrunningex.
        MESSAGE 'Crawler is already running.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartCrawlerin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue
```

```
    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise
 `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }
```

- Einzelheiten zur API finden Sie StartCrawlerin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`StartJobRun`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartJobRun` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();
```

```
                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                          jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
```

```
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                         clientConfig);
            return false;
        }
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Die Auftragsausführung starten

Das folgende Beispiel für `start-job-run` startet die Ausführung eines Auftrags.

```
aws glue start-job-run \
    --job-name my-job
```

Ausgabe:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

Weitere Informationen finden Sie unter Autorisieren von Aufträgen im Entwicklerhandbuch zu AWS Glue.

- Einzelheiten zur API finden Sie StartJobRunin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                               String outBucket) {
       try {
           Map<String, String> myMap = new HashMap<>();
           myMap.put("--input_database", inputDatabase);
           myMap.put("--input_table", inputTable);
           myMap.put("--output_bucket_url", outBucket);

           StartJobRunRequest runRequest = StartJobRunRequest.builder()
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .arguments(myMap)
               .jobName(jobName)
               .build();

           StartJobRunResponse response = glueClient.startJobRun(runRequest);
           System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

       } catch (GlueException e) {
           throw e;
       }
   }
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;
```

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl): Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return response["JobRunId"]
```

- Einzelheiten zur API finden Sie StartJobRunin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
 job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
```

```
        '--input_table': input_table,
        '--output_bucket_url': "s3:///#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
```

```
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

- Einzelheiten zur API finden Sie StartJobRunin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_input_database = 'my-database'
        " iv_input_table = 'my-table'
        " iv_output_bucket_url = 's3://example-output-bucket/'

        DATA lt_arguments TYPE /aws1/cl_glugenericmap_w=>tt_genericmap.
        lt_arguments = VALUE #(
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_database'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_input_database ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_table'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_table ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--output_bucket_url'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_output_bucket_url ) ) ) ).

        DATA(oo_result) = lo_glu->startjobrun(
          iv_jobname = iv_job_name
          it_arguments = lt_arguments ).
        ov_job_run_id = oo_result->get_jobrunid( ).
        MESSAGE 'Job run started successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_gluconcurrentrunsex00.
        MESSAGE 'Maximum concurrent runs exceeded.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Job does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartJobRunin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
```

```
   ///   - databaseName: The name of the AWS Glue database to run the job
against.
   ///   - tableName: The name of the table in the database to run the job
against.
   ///   - outputURL: The AWS S3 URI of the bucket location into which to
   ///     write the resulting output.
   ///
   /// - Returns: `true` if the job run is started successfully, otherwise
`false`.
   func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                    tableName: String, outputURL: String) async -> String? {
       do {
           let output = try await glueClient.startJobRun(
               input: StartJobRunInput(
                   arguments: [
                       "--input_database": databaseName,
                       "--input_table": tableName,
                       "--output_bucket_url": outputURL
                   ],
                   jobName: jobName,
                   numberOfWorkers: 10,
                   workerType: .g1x
               )
           )

           guard let id = output.jobRunId else {
               return nil
           }

           return id
       } catch {
           return nil
       }
   }
```

- Einzelheiten zur API finden Sie StartJobRunin der API-Referenz zum AWS SDK für Swift.

Szenarien für AWS Glue verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS Glue mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS Glue oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit dem Datenkatalog

Erste Schritte mit dem Datenkatalog

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Glue Data Catalog Tutorial Script
# This script demonstrates how to create and manage AWS Glue Data Catalog
 resources using the AWS CLI
```

```
# Cost improvements: Reduced API calls, optimized queries, eliminated redundant
 operations
# Reliability improvements: Enhanced error handling, input validation, resource
 tracking

set -euo pipefail

# Setup logging
LOG_FILE="glue-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Glue Data Catalog tutorial script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Generate a unique identifier for resource names
UNIQUE_ID=$(openssl rand -hex 4)
DB_NAME="tutorial-db-${UNIQUE_ID}"
TABLE_NAME="flights-data-${UNIQUE_ID}"
TABLE_INPUT_FILE="table-input-${UNIQUE_ID}.json"

# Track created resources
declare -a CREATED_RESOURCES=()

# Set default region if not provided
AWS_REGION="${AWS_REGION:-us-east-1}"

# Flag to track if database was successfully created
DATABASE_CREATED=false

# Trap to ensure cleanup on exit
trap cleanup_resources EXIT

# Function to check command status
check_status() {
    if [ $? -ne 0 ]; then
        echo "ERROR: $1 failed." >&2
        exit 1
    fi
}

# Function to cleanup resources
cleanup_resources() {
    local exit_code=$?
    echo "Attempting to clean up resources..."
```

```
    # Delete resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource=${CREATED_RESOURCES[$i]}
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_name=$(echo "$resource" | cut -d':' -f2)

        echo "Deleting $resource_type: $resource_name"

        case $resource_type in
            "table")
                if [ "$DATABASE_CREATED" = true ]; then
                    aws glue delete-table \
                        --database-name "$DB_NAME" \
                        --name "$resource_name" \
                        --region "$AWS_REGION" \
                        2>/dev/null || echo "Warning: Failed to delete table
$resource_name"
                fi
                ;;
            "database")
                aws glue delete-database \
                    --name "$resource_name" \
                    --region "$AWS_REGION" \
                    2>/dev/null || echo "Warning: Failed to delete database
$resource_name"
                ;;
            *)
                echo "Unknown resource type: $resource_type" >&2
                ;;
        esac
    done

    # Clean up temporary files securely
    if [ -f "$TABLE_INPUT_FILE" ]; then
        if command -v shred &> /dev/null; then
            shred -vfz -n 3 "$TABLE_INPUT_FILE" 2>/dev/null || rm -f
"$TABLE_INPUT_FILE"
        else
            rm -f "$TABLE_INPUT_FILE"
        fi
    fi

    echo "Cleanup completed."
```

```
    exit $exit_code
}

# Function to validate prerequisites
validate_prerequisites() {
    # Validate AWS CLI is available
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS CLI version
    local AWS_CLI_VERSION
    AWS_CLI_VERSION=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2 | cut -
d'.' -f1)
    if [ "$AWS_CLI_VERSION" -lt 1 ]; then
        echo "ERROR: AWS CLI is required" >&2
        exit 1
    fi

    # Validate jq is available for JSON validation
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS credentials and get account identity in single call (cost
 optimization)
    local CALLER_IDENTITY
    CALLER_IDENTITY=$(aws sts get-caller-identity --region "$AWS_REGION" --query
 'Account' --output text 2>/dev/null) || {
        echo "ERROR: Failed to get AWS caller identity. Check credentials and
 permissions." >&2
        exit 1
    }

    if [ -z "$CALLER_IDENTITY" ] || [ "$CALLER_IDENTITY" == "None" ]; then
        echo "ERROR: Unable to determine AWS account identity" >&2
        exit 1
    fi
    echo "Using AWS Account: $CALLER_IDENTITY"
    echo "Using Region: $AWS_REGION"
}
```

```
# Function to create database with verification
create_database() {
    echo "Step 1: Creating a database named $DB_NAME"

    if ! aws glue create-database \
        --database-input "Name=$DB_NAME,Description=Database for AWS Glue
 tutorial" \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create database $DB_NAME" >&2
        exit 1
    fi

    ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:database/
${DB_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    DATABASE_CREATED=true
    CREATED_RESOURCES+=("database:$DB_NAME")
    echo "Database $DB_NAME created successfully."
}

# Function to prepare table input JSON
prepare_table_input() {
    # Create a temporary JSON file for table input with restricted permissions
    if ! touch "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to create temporary file $TABLE_INPUT_FILE" >&2
        exit 1
    fi

    if ! chmod 600 "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to set permissions on $TABLE_INPUT_FILE" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    cat > "$TABLE_INPUT_FILE" << 'EOF'
{
  "Name": "TABLE_NAME_PLACEHOLDER",
  "StorageDescriptor": {
```

```
    "Columns": [
      {
        "Name": "year",
        "Type": "bigint"
      },
      {
        "Name": "quarter",
        "Type": "bigint"
      }
    ],
    "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
    "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
    "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
    "Compressed": false,
    "NumberOfBuckets": -1,
    "SerdeInfo": {
      "SerializationLibrary":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
      "Parameters": {
        "field.delim": ",",
        "serialization.format": ","
      }
    }
  },
  "PartitionKeys": [
    {
      "Name": "mon",
      "Type": "string"
    }
  ],
  "TableType": "EXTERNAL_TABLE",
  "Parameters": {
    "EXTERNAL": "TRUE",
    "classification": "csv",
    "columnsOrdered": "true",
    "compressionType": "none",
    "delimiter": ",",
    "skip.header.line.count": "1",
    "typeOfData": "file"
  }
}
EOF

    # Replace placeholder with actual table name
```

```
    if ! sed -i "s/TABLE_NAME_PLACEHOLDER/$TABLE_NAME/g" "$TABLE_INPUT_FILE" 2>/
dev/null; then
        echo "ERROR: Failed to substitute table name in JSON file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    # Validate JSON syntax before using it
    if ! jq empty "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Invalid JSON in table input file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi
}

# Function to create table
create_table() {
    echo "Step 2: Creating a table named $TABLE_NAME in database $DB_NAME"

    prepare_table_input

    local TABLE_ARN
    if ! aws glue create-table \
        --database-name "$DB_NAME" \
        --table-input "file://${TABLE_INPUT_FILE}" \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create table $TABLE_NAME" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:table/
${DB_NAME}/${TABLE_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    CREATED_RESOURCES+=("table:$TABLE_NAME")
    echo "Table $TABLE_NAME created successfully."
}

# Function to get and display table details
```

```
display_table_details() {
    echo "Step 3: Getting details of table $TABLE_NAME"

    if ! aws glue get-table \
        --database-name "$DB_NAME" \
        --name "$TABLE_NAME" \
        --region "$AWS_REGION" \
        --output json; then
        echo "ERROR: Failed to retrieve table details" >&2
        exit 1
    fi
}

# Function to display summary
display_summary() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="
    echo "Database: $DB_NAME"
    echo "Table: $TABLE_NAME"
    echo "==========================================="
}

# Main execution flow
validate_prerequisites
create_database
create_table
display_table_details
display_summary

echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Starting cleanup process..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateDatabase

- CreateTable
- DeleteDatabase
- DeleteTable
- GetDatabase
- GetTable

Codebeispiele für AWS Health verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Health mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS Health Entwicklerhandbuch — Weitere Informationen zu AWS Health.
- AWS Health API-Referenz — Details zu allen verfügbaren AWS Health Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS Health verwenden AWS SDKs
  - Aktionen für AWS Health verwenden AWS SDKs
    - Verwendung von DescribeEvents mit einer CLI

Grundlegende Beispiele für AWS Health verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Health mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS Health verwenden AWS SDKs

- Verwendung von DescribeEvents mit einer CLI

Aktionen für AWS Health verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS Health Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Health -API-Referenz.

Beispiele

- Verwendung von DescribeEvents mit einer CLI

Verwendung von **`DescribeEvents`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEvents` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um AWS Gesundheitsereignisse aufzulisten

Das folgende `describe-events` Beispiel listet aktuelle AWS Gesundheitsereignisse auf.

```
aws health describe-events \
    --region us-east-1
```

Ausgabe:

```
{
    "events": [
        {
            "arn": "arn:aws:health:us-west-1::event/ECS/
AWS_ECS_OPERATIONAL_ISSUE/AWS_ECS_OPERATIONAL_ISSUE_KWQPY_EXAMPLE111",
            "service": "ECS",
            "eventTypeCode": "AWS_ECS_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-west-1",
            "startTime": 1589077890.53,
```

```
            "endTime": 1589086345.597,
            "lastUpdatedTime": 1589086345.905,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/BILLING/
AWS_BILLING_NOTIFICATION/AWS_BILLING_NOTIFICATION_6ce1d874-e995-40e2-99cd-
EXAMPLE1118b",
            "service": "BILLING",
            "eventTypeCode": "AWS_BILLING_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1588356000.0,
            "lastUpdatedTime": 1588356524.358,
            "statusCode": "open",
            "eventScopeCode": "ACCOUNT_SPECIFIC"
        },
        {
            "arn": "arn:aws:health:us-west-2::event/
CLOUDFORMATION/AWS_CLOUDFORMATION_OPERATIONAL_ISSUE/
AWS_CLOUDFORMATION_OPERATIONAL_ISSUE_OHTWY_EXAMPLE111",
            "service": "CLOUDFORMATION",
            "eventTypeCode": "AWS_CLOUDFORMATION_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-west-2",
            "startTime": 1588279630.761,
            "endTime": 1588284650.0,
            "lastUpdatedTime": 1588284691.941,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:ap-northeast-1::event/LAMBDA/
AWS_LAMBDA_OPERATIONAL_ISSUE/AWS_LAMBDA_OPERATIONAL_ISSUE_JZDND_EXAMPLE111",
            "service": "LAMBDA",
            "eventTypeCode": "AWS_LAMBDA_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "ap-northeast-1",
            "startTime": 1587379534.08,
            "endTime": 1587391771.0,
            "lastUpdatedTime": 1587395689.316,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
```

```
        },
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1586473044.284,
            "endTime": 1586479706.091,
            "lastUpdatedTime": 1586479706.153,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/SECURITY/
AWS_SECURITY_NOTIFICATION/AWS_SECURITY_NOTIFICATION_42007387-8129-42da-8c88-
EXAMPLE11139",
            "service": "SECURITY",
            "eventTypeCode": "AWS_SECURITY_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1585674000.0,
            "lastUpdatedTime": 1585674004.132,
            "statusCode": "open",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/CLOUDFRONT/
AWS_CLOUDFRONT_OPERATIONAL_ISSUE/
AWS_CLOUDFRONT_OPERATIONAL_ISSUE_FRQXG_EXAMPLE111",
            "service": "CLOUDFRONT",
            "eventTypeCode": "AWS_CLOUDFRONT_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "global",
            "startTime": 1585610898.589,
            "endTime": 1585617671.0,
            "lastUpdatedTime": 1585620638.869,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:us-east-1::event/SES/
AWS_SES_OPERATIONAL_ISSUE/AWS_SES_OPERATIONAL_ISSUE_URNDF_EXAMPLE111",
```

```
            "service": "SES",
            "eventTypeCode": "AWS_SES_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1585342008.46,
            "endTime": 1585344017.0,
            "lastUpdatedTime": 1585344355.989,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:global::event/IAM/
AWS_IAM_OPERATIONAL_NOTIFICATION/
AWS_IAM_OPERATIONAL_NOTIFICATION_b6771c34-6ecd-4aea-9d3e-EXAMPLE1117e",
            "service": "IAM",
            "eventTypeCode": "AWS_IAM_OPERATIONAL_NOTIFICATION",
            "eventTypeCategory": "accountNotification",
            "region": "global",
            "startTime": 1584978300.0,
            "lastUpdatedTime": 1584978553.572,
            "statusCode": "open",
            "eventScopeCode": "ACCOUNT_SPECIFIC"
        },
        {
            "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "ap-southeast-2",
            "startTime": 1583881487.483,
            "endTime": 1583885056.785,
            "lastUpdatedTime": 1583885057.052,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte mit dem AWS Personal Health Dashboard im AWS Health-Benutzerhandbuch.

Beispiel 2: Um AWS Gesundheitsereignisse nach Service und Ereignisstatuscode aufzulisten

Das folgende `describe-events` Beispiel listet AWS Health-Ereignisse für Amazon Elastic Compute Cloud (Amazon EC2) auf, bei denen der Ereignisstatus geschlossen ist.

```
aws health describe-events \
    --filter "services=EC2,eventStatusCodes=closed"
```

Ausgabe:

```
{
    "events": [
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_VKTXI_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1587462325.096,
            "endTime": 1587464204.774,
            "lastUpdatedTime": 1587464204.865,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:us-east-1::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_COBXJ_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "us-east-1",
            "startTime": 1586473044.284,
            "endTime": 1586479706.091,
            "lastUpdatedTime": 1586479706.153,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        },
        {
            "arn": "arn:aws:health:ap-southeast-2::event/EC2/
AWS_EC2_OPERATIONAL_ISSUE/AWS_EC2_OPERATIONAL_ISSUE_HNGHE_EXAMPLE111",
            "service": "EC2",
            "eventTypeCode": "AWS_EC2_OPERATIONAL_ISSUE",
            "eventTypeCategory": "issue",
            "region": "ap-southeast-2",
```

```
            "startTime": 1583881487.483,
            "endTime": 1583885056.785,
            "lastUpdatedTime": 1583885057.052,
            "statusCode": "closed",
            "eventScopeCode": "PUBLIC"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte mit dem AWS Personal Health Dashboard im AWS Health-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeEventsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt Ereignisse aus dem AWS Personal Health Dashboard zurück. Der Benutzer fügt den Parameter -Region hinzu, um Ereignisse anzuzeigen, die für den Service in der Region USA Ost (Nord-Virginia) verfügbar sind, aber der Filter_Region Parameter - filtert nach Ereignissen, die in den Regionen EU (London) und USA West (Oregon) (eu-west-2 und us-west-2) protokolliert werden. Der Filter_StartTime Parameter - filtert nach einem bestimmten Zeitbereich, zu dem Ereignisse beginnen können, während der Filter_EndTime Parameter - nach einem bestimmten Zeitbereich filtert, zu dem Ereignisse enden können. Das Ergebnis ist ein geplantes Wartungsereignis für RDS, das innerhalb des angegebenen Filter_StartTime Bereichs beginnt und innerhalb des geplanten Filter_EndTime Bereichs endet.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Ausgabe:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
```

```
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- Einzelheiten zur API finden Sie unter DescribeEvents AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt Ereignisse aus dem AWS Personal Health Dashboard zurück. Der Benutzer fügt den Parameter -Region hinzu, um Ereignisse anzuzeigen, die für den Service in der Region USA Ost (Nord-Virginia) verfügbar sind, aber der Filter_Region Parameter - filtert nach Ereignissen, die in den Regionen EU (London) und USA West (Oregon) (eu-west-2 und us-west-2) protokolliert werden. Der Filter_StartTime Parameter - filtert nach einem bestimmten Zeitbereich, zu dem Ereignisse beginnen können, während der Filter_EndTime Parameter - nach einem bestimmten Zeitbereich filtert, zu dem Ereignisse enden können. Das Ergebnis ist ein geplantes Wartungsereignis für RDS, das innerhalb des angegebenen Filter_StartTime Bereichs beginnt und innerhalb des geplanten Filter_EndTime Bereichs endet.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Ausgabe:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
```

```
StatusCode        : open
```

- Einzelheiten zur API finden Sie unter DescribeEvents AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Verwendung HealthImaging AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS HealthImaging mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- HealthImaging Entwicklerhandbuch — Weitere Informationen zu HealthImaging.
- HealthImaging API-Referenz — Details zu allen verfügbaren HealthImaging Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung HealthImaging AWS SDKs
  - Hallo HealthImaging
  - Aktionen zur HealthImaging Verwendung AWS SDKs
    - Verwenden Sie mit CopyImageSeteinem AWS SDK oder CLI
    - Verwenden Sie CreateDatastoremit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDatastoremit einem AWS SDK oder CLI
    - Verwenden Sie DeleteImageSetmit einem AWS SDK oder CLI
    - Verwenden Sie GetDICOMImportJobmit einem AWS SDK oder CLI
    - Verwenden Sie GetDatastoremit einem AWS SDK oder CLI

    - Verwenden Sie GetImageFramemit einem AWS SDK oder CLI
    - Verwenden Sie GetImageSetmit einem AWS SDK oder CLI
    - Verwenden Sie GetImageSetMetadatamit einem AWS SDK oder CLI
    - Verwenden Sie ListDICOMImportJobsmit einem AWS SDK oder CLI
    - Verwenden Sie ListDatastoresmit einem AWS SDK oder CLI
    - Verwenden Sie ListImageSetVersionsmit einem AWS SDK oder CLI
    - Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
    - Verwenden Sie SearchImageSetsmit einem AWS SDK oder CLI
    - Verwenden Sie StartDICOMImportJobmit einem AWS SDK oder CLI
    - Verwenden Sie TagResourcemit einem AWS SDK oder CLI
    - Verwenden Sie UntagResourcemit einem AWS SDK oder CLI
    - Verwenden Sie UpdateImageSetMetadatamit einem AWS SDK oder CLI
- Szenarien für die HealthImaging Verwendung AWS SDKs
  - Beginnen Sie mit HealthImaging Bildsätzen und Bildrahmen mithilfe eines AWS SDK
  - Kennzeichnen eines HealthImaging Datenspeichers mit einem AWS SDK
  - Einen HealthImaging Bilddatensatz mit einem taggen AWS SDK

Grundlegende Beispiele für die Verwendung HealthImaging AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS HealthImaging mit AWS SDKs verwendet werden.

Beispiele

- Hallo HealthImaging
- Aktionen zur HealthImaging Verwendung AWS SDKs
  - Verwenden Sie mit CopyImageSeteinem AWS SDK oder CLI
  - Verwenden Sie CreateDatastoremit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDatastoremit einem AWS SDK oder CLI
  - Verwenden Sie DeleteImageSetmit einem AWS SDK oder CLI
  - Verwenden Sie GetDICOMImportJobmit einem AWS SDK oder CLI
  - Verwenden Sie GetDatastoremit einem AWS SDK oder CLI
  - Verwenden Sie GetImageFramemit einem AWS SDK oder CLI

- Verwenden Sie GetImageSetmit einem AWS SDK oder CLI
- Verwenden Sie GetImageSetMetadatamit einem AWS SDK oder CLI
- Verwenden Sie ListDICOMImportJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListDatastoresmit einem AWS SDK oder CLI
- Verwenden Sie ListImageSetVersionsmit einem AWS SDK oder CLI
- Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
- Verwenden Sie SearchImageSetsmit einem AWS SDK oder CLI
- Verwenden Sie StartDICOMImportJobmit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UntagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UpdateImageSetMetadatamit einem AWS SDK oder CLI

Hallo HealthImaging

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von HealthImaging beginnen.

C++

SDK für C++

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})
```

```
if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executable location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_health_imaging.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_health_imaging.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 *  A "Hello HealthImaging" starter application which initializes an AWS
 HealthImaging (HealthImaging) client
 *  and lists the HealthImaging data stores in the current account.
 *
 *  main function
```

```
 *
 *  Usage: 'hello_health-imaging'
 *
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    //   Optional: change the log level for debugging.
    //   options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
 medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;

        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome
 listDatastoresOutcome =
                    medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 &dataStoreSummaries =

 listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                             dataStoreSummaries.cbegin(),
                                             dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
```

```
            else {
                std::cerr << "ListDatastores error: "
                          << listDatastoresOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allDataStoreSummaries.size() << " HealthImaging data "
                  << ((allDataStoreSummaries.size() == 1) ?
                      "store was retrieved." : "stores were retrieved.") <<
 std::endl;

        for (auto const &dataStoreSummary: allDataStoreSummaries) {
            std::cout << "  Datastore: " << dataStoreSummary.GetDatastoreName()
                      << std::endl;
            std::cout << "  Datastore ID: " << dataStoreSummary.GetDatastoreId()
                      << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Einzelheiten zur API finden Sie ListDatastoresunter AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import {
  ListDatastoresCommand,
  MedicalImagingClient,
} from "@aws-sdk/client-medical-imaging";
```

```
// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new MedicalImagingClient({});

export const helloMedicalImaging = async () => {
  const command = new ListDatastoresCommand({});

  const { datastoreSummaries } = await client.send(command);
  console.log("Datastores: ");
  console.log(datastoreSummaries.map((item) => item.datastoreName).join("\n"));
  return datastoreSummaries;
};
```

- Einzelheiten zur API finden Sie ListDatastoresin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_medical_imaging(medical_imaging_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS HealthImaging
    client and list the data stores in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param medical_imaging_client: A Boto3 AWS HealthImaging Client object.
```

```
    """
    print("Hello, Amazon Health Imaging! Let's list some of your data stores:\n")
    try:
        paginator = medical_imaging_client.get_paginator("list_datastores")
        page_iterator = paginator.paginate()
        datastore_summaries = []
        for page in page_iterator:
            datastore_summaries.extend(page["datastoreSummaries"])
        print("\tData Stores:")
        for ds in datastore_summaries:
            print(f"\t\tDatastore: {ds['datastoreName']} ID {ds['datastoreId']}")
    except ClientError as err:
        logger.error(
            "Couldn't list data stores. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise


if __name__ == "__main__":
    hello_medical_imaging(boto3.client("medical-imaging"))
```

- Einzelheiten zur API finden Sie ListDatastoresin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktionen zur HealthImaging Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne HealthImaging Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die HealthImaging API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die HealthImaging Verwendung AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS HealthImaging -API-Referenz.

Beispiele

- Verwenden Sie mit CopyImageSeteinem AWS SDK oder CLI
- Verwenden Sie CreateDatastoremit einem AWS SDK oder CLI
- Verwenden Sie DeleteDatastoremit einem AWS SDK oder CLI
- Verwenden Sie DeleteImageSetmit einem AWS SDK oder CLI
- Verwenden Sie GetDICOMImportJobmit einem AWS SDK oder CLI
- Verwenden Sie GetDatastoremit einem AWS SDK oder CLI
- Verwenden Sie GetImageFramemit einem AWS SDK oder CLI
- Verwenden Sie GetImageSetmit einem AWS SDK oder CLI
- Verwenden Sie GetImageSetMetadatamit einem AWS SDK oder CLI
- Verwenden Sie ListDICOMImportJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListDatastoresmit einem AWS SDK oder CLI
- Verwenden Sie ListImageSetVersionsmit einem AWS SDK oder CLI
- Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
- Verwenden Sie SearchImageSetsmit einem AWS SDK oder CLI
- Verwenden Sie StartDICOMImportJobmit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UntagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UpdateImageSetMetadatamit einem AWS SDK oder CLI

Verwenden Sie mit **`CopyImageSet`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CopyImageSet` verwendet wird.

CLI

AWS CLI

Beispiel 1: So kopieren Sie einen Bildsatz ohne Ziel.

Im folgenden Beispiel für `copy-image-set` wird eine doppelte Kopie eines Bildersatzes ohne Ziel erstellt.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" } }'
```

Ausgabe:

```
{
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042357.432,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042357.432,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 2: So kopieren Sie einen Bildersatz mit Ziel.

Im folgenden Beispiel für `copy-image-set` wird eine doppelte Kopie eines Bildersatzes mit Ziel erstellt.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet": {"latestVersionId": "1" },
 "destinationImageSet": { "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
 "latestVersionId": "1"} }'
```

Ausgabe:

```
{
```

```
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9a06fef182a5f992842f77f8e0868e5",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 3: So kopieren Sie eine Teilmenge von Instances aus einem Quellbildersatz in einen Zielbildersatz.

Im folgenden Beispiel für `copy-image-set` wird eine DICOM-Instance aus dem Quellbildersatz in den Zielbildersatz kopiert. Der force-Parameter dient dazu, Inkonsistenzen in den Attributen auf Patienten-, Studien- und Serienebene zu überschreiben.

```
aws medical-imaging copy-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --source-image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --copy-image-set-information '{"sourceImageSet":
 {"latestVersionId": "1","DICOMCopies": {"copiableAttributes":
 "{\"SchemaVersion\":\"1.1\",\"Study\":{\"Series\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3666.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3673257865.2104868982.1369432891697.3669.0\":
{}}}}}}"}},"destinationImageSet": {"imageSetId":
 "b9eb50d8ee682eb9fcf4acbf92f62bb7","latestVersionId": "1"}}' \
    --force
```

Ausgabe:

```
{
```

```
    "destinationImageSetProperties": {
        "latestVersionId": "2",
        "imageSetWorkflowStatus": "COPYING",
        "updatedAt": 1680042505.135,
        "imageSetId": "b9eb50d8ee682eb9fcf4acbf92f62bb7",
        "imageSetState": "LOCKED",
        "createdAt": 1680042357.432
    },
    "sourceImageSetProperties": {
        "latestVersionId": "1",
        "imageSetWorkflowStatus": "COPYING_WITH_READ_ONLY_ACCESS",
        "updatedAt": 1680042505.135,
        "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
        "imageSetState": "LOCKED",
        "createdAt": 1680027126.436
    },
    "datastoreId": "12345678901234567890123456789012"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Kopieren eines Bildsatzes.

- Einzelheiten zur API finden Sie CopyImageSetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    /**
     * Copy an AWS HealthImaging image set.
     *
     * @param medicalImagingClient  - The AWS HealthImaging client object.
     * @param datastoreId           - The datastore ID.
     * @param imageSetId            - The image set ID.
     * @param latestVersionId       - The version ID.
     * @param destinationImageSetId - The optional destination image set ID,
 ignored if null.
     * @param destinationVersionId  - The optional destination version ID,
 ignored if null.
     * @param force                 - The force flag.
     * @param subsets               - The optional subsets to copy, ignored if
 null.
```

```
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
               String subsetInstanceToCopy =
getCopiableAttributesJSON(imageSetId, subsets);

copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                       .copiableAttributes(subsetInstanceToCopy)
                       .build());
           }

           CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                   .sourceImageSet(copySourceImageSetInformation.build());

           // Optionally designate a destination image set.
           if (destinationImageSetId != null) {
               copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                       .imageSetId(destinationImageSetId)
                       .latestVersionId(destinationVersionId)
                       .build());
           }

           CopyImageSetRequest copyImageSetRequest =
CopyImageSetRequest.builder()
```

```
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Hilfsfunktion zum Erstellen kopierbarer Attribute.

```
    /**
     * Create a JSON string of copiable image instances.
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                                "
                                 """
        );

        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
```

```
                                ": {
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                         }
                       }
                     }
                   }
                 }
                 """);
        return subsetInstanceToCopy.toString();
    }
```

- Einzelheiten zur API finden Sie CopyImageSetin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

Hilfsfunktion zum Kopieren eines Imagesatzes.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination
 image set.
```

```
 * @param {string} destinationVersionId - The optional version ID of the
 destination image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
  sourceVersionId = "1",
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
      copyImageSetInformation: {
        sourceImageSet: { latestVersionId: sourceVersionId },
      },
      force: force,
    };
    if (destinationImageSetId !== "" && destinationVersionId !== "") {
      params.copyImageSetInformation.destinationImageSet = {
        imageSetId: destinationImageSetId,
        latestVersionId: destinationVersionId,
      };
    }

    if (copySubsets.length > 0) {
      let copySubsetsJson;
      copySubsetsJson = {
        SchemaVersion: 1.1,
        Study: {
          Series: {
            imageSetId: {
              Instances: {},
            },
          },
        },
      };

      for (let i = 0; i < copySubsets.length; i++) {
```

```
        copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
      }

      params.copyImageSetInformation.dicomCopies = copySubsetsJson;
    }

    const response = await medicalImagingClient.send(
      new CopyImageSetCommand(params),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //         httpStatusCode: 200,
    //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    //      },
    //       datastoreId: 'xxxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
```

```
  }
};
```

Kopieren Sie einen Imagesatz ohne Ziel.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
  );
```

Kopieren Sie einen Imagesatz mit Ziel.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    false,
  );
```

Kopieren Sie eine Teilmenge eines Imagesatzes mit einem Ziel und erzwingen Sie den Kopiervorgang.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    true,
    ["12345678901234567890123456789012", "11223344556677889900112233445566"],
  );
```

- Einzelheiten zur API finden Sie CopyImageSetin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

Hilfsfunktion zum Kopieren eines Imagesatzes.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
 set.
        :param destination_version_id: The ID of the optional destination image
 set version.
        :param force: Force the copy.
```

```
        :param subsets: The optional subsets to copy. For example:
 ["12345678901234567890123456789012"].
        :return: The copied image set ID.
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Kopieren Sie einen Imagesatz ohne Ziel.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Kopieren Sie einen Imagesatz mit Ziel.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Kopieren Sie eine Teilmenge eines Imagesatzes.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
```

```
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"]
[
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie CopyImageSetin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_source_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678 90' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CopyImageSetin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`CreateDatastore`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDatastore` verwendet wird.

Bash

AWS CLI mit Bash-Skript

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_create_datastore
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
```

```
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation
 failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie CreateDatastorein der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

Beispiel 1: Um einen Datenspeicher zu erstellen

Im folgenden Codebeispiel für `create-datastore` wird ein Datenspeicher mit dem Namen `my-datastore` erstellt. Wenn Sie einen Datenspeicher ohne Angabe von a erstellen`--lossless-storage-format`, wird AWS HealthImaging standardmäßig HTJ2K (High Throughput JPEG 2000) verwendet.

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore"
```

Ausgabe:

```
{
```

```
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Beispiel 2: So erstellen Sie einen Datenspeicher mit dem verlustfreien JPEG 2000-Speicherformat

Ein Datenspeicher, der mit dem verlustfreien Speicherformat JPEG 2000 konfiguriert ist, transkodiert und speichert verlustfreie Bildframes im JPEG 2000-Format. Bildframes können dann ohne Transcodierung im Format JPEG 2000 Lossless abgerufen werden. Im folgenden `create-datastore` Codebeispiel wird ein für das verlustfreies JPEG 2000-Speicherformat konfigurierter Datenspeicher mit dem Namen `my-datastore` erstellt.

```
aws medical-imaging create-datastore \
    --datastore-name "my-datastore" \
    --lossless-storage-format JPEG_2000_LOSSLESS
```

Ausgabe:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "CREATING"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Erstellen eines Datenspeichers.

- Einzelheiten zur API finden Sie CreateDatastoreunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static String createMedicalImageDatastore(MedicalImagingClient
 medicalImagingClient,
            String datastoreName) {
        try {
            CreateDatastoreRequest datastoreRequest =
 CreateDatastoreRequest.builder()
                    .datastoreName(datastoreName)
                    .build();
```

```
            CreateDatastoreResponse response =
 medicalImagingClient.createDatastore(datastoreRequest);
            return response.datastoreId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- Einzelheiten zur API finden Sie CreateDatastorein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
```

```
  //       totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- Einzelheiten zur API finden Sie CreateDatastorein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
 self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie CreateDatastorein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_name = 'my-datastore-name'
        oo_result = lo_mig->createdatastore( iv_datastorename =
iv_datastore_name ).
        DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
        MESSAGE 'Data store created.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDatastorein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`DeleteDatastore`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDatastore` verwendet wird.

Bash

AWS CLI mit Bash-Skript

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function imaging_delete_datastore() {
  local datastore_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_delete_datastore"
    echo "Deletes an AWS HealthImaging data store."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation
 failed.$response"
    return 1
  fi

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteDatastorein der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So löschen Sie einen Datenspeicher

Im folgenden Codebeispiel für `delete-datastore` wird ein Datenspeicher gelöscht.

```
aws medical-imaging delete-datastore \
    --datastore-id "12345678901234567890123456789012"
```

Ausgabe:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "datastoreStatus": "DELETING"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Löschen eines Datenspeichers.

- Einzelheiten zur API finden Sie DeleteDatastoreunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static void deleteMedicalImagingDatastore(MedicalImagingClient
 medicalImagingClient,
            String datastoreID) {
        try {
            DeleteDatastoreRequest datastoreRequest =
 DeleteDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            medicalImagingClient.deleteDatastore(datastoreRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDatastorein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
```

```
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
  //           cfId: undefined,
  //           attempts: 1,
  //           totalRetryDelay: 0
  //        },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie DeleteDatastorein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
```

```
        try:
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie DeleteDatastorein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678 90'
    oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
    MESSAGE 'Data store deleted.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
```

```
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Data store not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDatastorein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`DeleteImageSet`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteImageSet` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
```

```
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data
 store "
                  << dataStoreID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteImageSetin der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So löschen Sie einen Bildersatz

Im folgenden Codebeispiel für `delete-image-set` wird ein Bildersatz gelöscht.

```
aws medical-imaging delete-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Ausgabe:

```
{
    "imageSetWorkflowStatus": "DELETING",
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "datastoreId": "12345678901234567890123456789012"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Löschen eines Bilddatensatzes.

- Einzelheiten zur API finden Sie DeleteImageSetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           medicalImagingClient.deleteImageSet(deleteImageSetRequest);

           System.out.println("The image set was deleted.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DeleteImageSetin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //        httpStatusCode: 200,
  //        requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
  // }
```

```
  return response;
};
```

- Einzelheiten zur API finden Sie DeleteImageSetin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return delete_results
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie DeleteImageSetin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        " iv_image_set_id = '1234567890123456789012345678901234567890'
        oo_result = lo_mig->deleteimageset(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
        MESSAGE 'Image set deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteImageSetin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`GetDICOMImportJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDICOMImportJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
```

```
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

- Einzelheiten zur API finden Sie GetDICOMImportJobin der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So rufen Sie die Eigenschaften eines DICOM-Importauftrags ab

Im folgenden Codebeispiel für `get-dicom-import-job` werden die Eigenschaften eines DICOM-Importauftrags abgerufen.

```
aws medical-imaging get-dicom-import-job \
    --datastore-id "12345678901234567890123456789012" \
    --job-id "09876543210987654321098765432109"
```

Ausgabe:

```
{
    "jobProperties": {
        "jobId": "09876543210987654321098765432109",
        "jobName": "my-job",
        "jobStatus": "COMPLETED",
        "datastoreId": "12345678901234567890123456789012",
```

```
        "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
        "endedAt": "2022-08-12T11:29:42.285000+00:00",
        "submittedAt": "2022-08-12T11:28:11.152000+00:00",
        "inputS3Uri": "s3://medical-imaging-dicom-input/dicom_input/",
        "outputS3Uri": "s3://medical-imaging-output/
job_output/12345678901234567890123456789012-
DicomImport-09876543210987654321098765432109/"
    }
}
```

Weitere Informationen finden Sie im AWS HealthImaging Developer Guide unter Abrufen der Eigenschaften von Importaufträgen.

- Einzelheiten zur API finden Sie GetDICOMImportJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
 medicalImagingClient,
           String datastoreId,
           String jobId) {

       try {
           GetDicomImportJobRequest getDicomImportJobRequest =
 GetDicomImportJobRequest.builder()
                   .datastoreId(datastoreId)
                   .jobId(jobId)
                   .build();
           GetDicomImportJobResponse response =
 medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
           return response.jobProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Einzelheiten zur API finden Sie GetDICOMImportJobin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
```

```
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie GetDICOMImportJobin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie GetDICOMImportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
TRY.
    " iv_datastore_id = '12345678901234567890123456789012345678 90'
    " iv_job_id = '12345678901234567890123456789012'
    oo_result = lo_mig->getdicomimportjob(
      iv_datastoreid = iv_datastore_id
      iv_jobid = iv_job_id ).
    DATA(lo_job_props) = oo_result->get_jobproperties( ).
    DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
    MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
```

```
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Job not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetDICOMImportJobin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`GetDatastore`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDatastore` verwendet wird.

Bash

AWS CLI mit Bash-Skript

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_get_datastore
```

```
#
# Get a data store's properties.
#
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,
 created_at, updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a
 function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
```

```
    usage
    return 1
  fi

  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
 datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
 datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
 datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie GetDatastorein der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

Beispiel 1: Um die Eigenschaften eines Datenspeichers abzurufen

Im folgenden Codebeispiel für `get-datastore` werden die Eigenschaften eines Datenspeichers abgerufen.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Ausgabe:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "HTJ2K"
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
    }
}
```

Beispiel 2: Um die Eigenschaften eines Datenspeichers für JPEG2000 zu konfigurieren

Im folgenden `get-datastore` Codebeispiel werden die Eigenschaften eines Datenspeichers für einen Datenspeicher abgerufen, der für das Format JPEG 2000 Lossless konfiguriert ist.

```
aws medical-imaging get-datastore \
    --datastore-id 12345678901234567890123456789012
```

Ausgabe:

```
{
    "datastoreProperties": {
        "datastoreId": "12345678901234567890123456789012",
        "datastoreName": "TestDatastore123",
        "datastoreStatus": "ACTIVE",
        "losslessStorageFormat": "JPEG_2000_LOSSLESS",
        "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
        "createdAt": "2022-11-15T23:33:09.643000+00:00",
        "updatedAt": "2022-11-15T23:33:09.643000+00:00"
```

```
    }
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Abrufen von Datenspeichereigenschaften.

- Einzelheiten zur API finden Sie GetDatastorein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static DatastoreProperties
getMedicalImageDatastore(MedicalImagingClient medicalImagingClient,
           String datastoreID) {
       try {
           GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
           return response.datastoreProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Einzelheiten zur API finden Sie GetDatastorein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxx:datastore/xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- Einzelheiten zur API finden Sie GetDatastorein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreProperties"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
```

```
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie GetDatastorein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
        DATA(lo_properties) = oo_result->get_datastoreproperties( ).
        DATA(lv_name) = lo_properties->get_datastorename( ).
        DATA(lv_status) = lo_properties->get_datastorestatus( ).
        MESSAGE 'Data store properties retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Data store not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetDatastorein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`GetImageFrame`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetImageFrame` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
    imageFrameInformation.SetImageFrameId(frameID);
```

```
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome =
 client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetImageFramein der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So rufen Sie Pixeldaten des Bildersatzes ab

Im folgenden Codebeispiel für `get-image-frame` wird ein Image-Frame abgerufen.

```
aws medical-imaging get-image-frame \
    --datastore-id "12345678901234567890123456789012" \
    --image-set-id "98765412345612345678907890789012" \
```

```
    --image-frame-information imageFrameId=3abf5d5d7ae72f80a0ec81b2c0de3ef4 \
    imageframe.jph
```

Hinweis: Dieses Codebeispiel beinhaltet keine Ausgabe, da die GetImageFrame Aktion einen Stream von Pixeldaten an die Datei imageframe.jph zurückgibt. Informationen zum Dekodieren und Anzeigen von Bildrahmen finden Sie unter „HTJ2K decoding libraries“.

Weitere Informationen finden Sie im Entwicklerhandbuch unter Abrufen von Pixeldaten von Bilddatensätzen.AWS HealthImaging

- Einzelheiten zur API finden Sie GetImageFramein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
        public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                        String destinationPath,
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()

.imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
```

```
                        System.exit(1);
                }
        }
```

- Einzelheiten zur API finden Sie GetImageFramein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-
encoded image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);
```

```
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
};
```

- Einzelheiten zur API finden Sie GetImageFramein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.
```

```
        :param file_path_to_write: The path to write the image frame's HTJ2K
encoded pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie GetImageFramein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        " iv_image_set_id = '1234567890123456789012345678901234567890'
        " iv_image_frame_id = '1234567890123456789012345678901234567890'
        oo_result = lo_mig->getimageframe(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
            iv_imageframeid = iv_image_frame_id ) ).
        DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
        MESSAGE 'Image frame retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image frame not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetImageFramein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`GetImageSet`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetImageSet` verwendet wird.

CLI

AWS CLI

So rufen Sie Bildsatzeigenschaften ab

Im folgenden Codebeispiel für `get-image-set` werden die Eigenschaften für einen Bildsatz abgerufen.

```
aws medical-imaging get-image-set \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 18f88ac7870584f58d56256646b4d92b \
    --version-id 1
```

Ausgabe:

```
{
    "versionId": "1",
    "imageSetWorkflowStatus": "COPIED",
    "updatedAt": 1680027253.471,
    "imageSetId": "18f88ac7870584f58d56256646b4d92b",
    "imageSetState": "ACTIVE",
    "createdAt": 1679592510.753,
    "datastoreId": "12345678901234567890123456789012"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Abrufen von Bilddatensatz-Eigenschaften.

- Einzelheiten zur API finden Sie GetImageSetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
 medicalImagingClient,
            String datastoreId,
            String imagesetId,
            String versionId) {
        try {
```

```
            GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId);

            if (versionId != null) {
                getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
            }

            return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Einzelheiten zur API finden Sie GetImageSetin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
 *
 */
```

```
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie GetImageSetin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return image_set
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie GetImageSetin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '1234567890123456789012345678901234567890'
        " iv_image_set_id = '1234567890123456789012345678901234567890'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimageset(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_state) = oo_result->get_imagesetstate( ).
        MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetImageSetin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`GetImageSetMetadata`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetImageSetMetadata` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
```

```
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 "", outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." <<
 std::endl;
```

```
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 versionID, outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." <<
 std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

- Einzelheiten zur API finden Sie GetImageSetMetadatain der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

Beispiel 1: So rufen Sie Bildsatz-Metadaten ohne Version ab

Im folgenden Codebeispiel für `get-image-set-metadata` werden Metadaten für einen Bildsatz abgerufen, ohne eine Version anzugeben.

Anmerkung: Der Parameter `outfile` ist erforderlich.

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    studymetadata.json.gz
```

Die zurückgegebenen Metadaten werden mit GZIP komprimiert und in der Datei „studymetadata.json.gz“ gespeichert. Um den Inhalt des zurückgegebenen JSON-Objekts anzuzeigen, müssen Sie es zuerst dekomprimieren.

Ausgabe:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Beispiel 2: So rufen Sie Bildsatz-Metadaten mit Version ab

Im folgenden Codebeispiel für `get-image-set-metadata` werden Metadaten für einen Bildsatz mit einer bestimmten Version abgerufen.

Anmerkung: Der Parameter `outfile` ist erforderlich.

```
aws medical-imaging get-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --version-id 1 \
    studymetadata.json.gz
```

Die zurückgegebenen Metadaten werden mit GZIP komprimiert und in der Datei „studymetadata.json.gz“ gespeichert. Um den Inhalt des zurückgegebenen JSON-Objekts anzuzeigen, müssen Sie es zuerst dekomprimieren.

Ausgabe:

```
{
    "contentType": "application/json",
    "contentEncoding": "gzip"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Abrufen von Bilddatensatz-Metadaten.

- Einzelheiten zur API finden Sie GetImageSetMetadatain der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
           String destinationPath,
           String datastoreId,
           String imagesetId,
           String versionId) {

       try {
           GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder
= GetImageSetMetadataRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
           }


medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                   FileSystems.getDefault().getPath(destinationPath));

           System.out.println("Metadata downloaded to " + destinationPath);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie GetImageSetMetadatain der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";

/**
 * @param {string} metadataFileName - The name of the file for the gzipped
 metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
```

```
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
  try {
    await getImageSetMetadata(
      "metadata.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
  try {
    await getImageSetMetadata(
      "metadata2.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
      "1",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

- Einzelheiten zur API finden Sie GetImageSetMetadatain der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:

                image_set_metadata =
 self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
```

```
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie GetImageSetMetadatain AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_version_id = '1' (optional)
        IF iv_version_id IS NOT INITIAL.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id
            iv_versionid = iv_version_id ).
        ELSE.
          oo_result = lo_mig->getimagesetmetadata(
            iv_datastoreid = iv_datastore_id
            iv_imagesetid = iv_image_set_id ).
        ENDIF.
        DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
        MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetImageSetMetadatain der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`ListDICOMImportJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDICOMImportJobs` verwendet wird.

CLI

AWS CLI

So listen Sie DICOM-Importaufträge auf

Im folgenden Codebeispiel für `list-dicom-import-jobs` werden die DICOM-Importaufträge aufgelistet.

```
aws medical-imaging list-dicom-import-jobs \
    --datastore-id "12345678901234567890123456789012"
```

Ausgabe:

```
{
    "jobSummaries": [
        {
            "jobId": "09876543210987654321098765432109",
            "jobName": "my-job",
            "jobStatus": "COMPLETED",
            "datastoreId": "12345678901234567890123456789012",
            "dataAccessRoleArn": "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole",
            "endedAt": "2022-08-12T11:21:56.504000+00:00",
            "submittedAt": "2022-08-12T11:20:21.734000+00:00"
        }
```

```
    ]
}
```

Weitere Informationen finden Sie im AWS HealthImaging Developer Guide unter Auflisten von Importaufträgen.

- Einzelheiten zur API finden Sie ListDICOMImportJobsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
           String datastoreId) {

       try {
           ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                   .datastoreId(datastoreId)
                   .build();
           ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
           return response.jobSummaries();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return new ArrayList<>();
   }
```

- Einzelheiten zur API finden Sie ListDICOMImportJobsin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/
dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
```

```
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
  // }
  // ]}

  return jobSummaries;
};
```

- Einzelheiten zur API finden Sie ListDICOMImportJobsin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_dicom_import_jobs(self, datastore_id):
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
```

```
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie ListDICOMImportJobsin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->listdicomimportjobs( iv_datastoreid =
iv_datastore_id ).
        DATA(lt_jobs) = oo_result->get_jobsummaries( ).
        DATA(lv_count) = lines( lt_jobs ).
        MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
```

```
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListDICOMImportJobsin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`ListDatastores`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDatastores` verwendet wird.

Bash

AWS CLI mit Bash-Skript

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
```

```
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_list_datastores() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi
```

```
  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie ListDatastoresin der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So listen Sie Datenspeicher auf

Im folgenden Codebeispiel für `list-datastores` werden die verfügbaren Datenspeicher aufgelistet.

```
aws medical-imaging list-datastores
```

Ausgabe:

```
{
    "datastoreSummaries": [
        {
            "datastoreId": "12345678901234567890123456789012",
            "datastoreName": "TestDatastore123",
            "datastoreStatus": "ACTIVE",
            "datastoreArn": "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012",
            "createdAt": "2022-11-15T23:33:09.643000+00:00",
            "updatedAt": "2022-11-15T23:33:09.643000+00:00"
        }
    ]
}
```

Weitere Informationen finden Sie im AWS HealthImaging Developer Guide unter Auflisten von Datenspeichern.

- Einzelheiten zur API finden Sie ListDatastoresin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest =
ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- Einzelheiten zur API finden Sie ListDatastoresin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
```

```
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- Einzelheiten zur API finden Sie ListDatastoresin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
        """
        try:
            paginator =
 self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list data stores. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie ListDatastoresin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
TRY.
    oo_result = lo_mig->listdatastores( ).
    DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
    DATA(lv_count) = lines( lt_datastores ).
    MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListDatastoresin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`ListImageSetVersions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListImageSetVersions` verwendet wird.

CLI

AWS CLI

So listen Sie Versionen von Bildsätzen auf

Im folgendem Codebeispiel für `list-image-set-versions` wird der Versionsverlauf für einen Bildsatz aufgelistet.

```
aws medical-imaging list-image-set-versions \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e
```

Ausgabe:

```
{
    "imageSetPropertiesList": [
        {
            "ImageSetWorkflowStatus": "UPDATED",
            "versionId": "4",
            "updatedAt": 1680029436.304,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "UPDATED",
```

```
            "versionId": "3",
            "updatedAt": 1680029163.325,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "createdAt": 1680027126.436
        },
        {
            "ImageSetWorkflowStatus": "COPY_FAILED",
            "versionId": "2",
            "updatedAt": 1680027455.944,
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "message": "INVALID_REQUEST:  Series of SourceImageSet and
 DestinationImageSet don't match.",
            "createdAt": 1680027126.436
        },
        {
            "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
            "imageSetState": "ACTIVE",
            "versionId": "1",
            "ImageSetWorkflowStatus": "COPIED",
            "createdAt": 1680027126.436
        }
    ]
}
```

Weitere Informationen finden Sie im AWS HealthImaging Developer Guide unter Auflisten von Imageset-Versionen.

- Einzelheiten zur API finden Sie ListImageSetVersionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static List<ImageSetProperties>
 listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
            String datastoreId,
            String imagesetId) {
        try {
            ListImageSetVersionsRequest getImageSetRequest =
 ListImageSetVersionsRequest.builder()
                    .datastoreId(datastoreId)
```

```
                    .imageSetId(imagesetId)
                    .build();

            ListImageSetVersionsIterable responses = medicalImagingClient
                    .listImageSetVersionsPaginator(getImageSetRequest);
            List<ImageSetProperties> imageSetProperties = new ArrayList<>();
            responses.stream().forEach(response ->
 imageSetProperties.addAll(response.imageSetPropertiesList()));

            return imageSetProperties;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Einzelheiten zur API finden Sie ListImageSetVersionsin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
export const listImageSetVersions = async (
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
```

```
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- Einzelheiten zur API finden Sie ListImageSetVersionsin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set_properties_list
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie ListImageSetVersionsin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
TRY.
    " iv_datastore_id = '1234567890123456789012345678901234567890'
    " iv_image_set_id = '1234567890123456789012345678901234567890'
    oo_result = lo_mig->listimagesetversions(
      iv_datastoreid = iv_datastore_id
      iv_imagesetid = iv_image_set_id ).
    DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
    DATA(lv_count) = lines( lt_versions ).
    MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Image set not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
```

```
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListImageSetVersionsin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`ListTagsForResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForResource` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Taggen eines Datenspeichers
- Taggen eines Imagesatzes

CLI

AWS CLI

Beispiel 1: So listen Sie Ressourcen-Tags für einen Datenspeicher auf

Im folgenden Codebeispiel für `list-tags-for-resource` werden die Tags für einen Datenspeicher aufgelistet.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

Ausgabe:

```
{
    "tags":{
        "Deployment":"Development"
```

```
    }
}
```

Beispiel 2: So listen Sie Ressourcen-Tags für einen Bildsatz auf

Im folgenden Codebeispiel für `list-tags-for-resource` werden die Tags für einen Bildsatz aufgelistet.

```
aws medical-imaging list-tags-for-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b"
```

Ausgabe:

```
{
    "tags":{
        "Deployment":"Development"
    }
}
```

Weitere Informationen finden Sie AWS HealthImaging im AWS HealthImaging Entwicklerhandbuch unter Ressourcen taggen mit.

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
```

```
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie ListTagsForResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn =
 iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`SearchImageSets`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchImageSets` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
```

```
                                                Aws::Vector<Aws::String>
 &imageSetResults,
                                                const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {

 imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}
```

Anwendungsfall 1: EQUAL-Operator.

```
        Aws::Vector<Aws::String> imageIDsForPatientID;
        Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
```

```
        Aws::Vector<Aws::MedicalImaging::Model::SearchFilter>
 patientIDSearchFilters = {

 Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Oper

 .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(pat
        };

        searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
        bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

 searchCriteriaEqualsPatientID,

 imageIDsForPatientID,
                                                                  clientConfig);
        if (result) {
            std::cout << imageIDsForPatientID.size() << " image sets found for
 the patient with ID '"
            <<  patientID << "'." << std::endl;
            for (auto &imageSetResult : imageIDsForPatientID) {
                std::cout << "  Image set with ID '" << imageSetResult <<
 std::endl;
            }
        }
```

Anwendungsfall #2: BETWEEN-Operator, der DICOMStudyDate und verwendet DICOMStudyTime.

```
         Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

 useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAnd
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

 useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTi

 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeSt
%m%d"))
        .WithDICOMStudyTime("000000.000"));
```

```
        Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
        useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
        useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

        Aws::Vector<Aws::String> usesCase2Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase2SearchCriteria,
                                                          usesCase2Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase2Results.size() << " image sets found for
between 1999/01/01 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase2Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
        useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
        useCase3SearchCriteria.SetFilters({useCase3SearchFilter});
```

```
        Aws::Vector<Aws::String> usesCase3Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase3SearchCriteria,
                                                          usesCase3Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase3Results.size() << " image sets found for
created between 2023/11/30 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase3Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

Anwendungsfall #4: EQUAL-Operator on DICOMSeriesInstanceUID und BETWEEN für updatedAt und sortiere die Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::Da

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
        useCase4SearchFilterBetween.SetValues({useCase4StartDate,
useCase4EndDate});

useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
        seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

        Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
        useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

        Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
```

```
        useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

        Aws::MedicalImaging::Model::Sort useCase4Sort;

useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
        useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

        useCase4SearchCriteria.SetSort(useCase4Sort);

        Aws::Vector<Aws::String> usesCase4Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase4SearchCriteria,
                                                          usesCase4Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
            << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort
response\n"
            <<  "in ASC order on updatedAt field." <<  std::endl;
            for (auto &imageSetResult : usesCase4Results) {
                std::cout << "  Image set with ID '" << imageSetResult <<
std::endl;
            }
        }
```

- Einzelheiten zur API finden Sie in der API-Referenz. SearchImageSetsAWS SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

Beispiel 1: So suchen Sie nach Bilddatensätzen mit einem EQUAL-Operator

Im folgenden Codebeispiel für `search-image-sets` wird der EQUAL-Operator verwendet, um Bildsätze auf der Grundlage eines bestimmten Werts zu durchsuchen.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Inhalt von `search-criteria.json`

```
{
    "filters": [{
        "values": [{"DICOMPatientId" : "SUBJECT08701"}],
        "operator": "EQUAL"
    }]
}
```

Ausgabe:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
             "DICOMPatientSex": "F",
             "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
             "DICOMPatientBirthDate": "19201120",
             "DICOMStudyDescription": "UNKNOWN",
             "DICOMPatientId": "SUBJECT08701",
             "DICOMPatientName": "Melissa844 Huel628",
             "DICOMNumberOfStudyRelatedInstances": 1,
             "DICOMStudyTime": "140728",
             "DICOMNumberOfStudyRelatedSeries": 1
            },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Beispiel 2: Um Bilddatensätze mit einem BETWEEN-Operator mithilfe von DICOMStudyDate und zu durchsuchen DICOMStudyTime

Im folgenden Codebeispiel für `search-image-sets` wird nach Bildsätzen mit DICOM-Studien gesucht, die zwischen dem 1. Januar 1990 (12:00 Uhr) und dem 1. Januar 2023 (12:00 Uhr) generiert wurden.

Hinweis: DICOMStudyTime ist optional. Wenn der Parameter nicht vorhanden ist, ist 12:00 Uhr (Beginn des Tages) der Zeitwert für die Datumsangaben, die für die Filterung bereitgestellt werden.

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria file://search-criteria.json
```

Inhalt von `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "19900101",
                "DICOMStudyTime": "000000"
            }
        },
        {
            "DICOMStudyDateAndTime": {
                "DICOMStudyDate": "20230101",
                "DICOMStudyTime": "000000"
            }
        }],
        "operator": "BETWEEN"
    }]
}
```

Ausgabe:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
```

```
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "updatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Beispiel 3: So suchen Sie Bildsätze mit einem BETWEEN-Operator mithilfe von createdAt (Zeitstudien wurden bisher fortgeführt)

Im folgenden `search-image-sets` Codebeispiel wird nach Bilddatensätzen gesucht, bei denen DICOM-Studien HealthImaging zwischen den Zeitbereichen in der UTC-Zeitzone persistiert wurden.

Hinweis: Geben Sie createdAt im Beispielformat an („T23:20:50.52Z1985-04-12 „).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Inhalt von `search-criteria.json`

```
{
    "filters": [{
        "values": [{
            "createdAt": "1985-04-12T23:20:50.52Z"
        },
        {
            "createdAt": "2022-04-12T23:20:50.52Z"
        }],
        "operator": "BETWEEN"
    }]
}
```

Ausgabe:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Beispiel 4: Um Bilddatensätze mit einem EQUAL-Operator on DICOMSeriesInstanceUID und BETWEEN auf updatedAt zu durchsuchen und die Antwort in ASC-Reihenfolge im Feld updatedAt zu sortieren

Das folgende `search-image-sets` Codebeispiel sucht nach Bilddatensätzen mit einem EQUAL-Operator für DICOMSeriesInstanceUID und BETWEEN für updatedAt und der Sortierantwort in ASC-Reihenfolge für das Feld updatedAt.

Hinweis: Geben Sie updatedAt im Beispielformat an („T23:20:50.52Z1985-04-12 „).

```
aws medical-imaging search-image-sets \
    --datastore-id 12345678901234567890123456789012 \
    --search-criteria  file://search-criteria.json
```

Inhalt von `search-criteria.json`

```
{
    "filters": [{
        "values": [{
```

```
            "updatedAt": "2024-03-11T15:00:05.074000-07:00"
        }, {
            "updatedAt": "2024-03-11T16:00:05.074000-07:00"
        }],
        "operator": "BETWEEN"
    }, {
        "values": [{
            "DICOMSeriesInstanceUID": "1.2.840.99999999.84710745.943275268089"
        }],
        "operator": "EQUAL"
    }],
    "sort": {
        "sortField": "updatedAt",
        "sortOrder": "ASC"
    }
}
```

Ausgabe:

```
{
    "imageSetsMetadataSummaries": [{
        "imageSetId": "09876543210987654321098765432109",
        "createdAt": "2022-12-06T21:40:59.429000+00:00",
        "version": 1,
        "DICOMTags": {
            "DICOMStudyId": "2011201407",
            "DICOMStudyDate": "19991122",
            "DICOMPatientSex": "F",
            "DICOMStudyInstanceUID": "1.2.840.99999999.84710745.943275268089",
            "DICOMPatientBirthDate": "19201120",
            "DICOMStudyDescription": "UNKNOWN",
            "DICOMPatientId": "SUBJECT08701",
            "DICOMPatientName": "Melissa844 Huel628",
            "DICOMNumberOfStudyRelatedInstances": 1,
            "DICOMStudyTime": "140728",
            "DICOMNumberOfStudyRelatedSeries": 1
        },
        "lastUpdatedAt": "2022-12-06T21:40:59.429000+00:00"
    }]
}
```

Weitere Informationen finden Sie im Entwicklerhandbuch unter Suchen von Bilddatensätzen.AWS HealthImaging

- Einzelheiten zur API finden Sie SearchImageSetsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
    public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
            MedicalImagingClient medicalImagingClient,
            String datastoreId, SearchCriteria searchCriteria) {
        try {
            SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                    .datastoreId(datastoreId)
                    .searchCriteria(searchCriteria)
                    .build();
            SearchImageSetsIterable responses = medicalImagingClient
                    .searchImageSetsPaginator(datastoreRequest);
            List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();

            responses.stream().forEach(response -> imageSetsMetadataSummaries
                    .addAll(response.imageSetsMetadataSummaries()));

            return imageSetsMetadataSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Anwendungsfall 1: EQUAL-Operator.

```
        List<SearchFilter> searchFilters =
Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
```

```
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:
\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudyDate und DICOMStudyTime.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

 .dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
```

```
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator
using createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall #4: EQUAL-Operator on DICOMSeriesInstanceUID und BETWEEN für updatedAt und sortiere die Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- Einzelheiten zur API finden Sie in der API-Referenz. SearchImageSetsAWS SDK for Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters -
 The search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };

  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
```

```
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Anwendungsfall 1: EQUAL-Operator.

```
  const datastoreId = "12345678901234567890123456789012";

  try {
    const searchCriteria = {
      filters: [
        {
          values: [{ DICOMPatientId: "1234567" }],
          operator: "EQUAL",
        },
      ],
    };
```

```
  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall #2: BETWEEN-Operator verwendet DICOMStudyDate und DICOMStudyTime.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
const datastoreId = "12345678901234567890123456789012";
```

```
try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall #4: EQUAL-Operator on DICOMSeriesInstanceUID und BETWEEN für updatedAt und sortiere die Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
          { updatedAt: new Date() },
        ],
        operator: "BETWEEN",
      },
      {
        values: [
          {
            DICOMSeriesInstanceUID:
              "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
          },
        ],
        operator: "EQUAL",
      },
```

```
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- Einzelheiten zur API finden Sie in der API-Referenz. SearchImageSetsAWS SDK für JavaScript

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
 [{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
```

```
        """
        try:
            paginator =
self.health_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries
```

Anwendungsfall 1: EQUAL-Operator.

```
        search_filter = {
            "filters": [
                {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Anwendungsfall #2: BETWEEN-Operator verwendet DICOMStudyDate und DICOMStudyTime.

```
        search_filter = {
            "filters": [
                {
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
```

```
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
DICOMStudyTime\n{image_sets}"
        )
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                }
            ]
        }
```

```
        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Anwendungsfall #4: EQUAL-Operator on DICOMSeriesInstanceUID und BETWEEN für updatedAt und sortiere die Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
 BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

Der folgende Code instanziiert das Objekt. MedicalImagingWrapper

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie SearchImageSetsin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
TRY.
    " iv_datastore_id = '1234567890123456789012345678901234567890'
    oo_result = lo_mig->searchimagesets(
      iv_datastoreid = iv_datastore_id
      io_searchcriteria = io_search_criteria ).
    DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
    DATA(lv_count) = lines( lt_imagesets ).
    MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict error.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie SearchImageSetsin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`StartDICOMImportJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartDICOMImportJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Bildsätzen und Bild-Frames

C++

SDK für C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
```

```
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie StartDICOMImportJobin der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

CLI

AWS CLI

So starten Sie einen DICOM-Importauftrag

Im folgenden Codebeispiel für `start-dicom-import-job` wird ein DICOM-Importauftrage gestartet.

```
aws medical-imaging start-dicom-import-job \
    --job-name "my-job" \
    --datastore-id "12345678901234567890123456789012" \
    --input-s3-uri "s3://medical-imaging-dicom-input/dicom_input/" \
    --output-s3-uri "s3://medical-imaging-output/job_output/" \
    --data-access-role-arn "arn:aws:iam::123456789012:role/
ImportJobDataAccessRole"
```

Ausgabe:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "jobId": "09876543210987654321098765432109",
    "jobStatus": "SUBMITTED",
    "submittedAt": "2022-08-12T11:28:11.152000+00:00"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Einen Importauftrag starten.

- Einzelheiten zur API finden Sie StartDICOMImportJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static String startDicomImportJob(MedicalImagingClient
 medicalImagingClient,
            String jobName,
            String datastoreId,
            String dataAccessRoleArn,
            String inputS3Uri,
```

```
            String outputS3Uri) {

        try {
            StartDicomImportJobRequest startDicomImportJobRequest =
 StartDicomImportJobRequest.builder()
                    .jobName(jobName)
                    .datastoreId(datastoreId)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .inputS3Uri(inputS3Uri)
                    .outputS3Uri(outputS3Uri)
                    .build();
            StartDicomImportJobResponse response =
 medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
            return response.jobId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- Einzelheiten zur API finden Sie StartDICOMImportJobin der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
 * @param {string} datastoreId - The ID of the data store.
```

```
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input
 files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files
 are stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- Einzelheiten zur API finden Sie StartDICOMImportJobin der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.

        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for
 the job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
 files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie StartDICOMImportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_job_name = 'import-job-1'
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
        " iv_input_s3_uri = 's3://my-bucket/input/'
        " iv_output_s3_uri = 's3://my-bucket/output/'
        oo_result = lo_mig->startdicomimportjob(
          iv_jobname = iv_job_name
          iv_datastoreid = iv_datastore_id
          iv_dataaccessrolearn = iv_role_arn
          iv_inputs3uri = iv_input_s3_uri
```

```
          iv_outputs3uri = iv_output_s3_uri ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartDICOMImportJobin der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`TagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Taggen eines Datenspeichers
- Taggen eines Imagesatzes

CLI

AWS CLI

Beispiel 1: So markieren Sie einen Datenspeicher

In den folgenden Codebeispielen für `tag-resource` wird ein Datenspeicher mit Tags versehen.

```
aws medical-imaging tag-resource \
  --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
  --tags '{"Deployment":"Development"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So taggen Sie einen Bildsatz

In den folgenden Codebeispielen für `tag-resource` wird ein Bildsatz mit Tags versehen.

```
aws medical-imaging tag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tags '{"Deployment":"Development"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie AWS HealthImaging im AWS HealthImaging Entwicklerhandbuch unter Ressourcen taggen mit.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
```

```
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
```

```
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //       httpStatusCode: 204,
  //        requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
```

```
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
```

```
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie TagResourcein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`UntagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResource` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Taggen eines Datenspeichers
- Taggen eines Imagesatzes

CLI

AWS CLI

Beispiel 1: So heben Sie die Markierung eines Datenspeichers auf

Im folgenden Codebeispiel für `untag-resource` werden Tags aus einem Datenspeicher entfernt.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012" \
```

```
    --tag-keys '["Deployment"]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So heben Sie das Tagging eines Bildsatzes auf

Im folgenden Codebeispiel für `untag-resource` werden Tags aus einem Bildsatz entfernt.

```
aws medical-imaging untag-resource \
    --resource-arn "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/18f88ac7870584f58d56256646b4d92b" \
    --tag-keys '["Deployment"]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie AWS HealthImaging im AWS HealthImaging Entwicklerhandbuch unter Ressourcen taggen mit.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
           UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tagKeys(tagKeys)
                   .build();

           medicalImagingClient.untagResource(untagResourceRequest);

           System.out.println("Tags have been removed from the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie UntagResourcein der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
```

```
  // }

  return response;
};
```

- Einzelheiten zur API finden Sie UntagResourcein der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Einzelheiten zur API finden Sie UntagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UntagResourcein der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie **`UpdateImageSetMetadata`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateImageSetMetadata` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie ein Attribut in Bildsatz-Metadaten ein oder aktualisieren es

Im folgenden Beispiel für `update-image-set-metadata` wird ein Attribut in die Metadaten eines Bildsatzes eingefügt.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":
{\"PatientName\":\"MX^MX\"}}}"
    }
}
```

Ausgabe:

```
{
```

```
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 2: So entfernen Sie ein Attribut aus den Metadaten eines Bildsatzes

Im folgenden Beispiel für `update-image-set-metadata` wird ein Attribut aus den Metadaten eines Bildsatzes entfernt.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"DICOM\":
{\"StudyDescription\":\"CHEST\"}}}"
    }
}
```

Ausgabe:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 3: So entfernen Sie eine Instance aus den Metadaten eines Bildsatzes

Im folgenden Beispiel für `update-image-set-metadata` wird eine Instance aus den Metadaten eines Bildsatzes entfernt.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id ea92b0d8838c72a3f25d00d13616f87e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates file://metadata-updates.json \
    --force
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "removableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances\":
 {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {}}}}}}"
    }
}
```

Ausgabe:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "ea92b0d8838c72a3f25d00d13616f87e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 4: So setzen Sie einen Bildsatz auf eine frühere Version zurück

Das folgende `update-image-set-metadata` Beispiel zeigt, wie ein Bilddatensatz auf eine frühere Version zurückgesetzt wird. CopyImageSet und UpdateImageSetMetadata Aktionen erstellen neue Versionen von Bilddatensätzen.

```
aws medical-imaging update-image-set-metadata \
```

```
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 3 \
    --cli-binary-format raw-in-base64-out \
    --update-image-set-metadata-updates '{"revertToVersionId": "1"}'
```

Ausgabe:

```
{
    "datastoreId": "12345678901234567890123456789012",
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "latestVersionId": "4",
    "imageSetState": "LOCKED",
    "imageSetWorkflowStatus": "UPDATING",
    "createdAt": 1680027126.436,
    "updatedAt": 1680042257.908
}
```

Beispiel 5: So fügen Sie einer Instance ein privates DICOM-Datenelement hinzu

Das folgende Beispiel für `update-image-set-metadata` zeigt, wie ein privates Element zu einer angegebenen Instance innerhalb eines Bildsatzes hinzugefügt wird. Der DICOM-Standard erlaubt private Datenelemente für die Übertragung von Informationen, die nicht in Standarddatenelementen enthalten sein dürfen. Mit der UpdateImageSetMetadata Aktion können Sie private Datenelemente erstellen, aktualisieren und löschen.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"001910F9\": \"97\"},\"DICOMVRs\": {\"001910F9\": \"DS\"}}}}}}"
```

```
    }
}
```

Ausgabe:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 6: So aktualisieren Sie ein privates DICOM-Datenelement in einer Instance

Das folgende Beispiel für `update-image-set-metadata` zeigt, wie der Wert eines privates Datenelements, das zu einer Instance eines Bildsatzes gehört, aktualisiert wird.

```
aws medical-imaging update-image-set-metadata \
    --datastore-id 12345678901234567890123456789012 \
    --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
    --latest-version-id 1 \
    --cli-binary-format raw-in-base64-out \
    --force \
    --update-image-set-metadata-updates file://metadata-updates.json
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\": 1.1,\"Study\": {\"Series
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"Instances
\": {\"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1\": {\"DICOM\":
 {\"00091001\": \"GE_GENESIS_DD\"}}}}}}"
    }
}
```

Ausgabe:

```
{
```

```
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Beispiel 7: So aktualisieren Sie eine SOPInstanceUID mit dem Force-Parameter

Im folgenden Beispiel für `update-image-set-metadata` wird gezeigt, wie eine SOPInstanceUID unter Verwendung des Force-Parameter aktualisiert wird, um die DICOM-Metadatenbeschränkungen zu überschreiben.

```
aws medical-imaging update-image-set-metadata \
        --datastore-id 12345678901234567890123456789012 \
        --image-set-id 53d5fdb05ca4d46ac7ca64b06545c66e \
        --latest-version-id 1 \
        --cli-binary-format raw-in-base64-out \
        --force \
        --update-image-set-metadata-updates file://metadata-updates.json
```

Inhalt von `metadata-updates.json`

```
{
    "DICOMUpdates": {
        "updatableAttributes": "{\"SchemaVersion\":1.1,\"Study\":{\"Series
\":{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3656.0\":
{\"Instances\":
{\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.0\":{\"DICOM\":
{\"SOPInstanceUID\":
\"1.3.6.1.4.1.5962.99.1.3633258862.2104868982.1369432891697.3659.9\"}}}}}}}"
    }
}
```

Ausgabe:

```
{
    "latestVersionId": "2",
    "imageSetWorkflowStatus": "UPDATING",
    "updatedAt": 1680042257.908,
```

```
    "imageSetId": "53d5fdb05ca4d46ac7ca64b06545c66e",
    "imageSetState": "LOCKED",
    "createdAt": 1680027126.436,
    "datastoreId": "12345678901234567890123456789012"
}
```

Weitere Informationen finden Sie im AWS HealthImaging Entwicklerhandbuch unter Aktualisieren von Bilddatensatz-Metadaten.

- Einzelheiten zur API finden Sie UpdateImageSetMetadatain der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service
exceptions thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
           UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                   .builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imageSetId)
```

```
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .force(force)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                              }
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
```

```
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeInstance

.getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- Einzelheiten zur API finden Sie UpdateImageSetMetadatain der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
```

```
/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set
 version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
```

```
  } catch (err) {
    console.error(err);
  }
};
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es und erzwingen Sie das Update.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        updatableAttributes: new TextEncoder().encode(insertAttributes),
      },
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
      true,
    );
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
    // Attribute key and value must match the existing attribute.
    const remove_attribute = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
```

```
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_attribute),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
const remove_instance = JSON.stringify({
  SchemaVersion: 1.1,
  Study: {
    Series: {
      "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
        Instances: {
          "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
        },
      },
    },
  },
});

const updateMetadata = {
  DICOMUpdates: {
    removableAttributes: new TextEncoder().encode(remove_instance),
  },
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
const updateMetadata = {
  revertToVersionId: "1",
};

await updateImageSetMetadata(
  datastoreID,
  imageSetID,
  versionID,
  updateMetadata,
);
```

- Einzelheiten zur API finden Sie UpdateImageSetMetadatain der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
```

```
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata =
 self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
```

```
                }
            }"""
        metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

        self.update_image_set_metadata(
            data_store_id, image_set_id, version_id, metadata, force
        )
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
        # Attribute key and value must match the existing attribute.
        attributes = """{
                "SchemaVersion": 1.1,
                "Study": {
                    "DICOM": {
                        "StudyDescription": "CT CHEST"
                    }
                }
            }"""
        metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

        self.update_image_set_metadata(
            data_store_id, image_set_id, version_id, metadata, force
        )
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
        attributes = """{
                "SchemaVersion": 1.1,
                "Study": {
                    "Series": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                            }
                    }
                }
            }"""
```

```
            metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
            metadata = {"revertToVersionId": "1"}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

- Einzelheiten zur API finden Sie UpdateImageSetMetadatain AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_latest_version_id = '1'
        " iv_force = abap_false
        oo_result = lo_mig->updateimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_latestversionid = iv_latest_version_id
          io_updateimagesetmetupdates = io_metadata_updates
          iv_force = iv_force ).
```

```
        DATA(lv_new_version) = oo_result->get_latestversionid( ).
        MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateImageSetMetadatain der AWS API-Referenz zum SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien für die HealthImaging Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien HealthImaging mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb HealthImaging oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Beginnen Sie mit HealthImaging Bildsätzen und Bildrahmen mithilfe eines AWS SDK
- Kennzeichnen eines HealthImaging Datenspeichers mit einem AWS SDK
- Einen HealthImaging Bilddatensatz mit einem taggen AWS SDK

Beginnen Sie mit HealthImaging Bildsätzen und Bildrahmen mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie DICOM-Dateien importieren und Bildrahmen herunterladen. HealthImaging

Die Implementierung ist als Befehlszeilenanwendung strukturiert.

- Richten Sie Ressourcen für einen DICOM-Import ein.
- Importieren Sie DICOM-Dateien in einen Datenspeicher.
- Rufen Sie die Bildsatz-IDs für den Importauftrag ab.
- Rufen Sie die Bild-Frame-IDs für den Importauftrag ab.
- Laden Sie die Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.
- Bereinigen Sie die Ressourcen.

C++

SDK für C++

Erstellen Sie einen CloudFormation Stapel mit den erforderlichen Ressourcen.

```
    Aws::String inputBucketName;
    Aws::String outputBucketName;
    Aws::String dataStoreId;
    Aws::String roleArn;
    Aws::String stackName;

    if (askYesNoQuestion(
            "Would you like to let this workflow create the resources for you?
(y/n) ")) {
        stackName = askQuestion(
                "Enter a name for the AWS CloudFormation stack to create. ");
        Aws::String dataStoreName = askQuestion(
                "Enter a name for the HealthImaging datastore to create. ");

        Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
```

```
                stackName,
                dataStoreName,
                clientConfiguration);

        if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
outputBucketName,
                             roleArn)) {
            return false;
        }

        std::cout << "The following resources have been created." << std::endl;
        std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
                  << std::endl;
        std::cout << "An Amazon S3 input bucket named: " << inputBucketName <<
"."
                  << std::endl;
        std::cout << "An Amazon S3 output bucket named: " << outputBucketName <<
"."
                  << std::endl;
        std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
        askQuestion("Enter return to continue.", alwaysTrueTest);
    }
    else {
        std::cout << "You have chosen to use preexisting resources:" <<
std::endl;
        dataStoreId = askQuestion(
                "Enter the data store ID of the HealthImaging datastore you wish
to use: ");
        inputBucketName = askQuestion(
                "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
                "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
                "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
```

Kopieren Sie DICOM-Dateien in den Amazon-S3-Import-Bucket.

```
    std::cout
            << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
```

```
            << "Commons (IDC) Collections." << std::endl;
   std::cout << "Here is the link to their website." << std::endl;
   std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
   std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
             << std::endl;
   std::cout
            << "First one of the DICOM folders in the IDC collection must be
copied to your\n"
              "input S3 bucket."
            << std::endl;
   std::cout << "You have the choice of one of the following "
             << IDC_ImageChoices.size() << " folders to copy." << std::endl;

   int index = 1;
   for (auto &idcChoice: IDC_ImageChoices) {
       std::cout << index << " - " << idcChoice.mDescription << std::endl;
       index++;
   }
   int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

   Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
   Aws::String inputDirectory = "input";

   std::cout << "The files in the directory '" << fromDirectory << "' in the
bucket '"
             << IDC_S3_BucketName << "' will be copied " << std::endl;
   std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
             << "' in the bucket '" << inputBucketName << "'." << std::endl;
   askQuestion("Enter return to start the copy.", alwaysTrueTest);

   if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
           IDC_S3_BucketName,
           fromDirectory,
           inputBucketName,
           inputDirectory, clientConfiguration)) {
       std::cerr << "This workflow will exit because of an error." << std::endl;
       cleanup(stackName, dataStoreId, clientConfiguration);
       return false;
   }
```

Importieren Sie die DICOM-Dateien in den Amazon-S3-Datenspeicher.

```
bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                               const Aws::String
 &inputBucketName,
                                               const Aws::String &inputDirectory,
                                               const Aws::String
 &outputBucketName,
                                               const Aws::String
 &outputDirectory,
                                               const Aws::String &roleArn,
                                               Aws::String &importJobId,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = false;
    if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                            outputBucketName, outputDirectory, roleArn,
 importJobId,
                            clientConfiguration)) {
        std::cout << "DICOM import job started with job ID " << importJobId <<
 "."
                  << std::endl;
        result = waitImportJobCompleted(dataStoreID, importJobId,
 clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;

        }
    }

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM
 files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
```

```
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory +
 "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}


//! Routine which waits for a DICOM import job to complete.
/*!
 * @param dataStoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return  bool: Function succeeded.
```

```
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String
 &datastoreID,
                                                      const Aws::String
 &importJobId,
                                                      const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
 Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
 getDicomImportJobOutcome = getDICOMImportJob(
                datastoreID, importJobId,
                clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

            std::cout << "DICOM import job status: " <<

 Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                              jobStatus) << std::endl;
        }
        else {
            std::cerr << "Failed to get import job status because "
                      << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    }

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
```

```
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

Rufen Sie Bildsätze ab, die durch den DICOM-Importauftrag erstellt wurden.

```
bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
 &datastoreID,
                                                       const Aws::String
 &importJobId,
                                                       Aws::Vector<Aws::String>
 &imageSets,
                                                       const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome
 = getDICOMImportJob(
            datastoreID, importJobId, clientConfiguration);
    bool result = false;
    if (getDicomImportJobOutcome.IsSuccess()) {
        auto outputURI =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
        Aws::Http::URI uri(outputURI);
        const Aws::String &bucket = uri.GetAuthority();
        Aws::String key = uri.GetPath();
```

```
        Aws::S3::S3Client s3Client(clientConfiguration);
        Aws::S3::Model::GetObjectRequest objectRequest;
        objectRequest.SetBucket(bucket);
        objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

        auto getObjectOutcome = s3Client.GetObject(objectRequest);
        if (getObjectOutcome.IsSuccess()) {
            auto &data = getObjectOutcome.GetResult().GetBody();

            std::stringstream stringStream;
            stringStream << data.rdbuf();

            try {
                // Use JMESPath to extract the image set IDs.
                // https://jmespath.org/specification.html
                std::string jmesPathExpression =
"jobSummary.imageSetsSummary[].imageSetId";
                jsoncons::json doc = jsoncons::json::parse(stringStream.str());

                jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,

jmesPathExpression);\
                for (auto &imageSet: imageSetsJson.array_range()) {
                    imageSets.push_back(imageSet.as_string());
                }

                result = true;
            }
            catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }

        }
        else {
            std::cerr << "Failed to get object because "
                      << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
                  << getDicomImportJobOutcome.GetError().GetMessage() <<
std::endl;
```

```
    }

    return result;
}
```

Rufen Sie Bild-Frame-Informationen für Bildsätze ab.

```
bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
 &dataStoreID,
                                                         const Aws::String
 &imageSetID,
                                                         const Aws::String
 &outDirectory,

 Aws::Vector<ImageFrameInfo> &imageFrames,
                                                         const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID +
 "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for
 version ID.
                            fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {
                    throw std::runtime_error("Failed to open file " + fileName);
                }

                std::stringstream stringStream;
                stringStream << inFileStream.rdbuf();
                metadataGZip = stringStream.str();
            }
            std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                         metadataGZip.size());
            // Use JMESPath to extract the image set IDs.
            // https://jmespath.org/specification.html
            jsoncons::json doc = jsoncons::json::parse(metadataJson);
            std::string jmesPathExpression = "Study.Series.*.Instances[].*[]";
            jsoncons::json instances = jsoncons::jmespath::search(doc,
```

```
jmesPathExpression);
           for (auto &instance: instances.array_range()) {
               jmesPathExpression = "DICOM.RescaleSlope";
               std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
               jmesPathExpression = "DICOM.RescaleIntercept";
               std::string rescaleIntercept =
jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

               jmesPathExpression = "ImageFrames[][]";
               jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);

               for (auto &imageFrame: imageFramesJson.array_range()) {
                   ImageFrameInfo imageFrameIDs;
                   imageFrameIDs.mImageSetId = imageSetID;
                   imageFrameIDs.mImageFrameId = imageFrame.find(
                           "ID")->value().as_string();
                   imageFrameIDs.mRescaleIntercept = rescaleIntercept;
                   imageFrameIDs.mRescaleSlope = rescaleSlope;
                   imageFrameIDs.MinPixelValue = imageFrame.find(
                           "MinPixelValue")->value().as_string();
                   imageFrameIDs.MaxPixelValue = imageFrame.find(
                           "MaxPixelValue")->value().as_string();

                   jmesPathExpression =
"max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
                   jsoncons::json checksumJson =
jsoncons::jmespath::search(imageFrame,

jmesPathExpression);
                   imageFrameIDs.mFullResolutionChecksum =
checksumJson.as_integer<uint32_t>();

                   imageFrames.emplace_back(imageFrameIDs);
               }
           }
```

```
            result = true;
        }
        catch (const std::exception &e) {
            std::cerr << "getImageFramesForImageSet failed because " << e.what()
                      << std::endl;
        }
    }

    return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped
 json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String
 &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
```

```
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Laden Sie Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.

```
bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
        const Aws::String &dataStoreID,
        const Aws::Vector<ImageFrameInfo> &imageFrames,
        const Aws::String &outDirectory,
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
            clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
 outDirectory](
                const Aws::MedicalImaging::MedicalImagingClient *client,
                const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
                Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext>
 &context) {
```

```
                if (!handleGetImageFrameResult(outcome, outDirectory,
 imageFrame)) {
                    std::cerr << "Failed to download and convert image frame: "
                              << imageFrame.mImageFrameId << " from image set: "
                              << imageFrame.mImageSetId << std::endl;
                    result = false;
                }

                count--;
                if (count <= 0) {

                    semaphore.ReleaseAll();
                }
        }; // End of 'getImageFrameAsyncLambda' lambda.

        medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                                getImageFrameAsyncLambda);
    }

    if (count > 0) {
        semaphore.WaitOne();
    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
                  << std::endl;
    }

    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
        const Aws::String &jphFile,
        uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected
 value."
```

```
                    << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
 std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
                     OPJ_PATH_LEN);

        inFileStream = opj_stream_create_default_file_stream(
                decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                    "Unable to create input file stream for file '" + jphFile +
 "'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }

        int decodeMessageLevel = 1;
        if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
            std::cerr << "Failed to setup codec logging." << std::endl;
```

```
        }

        if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
            throw std::runtime_error("Failed to setup decompression codec.");
        }
        if (!opj_codec_set_threads(decompressorCodec, 4)) {
            throw std::runtime_error("Failed to set decompression codec
threads.");
        }

        if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
            throw std::runtime_error("Failed to read header.");
        }

        if (!opj_decode(decompressorCodec, inFileStream,
                        outputImage)) {
            throw std::runtime_error("Failed to decode.");
        }

        if (DEBUGGING) {
            std::cout << "image width : " << outputImage->x1 - outputImage->x0
                      << std::endl;
            std::cout << "image height : " << outputImage->y1 - outputImage->y0
                      << std::endl;
            std::cout << "number of channels: " << outputImage->numcomps
                      << std::endl;
            std::cout << "colorspace : " << outputImage->color_space <<
std::endl;
        }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }
```

```
    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved
 image bitmap and
//! then verifies the checksum of the bitmap.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
                uint32_t fromIndex = fromRowStart + col / image-
>comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                        buffer.size() * sizeof(myType));
```

```
    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                  << crc32Checksum << ", actual - " << crc32.checksum()
                  << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                     uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;

        if (signedData) {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,

 crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,

 crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,

 crc32Checksum);
                    break;
                default:
```

```
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
signed bytes - "
                            << bytes << std::endl;
                    break;
            }
        }
        else {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<uint8_t>(image,

crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<uint16_t>(image,

crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,

crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
unsigned bytes - "
                            << bytes << std::endl;
                    break;
            }
        }

        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                      << " signed "
                      << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
                  << std::endl;
    }
    return result;
```

```
}
```

Bereinigen von Ressourcen.

```
bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                      const Aws::String &dataStoreId,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
            "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID,
 clientConfiguration);
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

Orchestrieren Sie die Schritte (index.js).

```
import {
  parseScenarioArgs,
  Scenario,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  saveState,
  loadState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import {
  createStack,
  deployStack,
  getAccountId,
  getDatastoreName,
  getStackName,
  outputState,
  waitForStackCreation,
} from "./deploy-steps.js";
```

```
import {
  doCopy,
  selectDataset,
  copyDataset,
  outputCopiedObjects,
} from "./dataset-steps.js";
import {
  doImport,
  outputImportJobStatus,
  startDICOMImport,
  waitForImportJobCompletion,
} from "./import-steps.js";
import {
  getManifestFile,
  outputImageSetIds,
  parseManifestFile,
} from "./image-set-steps.js";
import {
  getImageSetMetadata,
  outputImageFrameIds,
} from "./image-frame-steps.js";
import { decodeAndVerifyImages, doVerify } from "./verify-steps.js";
import {
  confirmCleanup,
  deleteImageSets,
  deleteStack,
} from "./clean-up-steps.js";

const context = {};

const scenarios = {
  deploy: new Scenario(
    "Deploy Resources",
    [
      deployStack,
      getStackName,
      getDatastoreName,
      getAccountId,
      createStack,
      waitForStackCreation,
      outputState,
      saveState,
    ],
    context,
```

```
  ),
  demo: new Scenario(
    "Run Demo",
    [
      loadState,
      doCopy,
      selectDataset,
      copyDataset,
      outputCopiedObjects,
      doImport,
      startDICOMImport,
      waitForImportJobCompletion,
      outputImportJobStatus,
      getManifestFile,
      parseManifestFile,
      outputImageSetIds,
      getImageSetMetadata,
      outputImageFrameIds,
      doVerify,
      decodeAndVerifyImages,
      saveState,
    ],
    context,
  ),
  destroy: new Scenario(
    "Clean Up Resources",
    [loadState, confirmCleanup, deleteImageSets, deleteStack],
    context,
  ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Health Imaging Workflow",
    description:
      "Work with DICOM images using an AWS Health Imaging data store.",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
  });
}
```

Stellen Sie Ressourcen bereit (deploy-steps.js).

```
import fs from "node:fs/promises";
import path from "node:path";

import {
  CloudFormationClient,
  CreateStackCommand,
  DescribeStacksCommand,
} from "@aws-sdk/client-cloudformation";
import { STSClient, GetCallerIdentityCommand } from "@aws-sdk/client-sts";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const cfnClient = new CloudFormationClient({});
const stsClient = new STSClient({});

const __dirname = path.dirname(new URL(import.meta.url).pathname);
const cfnTemplatePath = path.join(
  __dirname,
  "../../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml",
);

export const deployStack = new ScenarioInput(
  "deployStack",
  "Do you want to deploy the CloudFormation stack?",
  { type: "confirm" },
);

export const getStackName = new ScenarioInput(
  "getStackName",
  "Enter a name for the CloudFormation stack:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

```
export const getDatastoreName = new ScenarioInput(
  "getDatastoreName",
  "Enter a name for the HealthImaging datastore:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getAccountId = new ScenarioAction(
  "getAccountId",
  async (/** @type {{}} */ state) => {
    const command = new GetCallerIdentityCommand({});
    const response = await stsClient.send(command);
    state.accountId = response.Account;
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const createStack = new ScenarioAction(
  "createStack",
  async (/** @type {{}} */ state) => {
    const stackName = state.getStackName;
    const datastoreName = state.getDatastoreName;
    const accountId = state.accountId;

    const command = new CreateStackCommand({
      StackName: stackName,
      TemplateBody: await fs.readFile(cfnTemplatePath, "utf8"),
      Capabilities: ["CAPABILITY_IAM"],
      Parameters: [
        {
          ParameterKey: "datastoreName",
          ParameterValue: datastoreName,
        },
        {
          ParameterKey: "userAccountID",
          ParameterValue: accountId,
        },
      ],
    });

    const response = await cfnClient.send(command);
    state.stackId = response.StackId;
  },
```

```
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const waitForStackCreation = new ScenarioAction(
  "waitForStackCreation",
  async (/** @type {{}} */ state) => {
    const command = new DescribeStacksCommand({
      StackName: state.stackId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await cfnClient.send(command);
      const stack = response.Stacks?.find(
        (s) => s.StackName === state.getStackName,
      );
      if (!stack || stack.StackStatus === "CREATE_IN_PROGRESS") {
        throw new Error("Stack creation is still in progress");
      }
      if (stack.StackStatus === "CREATE_COMPLETE") {
        state.stackOutputs = stack.Outputs?.reduce((acc, output) => {
          acc[output.OutputKey] = output.OutputValue;
          return acc;
        }, {});
      } else {
        throw new Error(
          `Stack creation failed with status: ${stack.StackStatus}`,
        );
      }
    });
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const outputState = new ScenarioOutput(
  "outputState",
  (/** @type {{}} */ state) => {
    /**
     * @type {{ stackOutputs: { DatastoreID: string, BucketName: string, RoleArn:
 string }}}
     */
    const { stackOutputs } = state;
    return `Stack creation completed. Output values:
```

```
Datastore ID: ${stackOutputs?.DatastoreID}
Bucket Name: ${stackOutputs?.BucketName}
Role ARN: ${stackOutputs?.RoleArn}
    `;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

Kopieren Sie DICOM-Dateien (dataset-steps.js).

```
import {
  S3Client,
  CopyObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const s3Client = new S3Client({});

const datasetOptions = [
  {
    name: "CT of chest (2 images)",
    value: "00029d25-fb18-4d42-aaa5-a0897d1ac8f7",
  },
  {
    name: "CT of pelvis (57 images)",
    value: "00025d30-ef8f-4135-a35a-d83eff264fc1",
  },
  {
    name: "MRI of head (192 images)",
    value: "0002d261-8a5d-4e63-8e2e-0cbfac87b904",
  },
  {
    name: "MRI of breast (92 images)",
    value: "0002dd07-0b7f-4a68-a655-44461ca34096",
  },
];
```

```
/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   doCopy: boolean
 * }}} State
 */

export const selectDataset = new ScenarioInput(
  "selectDataset",
  (state) => {
    if (!state.doCopy) {
      process.exit(0);
    }
    return "Select a DICOM dataset to import:";
  },
  {
    type: "select",
    choices: datasetOptions,
  },
);

export const doCopy = new ScenarioInput(
  "doCopy",
  "Do you want to copy images from the public dataset into your bucket?",
  {
    type: "confirm",
  },
);

export const copyDataset = new ScenarioAction(
  "copyDataset",
  async (/** @type { State } */ state) => {
    const inputBucket = state.stackOutputs.BucketName;
    const inputPrefix = "input/";
    const selectedDatasetId = state.selectDataset;

    const sourceBucket = "idc-open-data";
    const sourcePrefix = `${selectedDatasetId}`;

    const listObjectsCommand = new ListObjectsV2Command({
      Bucket: sourceBucket,
      Prefix: sourcePrefix,
```

```
    });

    const objects = await s3Client.send(listObjectsCommand);

    const copyPromises = objects.Contents.map((object) => {
      const sourceKey = object.Key;
      const destinationKey = `${inputPrefix}${sourceKey
        .split("/")
        .slice(1)
        .join("/")}`;

      const copyCommand = new CopyObjectCommand({
        Bucket: inputBucket,
        CopySource: `/${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });

      return s3Client.send(copyCommand);
    });

    const results = await Promise.all(copyPromises);
    state.copiedObjects = results.length;
  },
);

export const outputCopiedObjects = new ScenarioOutput(
  "outputCopiedObjects",
  (state) => `${state.copiedObjects} DICOM files were copied.`,
);
```

Starten Sie den Import in den Datenspeicher (import-steps.js).

```
import {
  MedicalImagingClient,
  StartDICOMImportJobCommand,
  GetDICOMImportJobCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioOutput,
  ScenarioInput,
```

```
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }}} State
 */

export const doImport = new ScenarioInput(
  "doImport",
  "Do you want to import DICOM images into your datastore?",
  {
    type: "confirm",
    default: true,
  },
);

export const startDICOMImport = new ScenarioAction(
  "startDICOMImport",
  async (/** @type {State} */ state) => {
    if (!state.doImport) {
      process.exit(0);
    }
    const medicalImagingClient = new MedicalImagingClient({});
    const inputS3Uri = `s3://${state.stackOutputs.BucketName}/input/`;
    const outputS3Uri = `s3://${state.stackOutputs.BucketName}/output/`;

    const command = new StartDICOMImportJobCommand({
      dataAccessRoleArn: state.stackOutputs.RoleArn,
      datastoreId: state.stackOutputs.DatastoreID,
      inputS3Uri,
      outputS3Uri,
    });

    const response = await medicalImagingClient.send(command);
    state.importJobId = response.jobId;
  },
);

export const waitForImportJobCompletion = new ScenarioAction(
  "waitForImportJobCompletion",
```

```
  async (/** @type {State} */ state) => {
    const medicalImagingClient = new MedicalImagingClient({});
    const command = new GetDICOMImportJobCommand({
      datastoreId: state.stackOutputs.DatastoreID,
      jobId: state.importJobId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await medicalImagingClient.send(command);
      const jobStatus = response.jobProperties?.jobStatus;
      if (!jobStatus || jobStatus === "IN_PROGRESS") {
        throw new Error("Import job is still in progress");
      }
      if (jobStatus === "COMPLETED") {
        state.importJobOutputS3Uri = response.jobProperties.outputS3Uri;
      } else {
        throw new Error(`Import job failed with status: ${jobStatus}`);
      }
    });
  },
);

export const outputImportJobStatus = new ScenarioOutput(
  "outputImportJobStatus",
  (state) =>
    `DICOM import job completed. Output location: ${state.importJobOutputS3Uri}`,
);
```

Rufen Sie die Imagesatz-IDs ab (image-set-steps.js -).

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
```

```
 * }, importJobId: string,
 * importJobOutputS3Uri: string,
 * imageSetIds: string[],
 * manifestContent: { jobSummary: { imageSetsSummary: { imageSetId: string }
[] } }
 * }} State
 */

const s3Client = new S3Client({});

export const getManifestFile = new ScenarioAction(
  "getManifestFile",
  async (/** @type {State} */ state) => {
    const bucket = state.stackOutputs.BucketName;
    const prefix = `output/${state.stackOutputs.DatastoreID}-DicomImport-
${state.importJobId}/`;
    const key = `${prefix}job-output-manifest.json`;

    const command = new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    });

    const response = await s3Client.send(command);
    const manifestContent = await response.Body.transformToString();
    state.manifestContent = JSON.parse(manifestContent);
  },
);

export const parseManifestFile = new ScenarioAction(
  "parseManifestFile",
  (/** @type {State} */ state) => {
    const imageSetIds =
      state.manifestContent.jobSummary.imageSetsSummary.reduce((ids, next) => {
        return Object.assign({}, ids, {
          [next.imageSetId]: next.imageSetId,
        });
      }, {});
    state.imageSetIds = Object.keys(imageSetIds);
  },
);

export const outputImageSetIds = new ScenarioOutput(
  "outputImageSetIds",
```

```
  (/** @type {State} */ state) =>
    `The image sets created by this import job are: \n${state.imageSetIds
      .map((id) => `Image set: ${id}`)
      .join("\n")}`,
);
```

Rufen Sie die Image-Rahmen-IDs ab (image-frame-steps.js).

```
import {
  MedicalImagingClient,
  GetImageSetMetadataCommand,
} from "@aws-sdk/client-medical-imaging";
import { gunzip } from "node:zlib";
import { promisify } from "node:util";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const gunzipAsync = promisify(gunzip);

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
```

```
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetIds: string[] }} State
 */

const medicalImagingClient = new MedicalImagingClient({});

export const getImageSetMetadata = new ScenarioAction(
  "getImageSetMetadata",
  async (/** @type {State} */ state) => {
    const outputMetadata = [];
```

```
    for (const imageSetId of state.imageSetIds) {
      const command = new GetImageSetMetadataCommand({
        datastoreId: state.stackOutputs.DatastoreID,
        imageSetId,
      });

      const response = await medicalImagingClient.send(command);
      const compressedMetadataBlob =
        await response.imageSetMetadataBlob.transformToByteArray();
      const decompressedMetadata = await gunzipAsync(compressedMetadataBlob);
      const imageSetMetadata = JSON.parse(decompressedMetadata.toString());

      outputMetadata.push(imageSetMetadata);
    }

    state.imageSetMetadata = outputMetadata;
  },
);

export const outputImageFrameIds = new ScenarioOutput(
  "outputImageFrameIds",
  (/** @type {State & { imageSetMetadata: ImageSetMetadata[] }} */ state) => {
    let output = "";

    for (const metadata of state.imageSetMetadata) {
      const imageSetId = metadata.ImageSetID;
      /** @type {DICOMMetadata[]} */
      const instances = Object.values(metadata.Study.Series).flatMap(
        (series) => {
          return Object.values(series.Instances);
        },
      );
      const imageFrameIds = instances.flatMap((instance) =>
        instance.ImageFrames.map((frame) => frame.ID),
      );

      output += `Image set ID: ${imageSetId}\nImage frame IDs:\n
${imageFrameIds.join(
        "\n",
      )}\n\n`;
    }

    return output;
```

```
  },
);
```

Überprüfen Sie die Image-Rahmen (verify-steps.js). Die Bibliothek zur Überprüfung von AWS HealthImaging Pixeldaten wurde zur Überprüfung verwendet.

```
import { spawn } from "node:child_process";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */
```

```
/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

export const doVerify = new ScenarioInput(
  "doVerify",
  "Do you want to verify the imported images?",
  {
    type: "confirm",
    default: true,
  },
);

export const decodeAndVerifyImages = new ScenarioAction(
  "decodeAndVerifyImages",
  async (/** @type {State} */ state) => {
    if (!state.doVerify) {
      process.exit(0);
    }
```

```
    const verificationTool = "./pixel-data-verification/index.js";

    for (const metadata of state.imageSetMetadata) {
      const datastoreId = state.stackOutputs.DatastoreID;
      const imageSetId = metadata.ImageSetID;

      for (const [seriesInstanceUid, series] of Object.entries(
        metadata.Study.Series,
      )) {
        for (const [sopInstanceUid, _] of Object.entries(series.Instances)) {
          console.log(
            `Verifying image set ${imageSetId} with series ${seriesInstanceUid}
and sop ${sopInstanceUid}`,
          );
          const child = spawn(
            "node",
            [
              verificationTool,
              datastoreId,
              imageSetId,
              seriesInstanceUid,
              sopInstanceUid,
            ],
            { stdio: "inherit" },
          );

          await new Promise((resolve, reject) => {
            child.on("exit", (code) => {
              if (code === 0) {
                resolve();
              } else {
                reject(
                  new Error(
                    `Verification tool exited with code ${code} for image set
${imageSetId}`,
                  ),
                );
              }
            });
          });
        }
      }
    }
  },
```

```
);
```

Vernichten Sie Ressourcen (clean-up-steps.js).

```
import {
  CloudFormationClient,
  DeleteStackCommand,
} from "@aws-sdk/client-cloudformation";
import {
  MedicalImagingClient,
  DeleteImageSetCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */
```

```
/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

const cfnClient = new CloudFormationClient({});
const medicalImagingClient = new MedicalImagingClient({});

export const confirmCleanup = new ScenarioInput(
  "confirmCleanup",
  "Do you want to delete the created resources?",
  { type: "confirm" },
);

export const deleteImageSets = new ScenarioAction(
```

```
  "deleteImageSets",
  async (/** @type {State} */ state) => {
    const datastoreId = state.stackOutputs.DatastoreID;

    for (const metadata of state.imageSetMetadata) {
      const command = new DeleteImageSetCommand({
        datastoreId,
        imageSetId: metadata.ImageSetID,
      });

      try {
        await medicalImagingClient.send(command);
        console.log(`Successfully deleted image set ${metadata.ImageSetID}`);
      } catch (e) {
        if (e instanceof Error) {
          if (e.name === "ConflictException") {
            console.log(`Image set ${metadata.ImageSetID} already deleted`);
          }
        }
      }
    }
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);

export const deleteStack = new ScenarioAction(
  "deleteStack",
  async (/** @type {State} */ state) => {
    const stackName = state.getStackName;

    const command = new DeleteStackCommand({
      StackName: stackName,
    });

    await cfnClient.send(command);
    console.log(`Stack ${stackName} deletion initiated`);
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

Erstellen Sie einen CloudFormation Stack mit den erforderlichen Ressourcen.

```
    def deploy(self):
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `setup.yaml` AWS CloudFormation script and are
deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
        """

        print("\t\tLet's deploy the stack for resource creation.")
        stack_name = q.ask("\t\tEnter a name for the stack: ", q.non_empty)

        data_store_name = q.ask(
            "\t\tEnter a name for the Health Imaging Data Store: ", q.non_empty
        )

        account_id = boto3.client("sts").get_caller_identity()["Account"]
```

```
        with open(
            "../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml"
        ) as setup_file:
            setup_template = setup_file.read()
        print(f"\t\tCreating {stack_name}.")
        stack = self.cf_resource.create_stack(
            StackName=stack_name,
            TemplateBody=setup_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=[
                {
                    "ParameterKey": "datastoreName",
                    "ParameterValue": data_store_name,
                },
                {
                    "ParameterKey": "userAccountID",
                    "ParameterValue": account_id,
                },
            ],
        )
        print("\t\tWaiting for stack to deploy. This typically takes a minute or
 two.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_create_complete")
        waiter.wait(StackName=stack.name)
        stack.load()
        print(f"\t\tStack status: {stack.stack_status}")

        outputs_dictionary = {
            output["OutputKey"]: output["OutputValue"] for output in
 stack.outputs
        }
        self.input_bucket_name = outputs_dictionary["BucketName"]
        self.output_bucket_name = outputs_dictionary["BucketName"]
        self.role_arn = outputs_dictionary["RoleArn"]
        self.data_store_id = outputs_dictionary["DatastoreID"]
        return stack
```

Kopieren Sie DICOM-Dateien in den Amazon-S3-Import-Bucket.

```
    def copy_single_object(self, key, source_bucket, target_bucket,
target_directory):
        """
        Copies a single object from a source to a target bucket.

        :param key: The key of the object to copy.
        :param source_bucket: The source bucket for the copy.
        :param target_bucket: The target bucket for the copy.
        :param target_directory: The target directory for the copy.
        """
        new_key = target_directory + "/" + key
        copy_source = {"Bucket": source_bucket, "Key": key}
        self.s3_client.copy_object(
            CopySource=copy_source, Bucket=target_bucket, Key=new_key
        )
        print(f"\n\t\tCopying {key}.")

    def copy_images(
        self, source_bucket, source_directory, target_bucket, target_directory
    ):
        """
        Copies the images from the source to the target bucket using multiple
threads.

        :param source_bucket: The source bucket for the images.
        :param source_directory: Directory within the source bucket.
        :param target_bucket: The target bucket for the images.
        :param target_directory: Directory within the target bucket.
        """

        # Get list of all objects in source bucket.
        list_response = self.s3_client.list_objects_v2(
            Bucket=source_bucket, Prefix=source_directory
        )
        objs = list_response["Contents"]
        keys = [obj["Key"] for obj in objs]

        # Copy the objects in the bucket.
        for key in keys:
            self.copy_single_object(key, source_bucket, target_bucket,
target_directory)

        print("\t\tDone copying all objects.")
```

Importieren Sie die DICOM-Dateien in den Amazon-S3-Datenspeicher.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def start_dicom_import_job(
        self,
        data_store_id,
        input_bucket_name,
        input_directory,
        output_bucket_name,
        output_directory,
        role_arn,
    ):
        """
        Routine which starts a HealthImaging import job.

        :param data_store_id: The HealthImaging data store ID.
        :param input_bucket_name: The name of the Amazon S3 bucket containing the
 DICOM files.
        :param input_directory: The directory in the S3 bucket containing the
 DICOM files.
        :param output_bucket_name: The name of the S3 bucket for the output.
```

```
        :param output_directory: The directory in the S3 bucket to store the
 output.
        :param role_arn: The ARN of the IAM role with permissions for the import.
        :return: The job ID of the import.
        """

        input_uri = f"s3://{input_bucket_name}/{input_directory}/"
        output_uri = f"s3://{output_bucket_name}/{output_directory}/"
        try:
            job = self.medical_imaging_client.start_dicom_import_job(
                jobName="examplejob",
                datastoreId=data_store_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_uri,
                outputS3Uri=output_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Rufen Sie Bildsätze ab, die durch den DICOM-Importauftrag erstellt wurden.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client
```

```
    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_sets_for_dicom_import_job(self, datastore_id, import_job_id):
        """
        Retrieves the image sets created for an import job.

        :param datastore_id: The HealthImaging data store ID
        :param import_job_id: The import job ID
        :return: List of image set IDs
        """

        import_job = self.medical_imaging_client.get_dicom_import_job(
            datastoreId=datastore_id, jobId=import_job_id
        )

        output_uri = import_job["jobProperties"]["outputS3Uri"]

        bucket = output_uri.split("/")[2]
        key = "/".join(output_uri.split("/")[3:])

        # Try to get the manifest.
        retries = 3
        while retries > 0:
            try:
                obj = self.s3_client.get_object(
                    Bucket=bucket, Key=key + "job-output-manifest.json"
                )
                body = obj["Body"]
                break
            except ClientError as error:
                retries = retries - 1
                time.sleep(3)
        try:
            data = json.load(body)
            expression =
jmespath.compile("jobSummary.imageSetsSummary[].imageSetId")
            image_sets = expression.search(data)
        except json.decoder.JSONDecodeError as error:
            image_sets = import_job["jobProperties"]
```

```
        return image_sets


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Rufen Sie Bild-Frame-Informationen für Bildsätze ab.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""
```

```
    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_frames_for_image_set(self, datastore_id, image_set_id,
out_directory):
        """
        Get the image frames for an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param out_directory: The directory to save the file.
        :return: The image frames.
        """
        image_frames = []
        file_name = os.path.join(out_directory,
f"{image_set_id}_metadata.json.gzip")
        file_name = file_name.replace("/", "\\\\")
        self.get_image_set_metadata(file_name, datastore_id, image_set_id)
        try:
            with gzip.open(file_name, "rb") as f_in:
                doc = json.load(f_in)
            instances = jmespath.search("Study.Series.*.Instances[].*[]", doc)
            for instance in instances:
                rescale_slope = jmespath.search("DICOM.RescaleSlope", instance)
                rescale_intercept = jmespath.search("DICOM.RescaleIntercept",
instance)
                image_frames_json = jmespath.search("ImageFrames[][]", instance)
                for image_frame in image_frames_json:
                    checksum_json = jmespath.search(
                        "max_by(PixelDataChecksumFromBaseToFullResolution,
&Width)",
                        image_frame,
```

```
                    )
                    image_frame_info = {
                        "imageSetId": image_set_id,
                        "imageFrameId": image_frame["ID"],
                        "rescaleIntercept": rescale_intercept,
                        "rescaleSlope": rescale_slope,
                        "minPixelValue": image_frame["MinPixelValue"],
                        "maxPixelValue": image_frame["MaxPixelValue"],
                        "fullResolutionChecksum": checksum_json["Checksum"],
                    }
                    image_frames.append(image_frame_info)
            return image_frames
        except TypeError:
            return {}
        except ClientError as err:
            logger.error(
                "Couldn't get image frames for image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return image_frames


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """

        try:
            if version_id:
                image_set_metadata =
self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

```
            else:
                image_set_metadata =
 self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            with open(metadata_file, "wb") as f:
                for chunk in
 image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Laden Sie Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)
```

```
    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K
encoded pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.medical_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def download_decode_and_check_image_frames(
        self, data_store_id, image_frames, out_directory
    ):
        """
        Downloads image frames, decodes them, and uses the checksum to validate
        the decoded images.

        :param data_store_id: The HealthImaging data store ID.
        :param image_frames: A list of dicts containing image frame information.
        :param out_directory: A directory for the downloaded images.
        :return: True if the function succeeded; otherwise, False.
        """
        total_result = True
        for image_frame in image_frames:
```

```
            image_file_path = f"{out_directory}/
image_{image_frame['imageFrameId']}.jph"
            self.get_pixel_data(
                image_file_path,
                data_store_id,
                image_frame["imageSetId"],
                image_frame["imageFrameId"],
            )

            image_array = self.jph_image_to_opj_bitmap(image_file_path)
            crc32_checksum = image_frame["fullResolutionChecksum"]
            # Verify checksum.
            crc32_calculated = zlib.crc32(image_array)
            image_result = crc32_checksum == crc32_calculated
            print(
                f"\t\tImage checksum verified for {image_frame['imageFrameId']}:
 {image_result }"
            )
            total_result = total_result and image_result
        return total_result

    @staticmethod
    def jph_image_to_opj_bitmap(jph_file):
        """
        Decode the image to a bitmap using an OPENJPEG library.
        :param jph_file: The file to decode.
        :return: The decoded bitmap as an array.
        """
        # Use format 2 for the JPH file.
        params = openjpeg.utils.get_parameters(jph_file, 2)
        print(f"\n\t\tImage parameters for {jph_file}: \n\t\t{params}")

        image_array = openjpeg.utils.decode(jph_file, 2)

        return image_array
```

Bereinigen Sie die Ressourcen.

```
    def destroy(self, stack):
        """
```

```
        Destroys the resources managed by the CloudFormation stack, and the
 CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
 resources.
        """

        print(f"\t\tCleaning up resources and {stack.name}.")
        data_store_id = None
        for oput in stack.outputs:
            if oput["OutputKey"] == "DatastoreID":
                data_store_id = oput["OutputValue"]
        if data_store_id is not None:
            print(f"\t\tDeleting image sets in data store {data_store_id}.")
            image_sets = self.medical_imaging_wrapper.search_image_sets(
                data_store_id, {}
            )
            image_set_ids = [image_set["imageSetId"] for image_set in image_sets]

            for image_set_id in image_set_ids:
                self.medical_imaging_wrapper.delete_image_set(
                    data_store_id, image_set_id
                )
                print(f"\t\tDeleted image set with id : {image_set_id}")

        print(f"\t\tDeleting {stack.name}.")
        stack.delete()
        print("\t\tWaiting for stack removal. This may take a few minutes.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_delete_complete")
        waiter.wait(StackName=stack.name)
        print("\t\tStack delete complete.")




class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
```

```
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.medical_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.
```

```
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        """
        try:
            delete_results = self.medical_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kennzeichnen eines HealthImaging Datenspeichers mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie ein HealthImaging Datenspeicher mit Tags versehen wird.

Java

SDK für Java 2.x

So taggen Sie einen Datenspeicher.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

So listen Sie Tags für einen Datenspeicher auf.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
```

```
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " +
result.tags());
                }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
   public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
           String resourceArn) {
       try {
           ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .build();

           return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

So entfernen Sie Tags von einem Datenspeicher.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
                                Collections.singletonList("Deployment"));
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
           UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .tagKeys(tagKeys)
                   .build();

           medicalImagingClient.untagResource(untagResourceRequest);

           System.out.println("Tags have been removed from the resource.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

So taggen Sie einen Datenspeicher.

```
 try {
```

```
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(datastoreArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }
```

```
  return response;
};
```

So listen Sie Tags für einen Datenspeicher auf.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(datastoreArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
```

```
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

So entfernen Sie Tags von einem Datenspeicher.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(datastoreArn, keys);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
```

```
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

So taggen Sie einen Datenspeicher.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.tag_resource(data_store_arn, {"Deployment":
 "Development"})
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
```

```
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

So listen Sie Tags für einen Datenspeicher auf.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.list_tags_for_resource(data_store_arn)
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
```

```
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

So entfernen Sie Tags von einem Datenspeicher.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.untag_resource(data_store_arn, ["Deployment"])
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
```

```
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einen HealthImaging Bilddatensatz mit einem taggen AWS SDK

Die folgenden Codebeispiele zeigen, wie ein HealthImaging Bilddatensatz mit Tags versehen wird.

Java

SDK für Java 2.x

So taggen Sie einen Imagesatz.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

So listen Sie Tags für einen Imagesatz auf.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
```

```
                        System.out.println("Tags for resource: " +
 result.tags());
                }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

So entfernen Sie Tags von einem Imagesatz.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
```

```
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

JavaScript

SDK für JavaScript (v3)

So taggen Sie einen Imagesatz.

```
try {
  const imagesetArn =
```

```
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(imagesetArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }
```

```
  return response;
};
```

So listen Sie Tags für einen Imagesatz auf.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(imagesetArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/
ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
```

```
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

So entfernen Sie Tags von einem Imagesatz.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(imagesetArn, keys);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data
 store or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/
imageset/xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
```

```
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Python

SDK für Python (Boto3)

So taggen Sie einen Imagesatz.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.tag_resource(image_set_arn, {"Deployment":
 "Development"})
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

So listen Sie Tags für einen Imagesatz auf.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.list_tags_for_resource(image_set_arn)
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
```

```
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

So entfernen Sie Tags von einem Imagesatz.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.untag_resource(image_set_arn, ["Deployment"])
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Codebeispiele für die HealthLake Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS HealthLake mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- HealthLake Entwicklerhandbuch — Weitere Informationen zu HealthLake.
- HealthLake API-Referenz — Details zu allen verfügbaren HealthLake Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung HealthLake AWS SDKs
  - Aktionen zur Verwendung HealthLake AWS SDKs
    - Verwenden Sie CreateFhirDataStore mit einem AWS SDK oder CLI
    - Verwenden Sie DeleteFhirDataStore mit einem AWS SDK oder CLI
    - Verwenden Sie DescribeFhirDataStore mit einem AWS SDK oder CLI
    - DescribeFHIRExportJobVerwenden Sie mit einem AWS SDK oder CLI
    - Verwenden Sie DescribeFHIRImportJobmit einem AWS SDK oder CLI
    - Verwenden Sie ListFhirDataStores mit einem AWS SDK oder CLI
    - ListFHIRExportJobsVerwenden Sie mit einem AWS SDK oder CLI
    - Verwenden Sie ListFHIRImportJobsmit einem AWS SDK oder CLI
    - Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
    - Verwenden Sie StartFHIRExportJobmit einem AWS SDK oder CLI
    - Verwenden Sie StartFHIRImportJobmit einem AWS SDK oder CLI
    - Verwenden Sie TagResourcemit einem AWS SDK oder CLI
    - Verwenden Sie UntagResourcemit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung HealthLake AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS HealthLake mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung HealthLake AWS SDKs
  - Verwenden Sie CreateFhirDataStore mit einem AWS SDK oder CLI
  - Verwenden Sie DeleteFhirDataStore mit einem AWS SDK oder CLI
  - Verwenden Sie DescribeFhirDataStore mit einem AWS SDK oder CLI
  - DescribeFHIRExportJobVerwenden Sie mit einem AWS SDK oder CLI
  - Verwenden Sie DescribeFHIRImportJobmit einem AWS SDK oder CLI
  - Verwenden Sie ListFhirDataStores mit einem AWS SDK oder CLI
  - ListFHIRExportJobsVerwenden Sie mit einem AWS SDK oder CLI
  - Verwenden Sie ListFHIRImportJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
  - Verwenden Sie StartFHIRExportJobmit einem AWS SDK oder CLI
  - Verwenden Sie StartFHIRImportJobmit einem AWS SDK oder CLI
  - Verwenden Sie TagResourcemit einem AWS SDK oder CLI
  - Verwenden Sie UntagResourcemit einem AWS SDK oder CLI

Aktionen zur Verwendung HealthLake AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne HealthLake Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS HealthLake -API-Referenz.

Beispiele

- Verwenden Sie CreateFhirDataStore mit einem AWS SDK oder CLI
- Verwenden Sie DeleteFhirDataStore mit einem AWS SDK oder CLI
- Verwenden Sie DescribeFhirDataStore mit einem AWS SDK oder CLI

- DescribeFHIRExportJobVerwenden Sie mit einem AWS SDK oder CLI
- Verwenden Sie DescribeFHIRImportJobmit einem AWS SDK oder CLI
- Verwenden Sie ListFhirDataStores mit einem AWS SDK oder CLI
- ListFHIRExportJobsVerwenden Sie mit einem AWS SDK oder CLI
- Verwenden Sie ListFHIRImportJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListTagsForResourcemit einem AWS SDK oder CLI
- Verwenden Sie StartFHIRExportJobmit einem AWS SDK oder CLI
- Verwenden Sie StartFHIRImportJobmit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UntagResourcemit einem AWS SDK oder CLI

Verwenden Sie **`CreateFhirDataStore`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFHIRDatastore` verwendet wird.

CLI

AWS CLI

Beispiel 1: Erstellen Sie einen SigV4-enabled HealthLake Datenspeicher

Das folgende `create-fhir-datastore` Beispiel zeigt, wie Sie einen neuen Datenspeicher in erstellen AWS HealthLake.

```
aws healthlake create-fhir-datastore \
    --datastore-type-version R4 \
    --datastore-name "FhirTestDatastore"
```

Ausgabe:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
```

```
}
```

Beispiel 2: Erstellen Sie einen FHIR-enabled HealthLake SMART-Datenspeicher

Das folgende `create-fhir-datastore` Beispiel zeigt, wie ein neuer FHIR-enabled SMART-On-Datenspeicher in erstellt wird AWS HealthLake.

```
aws healthlake create-fhir-datastore \
    --datastore-name "your-data-store-name" \
    --datastore-type-version R4 \
    --preload-data-config PreloadDataType="SYNTHEA" \
    --sse-configuration '{ "KmsEncryptionConfig": {  "CmkType":
 "CUSTOMER_MANAGED_KMS_KEY", "KmsKeyId": "arn:aws:kms:us-east-1:your-account-
id:key/your-key-id" } }' \
    --identity-provider-configuration  file://
identity_provider_configuration.json
```

Inhalt von `identity_provider_configuration.json`:

```
{
    "AuthorizationStrategy": "SMART_ON_FHIR_V1",
    "FineGrainedAuthorizationEnabled": true,
    "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-id:function:your-
lambda-name",
    "Metadata": "{\"issuer\":\"https://ehr.example.com\", \"jwks_uri\":
\"https://ehr.example.com/.well-known/jwks.json\",\"authorization_endpoint
\":\"https://ehr.example.com/auth/authorize\",\"token_endpoint\":\"https://
ehr.token.com/auth/token\",\"token_endpoint_auth_methods_supported\":
[\"client_secret_basic\",\"foo\"],\"grant_types_supported\":[\"client_credential
\",\"foo\"],\"registration_endpoint\":\"https://ehr.example.com/auth/
register\",\"scopes_supported\":[\"openId\",\"profile\",\"launch\"],
\"response_types_supported\":[\"code\"],\"management_endpoint\":\"https://
ehr.example.com/user/manage\",\"introspection_endpoint\":\"https://
ehr.example.com/user/introspect\",\"revocation_endpoint\":\"https://
ehr.example.com/user/revoke\",\"code_challenge_methods_supported\":[\"S256\"],
\"capabilities\":[\"launch-ehr\",\"sso-openid-connect\",\"client-public\"]}"
}
```

Ausgabe:

```
{
```

```
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "CREATING",
    "DatastoreId": "(Data store ID)"
}
```

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Erstellen und Überwachen eines FHIR-Datenspeichers.

- Weitere API-Informationen finden Sie unter CreateFHIRDatastore in der AWS CLI - Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
        Creates a new HealthLake data store.
        When creating a SMART on FHIR data store, the following parameters are
 required:
        - sse_configuration: The server-side encryption configuration for a SMART
 on FHIR-enabled data store.
```

```
        - identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.

        :param datastore_name: The name of the data store.
        :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
        :param identity_provider_configuration: The identity provider
configuration for a SMART on FHIR-enabled data store.
        :return: A dictionary containing the data store information.
        """
        try:
            parameters = {"DatastoreName": datastore_name,
"DatastoreTypeVersion": "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response =
self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code zeigt ein Beispiel für Parameter für einen FHIR-enabled HealthLake SMART-On-Datenspeicher.

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
```

```
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/
authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/
introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
                "Metadata": json.dumps(metadata),
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration,
 identity_provider_configuration
            )
```

- Weitere API-Informationen finden Sie unter CreateFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter CreateFhirDataStore in der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie DeleteFhirDataStore mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFHIRDatastore` verwendet wird.

CLI

AWS CLI

So löschen Sie einen FHIR-Datenspeicher

Das folgende `delete-fhir-datastore` Beispiel zeigt, wie Sie einen Datenspeicher und seinen gesamten Inhalt in löschen AWS HealthLake.

```
aws healthlake delete-fhir-datastore \
    --datastore-id (Data store ID)
```

Ausgabe:

```
{
    "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/datastore/
(Data store ID)/r4/",
    "DatastoreArn": "arn:aws:healthlake:us-east-1:(AWS Account ID):datastore/
(Data store ID)",
    "DatastoreStatus": "DELETING",
    "DatastoreId": "(Data store ID)"
}
```

Weitere Informationen finden Sie unter Erstellen und Überwachen eines FHIR-Datenspeichers < https://docs.aws.amazon.com/healthlake/latest/devguide/working-with-FHIR-healthlake.html > im AWS HealthLake Entwicklerhandbuch.

- Weitere API-Informationen finden Sie unter DeleteFHIRDatastore in der AWS CLI - Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:

self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter DeleteFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter DeleteFhirDataStore in der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie DescribeFhirDataStore mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRDatastore` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Datenspeicher

Das folgende `describe-fhir-datastore` Beispiel zeigt, wie Sie die Eigenschaften eines Datenspeichers in finden AWS HealthLake.

```
aws healthlake describe-fhir-datastore \
    --datastore-id "1f2f459836ac6c513ce899f9e4f66a59"
```

Ausgabe:

```
{
    "DatastoreProperties": {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
}
```

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Erstellen und Überwachen eines FHIR-Datenspeichers.

- Weitere API-Informationen finden Sie unter DescribeFHIRDatastore in der AWS CLI - Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
            )
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter DescribeFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties-
>get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
```

```
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter DescribeFhirDataStore in der API-Referenz zum AWS SDK für SAP ABAP.

`DescribeFHIRExportJob`Verwenden Sie mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRExportJob` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Exportauftrag

Das folgende `describe-fhir-export-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Exportauftrags in AWS HealthLake finden.

```
aws healthlake describe-fhir-export-job \
    --datastore-id (Data store ID) \
    --job-id 9b9a51943afaedd0a8c0c26c49135a31
```

Ausgabe:

```
{
    "ExportJobProperties": {
        "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
        "JobStatus": "IN_PROGRESS",
        "JobId": "9009813e9d69ba7cf79bcb3468780f16",
        "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
        "EndTime": "2024-11-20T11:34:01.636000-05:00",
        "OutputDataConfig": {
            "S3Configuration": {
            "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
            "KmsKeyId": "arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"
```

```
        }

        },
        "DatastoreId": "(Data store ID)"
    }
}
```

Weitere Informationen finden Sie unter Exportieren von Dateien aus einem FHIR-Datenspeicher im AWS HealthLake Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DescribeFHIRExportJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
```

```
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeFHIRExportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
```

```
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeFHIRExportJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeFHIRImportJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeFHIRImportJob` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen FHIR-Importauftrag

Das folgende `describe-fhir-import-job` Beispiel zeigt, wie Sie die Eigenschaften eines FHIR-Importjobs mithilfe AWS HealthLake von lernen.

```
aws healthlake describe-fhir-import-job \
    --datastore-id (Data store ID) \
    --job-id c145fbb27b192af392f8ce6e7838e34f
```

Ausgabe:

```
{
    "ImportJobProperties": {
    "InputDataConfig": {
        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
        { "arrayitem2": 2 }
    },
```

```
    "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)",
    "JobStatus": "COMPLETED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f",
    "SubmitTime": 1606272542.161,
    "EndTime": 1606272609.497,
    "DatastoreId": "(Data store ID)"
    }
}
```

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Dateien in einen FHIR-Datenspeicher importieren.

- Einzelheiten zur API finden Sie DescribeFHIRImportJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
                DatastoreId=datastore_id, JobId=job_id
```

```
            )
            return response["ImportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeFHIRImportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
        IF lo_import_job_properties IS BOUND.
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
```

```
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeFHIRImportJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListFhirDataStores`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRDatastores` verwendet wird.

CLI

AWS CLI

So listen Sie FHIR-Datenspeicher auf

Das folgende `list-fhir-datastores` Beispiel zeigt, wie der Befehl verwendet wird und wie Benutzer Ergebnisse anhand des Datenspeicherstatus in filtern können AWS HealthLake.

```
aws healthlake list-fhir-datastores \
    --filter DatastoreStatus=ACTIVE
```

Ausgabe:

```
{
    "DatastorePropertiesList": [
    {
        "PreloadDataConfig": {
            "PreloadDataType": "SYNTHEA"
        },
```

```
        "SseConfiguration": {
            "KmsEncryptionConfig": {
                "CmkType": "CUSTOMER_MANAGED_KMS_KEY",
                "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
            }
        },
        "DatastoreName": "Demo",
        "DatastoreArn": "arn:aws:healthlake:us-east-1:<AWS Account ID>:datastore/
<Data store ID>",
        "DatastoreEndpoint": "https://healthlake.us-east-1.amazonaws.com/
datastore/<Data store ID>/r4/",
        "DatastoreStatus": "ACTIVE",
        "DatastoreTypeVersion": "R4",
        "CreatedAt": 1603761064.881,
        "DatastoreId": "<Data store ID>",
        "IdentityProviderConfiguration": {
            "AuthorizationStrategy": "AWS_AUTH",
            "FineGrainedAuthorizationEnabled": false
        }
    }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen und Überwachen eines AWS HealthLake FHIR-Datenspeichers im Entwicklerhandbuch.

- Weitere API-Informationen finden Sie unter ListFHIRDatastores in der AWS CLI - Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
```

```
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
 self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
            )
            raise
```

- Weitere API-Informationen finden Sie unter ListFHIRDatastores in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter ListFhirDataStores in der API-Referenz zum AWS SDK für SAP ABAP.

`ListFHIRExportJobs`Verwenden Sie mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRExportJobs` verwendet wird.

CLI

AWS CLI

So listen Sie alle FHIR-Exportaufträge auf

Das folgende Beispiel für `list-fhir-export-jobs` zeigt, wie Sie mit dem Befehl eine Liste von Exportjobs anzeigen, die mit einem Konto verknüpft sind.

```
aws healthlake list-fhir-export-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z)\
    --submitted-after (DATE like 2020-10-13T19:00:00Z )\
    --job-name "FHIR-EXPORT" \
    --job-status SUBMITTED  \
    --max-results (Integer between 1 and 500)
```

Ausgabe:

```
{
    "ExportJobPropertiesList": [
        {
            "ExportJobProperties": {
                "OutputDataConfig": {
                    "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                    "S3Configuration": {
                        "S3Uri": "s3://(Bucket Name)/(Prefix Name)/",
                        "KmsKeyId": "(KmsKey Id)"
                    }
                },
                "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role
 Name)",
                "JobStatus": "COMPLETED",
                "JobId": "c145fbb27b192af392f8ce6e7838e34f",
                "JobName": "FHIR-EXPORT",
                "SubmitTime": "2024-11-20T11:31:46.672000-05:00",
                "EndTime": "2024-11-20T11:34:01.636000-05:00",
                "DatastoreId": "(Data store ID)"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Exportieren von Dateien aus einem FHIR-Datenspeicher im AWS HealthLake Entwicklerhandbuch.

- Einzelheiten zur API finden Sie unter ListFHIRExportJobs AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified
date.
        :param submitted_after: The export job submitted after the specified
date.
        :return: A list of export jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
```

```
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListFHIRExportJobsin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListFHIRExportJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListFHIRImportJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFHIRImportJobs` verwendet wird.

CLI

AWS CLI

So listen Sie alle FHIR-Importaufträge auf

Das folgende Beispiel für `list-fhir-import-jobs` zeigt, wie Sie mit dem Befehl eine Liste aller Importjobs anzeigen, die mit einem Konto verknüpft sind.

```
aws healthlake list-fhir-import-jobs \
    --datastore-id (Data store ID) \
    --submitted-before (DATE like 2024-10-13T19:00:00Z) \
    --submitted-after (DATE like 2020-10-13T19:00:00Z ) \
    --job-name "FHIR-IMPORT" \
    --job-status SUBMITTED  \
    -max-results (Integer between 1 and 500)
```

Ausgabe:

```
{
    "ImportJobPropertiesList": [
        {
            "JobId": "c0fddbf76f238297632d4aebdbfc9ddf",
            "JobStatus": "COMPLETED",
            "SubmitTime": "2024-11-20T10:08:46.813000-05:00",
            "EndTime": "2024-11-20T10:10:09.093000-05:00",
            "DatastoreId": "(Data store ID)",
            "InputDataConfig": {
                "S3Uri": "s3://(Bucket Name)/(Prefix Name)/"
            },
            "JobOutputDataConfig": {
                "S3Configuration": {
                    "S3Uri": "s3://(Bucket Name)/
import/6407b9ae4c2def3cb6f1a46a0c599ec0-FHIR_IMPORT-
c0fddbf76f238297632d4aebdbfc9ddf/",
                    "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/b7f645cb-
e564-4981-8672-9e012d1ff1a0"
                }
            },
```

```
            "JobProgressReport": {
                "TotalNumberOfScannedFiles": 1,
                "TotalSizeOfScannedFilesInMB": 0.001798,
                "TotalNumberOfImportedFiles": 1,
                "TotalNumberOfResourcesScanned": 1,
                "TotalNumberOfResourcesImported": 1,
                "TotalNumberOfResourcesWithCustomerError": 0,
                "TotalNumberOfFilesReadWithCustomerError": 0,
                "Throughput": 0.0
            },
            "DataAccessRoleArn": "arn:aws:iam::(AWS Account ID):role/(Role Name)"
        }
    ]
}
```

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Dateien in den FHIR-Datenspeicher importieren.

- Einzelheiten zur API finden Sie unter ListFHIRImportJobs AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
```

```
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified
date.
        :param submitted_after: The import job submitted after the specified
date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListFHIRImportJobsin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
```

```
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListFHIRImportJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTagsForResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForResource` verwendet wird.

CLI

AWS CLI

So listen Sie Tags für einen Datenspeicher auf

Im folgenden Beispiel für `list-tags-for-resource` werden die Tags aufgelistet, die mit dem angegebenen Datenspeicher verknüpft sind:

```
aws healthlake list-tags-for-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe"
```

Ausgabe:

```
{
    "tags": {
        "key": "value",
        "key1": "value1"
    }
}
```

Weitere Informationen finden Sie AWS HealthLakeim AWS HealthLake Entwicklerhandbuch unter Ressourcen taggen.

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListTagsForResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTagsForResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartFHIRExportJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartFHIRExportJob` verwendet wird.

CLI

AWS CLI

Um einen FHIR-Exportauftrag zu starten

Das folgende `start-fhir-export-job` Beispiel zeigt, wie Sie einen FHIR-Exportauftrag mit AWS HealthLake starten.

```
aws healthlake start-fhir-export-job \
    --output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/
(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn arn:aws:iam::(AWS Account ID):role/(Role Name)
```

Ausgabe:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "9b9a51943afaedd0a8c0c26c49135a31"
}
```

Weitere Informationen finden Sie unter Exportieren von Dateien aus einem FHIR-Datenspeicher im AWS HealthLake Entwicklerhandbuch.

- Einzelheiten zur API finden Sie StartFHIRExportJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_export_job(
        self,
        job_name: str,
        datastore_id: str,
        output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake export job.
        :param job_name: The export job name.
        :param datastore_id: The data store ID.
        :param output_s3_uri: The output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The export job.
        """
        try:
            response = self.health_lake_client.start_fhir_export_job(
                OutputDataConfig={
                    "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
                JobName=job_name,
            )

            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie StartFHIRExportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
```

```
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartFHIRExportJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartFHIRImportJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartFHIRImportJob` verwendet wird.

CLI

AWS CLI

So starten Sie einen FHIR-Importauftrag

Das folgende `start-fhir-import-job` Beispiel zeigt, wie Sie einen FHIR-Importjob mit AWS HealthLake starten.

```
aws healthlake start-fhir-import-job \
    --input-data-config S3Uri="s3://(Bucket Name)/(Prefix Name)/" \
    --job-output-data-config '{"S3Configuration": {"S3Uri":"s3://(Bucket Name)/
(Prefix Name)/","KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/d330e7fc-
b56c-4216-a250-f4c43ef46e83"}}' \
    --datastore-id (Data store ID) \
    --data-access-role-arn "arn:aws:iam::(AWS Account ID):role/(Role Name)"
```

Ausgabe:

```
{
    "DatastoreId": "(Data store ID)",
    "JobStatus": "SUBMITTED",
    "JobId": "c145fbb27b192af392f8ce6e7838e34f"
}
```

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Dateien in einen FHIR-Datenspeicher importieren.

- Einzelheiten zur API finden Sie StartFHIRImportJobin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
```

```
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie StartFHIRImportJobin AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartFHIRImportJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`TagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie ein Tag zum Datenspeicher hinzu

Im folgenden Beispiel für `tag-resource` wird gezeigt, wie Sie ein Tag zu einem Datenspeicher hinzufügen.

```
aws healthlake tag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/0725c83f4307f263e16fd56b6d8ebdbe" \
    --tags '[{"Key": "key1", "Value": "value1"}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Hinzufügen eines Tags zu einem Datenspeicher. .

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
```

```
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) ->
None:
        """
        Tags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tags: The tags to add to the resource.
        """
        try:
            self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
        except ClientError as err:
            logger.exception(
                "Couldn't tag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UntagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResource` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einem Datenspeicher.

Das folgende Beispiel für `untag-resource` zeigt, wie Sie Tags aus einem Datenspeicher entfernen.

```
aws healthlake untag-resource \
    --resource-arn "arn:aws:healthlake:us-east-1:123456789012:datastore/fhir/
b91723d65c6fdeb1d26543a49d2ed1fa" \
    --tag-keys '["key1"]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie im AWS HealthLake Entwicklerhandbuch unter Entfernen von Tags aus einem Datenspeicher.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake
 client.

        :return: An instance of HealthLakeWrapper initialized with the default
 HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
```

```
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie UntagResourcein AWS SDK for Python (Boto3) API Reference.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
```

```
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UntagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für die Verwendung von IAM AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Identity and Access Management mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- IAM-Benutzerhandbuch – Weitere Informationen zu IAM.
- IAM-API-Referenz – Details zu allen verfügbaren IAM-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von IAM AWS SDKs
  - Hello IAM
  - Lernen Sie die Grundlagen von IAM mit einem AWS SDK
  - Aktionen für die Verwendung von IAM AWS SDKs
    - Verwendung von AddClientIdToOpenIdConnectProvider mit einer CLI
    - Verwendung von AddRoleToInstanceProfile mit einer CLI
    - Verwendung von AddUserToGroup mit einer CLI
    - Verwendung von AttachGroupPolicy mit einer CLI
    - Verwenden Sie mit einem AttachRolePolicy AWS SDK oder CLI
    - Verwenden Sie AttachUserPolicymit einem AWS SDK oder CLI
    - Verwendung von ChangePassword mit einer CLI
    - Verwenden Sie mit einem CreateAccessKey AWS SDK oder CLI
    - Verwenden Sie CreateAccountAliasmit einem AWS SDK oder CLI
    - Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem CreateInstanceProfile AWS SDK oder CLI
    - Verwendung von CreateLoginProfile mit einer CLI
    - Verwendung von CreateOpenIdConnectProvider mit einer CLI
    - Verwenden Sie mit einem CreatePolicy AWS SDK oder CLI
    - Verwenden Sie CreatePolicyVersionmit einem AWS SDK oder CLI
    - Verwenden Sie CreateRolemit einem AWS SDK oder CLI
    - Verwenden Sie CreateSAMLProvider mit einem AWS SDK oder CLI
    - Verwenden Sie CreateServiceLinkedRolemit einem AWS SDK oder CLI
    - Verwenden Sie CreateUsermit einem AWS SDK oder CLI
    - Verwendung von CreateVirtualMfaDevice mit einer CLI
    - Verwendung von DeactivateMfaDevice mit einer CLI
    - Verwenden Sie mit einem DeleteAccessKey AWS SDK oder CLI
    - Verwenden Sie DeleteAccountAliasmit einem AWS SDK oder CLI
    - Verwendung von DeleteAccountPasswordPolicy mit einer CLI
    - Verwenden Sie mit einem DeleteGroup AWS SDK oder CLI

- Verwendung von DeleteGroupPolicy mit einer CLI
- Verwenden Sie mit einem DeleteInstanceProfile AWS SDK oder CLI
- Verwendung von DeleteLoginProfile mit einer CLI
- Verwendung von DeleteOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem DeletePolicy AWS SDK oder CLI
- Verwenden Sie DeletePolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie DeleteRolemit einem AWS SDK oder CLI
- Verwendung von DeleteRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteRolePolicy AWS SDK oder CLI
- Verwenden Sie DeleteSAMLProvider mit einem AWS SDK oder CLI
- Verwenden Sie DeleteServerCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DeleteServiceLinkedRolemit einem AWS SDK oder CLI
- Verwendung von DeleteSigningCertificate mit einer CLI
- Verwenden Sie mit einem DeleteUser AWS SDK oder CLI
- Verwendung von DeleteUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteUserPolicy AWS SDK oder CLI
- Verwendung von DeleteVirtualMfaDevice mit einer CLI
- Verwendung von DetachGroupPolicy mit einer CLI
- Verwenden Sie mit einem DetachRolePolicy AWS SDK oder CLI
- Verwenden Sie DetachUserPolicymit einem AWS SDK oder CLI
- Verwendung von EnableMfaDevice mit einer CLI
- Verwenden Sie mit einem GenerateCredentialReport AWS SDK oder CLI
- Verwendung von GenerateServiceLastAccessedDetails mit einer CLI
- Verwenden Sie mit einem GetAccessKeyLastUsed AWS SDK oder CLI
- Verwenden Sie GetAccountAuthorizationDetailsmit einem AWS SDK oder CLI
- Verwenden Sie GetAccountPasswordPolicymit einem AWS SDK oder CLI
- Verwenden Sie GetAccountSummarymit einem AWS SDK oder CLI
- Verwendung von GetContextKeysForCustomPolicy mit einer CLI
- Verwendung von GetContextKeysForPrincipalPolicy mit einer CLI

- Verwenden Sie mit einem GetCredentialReport AWS SDK oder CLI

- Verwendung von GetGroup mit einer CLI
- Verwendung von GetGroupPolicy mit einer CLI
- Verwendung von GetInstanceProfile mit einer CLI
- Verwendung von GetLoginProfile mit einer CLI
- Verwendung von GetOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem GetPolicy AWS SDK oder CLI
- Verwenden Sie GetPolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie GetRolemit einem AWS SDK oder CLI
- Verwendung von GetRolePolicy mit einer CLI
- Verwendung von GetSamlProvider mit einer CLI
- Verwenden Sie mit einem GetServerCertificate AWS SDK oder CLI
- Verwendung von GetServiceLastAccessedDetails mit einer CLI
- Verwendung von GetServiceLastAccessedDetailsWithEntities mit einer CLI
- Verwenden Sie mit einem GetServiceLinkedRoleDeletionStatus AWS SDK oder CLI
- Verwenden Sie GetUsermit einem AWS SDK oder CLI
- Verwendung von GetUserPolicy mit einer CLI
- Verwenden Sie mit einem ListAccessKeys AWS SDK oder CLI
- Verwenden Sie ListAccountAliasesmit einem AWS SDK oder CLI
- Verwendung von ListAttachedGroupPolicies mit einer CLI
- Verwenden Sie mit einem ListAttachedRolePolicies AWS SDK oder CLI
- Verwendung von ListAttachedUserPolicies mit einer CLI
- Verwendung von ListEntitiesForPolicy mit einer CLI
- Verwendung von ListGroupPolicies mit einer CLI
- Verwenden Sie mit einem ListGroups AWS SDK oder CLI
- Verwendung von ListGroupsForUser mit einer CLI
- Verwendung von ListInstanceProfiles mit einer CLI
- Verwendung von ListInstanceProfilesForRole mit einer CLI
- Verwendung von ListMfaDevices mit einer CLI
- Verwendung von ListOpenIdConnectProviders mit einer CLI

- Verwenden Sie mit einem ListPolicies AWS SDK oder CLI

- Verwenden Sie ListPolicyVersionsmit einem AWS SDK oder CLI
- Verwenden Sie ListRolePoliciesmit einem AWS SDK oder CLI
- Verwendung von ListRoleTags mit einer CLI
- Verwenden Sie mit einem ListRoles AWS SDK oder CLI
- Verwenden Sie ListSamlProviders mit einem AWS SDK oder CLI
- ListServerCertificatesVerwenden Sie mit einem AWS SDK oder CLI
- Verwendung von ListSigningCertificates mit einer CLI
- Verwenden Sie mit einem ListUserPolicies AWS SDK oder CLI
- Verwendung von ListUserTags mit einer CLI
- Verwenden Sie mit einem ListUsers AWS SDK oder CLI
- Verwendung von ListVirtualMfaDevices mit einer CLI
- Verwendung von PutGroupPolicy mit einer CLI
- Verwendung von PutRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutRolePolicy AWS SDK oder CLI
- Verwendung von PutUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutUserPolicy AWS SDK oder CLI
- Verwendung von RemoveClientIdFromOpenIdConnectProvider mit einer CLI
- Verwendung von RemoveRoleFromInstanceProfile mit einer CLI
- Verwendung von RemoveUserFromGroup mit einer CLI
- Verwendung von ResyncMfaDevice mit einer CLI
- Verwenden Sie mit einem SetDefaultPolicyVersion AWS SDK oder CLI
- Verwendung von TagRole mit einer CLI
- Verwendung von TagUser mit einer CLI
- Verwendung von UntagRole mit einer CLI
- Verwendung von UntagUser mit einer CLI
- Verwenden Sie mit einem UpdateAccessKey AWS SDK oder CLI
- Verwendung von UpdateAccountPasswordPolicy mit einer CLI
- Verwendung von UpdateAssumeRolePolicy mit einer CLI
- Verwendung von UpdateGroup mit einer CLI

- Verwendung von UpdateLoginProfile mit einer CLI

    - Verwendung von UpdateOpenIdConnectProviderThumbprint mit einer CLI
    - Verwendung von UpdateRole mit einer CLI
    - Verwendung von UpdateRoleDescription mit einer CLI
    - Verwendung von UpdateSamlProvider mit einer CLI
    - Verwenden Sie mit einem UpdateServerCertificate AWS SDK oder CLI
    - Verwendung von UpdateSigningCertificate mit einer CLI
    - Verwenden Sie mit einem UpdateUser AWS SDK oder CLI
    - Verwenden Sie UploadServerCertificatemit einem AWS SDK oder CLI
    - Verwendung von UploadSigningCertificate mit einer CLI
- Szenarien für die Verwendung von IAM AWS SDKs
  - Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
  - Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
  - Erstellen Sie IAM-Benutzer mit Schreibzugriff und Lese-/Schreibzugriff mit einem AWS SDK
  - Einen Container-Service für VM-Instanzen erstellen
  - Einen verwalteten Monitoring-Workspace erstellen
  - Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
  - Erstellen Sie Ihre erste serverlose Funktion
  - Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
  - Erste Schritte mit dem Konfigurationsmanagement
  - Erste Schritte mit dem Geräteschutz für das Internet der Dinge
  - Erste Schritte mit Feature-Stores für maschinelles Lernen
  - Erste Schritte mit verwalteten Kubernetes-Clustern
  - Erste Schritte mit verwaltetem Streaming
  - Erste Schritte mit bereitgestellten Data Warehouse-Clustern
  - Erste Schritte mit der Workflow-Orchestrierung
  - Verwalten Sie IAM-Zugriffsschlüssel mit einem AWS SDK
  - Verwalten Sie IAM-Richtlinien mit einem AWS SDK

  - Verwalten Sie IAM-Rollen mit einem AWS SDK

- Verwalten Sie Ihr IAM-Konto mit einem AWS SDK
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher
- Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen
- Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation
- Richtlinie zur Aktivierung der Automatisierung für Ihr Konto
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation
- Führen Sie ein Rollback einer IAM-Richtlinienversion mithilfe eines AWS SDK
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2
- Systemmanagement einrichten
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2
- Arbeiten Sie mit der IAM Policy Builder-API unter Verwendung eines AWS SDK

Grundlegende Beispiele für die Verwendung von IAM AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von AWS Identity and Access Management mit AWS SDKs verwenden können.

Beispiele

- Hello IAM
- Lernen Sie die Grundlagen von IAM mit einem AWS SDK
- Aktionen für die Verwendung von IAM AWS SDKs
  - Verwendung von AddClientIdToOpenIdConnectProvider mit einer CLI
  - Verwendung von AddRoleToInstanceProfile mit einer CLI
  - Verwendung von AddUserToGroup mit einer CLI
  - Verwendung von AttachGroupPolicy mit einer CLI
  - Verwenden Sie mit einem AttachRolePolicy AWS SDK oder CLI
  - Verwenden Sie AttachUserPolicymit einem AWS SDK oder CLI
  - Verwendung von ChangePassword mit einer CLI
  - Verwenden Sie mit einem CreateAccessKey AWS SDK oder CLI
  - Verwenden Sie CreateAccountAliasmit einem AWS SDK oder CLI
  - Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem CreateInstanceProfile AWS SDK oder CLI
  - Verwendung von CreateLoginProfile mit einer CLI
  - Verwendung von CreateOpenIdConnectProvider mit einer CLI
  - Verwenden Sie mit einem CreatePolicy AWS SDK oder CLI
  - Verwenden Sie CreatePolicyVersionmit einem AWS SDK oder CLI
  - Verwenden Sie CreateRolemit einem AWS SDK oder CLI
  - Verwenden Sie CreateSAMLProvider mit einem AWS SDK oder CLI
  - Verwenden Sie CreateServiceLinkedRolemit einem AWS SDK oder CLI
  - Verwenden Sie CreateUsermit einem AWS SDK oder CLI
  - Verwendung von CreateVirtualMfaDevice mit einer CLI
  - Verwendung von DeactivateMfaDevice mit einer CLI
  - Verwenden Sie mit einem DeleteAccessKey AWS SDK oder CLI
  - Verwenden Sie DeleteAccountAliasmit einem AWS SDK oder CLI
  - Verwendung von DeleteAccountPasswordPolicy mit einer CLI
  - Verwenden Sie mit einem DeleteGroup AWS SDK oder CLI
  - Verwendung von DeleteGroupPolicy mit einer CLI

  - Verwenden Sie mit einem DeleteInstanceProfile AWS SDK oder CLI

- Verwendung von DeleteLoginProfile mit einer CLI
- Verwendung von DeleteOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem DeletePolicy AWS SDK oder CLI
- Verwenden Sie DeletePolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie DeleteRolemit einem AWS SDK oder CLI
- Verwendung von DeleteRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteRolePolicy AWS SDK oder CLI
- Verwenden Sie DeleteSAMLProvider mit einem AWS SDK oder CLI
- Verwenden Sie DeleteServerCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DeleteServiceLinkedRolemit einem AWS SDK oder CLI
- Verwendung von DeleteSigningCertificate mit einer CLI
- Verwenden Sie mit einem DeleteUser AWS SDK oder CLI
- Verwendung von DeleteUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteUserPolicy AWS SDK oder CLI
- Verwendung von DeleteVirtualMfaDevice mit einer CLI
- Verwendung von DetachGroupPolicy mit einer CLI
- Verwenden Sie mit einem DetachRolePolicy AWS SDK oder CLI
- Verwenden Sie DetachUserPolicymit einem AWS SDK oder CLI
- Verwendung von EnableMfaDevice mit einer CLI
- Verwenden Sie mit einem GenerateCredentialReport AWS SDK oder CLI
- Verwendung von GenerateServiceLastAccessedDetails mit einer CLI
- Verwenden Sie mit einem GetAccessKeyLastUsed AWS SDK oder CLI
- Verwenden Sie GetAccountAuthorizationDetailsmit einem AWS SDK oder CLI
- Verwenden Sie GetAccountPasswordPolicymit einem AWS SDK oder CLI
- Verwenden Sie GetAccountSummarymit einem AWS SDK oder CLI
- Verwendung von GetContextKeysForCustomPolicy mit einer CLI
- Verwendung von GetContextKeysForPrincipalPolicy mit einer CLI
- Verwenden Sie mit einem GetCredentialReport AWS SDK oder CLI
- Verwendung von GetGroup mit einer CLI

- Verwendung von GetGroupPolicy mit einer CLI

- Verwendung von GetInstanceProfile mit einer CLI
- Verwendung von GetLoginProfile mit einer CLI
- Verwendung von GetOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem GetPolicy AWS SDK oder CLI
- Verwenden Sie GetPolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie GetRolemit einem AWS SDK oder CLI
- Verwendung von GetRolePolicy mit einer CLI
- Verwendung von GetSamlProvider mit einer CLI
- Verwenden Sie mit einem GetServerCertificate AWS SDK oder CLI
- Verwendung von GetServiceLastAccessedDetails mit einer CLI
- Verwendung von GetServiceLastAccessedDetailsWithEntities mit einer CLI
- Verwenden Sie mit einem GetServiceLinkedRoleDeletionStatus AWS SDK oder CLI
- Verwenden Sie GetUsermit einem AWS SDK oder CLI
- Verwendung von GetUserPolicy mit einer CLI
- Verwenden Sie mit einem ListAccessKeys AWS SDK oder CLI
- Verwenden Sie ListAccountAliasesmit einem AWS SDK oder CLI
- Verwendung von ListAttachedGroupPolicies mit einer CLI
- Verwenden Sie mit einem ListAttachedRolePolicies AWS SDK oder CLI
- Verwendung von ListAttachedUserPolicies mit einer CLI
- Verwendung von ListEntitiesForPolicy mit einer CLI
- Verwendung von ListGroupPolicies mit einer CLI
- Verwenden Sie mit einem ListGroups AWS SDK oder CLI
- Verwendung von ListGroupsForUser mit einer CLI
- Verwendung von ListInstanceProfiles mit einer CLI
- Verwendung von ListInstanceProfilesForRole mit einer CLI
- Verwendung von ListMfaDevices mit einer CLI
- Verwendung von ListOpenIdConnectProviders mit einer CLI
- Verwenden Sie mit einem ListPolicies AWS SDK oder CLI
- Verwenden Sie ListPolicyVersionsmit einem AWS SDK oder CLI

- Verwenden Sie ListRolePoliciesmit einem AWS SDK oder CLI

- Verwendung von ListRoleTags mit einer CLI
- Verwenden Sie mit einem ListRoles AWS SDK oder CLI
- Verwenden Sie ListSamlProviders mit einem AWS SDK oder CLI
- ListServerCertificatesVerwenden Sie mit einem AWS SDK oder CLI
- Verwendung von ListSigningCertificates mit einer CLI
- Verwenden Sie mit einem ListUserPolicies AWS SDK oder CLI
- Verwendung von ListUserTags mit einer CLI
- Verwenden Sie mit einem ListUsers AWS SDK oder CLI
- Verwendung von ListVirtualMfaDevices mit einer CLI
- Verwendung von PutGroupPolicy mit einer CLI
- Verwendung von PutRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutRolePolicy AWS SDK oder CLI
- Verwendung von PutUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutUserPolicy AWS SDK oder CLI
- Verwendung von RemoveClientIdFromOpenIdConnectProvider mit einer CLI
- Verwendung von RemoveRoleFromInstanceProfile mit einer CLI
- Verwendung von RemoveUserFromGroup mit einer CLI
- Verwendung von ResyncMfaDevice mit einer CLI
- Verwenden Sie mit einem SetDefaultPolicyVersion AWS SDK oder CLI
- Verwendung von TagRole mit einer CLI
- Verwendung von TagUser mit einer CLI
- Verwendung von UntagRole mit einer CLI
- Verwendung von UntagUser mit einer CLI
- Verwenden Sie mit einem UpdateAccessKey AWS SDK oder CLI
- Verwendung von UpdateAccountPasswordPolicy mit einer CLI
- Verwendung von UpdateAssumeRolePolicy mit einer CLI
- Verwendung von UpdateGroup mit einer CLI
- Verwendung von UpdateLoginProfile mit einer CLI
- Verwendung von UpdateOpenIdConnectProviderThumbprint mit einer CLI

- Verwendung von UpdateRole mit einer CLI

- Verwendung von UpdateRoleDescription mit einer CLI
- Verwendung von UpdateSamlProvider mit einer CLI
- Verwenden Sie mit einem UpdateServerCertificate AWS SDK oder CLI
- Verwendung von UpdateSigningCertificate mit einer CLI
- Verwenden Sie mit einem UpdateUser AWS SDK oder CLI
- Verwenden Sie UploadServerCertificatemit einem AWS SDK oder CLI
- Verwendung von UploadSigningCertificate mit einer CLI

Hello IAM

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von IAM beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
```

```
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})
```

```
if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die iam.cpp-Quellendatei.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
```

```
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
```

```
                            policy.GetPolicyId() << std::setw(80) <<
 policy.GetArn() <<
                            std::setw(64) << policy.GetDescription() <<
 std::setw(12) <<
                            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                            std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListPoliciesunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
```

```
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      for (const policy of page.Policies) {
        console.log(`${policy.PolicyName}`);
        policyCount++;
      }
    }
  }
  console.log(`Found ${policyCount} policies.`);
```

```
};
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ListPoliciesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
```

```
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Von src/bin /hello.rs.

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
```

```
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

Von src/iam -service-lib.rs.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der API-Referenz zum AWS SDK für Rust.

Lernen Sie die Grundlagen von IAM mit einem AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
```

```
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
```

```
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
```

```
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
```

```
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
```

```
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
 { UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
```

```
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
```

```
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }
```

```
        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
```

```
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
```

```
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;
```

```
public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();
```

```
        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;
```

```
        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
```

```
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
            });
        }
```

```
        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
```

```
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
 { BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
```

```
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
 creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
```

```
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
```

```
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
```

```
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
```

```
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
```

```
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
  fi

  local assume_role_policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

  local assume_role_policy_name
  assume_role_policy_name=$(generate_random_name "test-assume-role-")

  # shellcheck disable=SC2181
  local assume_role_policy_arn
  assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
    return 1
  fi

  echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
  sleep 10
  echo_repeat "*" 88
  echo

  echo "Try to list buckets without the new user assuming the role."
  echo_repeat "*" 88
  echo

  # Set the environment variables for the created user.
  # bashsupport disable=BP2001
  export AWS_ACCESS_KEY_ID=$key_name
  # bashsupport disable=BP2001
  export AWS_SECRET_ACCESS_KEY=$key_secret
```

```
 local buckets
 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials

 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)
```

```
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
 succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
 "$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

Die in diesem Szenario verwendeten IAM-Funktionen.

```
##############################################################################
```

```
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
```

```
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
```

```
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-role operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
```

```
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
```

```
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
```

```
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi
```

```
  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy

- CreateAccessKey
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
```

```
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                                Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }
```

```
        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());
```

```
        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

  request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");
```

```
        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);
```

```
        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
```

```
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
```

```
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
 role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}
```

```
bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
 client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
```

```
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
```

```
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser(ctx)
 accessKey := scenario.CreateAccessKey(ctx, user)
 role := scenario.CreateRoleAndPolicies(ctx, user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
 scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
 scenario.Cleanup(ctx, user, role)
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(ctx, userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
```

```
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context,
 user *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"},
 "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}
```

```
// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
  ))
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

 s3Client := s3.NewFromConfig(noPermsConfig)
 _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
 } else {
  log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
   "but the call succeeded. Continuing the example anyway...")
```

```
 }
 log.Println(strings.Repeat("-", 88))
 return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
//  1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//     the user's access key credentials.
//  2. Gets temporary credentials by assuming the role that grants permission to
 list the
//     buckets.
//  3. Creates an Amazon S3 client from the temporary credentials.
//  4. Lists buckets for the account. Because the temporary credentials are
 generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx
 context.Context, noPermsConfig *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
  RoleArn:         role.Arn,
  RoleSessionName: aws.String("AssumeRoleExampleSession"),
  DurationSeconds: aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }
```

```
 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
   err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
 *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   log.Printf("Detached policy %v from role %v and deleted the policy.\n",
    *policy.PolicyName, *role.RoleName)
  }
  err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
```

```
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, userPol := range userPols {
   err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
  }
  keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, key := range keys {
   err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName,
 *key.AccessKeyId)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
  }
  err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
```

```
 Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

Definieren Sie eine Struktur, die Kontoaktionen umschließt.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}
```

```
// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

Definieren Sie eine Struktur, die Richtlinienaktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
```

```
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

Definieren Sie eine Struktur, die Rollenaktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
```

```
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
```

```
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
```

```
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
```

```
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

Definieren Sie eine Struktur, die Benutzeraktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
```

```
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
```

```
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
```

```
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
```

```
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
```

```
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
```

```
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
```

```
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
```

```
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
```

```
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();
```

```
            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn)
{

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
```

```
                .policyArn(polArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
                .policyArn(polArn)
                .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted " + polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKey(IamClient iam, String username, String
accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                .accessKeyId(accessKey)
                .userName(username)
                .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
                " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
```

```
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
 * Create a new IAM user. If the user already exists, give
```

```
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }
```

```
 // Create an access key. This key is used to authenticate the new user to
 // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
(AWS STS).
 // It's not best practice to use access keys. For more information, see
https://aws.amazon.com/iam/resources/best-practices/.
 const createAccessKeyResponse = await iamClient.send(
   new CreateAccessKeyCommand({ UserName: userName }),
 );

 if (
   !createAccessKeyResponse.AccessKey?.AccessKeyId ||
   !createAccessKeyResponse.AccessKey?.SecretAccessKey
 ) {
   throw new Error("Access key not created");
 }

 const {
   AccessKey: { AccessKeyId, SecretAccessKey },
 } = createAccessKeyResponse;

 let s3Client = new S3Client({
   credentials: {
     accessKeyId: AccessKeyId,
     secretAccessKey: SecretAccessKey,
   },
 });

 // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
 // thrown while the user and access keys are still stabilizing.
 await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
   try {
     return await listBuckets(s3Client);
   } catch (err) {
     if (err instanceof Error && err.name === "InvalidAccessKeyId") {
       throw err;
     }
   }
 });

 // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
 // is thrown while the user and access keys are still stabilizing.
 const { Role } = await retry(
```

```
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Action: ["s3:ListAllMyBuckets"],
            Resource: "*",
          },
        ],
      }),
      PolicyName: policyName,
    }),
```

```
  );

  if (!listBucketPolicy) {
    throw new Error("Policy not created");
  }

  // Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
  await iamClient.send(
    new AttachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  // Assume the role.
  const stsClient = new STSClient({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the assume role operation until it succeeds.
  const { Credentials } = await retry(
    { intervalInMs: 2000, maxRetries: 60 },
    () =>
      stsClient.send(
        new AssumeRoleCommand({
          RoleArn: Role.Arn,
          RoleSessionName: `iamBasicScenarioSession-${Math.floor(
            Math.random() * 1000000,
          )}`,
          DurationSeconds: 900,
        }),
      ),
  );

  if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
    throw new Error("Credentials not created");
  }

  s3Client = new S3Client({
    credentials: {
      accessKeyId: Credentials.AccessKeyId,
```

```
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
  }),
);

await iamClient.send(
  new DeleteRoleCommand({
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeleteAccessKeyCommand({
    UserName: userName,
    AccessKeyId,
  }),
);

await iamClient.send(
  new DeleteUserCommand({
    UserName: userName,
  }),
);
```

```
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
```

```
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
```

```
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
```

```
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
```

```
    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
```

```
    return jsonParser.parse(reader)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
```

```
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
```

```
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
```

```
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
```

```
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
"
            f"the role."
        )
    except ClientError as error:
        print(
```

```
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.
```

```
    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
```

```
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
```

```
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
```

```
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
 and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
 #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
 #{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
```

```
 # @param role [Aws::IAM::Role] The role that can be assumed.
 # @return [Aws::IAM::UserPolicy] The newly created policy.
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: '2012-10-17',
     Statement: [{
       Effect: 'Allow',
       Action: 'sts:AssumeRole',
       Resource: role.arn
     }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
```

```
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end
```

```
  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
 role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
 policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
```

```
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - AttachRolePolicy

- CreateAccessKey
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;
```

```
    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
```

```
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
```

```
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
```

```
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
```

```
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole

- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen für die Verwendung von IAM AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne IAM-Aktionen mit SDKs ausgeführt werden. AWS Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die IAM-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von IAM AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Identity and Access Management -API-Referenz.

Beispiele

- Verwendung von AddClientIdToOpenIdConnectProvider mit einer CLI
- Verwendung von AddRoleToInstanceProfile mit einer CLI
- Verwendung von AddUserToGroup mit einer CLI
- Verwendung von AttachGroupPolicy mit einer CLI
- Verwenden Sie mit einem AttachRolePolicy AWS SDK oder CLI
- Verwenden Sie AttachUserPolicymit einem AWS SDK oder CLI
- Verwendung von ChangePassword mit einer CLI
- Verwenden Sie mit einem CreateAccessKey AWS SDK oder CLI
- Verwenden Sie CreateAccountAliasmit einem AWS SDK oder CLI
- Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
- Verwenden Sie mit einem CreateInstanceProfile AWS SDK oder CLI
- Verwendung von CreateLoginProfile mit einer CLI
- Verwendung von CreateOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem CreatePolicy AWS SDK oder CLI

- Verwenden Sie CreatePolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie CreateRolemit einem AWS SDK oder CLI
- Verwenden Sie CreateSAMLProvider mit einem AWS SDK oder CLI
- Verwenden Sie CreateServiceLinkedRolemit einem AWS SDK oder CLI
- Verwenden Sie CreateUsermit einem AWS SDK oder CLI
- Verwendung von CreateVirtualMfaDevice mit einer CLI
- Verwendung von DeactivateMfaDevice mit einer CLI
- Verwenden Sie mit einem DeleteAccessKey AWS SDK oder CLI
- Verwenden Sie DeleteAccountAliasmit einem AWS SDK oder CLI
- Verwendung von DeleteAccountPasswordPolicy mit einer CLI
- Verwenden Sie mit einem DeleteGroup AWS SDK oder CLI
- Verwendung von DeleteGroupPolicy mit einer CLI
- Verwenden Sie mit einem DeleteInstanceProfile AWS SDK oder CLI
- Verwendung von DeleteLoginProfile mit einer CLI
- Verwendung von DeleteOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem DeletePolicy AWS SDK oder CLI
- Verwenden Sie DeletePolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie DeleteRolemit einem AWS SDK oder CLI
- Verwendung von DeleteRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteRolePolicy AWS SDK oder CLI
- Verwenden Sie DeleteSAMLProvider mit einem AWS SDK oder CLI
- Verwenden Sie DeleteServerCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DeleteServiceLinkedRolemit einem AWS SDK oder CLI
- Verwendung von DeleteSigningCertificate mit einer CLI
- Verwenden Sie mit einem DeleteUser AWS SDK oder CLI
- Verwendung von DeleteUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem DeleteUserPolicy AWS SDK oder CLI
- Verwendung von DeleteVirtualMfaDevice mit einer CLI
- Verwendung von DetachGroupPolicy mit einer CLI
- Verwenden Sie mit einem DetachRolePolicy AWS SDK oder CLI

- Verwenden Sie DetachUserPolicymit einem AWS SDK oder CLI
- Verwendung von EnableMfaDevice mit einer CLI
- Verwenden Sie mit einem GenerateCredentialReport AWS SDK oder CLI
- Verwendung von GenerateServiceLastAccessedDetails mit einer CLI
- Verwenden Sie mit einem GetAccessKeyLastUsed AWS SDK oder CLI
- Verwenden Sie GetAccountAuthorizationDetailsmit einem AWS SDK oder CLI
- Verwenden Sie GetAccountPasswordPolicymit einem AWS SDK oder CLI
- Verwenden Sie GetAccountSummarymit einem AWS SDK oder CLI
- Verwendung von GetContextKeysForCustomPolicy mit einer CLI
- Verwendung von GetContextKeysForPrincipalPolicy mit einer CLI
- Verwenden Sie mit einem GetCredentialReport AWS SDK oder CLI
- Verwendung von GetGroup mit einer CLI
- Verwendung von GetGroupPolicy mit einer CLI
- Verwendung von GetInstanceProfile mit einer CLI
- Verwendung von GetLoginProfile mit einer CLI
- Verwendung von GetOpenIdConnectProvider mit einer CLI
- Verwenden Sie mit einem GetPolicy AWS SDK oder CLI
- Verwenden Sie GetPolicyVersionmit einem AWS SDK oder CLI
- Verwenden Sie GetRolemit einem AWS SDK oder CLI
- Verwendung von GetRolePolicy mit einer CLI
- Verwendung von GetSamlProvider mit einer CLI
- Verwenden Sie mit einem GetServerCertificate AWS SDK oder CLI
- Verwendung von GetServiceLastAccessedDetails mit einer CLI
- Verwendung von GetServiceLastAccessedDetailsWithEntities mit einer CLI
- Verwenden Sie mit einem GetServiceLinkedRoleDeletionStatus AWS SDK oder CLI
- Verwenden Sie GetUsermit einem AWS SDK oder CLI
- Verwendung von GetUserPolicy mit einer CLI
- Verwenden Sie mit einem ListAccessKeys AWS SDK oder CLI
- Verwenden Sie ListAccountAliasesmit einem AWS SDK oder CLI
- Verwendung von ListAttachedGroupPolicies mit einer CLI

- Verwenden Sie mit einem ListAttachedRolePolicies AWS SDK oder CLI
- Verwendung von ListAttachedUserPolicies mit einer CLI
- Verwendung von ListEntitiesForPolicy mit einer CLI
- Verwendung von ListGroupPolicies mit einer CLI
- Verwenden Sie mit einem ListGroups AWS SDK oder CLI
- Verwendung von ListGroupsForUser mit einer CLI
- Verwendung von ListInstanceProfiles mit einer CLI
- Verwendung von ListInstanceProfilesForRole mit einer CLI
- Verwendung von ListMfaDevices mit einer CLI
- Verwendung von ListOpenIdConnectProviders mit einer CLI
- Verwenden Sie mit einem ListPolicies AWS SDK oder CLI
- Verwenden Sie ListPolicyVersionsmit einem AWS SDK oder CLI
- Verwenden Sie ListRolePoliciesmit einem AWS SDK oder CLI
- Verwendung von ListRoleTags mit einer CLI
- Verwenden Sie mit einem ListRoles AWS SDK oder CLI
- Verwenden Sie ListSamlProviders mit einem AWS SDK oder CLI
- ListServerCertificatesVerwenden Sie mit einem AWS SDK oder CLI
- Verwendung von ListSigningCertificates mit einer CLI
- Verwenden Sie mit einem ListUserPolicies AWS SDK oder CLI
- Verwendung von ListUserTags mit einer CLI
- Verwenden Sie mit einem ListUsers AWS SDK oder CLI
- Verwendung von ListVirtualMfaDevices mit einer CLI
- Verwendung von PutGroupPolicy mit einer CLI
- Verwendung von PutRolePermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutRolePolicy AWS SDK oder CLI
- Verwendung von PutUserPermissionsBoundary mit einer CLI
- Verwenden Sie mit einem PutUserPolicy AWS SDK oder CLI
- Verwendung von RemoveClientIdFromOpenIdConnectProvider mit einer CLI
- Verwendung von RemoveRoleFromInstanceProfile mit einer CLI
- Verwendung von RemoveUserFromGroup mit einer CLI

- Verwendung von ResyncMfaDevice mit einer CLI
- Verwenden Sie mit einem SetDefaultPolicyVersion AWS SDK oder CLI
- Verwendung von TagRole mit einer CLI
- Verwendung von TagUser mit einer CLI
- Verwendung von UntagRole mit einer CLI
- Verwendung von UntagUser mit einer CLI
- Verwenden Sie mit einem UpdateAccessKey AWS SDK oder CLI
- Verwendung von UpdateAccountPasswordPolicy mit einer CLI
- Verwendung von UpdateAssumeRolePolicy mit einer CLI
- Verwendung von UpdateGroup mit einer CLI
- Verwendung von UpdateLoginProfile mit einer CLI
- Verwendung von UpdateOpenIdConnectProviderThumbprint mit einer CLI
- Verwendung von UpdateRole mit einer CLI
- Verwendung von UpdateRoleDescription mit einer CLI
- Verwendung von UpdateSamlProvider mit einer CLI
- Verwenden Sie mit einem UpdateServerCertificate AWS SDK oder CLI
- Verwendung von UpdateSigningCertificate mit einer CLI
- Verwenden Sie mit einem UpdateUser AWS SDK oder CLI
- Verwenden Sie UploadServerCertificatemit einem AWS SDK oder CLI
- Verwendung von UploadSigningCertificate mit einer CLI

Verwendung von **`AddClientIdToOpenIdConnectProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddClientIdToOpenIdConnectProvider` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Open-ID Connect-Anbieter (OIDC) eine Client-ID (Audience) hinzu

Der folgende `add-client-id-to-open-id-connect-provider`-Befehl fügt die Client-ID `my-application-ID` zum OIDC-Anbieter mit dem Namen `server.example.com` hinzu.

```
aws iam add-client-id-to-open-id-connect-provider \
    --client-id my-application-ID \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `create-open-id-connect-provider`-Befehl, um einen OIDC-Anbieter zu erstellen.

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter AddClientIdToOpenIdConnectProvider AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Mit diesem Befehl wird die Client-ID (oder Zielgruppe) **`my-application-ID`** zum vorhandenen OIDC-Anbieter mit dem Namen **`server.example.com`** hinzugefügt.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Einzelheiten zur API finden Sie unter AddClientIdToOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl wird die Client-ID (oder Zielgruppe) **`my-application-ID`** zum vorhandenen OIDC-Anbieter mit dem Namen **`server.example.com`** hinzugefügt.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- Einzelheiten zur API finden Sie unter AddClientIdToOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AddRoleToInstanceProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddRoleToInstanceProfile` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Einen Containerdienst für VM-Instanzen erstellen
- Erste Schritte mit verwaltetem Streaming
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

CLI

AWS CLI

So fügen Sie eine Rolle zu einem Instance-Profil hinzu

Mit dem folgenden `add-role-to-instance-profile`-Befehl wird die Rolle mit dem Namen `S3Access` zum Instance-Profil mit dem Namen `Webserver` hinzugefügt.

```
aws iam add-role-to-instance-profile \
    --role-name S3Access \
    --instance-profile-name Webserver
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `create-instance-profile`-Befehl, um ein Instance-Profil zu erstellen.

Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie AddRoleToInstanceProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Mit diesem Befehl wird die Rolle mit dem Namen **`S3Access`** einem vorhandenen Instance-Profil mit dem Namen **`webserver`** hinzugefügt. Verwenden Sie den **`New-IAMInstanceProfile`**-Befehl, um das Instance-Profil zu erstellen. Nachdem Sie das

Instance-Profil erstellt und es mithilfe dieses Befehls einer Rolle zugeordnet haben, können Sie es an eine EC2 Instance anfügen. Verwenden Sie dazu das **`New-EC2Instance`** Cmdlet mit dem Parameter **`InstanceProfile_Arn`** oder **`InstanceProfile-Name`**, um die neue Instance zu starten.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- Einzelheiten zur API finden Sie unter AddRoleToInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl wird die Rolle mit dem Namen **`S3Access`** einem vorhandenen Instance-Profil mit dem Namen **`webserver`** hinzugefügt. Verwenden Sie den **`New-IAMInstanceProfile`**-Befehl, um das Instance-Profil zu erstellen. Nachdem Sie das Instance-Profil erstellt und es mithilfe dieses Befehls einer Rolle zugeordnet haben, können Sie es an eine EC2 Instance anfügen. Verwenden Sie dazu das **`New-EC2Instance`** Cmdlet mit dem Parameter **`InstanceProfile_Arn`** oder **`InstanceProfile-Name`**, um die neue Instance zu starten.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- Einzelheiten zur API finden Sie unter AddRoleToInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AddUserToGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddUserToGroup` verwendet wird.

CLI

AWS CLI

So fügen Sie einen Benutzer einer IAM-Gruppe hinzu

Mit dem folgenden `add-user-to-group`-Befehl wird ein Benutzer mit dem Namen `Bob` zur IAM-Gruppe mit dem Namen `Admins` hinzugefügt.

```
aws iam add-user-to-group \
```

```
    --user-name Bob \
    --group-name Admins
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Hinzufügen und Entfernen von Benutzern in einer IAM-Benutzergruppe im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie AddUserToGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Mit diesem Befehl wird der Benutzer mit dem Namen **Bob** der Gruppe mit dem Namen **Admins** hinzugefügt.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Einzelheiten zur API finden Sie unter AddUserToGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl wird der Benutzer mit dem Namen **Bob** der Gruppe mit dem Namen **Admins** hinzugefügt.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- Einzelheiten zur API finden Sie unter AddUserToGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`AttachGroupPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AttachGroupPolicy` verwendet wird.

CLI

AWS CLI

So fügen Sie einer IAM-Gruppe eine verwaltete Richtlinie hinzu

Mit dem folgenden `attach-group-policy` Befehl wird die benannte AWS verwaltete Richtlinie `ReadOnlyAccess` an die angegebene IAM-Gruppe angehängt. `Finance`

```
aws iam attach-group-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --group-name Finance
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen hierzu finden Sie unter Verwaltete Richtlinien und eingebundene Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter AttachGroupPolicy AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die vom Kunden verwaltete Richtlinie mit dem Namen **`TesterPolicy`** an die IAM-Gruppe **`Testers`** angefügt. Die in der Standardversion dieser Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`AdministratorAccess`** der IAM-Gruppe angehängt. **`Admins`** Die in der neuesten Version dieser Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Einzelheiten zur API finden Sie unter AttachGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die vom Kunden verwaltete Richtlinie mit dem Namen **`TesterPolicy`** an die IAM-Gruppe **`Testers`** angefügt. Die in der Standardversion dieser

Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen der **`AdministratorAccess`** IAM-Gruppe angehängt. **`Admins`** Die in der neuesten Version dieser Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- Einzelheiten zur API finden Sie unter AttachGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`AttachRolePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachRolePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Einen Containerdienst für VM-Instanzen erstellen
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern

- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten Sie Rollen
- Übertragung von hartcodierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Einrichtung der Systemverwaltung
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Arbeiten Sie mit Streams und Time-to-Live

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_attach_role_policy"
  echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam attach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
```

```
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                                return policy.GetPolicyArn() == policyArn;
                        })) {
            std::cout << "Policy " << policyArn <<
                      " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
 iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                  roleName << ": " << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role "
 <<
                  roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie einer IAM-Rolle eine verwaltete Richtlinie an

Mit dem folgenden `attach-role-policy` Befehl wird die benannte AWS verwaltete Richtlinie `ReadOnlyAccess` an die angegebene IAM-Rolle angehängt. `ReadOnlyRole`

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --role-name ReadOnlyRole
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen hierzu finden Sie unter Verwaltete Richtlinien und eingebundene Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter AttachRolePolicy AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
```

```
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
```

```
import
 software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }
```

```
    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Anfügen einer Richtlinie.

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
```

```
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
```

```
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`SecurityAudit`** der IAM-Rolle angehängt. **`CoSecurityAuditors`** Die Benutzer, die

diese Rolle übernehmen, sind unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Einzelheiten zur API finden Sie unter AttachRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen der **`SecurityAudit`** IAM-Rolle angehängt. **`CoSecurityAuditors`** Die Benutzer, die diese Rolle übernehmen, sind unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- Einzelheiten zur API finden Sie unter AttachRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie einer Rolle mithilfe des Boto3-Richtlinienobjekts eine Richtlinie an.

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
```

```
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

Fügen Sie einer Rolle mithilfe des Boto3-Rollenobjekts eine Richtlinie an.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- Einzelheiten zur API finden Sie AttachRolePolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->attachrolepolicy(
      iv_rolename = iv_role_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy attached to role successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role or policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func attachRolePolicy(role: String, policyArn: String) async throws {
        let input = AttachRolePolicyInput(
            policyArn: policyArn,
            roleName: role
        )
        do {
            _ = try await client.attachRolePolicy(input: input)
        } catch {
            print("ERROR: Attaching a role policy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`AttachUserPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachUserPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

CLI

AWS CLI

So fügen Sie einem IAM-Benutzer eine verwaltete Richtlinie an

Mit dem folgenden `attach-user-policy` Befehl wird die AWS verwaltete Richtlinie mit `AdministratorAccess` dem Namen des IAM-Benutzers verknüpft. `Alice`

```
aws iam attach-user-policy \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
```

```
    --user-name Alice
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen hierzu finden Sie unter Verwaltete Richtlinien und eingebundene Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter AttachUserPolicy AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen **`AmazonCognitoPowerUser`** des IAM-Benutzers angehängt. **`Bob`** Der Benutzer ist unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Einzelheiten zur API finden Sie unter AttachUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen des **`AmazonCognitoPowerUser`** IAM-Benutzers angehängt. **`Bob`** Der Benutzer ist unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- Einzelheiten zur API finden Sie unter AttachUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- Einzelheiten zur API finden Sie AttachUserPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- Einzelheiten zur API finden Sie AttachUserPolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;
```

```
    Ok(())
}
```

- Einzelheiten zur API finden Sie AttachUserPolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->attachuserpolicy(
          iv_username = iv_user_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy attached to user successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User or policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AttachUserPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`ChangePassword`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ChangePassword` verwendet wird.

CLI

AWS CLI

So ändern Sie das Passwort für Ihren IAM-Benutzer

Um das Passwort für Ihren IAM-Benutzer zu ändern, empfehlen wir, den `--cli-input-json`-Parameter zu verwenden, um eine JSON-Datei zu übergeben, die Ihr altes und Ihr neues Passwort enthält. Mit dieser Methode können Sie sichere Passwörter mit nicht-alphanumerischen Zeichen verwenden. Es kann schwierig sein, Passwörter mit nicht-alphanumerischen Zeichen zu verwenden, wenn Sie diese als Befehlszeilenparameter übergeben. Um den `--cli-input-json`-Parameter zu verwenden, verwenden Sie zunächst den `change-password`-Befehl mit dem `--generate-cli-skeleton`-Parameter, wie im folgenden Beispiel.

```
aws iam change-password \
    --generate-cli-skeleton > change-password.json
```

Mit dem vorherigen Befehl wird eine JSON-Datei mit dem Namen „change-password.json“ erstellt, die Sie zum Eingeben Ihrer alten und neuen Passwörter verwenden können. Die Datei könnte beispielsweise wie folgt aussehen.

```
{
    "OldPassword": "3s0K_;xh4~8XXI",
    "NewPassword": "]35d/{pB9Fo9wJ"
}
```

Um Ihr Passwort zu ändern, verwenden Sie als Nächstes den `change-password`-Befehl erneut und übergeben dieses Mal den `--cli-input-json`-Parameter zur Angabe Ihrer JSON-Datei. Der folgende `change-password`-Befehl verwendet den `--cli-input-json`-Parameter mit einer JSON-Datei mit dem Namen change-password.json.

```
aws iam change-password \
    --cli-input-json file://change-password.json
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Dieser Befehl kann nur von IAM-Benutzern aufgerufen werden. Wenn dieser Befehl mit AWS Kontoanmeldeinformationen (Root) aufgerufen wird, gibt der Befehl einen `InvalidUserType` Fehler zurück.

Weitere Informationen finden Sie unter So ändert ein IAM-Benutzer sein eigenes Passwort im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ChangePasswordin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Mit diesem Befehl wird das Passwort des Benutzers geändert, der den Befehl ausführt. Dieser Befehl kann nur von IAM-Benutzern aufgerufen werden. Wenn dieser Befehl aufgerufen wird, wenn Sie mit AWS Kontoanmeldeinformationen (Root) angemeldet sind, gibt der Befehl einen Fehler zurück. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Einzelheiten zur API finden Sie unter ChangePassword AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl wird das Passwort des Benutzers geändert, der den Befehl ausführt. Dieser Befehl kann nur von IAM-Benutzern aufgerufen werden. Wenn dieser Befehl aufgerufen wird, während Sie mit AWS Kontoanmeldeinformationen (Root) angemeldet sind, gibt der Befehl einen Fehler zurück. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- Einzelheiten zur API finden Sie unter ChangePassword AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateAccessKey`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAccessKey` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern
- Verwalten von Zugriffsschlüsseln

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
```

```
  echo "  -u user_name   The name of the IAM user."
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi
```

```
  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome =
 iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
```

```
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey()
 <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Zugriffsschlüssel für einen IAM-Benutzer

Mit dem folgenden `create-access-key`-Befehl wird ein Zugriffsschlüssel (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel) für den IAM-Benutzer mit dem Namen `Bob` erstellt.

```
aws iam create-access-key \
    --user-name Bob
```

Ausgabe:

```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

Speichern Sie den geheimen Zugriffsschlüssel an einem sicheren Ort. Bei Verlust kann er nicht wiederhergestellt werden und Sie müssen einen neuen Zugriffsschlüssel erstellen.

Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
```

```
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Zugriffsschlüssel.

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
```

```
    }
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Paar aus Zugriffsschlüssel und geheimem Zugriffsschlüssel erstellt und dem Benutzer **`David`** zugewiesen. Stellen Sie sicher, dass Sie die **`AccessKeyId`**- und **`SecretAccessKey`**-Werte in einer Datei speichern, da Sie den **`SecretAccessKey`** nur zu diesem Zeitpunkt abrufen können. Sie können es später nicht abrufen. Bei Verlust des geheimen Schlüssels müssen Sie ein neues Zugriffsschlüsselpaar erstellen.

```
New-IAMAccessKey -UserName David
```

Ausgabe:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Einzelheiten zur API finden Sie unter CreateAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Paar aus Zugriffsschlüssel und geheimem Zugriffsschlüssel erstellt und dem Benutzer **`David`** zugewiesen. Stellen Sie sicher, dass Sie die **`AccessKeyId`**- und **`SecretAccessKey`**-Werte in einer Datei speichern, da Sie den **`SecretAccessKey`** nur zu diesem Zeitpunkt abrufen können. Sie können es später nicht abrufen. Bei Verlust des geheimen Schlüssels müssen Sie ein neues Zugriffsschlüsselpaar erstellen.

```
New-IAMAccessKey -UserName David
```

Ausgabe:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- Einzelheiten zur API finden Sie unter CreateAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
```

```
        raise
    else:
        return key_pair
```

- Einzelheiten zur API finden Sie CreateAccessKeyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
```

```
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
```

```
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
```

```
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createaccesskey(
          iv_username = iv_user_name ).
        MESSAGE 'Access key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Maximum number of access keys reached.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createAccessKey(userName: String) async throws ->
 IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
        do {
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            print("ERROR: createAccessKey:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateAccountAlias`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAccountAlias` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Konto-Alias

Der folgende `create-account-alias` Befehl erstellt den Alias `examplecorp` für Ihr AWS Konto.

```
aws iam create-account-alias \
    --account-alias examplecorp
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ihre AWS Konto-ID und deren Alias im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter CreateAccountAlias AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example,
 myawsaccount).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request =
 CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Konto-Alias.

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird der Kontoalias für Ihr AWS Konto in geändert**mycompanyaws**. <accountidnumber>Die Adresse der Benutzeranmeldeseite wird in https://mycompanyaws.signin.aws.amazon.com/console. Die ursprüngliche URL geändert, wobei Ihre Konto-ID-Nummer anstelle des Alias verwendet wird (https://.signin.aws.amazon). com/console) funktioniert weiterhin. Alle zuvor definierten Alias-basierten URLs funktionieren jedoch nicht mehr.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter CreateAccountAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Kontoalias für Ihr AWS Konto in geändert. **mycompanyaws** <accountidnumber>Die Adresse der Benutzeranmeldeseite wird in https://mycompanyaws.signin.aws.amazon.com/console. Die ursprüngliche URL geändert, wobei Ihre Konto-ID-Nummer anstelle des Alias verwendet wird (https://.signin.aws.amazon). com/console) funktioniert weiterhin. Alle zuvor definierten Alias-basierten URLs funktionieren jedoch nicht mehr.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter CreateAccountAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- Einzelheiten zur API finden Sie CreateAccountAliasin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->createaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Account alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Account alias limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGroup` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine IAM-Gruppe

Mit dem folgenden `create-group`-Befehl wird eine IAM-Gruppe mit dem Namen `Admins` erstellt.

```
aws iam create-group \
    --group-name Admins
```

Ausgabe:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-03-09T20:30:24.940Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-Benutzergruppen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie CreateGroupin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue IAM-Gruppe mit dem Namen **`Developers`** erstellt.

```
New-IAMGroup -GroupName Developers
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue IAM-Gruppe mit dem Namen **`Developers`** erstellt.

```
New-IAMGroup -GroupName Developers
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateInstanceProfile`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateInstanceProfile` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Einen Containerdienst für VM-Instanzen erstellen
- Erste Schritte mit verwaltetem Streaming
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
```

```
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
```

```
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
```

```
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- Einzelheiten zur API finden Sie CreateInstanceProfilein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Instance-Profil

Der folgende `create-instance-profile`-Befehl erstellt ein Instance-Profil mit dem Namen `Webserver`.

```
aws iam create-instance-profile \
    --instance-profile-name Webserver
```

Ausgabe:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJMBYC7DLSPEXAMPLE",
        "Roles": [],
        "CreateDate": "2015-03-09T20:33:19.626Z",
        "InstanceProfileName": "Webserver",
        "Path": "/",
```

```
        "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
    }
}
```

Verwenden Sie den `add-role-to-instance-profile`-Befehl, um einem Instance-Profil eine Rolle hinzuzufügen.

Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Gewähren von Berechtigungen für Anwendungen, die in Amazon-EC2-Instances ausgeführt werden im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateInstanceProfilein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- Einzelheiten zur API finden Sie CreateInstanceProfilein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues IAM Instance-Profil mit dem Namen **ProfileForDevEC2Instance** erstellt. Sie müssen den Befehl **Add-IAMRoleToInstanceProfile** separat ausführen, um das Instance-Profil einer vorhandenen IAM-Rolle zuzuordnen, die Berechtigungen für die Instance bereitstellt. Fügen Sie abschließend das Instance-Profil einer EC2-Instance an, wenn Sie diese starten. Verwenden Sie dazu das Cmdlet **New-EC2Instance** entweder mit dem Parameter **InstanceProfile_Arn** oder **InstanceProfile_Name**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Einzelheiten zur API finden Sie unter CreateInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues IAM Instance-Profil mit dem Namen **ProfileForDevEC2Instance** erstellt. Sie müssen den Befehl **Add-IAMRoleToInstanceProfile** separat ausführen, um das Instance-Profil einer vorhandenen IAM-Rolle zuzuordnen, die Berechtigungen für die Instance bereitstellt. Fügen Sie abschließend das Instance-Profil einer EC2-Instance an, wenn Sie diese starten. Verwenden Sie dazu das Cmdlet **New-EC2Instance** entweder mit dem Parameter **InstanceProfile_Arn** oder **InstanceProfile_Name**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- Einzelheiten zur API finden Sie unter CreateInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel werden eine Richtlinie, eine Rolle und ein Instance-Profil erstellt und alle miteinander verknüpft.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.
```

```
        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
```

```
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
```

```
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

- Einzelheiten zur API finden Sie CreateInstanceProfilein AWS SDK for Python (Boto3) API Reference.

Verwendung von **`CreateLoginProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateLoginProfile` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein Passwort für einen IAM-Benutzer

Für die Erstellung eines Passworts für einen IAM-Benutzer empfehlen wir die Nutzung des Parameters `--cli-input-json` zur Übergabe einer JSON-Datei, die das Passwort enthält. Mit dieser Methode können Sie ein sicheres Passwort mit nicht alphanumerischen Zeichen erstellen. Es kann schwierig sein, ein Passwort mit nicht-alphanumerischen Zeichen zu erstellen, wenn Sie es als Befehlszeilenparameter übergeben.

Um den `--cli-input-json`-Parameter zu verwenden, verwenden Sie zunächst den `create-login-profile`-Befehl mit dem `--generate-cli-skeleton`-Parameter, wie im folgenden Beispiel.

```
aws iam create-login-profile \
    --generate-cli-skeleton > create-login-profile.json
```

Der vorherige Befehl erstellt eine JSON-Datei mit dem Namen create-login-profile.json, die Sie zur Eingabe der Informationen für einen nachfolgenden Befehl `create-login-profile` verwenden können. Beispiel:

```
{
    "UserName": "Bob",
    "Password": "&1-3a6u:RA0djs",
    "PasswordResetRequired": true
}
```

Verwenden Sie anschließend zur Erstellung eines Passworts für einen IAM-Benutzer erneut den Befehl `create-login-profile`, diesmal jedoch mit dem Parameter `--cli-input-json` zur Angabe Ihrer JSON-Datei. Der folgende Befehl `create-login-profile` verwendet den Parameter `--cli-input-json` mit einer JSON-Datei, die als create-login-profile.json aufgerufen wird.

```
aws iam create-login-profile \
    --cli-input-json file://create-login-profile.json
```

Ausgabe:

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2015-03-10T20:55:40.274Z",
        "PasswordResetRequired": true
    }
}
```

Wenn das neue Passwort gegen die Passwortrichtlinie des Kontos verstößt, gibt der Befehl einen `PasswordPolicyViolation`-Fehler zurück.

Um das Passwort für einen Benutzer zu ändern, der bereits über ein Passwort verfügt, verwenden Sie `update-login-profile`. Um eine Passwortrichtlinie für das Konto festzulegen, verwenden Sie den `update-account-password-policy`-Befehl.

Sofern die Passwortrichtlinie des Kontos dies zulässt, können IAM-Benutzer mit dem `change-password`-Befehl ihre eigenen Passwörter ändern.

Weitere Informationen finden Sie unter Verwalten von Passwörtern für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateLoginProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein (temporäres) Passwort für den IAM-Benutzer Bob erstellt und festgelegt, dass der Benutzer das Passwort bei der nächsten Anmeldung von **Bob** ändern muss.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Ausgabe:

```
CreateDate                    PasswordResetRequired                UserName
----------                    ---------------------                --------
4/14/2015 12:26:30 PM         True                                 Bob
```

- Einzelheiten zur API finden Sie unter CreateLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein (temporäres) Passwort für den IAM-Benutzer Bob erstellt und festgelegt, dass der Benutzer das Passwort bei der nächsten Anmeldung von **Bob** ändern muss.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Ausgabe:

```
CreateDate                    PasswordResetRequired                UserName
----------                    ---------------------                --------
4/14/2015 12:26:30 PM         True                                 Bob
```

- Einzelheiten zur API finden Sie unter CreateLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateOpenIdConnectProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOpenIdConnectProvider` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen OpenID Connect (OIDC)-Anbieter

Um einen OpenID Connect (OIDC)-Anbieter zu erstellen, empfehlen wir die Nutzung des Parameters `--cli-input-json` zur Übergabe einer JSON-Datei, die die erforderlichen Parameter enthält. Wenn Sie einen OIDC-Anbieter erstellen, müssen Sie die URL des Anbieters übergeben und die URL muss mit `https://` beginnen. Es kann schwierig sein, die URL als Befehlszeilenparameter zu übergeben, da der Doppelpunkt (:) und der Schrägstrich (/) in einigen Befehlszeilenumgebungen eine besondere Bedeutung haben. Durch die Verwendung des Parameters `--cli-input-json` wird diese Einschränkung umgangen.

Um den `--cli-input-json`-Parameter zu verwenden, verwenden Sie zunächst den `create-open-id-connect-provider`-Befehl mit dem `--generate-cli-skeleton`-Parameter, wie im folgenden Beispiel.

```
aws iam create-open-id-connect-provider \
    --generate-cli-skeleton > create-open-id-connect-provider.json
```

Der vorherige Befehl erstellt eine JSON-Datei mit dem Namen create-open-id-connect-provider.json, die Sie zum Ausfüllen der Informationen für einen nachfolgenden Befehl `create-open-id-connect-provider` verwenden können. Beispiel:

```
{
    "Url": "https://server.example.com",
    "ClientIDList": [
        "example-application-ID"
    ],
    "ThumbprintList": [
        "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"
    ]
```

```
}
```

Wenn die Discovery- und JWKS-Endpunkte Ihres OIDC-Anbieters unterschiedliche Zertifikate verwenden, fügen Sie Fingerabdrücke für beide Endpunkte in das. `ThumbprintList`

Verwenden Sie als Nächstes den Befehl `create-open-id-connect-provider` erneut, um den OpenID Connect (OIDC)-Anbieter zu erstellen, und übergeben Sie diesmal den Parameter `--cli-input-json`, um Ihre JSON-Datei anzugeben. Der folgende Befehl `create-open-id-connect-provider` verwendet den Parameter `--cli-input-json` mit einer JSON-Datei mit dem Namen create-open-id-connect-provider.json.

```
aws iam create-open-id-connect-provider \
    --cli-input-json file://create-open-id-connect-provider.json
```

Ausgabe:

```
{
    "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

Weitere Informationen zu OIDC-Anbietern finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

Weitere Informationen zum Abrufen von Fingerabdrücken für einen OIDC-Anbieter finden Sie unter Abrufen des Fingerabdrucks für einen OpenID Connect-Identitätsanbieter im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. CreateOpenIdConnectProviderAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein IAM-OIDC-Anbieter erstellt, der dem OIDC-kompatiblen Anbieter-Service zugeordnet ist, der unter der URL **`https://example.oidcprovider.com`** und der Client-ID **`my-testapp-1`** zu finden ist. Der OIDC-Anbieter stellt den Fingerabdruck bereit. Gehen Sie wie folgt vor, um den Fingerabdruck zu authentifizieren http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Ausgabe:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter CreateOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein IAM-OIDC-Anbieter erstellt, der dem OIDC-kompatiblen Anbieter-Service zugeordnet ist, der unter der URL **`https://example.oidcprovider.com`** und der Client-ID **`my-testapp-1`** zu finden ist. Der OIDC-Anbieter stellt den Fingerabdruck bereit. Gehen Sie wie folgt vor, um den Fingerabdruck zu authentifizieren http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Ausgabe:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter CreateOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreatePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erste Schritte mit verwaltetem Streaming

- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten von Richtlinien
- Richten Sie die Zugriffskontrolle ein Attribute-Based
- Einrichtung der Systemverwaltung
- Arbeiten mit der IAM-Policy-Builder-API

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
  --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-policy operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }

    return result;
```

```
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "       ],"
                 << "       \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";

    return stringStream.str();
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine vom Kunden verwaltete Richtlinie

Mit dem folgenden Befehl wird eine vom Kunden verwaltete Richtlinie mit dem Namen `my-policy` erstellt. Bei der Datei `policy.json` handelt es sich um ein JSON-Dokument im aktuellen `shared`-Ordner, das schreibgeschützten Zugriff auf den Ordner in einem Amazon-S3-Bucket mit dem Namen `amzn-s3-demo-bucket` gewährt.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json
```

Inhalt von policy.json:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/shared/*"
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "CreateDate": "2015-06-01T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
        "UpdateDate": "2015-06-01T19:31:18.620Z"
    }
```

```
}
```

Weitere Informationen zur Verwendung von Dateien als Eingabe für Zeichenkettenparameter finden Sie unter Angeben von Parameterwerten für die AWS CLI im AWS CLI-Benutzerhandbuch.

Beispiel 2: So erstellen Sie eine vom Kunden verwaltete Richtlinie mit einer Beschreibung

Der folgende Befehl erstellt eine vom Kunden verwaltete Richtlinie mit dem Namen `my-policy` und einer unveränderlichen Beschreibung.

Bei der `policy.json`-Datei handelt es sich um ein JSON-Dokument im aktuellen Ordner, das Zugriff auf alle Put-, List- und Get-Aktionen für einen Amazon-S3-Bucket mit dem Namen `amzn-s3-demo-bucket` gewährt.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --description "This policy grants access to all Put, Get, and List actions
 for amzn-s3-demo-bucket"
```

Inhalt von policy.json:

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T22:38:47+00:00",
        "UpdateDate": "2023-05-24T22:38:47+00:00"
    }
}
```

Weitere Informationen zu Idenity-based Richtlinien finden Sie unter Identity-based Richtlinien und ressourcenbasierte Richtlinien im AWS IAM-Benutzerhandbuch.

Beispiel 3: So erstellen Sie eine vom Kunden verwaltete Richtlinie mit Tags

Mit dem folgenden Befehl wird eine vom Kunden verwaltete Richtlinie mit dem Namen `my-policy` mit Tags erstellt. In diesem Beispiel wird der `--tags` Parameter mit den folgenden JSON-formatted Tags verwendet:. `'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'` Alternativ kann der Parameter `--tags` mit Tags im Kurzformat verwendet werden: `'Key=Department,Value=Accounting Key=Location,Value=Seattle'`.

Bei der `policy.json`-Datei handelt es sich um ein JSON-Dokument im aktuellen Ordner, das Zugriff auf alle Put-, List- und Get-Aktionen für einen Amazon-S3-Bucket mit dem Namen `amzn-s3-demo-bucket` gewährt.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Inhalt von policy.json:

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

Ausgabe:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::12345678012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T23:16:39+00:00",
        "UpdateDate": "2023-05-24T23:16:39+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
                "Key": "Location",
                "Value": "Seattle"
            {
        ]
    }
}
```

Weitere Informationen zu Tagging-Richtlinien finden Sie unter Tagging von vom Kunden verwalteten Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreatePolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
```

```
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
```

```
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: "
 + result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);
```

```
            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie die Richtlinie.

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyName
```

```
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
```

```
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "dynamodb:DeleteItem",
                    "dynamodb:GetItem",
                    "dynamodb:PutItem",
                    "dynamodb:Scan",
                    "dynamodb:UpdateItem"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName,
 $policyDocument) {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird im aktuellen AWS Konto eine neue IAM-Richtlinie mit dem Namen **MySamplePolicy** Die Datei **MySamplePolicy.json** enthält den Richtlinieninhalt erstellt. Beachten Sie, dass Sie den Switch-Parameter **-Raw** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Einzelheiten zur API finden Sie unter CreatePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird im aktuellen AWS Konto eine neue IAM-Richtlinie mit dem Namen **MySamplePolicy** Die Datei **MySamplePolicy.json** enthält den Richtlinieninhalt erstellt. Beachten Sie, dass Sie den Switch-Parameter **-Raw** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- Einzelheiten zur API finden Sie unter CreatePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
```

```
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Einzelheiten zur API finden Sie CreatePolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
```

```
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createpolicy(
          iv_policyname = iv_policy_name
```

```
          iv_policydocument = iv_policy_document
          iv_description = iv_description ).
        MESSAGE 'Policy created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Policy already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreatePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
```

```
        } catch {
            print("ERROR: createPolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreatePolicyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreatePolicyVersion`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePolicyVersion` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Richtlinien

CLI

AWS CLI

So erstellen Sie eine neue Version einer verwalteten Richtlinie

In diesem Beispiel wird eine neue `v2`-Version der IAM-Richtlinie erstellt, deren ARN `arn:aws:iam::123456789012:policy/MyPolicy` lautet, und sie zur Standardversion gemacht.

```
aws iam create-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --policy-document file://NewPolicyVersion.json \
    --set-as-default
```

Ausgabe:

```
{
    "PolicyVersion": {
        "CreateDate": "2015-06-16T18:56:03.721Z",
        "VersionId": "v2",
        "IsDefaultVersion": true
```

```
    }
}
```

Weitere Informationen finden Sie unter Versionsverwaltung von IAM-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreatePolicyVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue „v2“-Version der IAM-Richtlinie erstellt, deren ARN **`arn:aws:iam::123456789012:policy/MyPolicy`** lautet, und als standardmäßige Version festgelegt. Die Datei **`NewPolicyVersion.json`** enthält den Richtlinieninhalt. Beachten Sie, dass Sie den Switch-Parameter **`-Raw`** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Ausgabe:

```
CreateDate                          Document                  IsDefaultVersion
           VersionId
----------                          --------                  ----------------
           ---------
4/15/2015 10:54:54 AM                                         True
           v2
```

- Einzelheiten zur API finden Sie unter CreatePolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue „v2“-Version der IAM-Richtlinie erstellt, deren ARN **`arn:aws:iam::123456789012:policy/MyPolicy`** lautet, und als standardmäßige Version festgelegt. Die Datei **`NewPolicyVersion.json`** enthält den Richtlinieninhalt. Beachten Sie, dass Sie den Switch-Parameter **`-Raw`** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Ausgabe:

```
CreateDate                          Document                 IsDefaultVersion
           VersionId
----------                          --------                 ----------------
           ---------
4/15/2015 10:54:54 AM                                        True
           v2
```

- Einzelheiten zur API finden Sie unter CreatePolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
```

```
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- Einzelheiten zur API finden Sie CreatePolicyVersionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_policydocument = iv_policy_document
          iv_setasdefault = iv_set_as_default ).
        MESSAGE 'Policy version created successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy version limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreatePolicyVersionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateRole`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateRole` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Einen Containerdienst für VM-Instanzen erstellen
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Erste Schritte mit der Workflow-Orchestrierung

- Verwalten Sie Rollen
- Übertragung von hartcodierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Einrichtung der Systemverwaltung
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Arbeiten Sie mit Streams und Time-to-Live

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie CreateRolein der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
```

```
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine IAM-Rolle

Mit dem folgenden `create-role`-Befehl wird eine Rolle mit dem Namen `Test-Role` erstellt und ihr eine Vertrauensrichtlinie angefügt.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

Ausgabe:

```
{
    "Role": {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "CreateDate": "2013-06-07T20:43:32.821Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

Die Vertrauensrichtlinie ist in der Test-Role-Trust-Policy.jsonDatei als JSON-Dokument definiert. (Der Dateiname und die Erweiterung sind nicht von Bedeutung.) Die Vertrauensrichtlinie muss einen Prinzipal angeben.

Verwenden Sie den `put-role-policy`-Befehl, um die Berechtigungsrichtlinie der Rolle anzufügen.

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

Beispiel 2: So erstellen Sie eine IAM-Rolle mit angegebener maximaler Sitzungsdauer

Mit dem folgenden `create-role`-Befehl wird eine Rolle mit dem Namen `Test-Role` erstellt und eine maximale Sitzungsdauer von 7 200 Sekunden (2 Stunden) festgelegt.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --max-session-duration 7200
```

Ausgabe:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:role/Test-Role",
        "CreateDate": "2023-05-24T23:50:25+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::12345678012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Ändern der maximalen Sitzungsdauer (AWS API) einer Rolle.

Beispiel 3: So erstellen Sie eine IAM-Rolle mit Tags

Mit dem folgenden Befehl wird eine IAM-Rolle `Test-Role` mit Tags erstellt. In diesem Beispiel wird der `--tags` Parameter flag mit den folgenden JSON-formatted Tags verwendet:`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`. Alternativ kann das `--tags`-Flag mit Tags im Kurzformat `'Key=Department,Value=Accounting Key=Location,Value=Seattle'` verwendet werden.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Ausgabe:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role",
        "CreateDate": "2023-05-25T23:29:41+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
```

```
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Taggen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateRolein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
```

```
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-
roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
```

```
                    fileLocation - The location of the JSON document that
represents the trust policy.\s
               """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }

       String rolename = args[0];
       String fileLocation = args[1];
       Region region = Region.AWS_GLOBAL;
       IamClient iam = IamClient.builder()
               .region(region)
               .build();

       String result = createIAMRole(iam, rolename, fileLocation);
       System.out.println("Successfully created user: " + result);
       iam.close();
   }

   public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
       try {
           JSONObject jsonObject = (JSONObject)
readJsonSimpleDemo(fileLocation);
           CreateRoleRequest request = CreateRoleRequest.builder()
                   .roleName(rolename)
                   .assumeRolePolicyDocument(jsonObject.toJSONString())
                   .description("Created using the AWS SDK for Java")
                   .build();

           CreateRoleResponse response = iam.createRole(request);
           System.out.println("The ARN of the role is " +
response.role().arn());

       } catch (IamException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }

   public static Object readJsonSimpleDemo(String filename) throws Exception {
```

```
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie die Rolle.

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
```

```
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
```

```
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue Rolle mit dem Namen **`MyNewRole`** erstellt und die in der Datei **`NewRoleTrustPolicy.json`** gefundene Richtlinie daran angefügt. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten. Das in der Ausgabe angezeigte Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der **`UrlDecode`** .NET-Methode.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Ausgabe:

```
Arn                      : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
```

```
                              %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                              %20%5D%0D%0A%7D
CreateDate                  : 4/15/2015 11:04:23 AM
Path                        : /
RoleId                      : V5PAJI2KPN4EAEXAMPLE1
RoleName                    : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter CreateRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Rolle mit dem Namen **`MyNewRole`** erstellt und die in der Datei **`NewRoleTrustPolicy.json`** gefundene Richtlinie daran angefügt. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten. Das in der Ausgabe angezeigte Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der **`UrlDecode`** .NET-Methode.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Ausgabe:

```
Arn                         : arn:aws:iam::123456789012:role/MyNewRole
```

```
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate               : 4/15/2015 11:04:23 AM
Path                     : /
RoleId                   : V5PAJI2KPN4EAEXAMPLE1
RoleName                 : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter CreateRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- Einzelheiten zur API finden Sie CreateRolein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn

   policy_arns.each do |policy_arn|
     @iam_client.attach_role_policy(
       role_name: role_name,
       policy_arn: policy_arn
     )
   end

   role_arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating role: #{e.message}")
   nil
 end
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- Einzelheiten zur API finden Sie CreateRolein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createrole(
          iv_rolename = iv_role_name
          iv_assumerolepolicydocument = iv_assume_role_policy_document ).
        MESSAGE 'Role created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Role already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Assume role policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Role limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateRolein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3
```

```
    public func createRole(name: String, policyDocument: String) async throws ->
String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
            }
            return id
        } catch {
            print("ERROR: createRole:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateRolein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateSAMLProvider`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSAMLProvider` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen SAML-Anbieter

In diesem Beispiel wird in IAM ein neuer SAML-Anbieter mit dem Namen `MySAMLProvider` erstellt. Es wird durch das SAML-Metadatendokument beschrieben, das sich in der Datei `SAMLMetaData.xml` befindet.

```
aws iam create-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
```

```
    --name MySAMLProvider
```

Ausgabe:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-SAML-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Weitere API-Informationen finden Sie unter CreateSAMLProvider in der AWS CLI -Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import * as path from "node:path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
```

```
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter CreateSAMLProvider in der API-Referenz für AWS SDK für JavaScript .

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue SAML-Anbieter-Entität in IAM erstellt. Sie ist **`MySAMLProvider`** benannt und wird durch das SAML-Metadatendokument in der Datei **`SAMLMetaData.xml`** beschrieben, die separat von der Website des SAML-Service-Anbieters heruntergeladen wurde.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Ausgabe:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- API-Details finden Sie unter CreateSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue SAML-Anbieter-Entität in IAM erstellt. Sie ist **`MySAMLProvider`** benannt und wird durch das SAML-Metadatendokument in der Datei **`SAMLMetaData.xml`** beschrieben, die separat von der Website des SAML-Service-Anbieters heruntergeladen wurde.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Ausgabe:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Weitere API-Informationen finden Sie unter CreateSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Verwenden Sie **`CreateServiceLinkedRole`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateServiceLinkedRole` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
```

```
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine serviceverknüpfte Rolle

Im folgenden `create-service-linked-role` Beispiel wird eine dienstbezogene Rolle für den angegebenen AWS Dienst erstellt und die angegebene Beschreibung angehängt.

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazonaws.com \
    --description "My service-linked role to support Lex"
```

Ausgabe:

```
{
    "Role": {
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "RoleName": "AWSServiceRoleForLexBots",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
        "CreateDate": "2019-04-17T20:34:14+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
```

```
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "lex.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
}
```

Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter CreateServiceLinkedRole AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer serviceverknüpften Rolle.

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
```

```
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
 this account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
 $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
```

```
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine serviceverknüpfte Rolle für den Autoscaling-Service erstellt.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- Einzelheiten zur API finden Sie unter CreateServiceLinkedRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine serviceverknüpfte Rolle für den Autoscaling-Service erstellt.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- Einzelheiten zur API finden Sie unter CreateServiceLinkedRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
    else:
        return role
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates a service-linked role
 #
 # @param service_name [String] The service name to create the role for.
 # @param description [String] The description of the service-linked role.
 # @param suffix [String] Suffix for customizing role name.
 # @return [String] The name of the created role
 def create_service_linked_role(service_name, description, suffix)
   response = @iam_client.create_service_linked_role(
     aws_service_name: service_name, description: description, custom_suffix:
suffix
   )
   role_name = response.role.role_name
   @logger.info("Created service-linked role #{role_name}.")
   role_name
 rescue Aws::Errors::ServiceError => e
   @logger.error("Couldn't create service-linked role for #{service_name}.
Here's why:")
   @logger.error("\t#{e.code}: #{e.message}")
   raise
 end
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
 SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createServiceLinkedRole(service: String, suffix: String? = nil,
 description: String?)
                       async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
```

```
            return role
        } catch {
            print("ERROR: createServiceLinkedRole:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateServiceLinkedRolein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateUser`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateUser` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
```

```
        return response.User;
    }
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
```

```
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
```

```
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie CreateUserin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);

    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen IAM-Benutzer

Mit dem folgenden `create-user`-Befehl wird im aktuellen Konto ein IAM-Benutzer mit dem Namen Bob erstellt.

```
aws iam create-user \
    --user-name Bob
```

Ausgabe:

```
{
    "User": {
        "UserName": "Bob",
        "Path": "/",
        "CreateDate": "2023-06-08T03:20:41.270Z",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Bob"
    }
```

```
}
```

Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Einen IAM-Benutzer in Ihrem AWS Konto erstellen.AWS

Beispiel 2: So erstellen Sie einen IAM-Benutzer unter einem angegebenen Pfad

Mit dem folgenden `create-user`-Befehl wird im angegebenen Pfad ein IAM-Benutzer mit dem Namen Bob erstellt.

```
aws iam create-user \
    --user-name Bob \
    --path /division_abc/subdivision_xyz/
```

Ausgabe:

```
{
    "User": {
        "Path": "/division_abc/subdivision_xyz/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
        "CreateDate": "2023-05-24T18:20:17+00:00"
    }
}
```

Weitere Informationen finden Sie unter IAM-Kennungen im AWS -IAM-Benutzerhandbuch.

Beispiel 3: So erstellen Sie einen IAM-Benutzer mit Tags

Mit dem folgenden `create-user`-Befehl wird ein IAM-Benutzer mit dem Namen Bob mit Tags erstellt. In diesem Beispiel wird der `--tags` Parameter flag mit den folgenden JSON-formatted Tags verwendet:. `'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'` Alternativ kann das `--tags`-Flag mit Tags im Kurzformat `'Key=Department,Value=Accounting Key=Location,Value=Seattle'` verwendet werden.

```
aws iam create-user \
    --user-name Bob \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

Ausgabe:

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-25T17:14:21+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Tagging von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

Beispiel 3: So erstellen Sie einen IAM-Benutzer mit einer festgelegten Berechtigungsgrenze

Mit dem folgenden `create-user` Befehl wird ein IAM-Benutzer Bob mit der Berechtigungsgrenze von AmazonS3FullAccess erstellt.

```
aws iam create-user \
    --user-name Bob \
    --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

Ausgabe:

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-24T17:50:53+00:00",
```

```
        "PermissionsBoundary": {
        "PermissionsBoundaryType": "Policy",
        "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    }
}
```

Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateUserin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);
```

```
            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);

 waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Benutzer.

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
```

```
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
```

```
      data.User.UserId
    );
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- Einzelheiten zur API finden Sie CreateUserin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein IAM-Benutzer mit dem Namen **`Bob`** erstellt. Wenn Bob sich an der AWS Konsole anmelden muss, müssen Sie den Befehl separat ausführen, **`New-IAMLoginProfile`** um ein Anmeldeprofil mit einem Passwort zu erstellen. Wenn Bob

plattformübergreifende CLI-Befehle ausführen AWS PowerShell oder AWS API-Aufrufe tätigen muss, müssen Sie den **New-IAMAccessKey** Befehl separat ausführen, um Zugriffsschlüssel zu erstellen.

```
New-IAMUser -UserName Bob
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Einzelheiten zur API finden Sie unter CreateUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein IAM-Benutzer mit dem Namen **Bob** erstellt. Wenn Bob sich an der AWS Konsole anmelden muss, müssen Sie den Befehl separat ausführen, **New-IAMLoginProfile** um ein Anmeldeprofil mit einem Passwort zu erstellen. Wenn Bob plattformübergreifende CLI-Befehle ausführen AWS PowerShell oder AWS API-Aufrufe tätigen muss, müssen Sie den **New-IAMAccessKey** Befehl separat ausführen, um Zugriffsschlüssel zu erstellen.

```
New-IAMUser -UserName Bob
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- Einzelheiten zur API finden Sie unter CreateUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- Einzelheiten zur API finden Sie CreateUserin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates a user and their login profile
  #
  # @param user_name [String] The name of the user
  # @param initial_password [String] The initial password for the user
  # @return [String, nil] The ID of the user if created, or nil if an error
 occurred
  def create_user(user_name, initial_password)
    response = @iam_client.create_user(user_name: user_name)
    @iam_client.wait_until(:user_exists, user_name: user_name)
    @iam_client.create_login_profile(
      user_name: user_name,
      password: initial_password,
      password_reset_required: true
    )
    @logger.info("User '#{user_name}' created successfully.")
    response.user.user_id
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;
```

```
    Ok(response.user.unwrap())
}
```

- Einzelheiten zur API finden Sie CreateUserin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createuser(
          iv_username = iv_user_name ).
        MESSAGE 'User created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'User already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded for IAM users.' TYPE 'E'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Entity does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateUserin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createUser(name: String) async throws -> String {
        let input = CreateUserInput(
            userName: name
        )
        do {
            let output = try await client.createUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            guard let id = user.userId else {
                throw ServiceHandlerError.noSuchUser
            }
            return id
        } catch {
            print("ERROR: createUser:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateUserin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`CreateVirtualMfaDevice`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVirtualMfaDevice` verwendet wird.

CLI

AWS CLI

So erstellen Sie ein virtuelles MFA-Gerät

In diesem Beispiel wird ein neues virtuelles MFA-Gerät mit dem Namen `BobsMFADevice` erstellt. Es erstellt eine Datei mit dem Namen `QRCode.png`, die Bootstrap-Informationen enthält, und platziert sie im Verzeichnis `C:/`. Die in diesem Beispiel verwendete Bootstrap-Methode lautet `QRCodePNG`.

```
aws iam create-virtual-mfa-device \
    --virtual-mfa-device-name BobsMFADevice \
    --outfile C:/QRCode.png \
    --bootstrap-method QRCodePNG
```

Ausgabe:

```
{
    "VirtualMFADevice": {
        "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
}
```

Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateVirtualMfaDeviceein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **`Base32StringSeed`**, den das virtuelle MFA-Softwareprogramm zum Erstellen eines Kontos benötigt (als Alternative zum QR-Code), extrahiert. Nachdem Sie das Programm mit dem Wert konfiguriert haben, erhalten Sie vom Programm zwei aufeinanderfolgende Authentifizierungs-Codes. Verwenden Sie abschließend den letzten Befehl, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **`Bob`** zu verknüpfen und das Konto mit den beiden Authentifizierungscodes zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
```

```
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Ausgabe:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Beispiel 2: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **`QRCodePNG`** extrahiert und in eine Datei geschrieben. Dieses Bild kann vom virtuellen MFA-Softwareprogramm gescannt werden, um ein Konto zu erstellen (als Alternative zur manuellen Eingabe des Base32StringSeed Werts). Nachdem Sie das Konto in Ihrem virtuellen MFA-Programm erstellt haben, erhalten Sie zwei aufeinanderfolgende Authentifizierungscodes. Geben Sie diese in die letzten Befehle ein, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **`Bob`** zu verknüpfen und das Konto zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

Ausgabe:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Einzelheiten zur API finden Sie unter CreateVirtualMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **`Base32StringSeed`**, den das virtuelle MFA-Softwareprogramm zum Erstellen

eines Kontos benötigt (als Alternative zum QR-Code), extrahiert. Nachdem Sie das Programm mit dem Wert konfiguriert haben, erhalten Sie vom Programm zwei aufeinanderfolgende Authentifizierungs-Codes. Verwenden Sie abschließend den letzten Befehl, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **Bob** zu verknüpfen und das Konto mit den beiden Authentifizierungscodes zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Ausgabe:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Beispiel 2: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **QRCodePNG** extrahiert und in eine Datei geschrieben. Dieses Bild kann vom virtuellen MFA-Softwareprogramm gescannt werden, um ein Konto zu erstellen (als Alternative zur manuellen Eingabe des Base32StringSeed Werts). Nachdem Sie das Konto in Ihrem virtuellen MFA-Programm erstellt haben, erhalten Sie zwei aufeinanderfolgende Authentifizierungscodes. Geben Sie diese in die letzten Befehle ein, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **Bob** zu verknüpfen und das Konto zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

Ausgabe:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- Einzelheiten zur API finden Sie unter CreateVirtualMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeactivateMfaDevice`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeactivateMfaDevice` verwendet wird.

CLI

AWS CLI

So deaktivieren Sie ein MFA-Gerät

Dieser Befehl deaktiviert das virtuelle MFA-Gerät mit der ARN `arn:aws:iam::210987654321:mfa/BobsMFADevice`, die dem Benutzer Bob zugeordnet ist.

```
aws iam deactivate-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeactivateMfaDevicein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Mit diesem Befehl wird das Hardware-MFA-Gerät deaktiviert, das dem Benutzer **`Bob`** mit der Seriennummer **`123456789012`** zugeordnet ist.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Beispiel 2: Mit diesem Befehl wird das virtuelle MFA-Gerät deaktiviert, das dem Benutzer **`David`** mit der ARN **`arn:aws:iam::210987654321:mfa/David`** zugeordnet ist. Beachten

Sie, dass das virtuelle MFA-Gerät nicht aus dem Konto gelöscht wird. Das virtuelle Gerät ist noch vorhanden und wird in der Ausgabe des Befehls **`Get-IAMVirtualMFADevice`** angezeigt. Bevor Sie ein neues virtuelles MFA-Gerät für denselben Benutzer erstellen können, müssen Sie das alte mit dem Befehl **`Remove-IAMVirtualMFADevice`** löschen.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- Einzelheiten zur API finden Sie unter DeactivateMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl wird das Hardware-MFA-Gerät deaktiviert, das dem Benutzer **`Bob`** mit der Seriennummer **`123456789012`** zugeordnet ist.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Beispiel 2: Mit diesem Befehl wird das virtuelle MFA-Gerät deaktiviert, das dem Benutzer **`David`** mit der ARN **`arn:aws:iam::210987654321:mfa/David`** zugeordnet ist. Beachten Sie, dass das virtuelle MFA-Gerät nicht aus dem Konto gelöscht wird. Das virtuelle Gerät ist noch vorhanden und wird in der Ausgabe des Befehls **`Get-IAMVirtualMFADevice`** angezeigt. Bevor Sie ein neues virtuelles MFA-Gerät für denselben Benutzer erstellen können, müssen Sie das alte mit dem Befehl **`Remove-IAMVirtualMFADevice`** löschen.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- Einzelheiten zur API finden Sie unter DeactivateMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteAccessKey`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAccessKey` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern
- Verwalten von Zugriffsschlüsseln

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
```

```
  echo "  -k access_key   The access key to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
```

```
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie einen Zugriffsschlüssel für einen IAM-Benutzer

Mit dem folgenden `delete-access-key`-Befehl wird der angegebene Zugriffsschlüssel (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel) für den IAM-Benutzer mit dem Namen Bob gelöscht.

```
aws iam delete-access-key \
    --access-key-id AKIDPMS9RO4H3FEXAMPLE \
    --user-name Bob
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `list-access-keys`-Befehl, um die für einen IAM-Benutzer definierten Zugriffsschlüssel aufzulisten.

Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
```

```
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you
 want to delete.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String
 accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Zugriffsschlüssel.

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
```

```
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird das AWS Zugriffsschlüsselpaar mit der Schlüssel-ID des **`AKIAIOSFODNN7EXAMPLE`** angegebenen Benutzers gelöscht. **`Bob`**

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Einzelheiten zur API finden Sie unter DeleteAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das AWS Zugriffsschlüsselpaar mit der Schlüssel-ID des **`AKIAIOSFODNN7EXAMPLE`** angegebenen Benutzers gelöscht. **`Bob`**

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- Einzelheiten zur API finden Sie unter DeleteAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
 more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
```

```
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deleteaccesskey(
          iv_accesskeyid = iv_access_key_id
          iv_username = iv_user_name ).
        MESSAGE 'Access key deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Access key or user does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws
```

```
    {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
            print("ERROR: deleteAccessKey:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteAccountAlias`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAccountAlias` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie einen Konto-Alias

Mit dem folgenden `delete-account-alias`-Befehl wird der Alias `mycompany` für das aktuelle Konto entfernt.

```
aws iam delete-account-alias \
    --account-alias mycompany
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ihre AWS Konto-ID und deren Alias im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DeleteAccountAlias AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request =
 DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Konto-Alias.

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
```

```
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird der Account-Alias aus Ihrem entfernt AWS-Konto. Die Benutzeranmeldeseite mit dem Alias at funktioniert nicht https://mycompanyaws.signin.aws.amazon.com/console mehr. Sie müssen stattdessen die ursprüngliche URL mit Ihrer AWS-Konto ID-Nummer unter https://<accountidnumber>.signin.aws.amazon verwenden. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter DeleteAccountAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Account-Alias aus Ihrem AWS-Konto entfernt. Die Benutzeranmeldeseite mit dem Alias at funktioniert nicht https://mycompanyaws.signin.aws.amazon.com/console mehr. Sie müssen stattdessen die ursprüngliche URL mit Ihrer AWS-Konto ID-Nummer unter https://<accountidnumber>.signin.aws.amazon verwenden. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- Einzelheiten zur API finden Sie unter DeleteAccountAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deleteaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Account alias does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteAccountPasswordPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAccountPasswordPolicy` verwendet wird.

CLI

AWS CLI

So löschen Sie die Passwortrichtlinie für das aktuelle Konto

Mit dem folgenden `delete-account-password-policy`-Befehl wird die Passwortrichtlinie für das aktuelle Konto entfernt.

```
aws iam delete-account-password-policy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Festlegen einer Kontopasswortrichtlinie für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAccountPasswordPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Kennwortrichtlinie für gelöscht AWS-Konto und alle Werte auf ihre ursprünglichen Standardwerte zurückgesetzt. Wenn derzeit keine Kennwortrichtlinie existiert, wird die folgende Fehlermeldung angezeigt: Die Kontorichtlinie mit dem Namen PasswordPolicy kann nicht gefunden werden.

```
Remove-IAMAccountPasswordPolicy
```

- Einzelheiten zur API finden Sie unter DeleteAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Kennwortrichtlinie für gelöscht AWS-Konto und alle Werte auf ihre ursprünglichen Standardwerte zurückgesetzt. Wenn derzeit keine Kennwortrichtlinie existiert, wird die folgende Fehlermeldung angezeigt: Die Kontorichtlinie mit dem Namen PasswordPolicy kann nicht gefunden werden.

```
Remove-IAMAccountPasswordPolicy
```

- Einzelheiten zur API finden Sie unter DeleteAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteGroup`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteGroup` verwendet wird.

CLI

AWS CLI

So löschen Sie eine IAM-Gruppe

Mit dem folgenden `delete-group`-Befehl wird eine IAM-Gruppe mit dem Namen `MyTestGroup` gelöscht.

```
aws iam delete-group \
    --group-name MyTestGroup
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen einer IAM-Benutzergruppe im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteGroupin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
```

```
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie DeleteGroupin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe mit dem Namen **`MyTestGroup`** gelöscht. Mit dem ersten Befehl werden alle IAM-Benutzer entfernt, die Mitglieder der Gruppe sind, und mit dem zweiten Befehl wird die IAM-Gruppe gelöscht. Beide Befehle funktionieren ohne weitere Bestätigungsaufforderung.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe mit dem Namen **`MyTestGroup`** gelöscht. Mit dem ersten Befehl werden alle IAM-Benutzer entfernt, die Mitglieder der Gruppe sind, und mit dem zweiten Befehl wird die IAM-Gruppe gelöscht. Beide Befehle funktionieren ohne weitere Bestätigungsaufforderung.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteGroupPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteGroupPolicy` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Richtlinie aus einer IAM-Gruppe

Mit dem folgenden `delete-group-policy`-Befehl wird die Richtlinie mit dem Namen `ExamplePolicy` aus der Gruppe mit dem Namen `Admins` gelöscht.

```
aws iam delete-group-policy \
    --group-name Admins \
    --policy-name ExamplePolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `list-group-policies`-Befehl, um die einer Gruppe zugeordneten Richtlinien anzuzeigen.

Weitere Informationen finden Sie unter Verwalten von IAM-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteGroupPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`TesterPolicy`** aus der IAM-Gruppe **`Testers`** entfernt. Die Benutzer in dieser Gruppe verlieren sofort die in dieser Richtlinie definierten Berechtigungen.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Einzelheiten zur API finden Sie unter DeleteGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`TesterPolicy`** aus der IAM-Gruppe **`Testers`** entfernt. Die Benutzer in dieser Gruppe verlieren sofort die in dieser Richtlinie definierten Berechtigungen.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- Einzelheiten zur API finden Sie unter DeleteGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteInstanceProfile`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteInstanceProfile` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verwalten eines ausfallsicheren Services
- Erste Schritte mit verwaltetem Streaming
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
```

```
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
```

```
        }
    }
```

- Einzelheiten zur API finden Sie DeleteInstanceProfilein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie ein Instance-Profil

Mit dem folgenden `delete-instance-profile`-Befehl wird das Instance-Profil mit dem Namen `ExampleInstanceProfile` gelöscht.

```
aws iam delete-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteInstanceProfilein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
```

```
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
```

- Einzelheiten zur API finden Sie DeleteInstanceProfilein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das EC2-Instance-Profil mit dem Namen **`MyAppInstanceProfile`** gelöscht. Mit dem ersten Befehl werden alle Rollen vom Instance-Profil getrennt, und mit dem zweiten Befehl wird das Instance-Profil gelöscht.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Einzelheiten zur API finden Sie unter DeleteInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das EC2-Instance-Profil mit dem Namen **`MyAppInstanceProfile`** gelöscht. Mit dem ersten Befehl werden alle Rollen vom Instance-Profil getrennt, und mit dem zweiten Befehl wird das Instance-Profil gelöscht.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- Einzelheiten zur API finden Sie unter DeleteInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird die Rolle aus dem Instance-Profil entfernt, alle mit der Rolle verknüpften Richtlinien getrennt und alle Ressourcen gelöscht.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
```

```
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
```

```
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
```

- Einzelheiten zur API finden Sie DeleteInstanceProfilein AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DeleteLoginProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLoginProfile` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Passwort für einen IAM-Benutzer

Der folgende Befehl `delete-login-profile` löscht das Passwort für den IAM-Benutzer mit dem Namen Bob.

```
aws iam delete-login-profile \
    --user-name Bob
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwalten von Passwörtern für IAM-Benutzer im AWS - IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteLoginProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Anmeldeprofil des IAM-Benutzers **Bob** gelöscht. Dadurch wird verhindert, dass sich der Benutzer an der AWS Konsole anmeldet. Es verhindert nicht, dass der Benutzer AWS CLI- PowerShell oder API-Aufrufe mit AWS Zugriffsschlüsseln ausführt, die möglicherweise noch mit dem Benutzerkonto verknüpft sind.

```
Remove-IAMLoginProfile -UserName Bob
```

- Einzelheiten zur API finden Sie unter DeleteLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Anmeldeprofil des IAM-Benutzers **Bob** gelöscht. Dadurch wird verhindert, dass sich der Benutzer an der Konsole anmeldet. AWS Es verhindert nicht, dass der Benutzer AWS CLI- PowerShell oder API-Aufrufe mit AWS Zugriffsschlüsseln ausführt, die möglicherweise noch mit dem Benutzerkonto verknüpft sind.

```
Remove-IAMLoginProfile -UserName Bob
```

- Einzelheiten zur API finden Sie unter DeleteLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteOpenIdConnectProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteOpenIdConnectProvider` verwendet wird.

CLI

AWS CLI

So löschen Sie einen IAM-OpenID-Connect-Identitätsanbieter

In diesem Beispiel wird der IAM-OIDC-Anbieter gelöscht, der eine Verbindung zum Anbieter `example.oidcprovider.com` herstellt.

```
aws iam delete-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteOpenIdConnectProviderin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der IAM-OIDC-Anbieter gelöscht, der eine Verbindung zum Anbieter **`example.oidcprovider.com`** herstellt. Stellen Sie sicher, dass Sie alle Rollen aktualisieren oder löschen, die im **`Principal`**-Element der Vertrauensrichtlinie der Rolle auf diesen Anbieter verweisen.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter DeleteOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-OIDC-Anbieter gelöscht, der eine Verbindung zum Anbieter **`example.oidcprovider.com`** herstellt. Stellen Sie sicher, dass Sie alle Rollen aktualisieren oder löschen, die im **`Principal`**-Element der Vertrauensrichtlinie der Rolle auf diesen Anbieter verweisen.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter DeleteOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeletePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten von Richtlinien
- Einrichtung der Systemverwaltung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
```

```
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine IAM-Richtlinie

In diesem Beispiel wird die Richtlinie, deren ARN `arn:aws:iam::123456789012:policy/MySamplePolicy` lautet, gelöscht.

```
aws iam delete-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeletePolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Richtlinie.

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die Richtlinie gelöscht, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet. Bevor Sie die Richtlinie löschen können, müssen Sie zunächst alle Versionen außer der Standardversion löschen, indem Sie **`Remove-IAMPolicyVersion`** ausführen. Sie müssen die Richtlinie auch von allen IAM-Benutzern, -Gruppen oder -Rollen trennen.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen gelöscht werden, die Richtlinie dann von allen zugeordneten IAM-Entitäten getrennt wird und schließlich die Richtlinie selbst gelöscht wird. In der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßige Version gekennzeichnet sind, in einer Sammlung erfasst und dann wird jede Richtlinie in der Sammlung gelöscht. In der dritten Zeile werden alle Benutzer, Gruppen und Rollen des IAM abgerufen, denen die Richtlinie angefügt ist. Die Zeilen vier bis sechs trennen die Richtlinie von jeder angefügten Entität. In der letzten Zeile wird dieser Befehl zum Entfernen der verwalteten Richtlinie sowie der verbleibenden Standardversion verwendet. Das Beispiel schließt den **`-Force`**-Switch-Parameter in jede Zeile ein, in der er benötigt wird, um Bestätigungsaufforderungen zu unterdrücken.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Richtlinie gelöscht, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet. Bevor Sie die Richtlinie

löschen können, müssen Sie zunächst alle Versionen außer der Standardversion löschen, indem Sie **`Remove-IAMPolicyVersion`** ausführen. Sie müssen die Richtlinie auch von allen IAM-Benutzern, -Gruppen oder -Rollen trennen.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen gelöscht werden, die Richtlinie dann von allen zugeordneten IAM-Entitäten getrennt wird und schließlich die Richtlinie selbst gelöscht wird. In der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßige Version gekennzeichnet sind, in einer Sammlung erfasst und dann wird jede Richtlinie in der Sammlung gelöscht. In der dritten Zeile werden alle Benutzer, Gruppen und Rollen des IAM abgerufen, denen die Richtlinie angefügt ist. Die Zeilen vier bis sechs trennen die Richtlinie von jeder angefügten Entität. In der letzten Zeile wird dieser Befehl zum Entfernen der verwalteten Richtlinie sowie der verbleibenden Standardversion verwendet. Das Beispiel schließt den **`-Force`**-Switch-Parameter in jede Zeile ein, in der er benötigt wird, um Bestätigungsaufforderungen zu unterdrücken.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- Einzelheiten zur API finden Sie unter DeletePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- Einzelheiten zur API finden Sie DeletePolicyin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deletepolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Policy cannot be deleted due to attachments.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeletePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
```

```
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            print("ERROR: deletePolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeletePolicyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeletePolicyVersion`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePolicyVersion` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten von Richtlinien
- Zurücksetzen einer Richtlinienversion

CLI

AWS CLI

So löschen Sie eine Version einer verwalteten Richtlinie

Dieses Beispiel löscht die als `v2` identifizierte Version aus der Richtlinie, deren ARN `arn:aws:iam::123456789012:policy/MySamplePolicy` lautet.

```
aws iam delete-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeletePolicyVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die als **v2** identifizierte Version aus der Richtlinie gelöscht, deren ARN **arn:aws:iam::123456789012:policy/MySamplePolicy** lautet.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen und dann die Richtlinie selbst gelöscht werden. In der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßig gekennzeichnet sind, in einer Sammlung abgerufen und dann wird dieser Befehl zur Löschung jeder einzelnen Richtlinie in der Sammlung verwendet. In der letzten Zeile wird die Richtlinie selbst sowie die verbleibende Standardversion entfernt. Beachten Sie, dass Sie zum erfolgreichen Löschen einer verwalteten Richtlinie die Richtlinie auch von allen Benutzern, Gruppen oder Rollen trennen müssen, indem Sie die Befehle **Unregister-IAMUserPolicy**, **Unregister-IAMGroupPolicy** und **Unregister-IAMRolePolicy** verwenden. Weitere Informationen finden Sie im Beispiel für das **Remove-IAMPolicy** Cmdlet.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Einzelheiten zur API finden Sie unter DeletePolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die als **v2** identifizierte Version aus der Richtlinie gelöscht, deren ARN **arn:aws:iam::123456789012:policy/MySamplePolicy** lautet.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen und dann die Richtlinie selbst gelöscht werden. In

der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßig gekennzeichnet sind, in einer Sammlung abgerufen und dann wird dieser Befehl zur Löschung jeder einzelnen Richtlinie in der Sammlung verwendet. In der letzten Zeile wird die Richtlinie selbst sowie die verbleibende Standardversion entfernt. Beachten Sie, dass Sie zum erfolgreichen Löschen einer verwalteten Richtlinie die Richtlinie auch von allen Benutzern, Gruppen oder Rollen trennen müssen, indem Sie die Befehle **`Unregister-IAMUserPolicy`**, **`Unregister-IAMGroupPolicy`** und **`Unregister-IAMRolePolicy`** verwenden. Weitere Informationen finden Sie im Beispiel für das **`Remove-IAMPolicy`** Cmdlet.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- Einzelheiten zur API finden Sie unter DeletePolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deletepolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Policy version deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Cannot delete default policy version.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeletePolicyVersionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteRole`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRole` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten Sie Rollen
- Übertragung von hartcodierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Einrichtung der Systemverwaltung
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
```

```
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
```

```
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS CLI Befehlsreferenz.

CLI

AWS CLI

So löschen Sie eine IAM-Rolle

Mit dem folgenden `delete-role`-Befehl wird die Rolle mit dem Namen `Test-Role` entfernt.

```
aws iam delete-role \
    --role-name Test-Role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Bevor Sie eine Rolle löschen können, müssen Sie die Rolle aus allen Instance-Profilen entfernen (`remove-role-from-instance-profile`), alle verwalteten Richtlinien entfernen (`detach-role-policy`) und alle eingebundenen Richtlinien, die der Rolle angefügt sind (`delete-role-policy`), löschen.

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen und Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteRolein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Rolle.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **`MyNewRole`** aus dem aktuellen IAM-Konto gelöscht. Bevor Sie die Rolle löschen können, müssen Sie zunächst mit dem **`Unregister-IAMRolePolicy`**-Befehl alle verwalteten Richtlinien entfernen. Inline-Richtlinien werden mit der Rolle gelöscht.

```
Remove-IAMRole -RoleName MyNewRole
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien von der Rolle mit dem Namen **`MyNewRole`** getrennt und dann die Rolle gelöscht. In der ersten Zeile werden alle verwalteten Richtlinien abgerufen, die der Rolle als Sammlung zugeordnet sind, und dann werden alle Richtlinien in der Sammlung von der Rolle getrennt. In der zweiten Zeile wird die Rolle selbst gelöscht. Inline-Richtlinien werden zusammen mit der Rolle gelöscht.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Einzelheiten zur API finden Sie unter DeleteRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **MyNewRole** aus dem aktuellen IAM-Konto gelöscht. Bevor Sie die Rolle löschen können, müssen Sie zunächst mit dem **Unregister-IAMRolePolicy**-Befehl alle verwalteten Richtlinien entfernen. Inline-Richtlinien werden mit der Rolle gelöscht.

```
Remove-IAMRole -RoleName MyNewRole
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien von der Rolle mit dem Namen **MyNewRole** getrennt und dann die Rolle gelöscht. In der ersten Zeile werden alle verwalteten Richtlinien abgerufen, die der Rolle als Sammlung zugeordnet sind, und dann werden alle Richtlinien in der Sammlung von der Rolle getrennt. In der zweiten Zeile wird die Rolle selbst gelöscht. Inline-Richtlinien werden zusammen mit der Rolle gelöscht.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- Einzelheiten zur API finden Sie unter DeleteRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
```

```
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- Einzelheiten zur API finden Sie DeleteRolein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
                                        role_name: role_name,
                                        policy_arn: policy.policy_arn
                                      })
        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?('aws:policy/')
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end
```

```
    # Delete the role
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Deleted role #{role_name}.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
 why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie DeleteRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
 {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteRolein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deleterole( iv_rolename = iv_role_name ).
        MESSAGE 'Role deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Role cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteRolein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
```

```
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            print("ERROR: deleteRole:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteRolein der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteRolePermissionsBoundary`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRolePermissionsBoundary` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Berechtigungsgrenze aus einer IAM-Rolle

Im folgenden Beispiel `delete-role-permissions-boundary` wird die Berechtigungsgrenze für die angegebene IAM-Rolle gelöscht. Verwenden Sie den Befehl `put-role-permissions-boundary`, um einer Rolle eine Berechtigungsgrenze anzuwenden.

```
aws iam delete-role-permissions-boundary \
    --role-name lambda-application-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteRolePermissionsBoundaryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie die an eine IAM-Rolle angefügte Berechtigung entfernt wird.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter DeleteRolePermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie die an eine IAM-Rolle angefügte Berechtigung entfernt wird.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter DeleteRolePermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteRolePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteRolePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteRolePolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So entfernen Sie eine Richtlinie aus einer IAM-Rolle

Mit dem folgenden `delete-role-policy`-Befehl wird die Richtlinie mit dem Namen `ExamplePolicy` aus der Rolle mit dem Namen `Test-Role` entfernt.

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteRolePolicyin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteRolePolicyin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie **`S3AccessPolicy`** gelöscht, die in die IAM-Rolle **`S3BackupRole`** eingebettet ist.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Einzelheiten zur API finden Sie unter DeleteRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie **`S3AccessPolicy`** gelöscht, die in die IAM-Rolle **`S3BackupRole`** eingebettet ist.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- Einzelheiten zur API finden Sie unter DeleteRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

`Verwenden Sie DeleteSAMLProvider mit einem` AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSAMLProvider` verwendet wird.

CLI

AWS CLI

So löschen Sie einen SAML-Anbieter

In diesem Beispiel wird der IAM SAML 2.0-Anbieter gelöscht, dessen ARN `arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider` lautet.

```
aws iam delete-saml-provider \
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von IAM-SAML-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Weitere API-Informationen finden Sie unter DeleteSAMLProvider in der AWS CLI -Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter DeleteSAMLProvider in der API-Referenz für AWS SDK für JavaScript .

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der IAM SAML 2.0-Anbieter gelöscht, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`** lautet.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- API-Details finden Sie unter DeleteSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM SAML 2.0-Anbieter gelöscht, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`** lautet.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Weitere API-Informationen finden Sie unter DeleteSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Verwenden Sie **`DeleteServerCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteServerCertificate` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
```

```
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " <<
 certificateName
                  << std::endl;
    }

    return result;
}
```

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um ein Serverzertifikat aus Ihrem AWS Konto zu löschen

Der folgende `delete-server-certificate` Befehl entfernt das angegebene Serverzertifikat aus Ihrem AWS Konto.

```
aws iam delete-server-certificate \
```

```
    --server-certificate-name myUpdatedServerCertificate
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `list-server-certificates` Befehl, um die in Ihrem AWS Konto verfügbaren Serverzertifikate aufzulisten.

Weitere Informationen finden Sie unter Verwaltung von Serverzertifikaten in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Serverzertifikat.

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Serverzertifikat mit dem Namen **`MyServerCert`** gelöscht.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Einzelheiten zur API finden Sie unter DeleteServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Serverzertifikat mit dem Namen **`MyServerCert`** gelöscht.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- Einzelheiten zur API finden Sie unter DeleteServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
```

```
                                                server_certificate_name: name,
                                                certificate_body: certificate_body,
                                                private_key: private_key
                                              })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
```

```
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`DeleteServiceLinkedRole`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteServiceLinkedRole` verwendet wird.

CLI

AWS CLI

So löschen Sie eine serviceverknüpfte Rolle

Im folgenden `delete-service-linked-role`-Beispiel wird die angegebene serviceverknüpfte Rolle, die Sie nicht mehr benötigen, gelöscht. Der Löschvorgang erfolgt asynchron. Sie können den Status des Löschvorgangs mithilfe des `get-service-linked-role-deletion-status`-Befehls überprüfen und bestätigen, wann der Vorgang abgeschlossen ist.

```
aws iam delete-service-linked-role \
    --role-name AWSServiceRoleForLexBots
```

Ausgabe:

```
{
    "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
```

```
}
```

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wurde die serviceverknüpfte Rolle gelöscht. Bitte beachten Sie, dass dieser Befehl fehlschlägt, wenn der Service diese Rolle weiterhin verwendet.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Einzelheiten zur API finden Sie unter DeleteServiceLinkedRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wurde die serviceverknüpfte Rolle gelöscht. Bitte beachten Sie, dass dieser Befehl fehlschlägt, wenn der Service diese Rolle weiterhin verwendet.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- Einzelheiten zur API finden Sie unter DeleteServiceLinkedRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a service-linked role.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_service_linked_role(role_name)
    response = @iam_client.delete_service_linked_role(role_name: role_name)
    task_id = response.deletion_task_id
    check_deletion_status(role_name, task_id)
  rescue Aws::Errors::ServiceError => e
    handle_deletion_error(e, role_name)
  end

  private
```

```
# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id
    )
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)

    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  return if e.code == 'NoSuchEntity'

  @logger.error("Couldn't delete #{role_name}. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteServiceLinkedRolein der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DeleteSigningCertificate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSigningCertificate` verwendet wird.

CLI

AWS CLI

So löschen Sie ein Signaturzertifikat für einen IAM-Benutzer

Der folgende `delete-signing-certificate`-Befehl löscht das angegebene Signaturzertifikat für den IAM-Benutzer mit dem Namen `Bob`.

```
aws iam delete-signing-certificate \
    --user-name Bob \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `list-signing-certificates`-Befehl, um die ID für ein Signaturzertifikat abzurufen.

Weitere Informationen finden Sie unter Verwaltung von Signaturzertifikaten im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie DeleteSigningCertificatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Signaturzertifikat mit der ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** des IAM-Benutzers mit dem Namen **`Bob`** gelöscht.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Einzelheiten zur API finden Sie unter DeleteSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Signaturzertifikat mit der ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** des IAM-Benutzers mit dem Namen **`Bob`** gelöscht.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- Einzelheiten zur API finden Sie unter DeleteSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteUser`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteUser` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
```

```
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
while getopts "u:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    User name:   $user_name"
iecho ""

# If the user does not exist, we don't want to try to delete it.
if (! iam_user_exists "$user_name"); then
  errecho "ERROR: A user with that name does not exist in the account."
  return 1
fi

response=$(aws iam delete-user \
  --user-name "$user_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-user operation failed.$response"
  return 1
fi

iecho "delete-user response:$response"
```

```
  iecho

  return 0
}
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
    }
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie einen IAM-Benutzer

Mit dem folgenden `delete-user`-Befehl wird der IAM-Benutzer mit dem Namen Bob aus dem aktuellen Konto entfernt.

```
aws iam delete-user \
    --user-name Bob
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines IAM-Benutzers im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteUserin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Benutzer.

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMUser(userNameVal: String) {
```

```
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DeleteUserin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **Bob** gelöscht.

```
Remove-IAMUser -UserName Bob
```

Beispiel 2: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **Theresa** zusammen mit allen Elementen gelöscht, die zuerst gelöscht werden müssen.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }
```

```
# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Einzelheiten zur API finden Sie unter DeleteUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **Bob** gelöscht.

```
Remove-IAMUser -UserName Bob
```

Beispiel 2: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **Theresa** zusammen mit allen Elementen gelöscht, die zuerst gelöscht werden müssen.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
```

```
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- Einzelheiten zur API finden Sie unter DeleteUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- Einzelheiten zur API finden Sie DeleteUserin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a user and their associated resources
```

```
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
```

```
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
}
```

- Einzelheiten zur API finden Sie DeleteUserin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deleteuser( iv_username = iv_user_name ).
        MESSAGE 'User deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'User cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteUserin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteUser(user: IAMClientTypes.User) async throws {
        let input = DeleteUserInput(
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUser(input: input)
        } catch {
            print("ERROR: deleteUser:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteUserin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteUserPermissionsBoundary`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteUserPermissionsBoundary` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Berechtigungsgrenze von einem IAM-Benutzer

Im folgenden `delete-user-permissions-boundary`-Beispiel wird die Berechtigungsgrenze gelöscht, die dem IAM-Benutzer mit dem Namen `intern` zugeordnet ist. Um einem Benutzer eine Berechtigungsgrenze zuzuweisen, verwenden Sie den Befehl `put-user-permissions-boundary`.

```
aws iam delete-user-permissions-boundary \
    --user-name intern
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteUserPermissionsBoundaryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze entfernt wird, die einem IAM-Benutzer zugeordnet ist.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Einzelheiten zur API finden Sie unter DeleteUserPermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze entfernt wird, die einem IAM-Benutzer zugeordnet ist.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- Einzelheiten zur API finden Sie unter DeleteUserPermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteUserPolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteUserPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So entfernen Sie eine Richtlinie von einem IAM-Benutzer

Mit dem folgenden `delete-user-policy`-Befehl wird die angegebene Richtlinie vom IAM-Benutzer mit dem Namen Bob entfernt.

```
aws iam delete-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Verwenden Sie den `list-user-policies`-Befehl, um eine Liste der Richtlinien für einen IAM-Benutzer abzurufen.

Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Einen IAM-Benutzer in Ihrem AWS Konto erstellen.AWS

- Einzelheiten zur API finden Sie unter DeleteUserPolicy AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`AccessToEC2Policy`** gelöscht, die in den IAM-Benutzer mit dem Namen **`Bob`** eingebettet ist.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Inline-Richtlinien gefunden, die in den IAM-Benutzer mit dem Namen **`Theresa`** eingebettet sind, und dann gelöscht.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Einzelheiten zur API finden Sie unter DeleteUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`AccessToEC2Policy`** gelöscht, die in den IAM-Benutzer mit dem Namen **`Bob`** eingebettet ist.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Inline-Richtlinien gefunden, die in den IAM-Benutzer mit dem Namen **`Theresa`** eingebettet sind, und dann gelöscht.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- Einzelheiten zur API finden Sie unter DeleteUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes a user and their associated resources
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
```

```
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
 throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            print("ERROR: deleteUserPolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteUserPolicyin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteVirtualMfaDevice`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVirtualMfaDevice` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein virtuelles MFA-Gerät

Mit dem folgenden `delete-virtual-mfa-device`-Befehl wird das angegebene MFA-Gerät vom aktuellen Konto entfernt.

```
aws iam delete-virtual-mfa-device \
    --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Deaktivierung von MFA-Geräten im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteVirtualMfaDeviceein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das virtuelle IAM-MFA-Gerät gelöscht, dessen ARN **`arn:aws:iam::123456789012:mfa/bob`** lautet.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Beispiel 2: In diesem Beispiel wird überprüft, ob der IAM-Benutzerin Theresa ein MFA-Gerät zugewiesen ist. Wenn eines gefunden wird, wird das Gerät für den IAM-Benutzer deaktiviert. Wenn das Gerät virtuell ist, wird es ebenfalls gelöscht.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- Einzelheiten zur API finden Sie unter DeleteVirtualMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das virtuelle IAM-MFA-Gerät gelöscht, dessen ARN **`arn:aws:iam::123456789012:mfa/bob`** lautet.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Beispiel 2: In diesem Beispiel wird überprüft, ob der IAM-Benutzerin Theresa ein MFA-Gerät zugewiesen ist. Wenn eines gefunden wird, wird das Gerät für den IAM-Benutzer deaktiviert. Wenn das Gerät virtuell ist, wird es ebenfalls gelöscht.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- Einzelheiten zur API finden Sie unter DeleteVirtualMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DetachGroupPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DetachGroupPolicy` verwendet wird.

CLI

AWS CLI

So trennen Sie eine Richtlinie von einer Gruppe

In diesem Beispiel wird die verwaltete Richtlinie mit dem ARN `arn:aws:iam::123456789012:policy/TesterAccessPolicy` aus der Gruppe mit dem Namen `Testers` entfernt.

```
aws iam detach-group-policy \
    --group-name Testers \
    --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwaltung von IAM-Benutzergruppen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DetachGroupPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** lautet, von der Gruppe mit dem Namen **`Testers`** getrennt.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Gruppe mit dem Namen **`Testers`** zugeordnet sind, und von der Gruppe getrennt.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Einzelheiten zur API finden Sie unter DetachGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** lautet, von der Gruppe mit dem Namen **`Testers`** getrennt.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Gruppe mit dem Namen **`Testers`** zugeordnet sind, und von der Gruppe getrennt.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- Einzelheiten zur API finden Sie unter DetachGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DetachRolePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetachRolePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Einen verwalteten Arbeitsbereich für die Überwachung erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten Sie Rollen
- Übertragung von hartcodierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Einrichtung der Systemverwaltung
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam detach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role
 "
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So trennen Sie eine Richtlinie von einer Rolle

In diesem Beispiel wird die verwaltete Richtlinie mit dem ARN `arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy` aus der Rolle mit dem Namen `FedTesterRole` entfernt.

```
aws iam detach-role-policy \
    --role-name FedTesterRole \
    --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
```

```
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

```
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
```

```
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Trennen Sie die Richtlinie.

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
```

```
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
```

```
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** lautet, von der Rolle mit dem Namen **`FedTesterRole`** getrennt.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Rolle mit dem Namen **`FedTesterRole`** zugeordnet sind, und von der Rolle getrennt.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- Einzelheiten zur API finden Sie unter DetachRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** lautet, von der Rolle mit dem Namen **`FedTesterRole`** getrennt.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Rolle mit dem Namen **`FedTesterRole`** zugeordnet sind, und von der Rolle getrennt.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- Einzelheiten zur API finden Sie unter DetachRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Trennen Sie eine Richtlinie von einer Rolle mithilfe des Boto3-Richtlinienobjekts.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

Trennen Sie eine Richtlinie von einer Rolle mithilfe des Boto3-Rollenobjekts.

```
def detach_policy(role_name, policy_arn):
```

```
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- Einzelheiten zur API finden Sie DetachRolePolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
```

```
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        lo_iam->detachrolepolicy(
          iv_rolename = iv_role_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy detached from role successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role or policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
 IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            print("ERROR: detachRolePolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DetachUserPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetachUserPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

CLI

AWS CLI

So trennen Sie eine Richtlinie von einem Benutzer

In diesem Beispiel wird die verwaltete Richtlinie mit dem ARN `arn:aws:iam::123456789012:policy/TesterPolicy` vom Benutzer Bob entfernt.

```
aws iam detach-user-policy \
    --user-name Bob \
    --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Änderung der Berechtigungen für einen IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DetachUserPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die verwaltete Richtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterPolicy`** lautet, vom IAM-Benutzer mit dem Namen **`Bob`** getrennt.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die dem IAM-Benutzer mit dem Namen **`Theresa`** zugeordnet sind, und diese Richtlinien werden vom Benutzer getrennt.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- Einzelheiten zur API finden Sie unter DetachUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die verwaltete Richtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterPolicy`** lautet, vom IAM-Benutzer mit dem Namen **`Bob`** getrennt.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die dem IAM-Benutzer mit dem Namen **`Theresa`** zugeordnet sind, und diese Richtlinien werden vom Benutzer getrennt.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- Einzelheiten zur API finden Sie unter DetachUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- Einzelheiten zur API finden Sie DetachUserPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Detaches a policy from a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The ARN of the policy to detach
  # @return [Boolean] true if the policy was successfully detached, false
 otherwise
  def detach_user_policy(user_name, policy_arn)
    @iam_client.detach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}':
 #{e.message}")
    false
  end
```

- Einzelheiten zur API finden Sie DetachUserPolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
```

```
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie DetachUserPolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->detachuserpolicy(
          iv_username = iv_user_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy detached from user successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User or policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetachUserPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`EnableMfaDevice`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `EnableMfaDevice` verwendet wird.

CLI

AWS CLI

So aktivieren Sie ein MFA-Gerät

Nachdem Sie mit dem `create-virtual-mfa-device`-Befehl ein neues virtuelles MFA-Gerät erstellt haben, können Sie das MFA-Gerät einem Benutzer zuweisen. Im folgenden `enable-mfa-device`-Beispiel wird dem Benutzer Bob das MFA-Gerät mit der Seriennummer `arn:aws:iam::210987654321:mfa/BobsMFADevice` zugewiesen. Der Befehl synchronisiert das Gerät auch mit, AWS indem er die ersten beiden Codes nacheinander vom virtuellen MFA-Gerät einfügt.

```
aws iam enable-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 789012
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Aktivieren eines virtuellen Geräts zur Multi-Faktor-Authentifizierung (MFA) im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie EnableMfaDevicein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Befehl wird das Hardware-MFA-Gerät mit der Seriennummer **`987654321098`** aktiviert und das Gerät dem Benutzer **`Bob`** zugeordnet. Er enthält nacheinander die ersten beiden Codes des Geräts.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Beispiel 2: In diesem Beispiel wird ein virtuelles MFA-Gerät erstellt und aktiviert. Mit dem ersten Befehl wird das virtuelle Gerät erstellt und die Objektdarstellung des Geräts in der Variablen **`$MFADevice`** zurückgegeben. Sie können die Eigenschaften **`.Base32StringSeed`** oder **`QRCodePng`** verwenden, um die Softwar-Aanwendung des Benutzers zu konfigurieren.

Mit dem letzten Befehl wird das Gerät dem Benutzer **David** zugewiesen und das Gerät anhand seiner Seriennummer identifiziert. Der Befehl synchronisiert das Gerät auch mit, AWS indem er die ersten beiden Codes nacheinander vom virtuellen MFA-Gerät einfügt.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Einzelheiten zur API finden Sie unter EnableMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Befehl wird das Hardware-MFA-Gerät mit der Seriennummer **987654321098** aktiviert und das Gerät dem Benutzer **Bob** zugeordnet. Er enthält nacheinander die ersten beiden Codes des Geräts.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Beispiel 2: In diesem Beispiel wird ein virtuelles MFA-Gerät erstellt und aktiviert. Mit dem ersten Befehl wird das virtuelle Gerät erstellt und die Objektdarstellung des Geräts in der Variablen **$MFADevice** zurückgegeben. Sie können die Eigenschaften **.Base32StringSeed** oder **QRCodePng** verwenden, um die Softwar-Aanwendung des Benutzers zu konfigurieren. Mit dem letzten Befehl wird das Gerät dem Benutzer **David** zugewiesen und das Gerät anhand seiner Seriennummer identifiziert. Der Befehl synchronisiert das Gerät auch mit, AWS indem er die ersten beiden Codes nacheinander vom virtuellen MFA-Gerät einfügt.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- Einzelheiten zur API finden Sie unter EnableMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GenerateCredentialReport`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateCredentialReport` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

CLI

AWS CLI

So erstellen Sie einen Bericht zu Anmeldeinformationen

Im folgenden Beispiel wird versucht, einen Anmeldeinformationsbericht für das AWS Konto zu generieren.

```
aws iam generate-credential-report
```

Ausgabe:

```
{
    "State":  "STARTED",
    "Description": "No report exists. Starting a new report generation task"
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Abrufen von Berichten zu Anmeldeinformationen für Ihr AWS Konto.

- Einzelheiten zur API finden Sie unter GenerateCredentialReport AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Erstellung eines neuen Berichts angefordert, was alle vier Stunden erfolgen kann. Wenn der letzte Bericht noch aktuell ist, lautet das Feld „Status“ **`COMPLETE`**. Verwenden Sie **`Get-IAMCredentialReport`**, um den vollständigen Bericht anzuzeigen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Einzelheiten zur API finden Sie unter GenerateCredentialReport AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Erstellung eines neuen Berichts angefordert, was alle vier Stunden erfolgen kann. Wenn der letzte Bericht noch aktuell ist, lautet das Feld „Status“ **`COMPLETE`**. Verwenden Sie **`Get-IAMCredentialReport`**, um den vollständigen Bericht anzuzeigen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                  State
-----------                                                  -----
No report exists. Starting a new report generation task      STARTED
```

- Einzelheiten zur API finden Sie unter GenerateCredentialReport AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response
```

- Einzelheiten zur API finden Sie GenerateCredentialReportin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->generatecredentialreport( ).
        MESSAGE 'Credential report generation started.' TYPE 'I'.
```

```
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Report generation limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when generating credential report.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GenerateCredentialReportin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GenerateServiceLastAccessedDetails`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateServiceLastAccessedDetails` verwendet wird.

CLI

AWS CLI

Beispiel 1: So generieren Sie einen Service-Zugriffsbericht für eine benutzerdefinierte Richtlinie

Im folgenden Beispiel `generate-service-last-accessed-details` wird ein Hintergrundauftrag gestartet, um einen Bericht zu generieren, der die Services auflistet, auf die IAM-Benutzer und andere Entitäten mit einer benutzerdefinierten Richtlinie mit dem Namen `intern-boundary` zugreifen. Sie können den Bericht anzeigen, nachdem er durch Ausführung des Befehls `get-service-last-accessed-details` erstellt wurde.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::123456789012:policy/intern-boundary
```

Ausgabe:

```
{
    "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"
}
```

Beispiel 2: Um einen Servicezugriffsbericht für die AWS verwaltete AdministratorAccess Richtlinie zu generieren

Im folgenden `generate-service-last-accessed-details` Beispiel wird ein Hintergrundjob gestartet, um einen Bericht zu generieren, der die Dienste auflistet, auf die IAM-Benutzer und andere Entitäten mit der AWS verwalteten `AdministratorAccess` Richtlinie zugreifen. Sie können den Bericht anzeigen, nachdem er durch Ausführung des Befehls `get-service-last-accessed-details` erstellt wurde.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::aws:policy/AdministratorAccess
```

Ausgabe:

```
{
    "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Verfeinerung von Berechtigungen bei der AWS Verwendung von Informationen, auf die zuletzt zugegriffen wurde.

- Einzelheiten zur API finden Sie GenerateServiceLastAccessedDetailsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel entspricht dem API-Cmdlet. GenerateServiceLastAccessedDetails Dies liefert eine Job-ID, die in und verwendet werden kann Get-IAMServiceLastAccessedDetail Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Einzelheiten zur API finden Sie unter GenerateServiceLastAccessedDetails AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel entspricht dem API-Cmdlet. GenerateServiceLastAccessedDetails Dies liefert eine Job-ID, die in und verwendet werden kann Get-IAMServiceLastAccessedDetail Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- Einzelheiten zur API finden Sie unter GenerateServiceLastAccessedDetails AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetAccessKeyLastUsed`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAccessKeyLastUsed` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffsschlüsseln

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
 iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
```

```
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen darüber ab, wann der angegebene Zugriffsschlüssel zuletzt verwendet wurde

Im folgenden Beispiel werden Informationen darüber abgerufen, wann der Zugriffsschlüssel `ABCDEXAMPLE` zuletzt verwendet wurde.

```
aws iam get-access-key-last-used \
    --access-key-id ABCDEXAMPLE
```

Ausgabe:

```
{
    "UserName":  "Bob",
    "AccessKeyLastUsed": {
        "Region": "us-east-1",
        "ServiceName": "iam",
        "LastUsedDate": "2015-06-16T22:45:00Z"
    }
}
```

Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie den Zugriffsschlüssel ab.

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });

  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt den Benutzernamen des Besitzers und die Informationen zur letzten Verwendung für den angegebenen Zugriffsschlüssel zurück.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Einzelheiten zur API finden Sie unter GetAccessKeyLastUsed AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt den Benutzernamen des Besitzers und die Informationen zur letzten Verwendung für den angegebenen Zugriffsschlüssel zurück.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- Einzelheiten zur API finden Sie unter GetAccessKeyLastUsed AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
```

```
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getaccesskeylastused(
          iv_accesskeyid = iv_access_key_id ).
        MESSAGE 'Retrieved access key last used information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Access key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetAccountAuthorizationDetails`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAccountAuthorizationDetails` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

CLI

AWS CLI

Um die IAM-Benutzer, -Gruppen, -Rollen und -Richtlinien eines AWS Kontos aufzulisten

Der folgende `get-account-authorization-details` Befehl gibt Informationen zu allen IAM-Benutzern, -Gruppen, -Rollen und -Richtlinien im AWS Konto zurück.

```
aws iam get-account-authorization-details
```

Ausgabe:

```
{
    "RoleDetailList": [
        {
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "RoleId": "AROA1234567890EXAMPLE",
```

```
            "CreateDate": "2014-07-30T17:09:20Z",
            "InstanceProfileList": [
                {
                    "InstanceProfileId": "AIPA1234567890EXAMPLE",
                    "Roles": [
                        {
                            "AssumeRolePolicyDocument": {
                                "Version":"2012-10-17",
                                "Statement": [
                                    {
                                        "Sid": "",
                                        "Effect": "Allow",
                                        "Principal": {
                                            "Service": "ec2.amazonaws.com"
                                        },
                                        "Action": "sts:AssumeRole"
                                    }
                                ]
                            },
                            "RoleId": "AROA1234567890EXAMPLE",
                            "CreateDate": "2014-07-30T17:09:20Z",
                            "RoleName": "EC2role",
                            "Path": "/",
                            "Arn": "arn:aws:iam::123456789012:role/EC2role"
                        }
                    ],
                    "CreateDate": "2014-07-30T17:09:20Z",
                    "InstanceProfileName": "EC2role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
                }
            ],
            "RoleName": "EC2role",
            "Path": "/",
            "AttachedManagedPolicies": [
                {
                    "PolicyName": "AmazonS3FullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
                },
                {
                    "PolicyName": "AmazonDynamoDBFullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
                }
```

```
            ],
            "RoleLastUsed": {
                "Region": "us-west-2",
                "LastUsedDate": "2019-11-13T17:30:00Z"
            },
            "RolePolicyList": [],
            "Arn": "arn:aws:iam::123456789012:role/EC2role"
        }
    ],
    "GroupDetailList": [
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            },
            "GroupName": "Admins",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "CreateDate": "2013-10-14T18:32:24Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "PowerUserAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
            },
            "GroupName": "Dev",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Dev",
            "CreateDate": "2013-10-14T18:33:55Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": [],
            "GroupName": "Finance",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Finance",
            "CreateDate": "2013-10-14T18:57:48Z",
            "GroupPolicyList": [
                {
                    "PolicyName": "policygen-201310141157",
```

```
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Action": "aws-portal:*",
                                "Sid": "Stmt1381777017000",
                                "Resource": "*",
                                "Effect": "Allow"
                            }
                        ]
                    }
                }
            ]
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
                                "aws-portal:*",
```

```
                                "iam:*"
                            ],
                            "Resource": "*"
                        }
                    }
                }
            ],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        },
        {
            "UserName": "Charlie",
            "GroupList": [
                "Dev"
            ],
            "CreateDate": "2013-10-14T18:33:56Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Charlie"
        }
    ],
    "Policies": [
        {
            "PolicyName": "create-update-delete-set-managed-policies",
            "CreateDate": "2015-02-06T19:58:34Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-02-06T19:58:34Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Allow",
                            "Action": [
                                "iam:CreatePolicy",
                                "iam:CreatePolicyVersion",
                                "iam:DeletePolicy",
```

```
                                "iam:DeletePolicyVersion",
                                "iam:GetPolicy",
                                "iam:GetPolicyVersion",
                                "iam:ListPolicies",
                                "iam:ListPolicyVersions",
                                "iam:SetDefaultPolicyVersion"
                            ],
                            "Resource": "*"
                        }
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
            "UpdateDate": "2015-02-06T19:58:34Z"
        },
        {
            "PolicyName": "S3-read-only-specific-bucket",
            "CreateDate": "2015-01-21T21:39:41Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-01-21T21:39:41Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:Get*",
                                    "s3:List*"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::amzn-s3-demo-bucket",
                                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                                ]
                            }
                        ]
```

```
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
            "UpdateDate": "2015-01-21T23:39:41Z"
        },
        {
            "PolicyName": "AmazonEC2FullAccess",
            "CreateDate": "2015-02-06T18:40:15Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2014-10-30T20:59:46Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Action": "ec2:*",
                                "Effect": "Allow",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "elasticloadbalancing:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "cloudwatch:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "autoscaling:*",
                                "Resource": "*"
                            }
                        ]
```

```
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
            "UpdateDate": "2015-02-06T18:40:15Z"
        }
    ],
    "Marker": "EXAMPLEkakv9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
    "IsTruncated": true
}
```

Weitere Informationen finden Sie unter AWS -Sicherheitsaudit-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetAccountAuthorizationDetails AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Autorisierungsdetails zu den Identitäten im AWS Konto abgerufen und die Elementliste des zurückgegebenen Objekts angezeigt, einschließlich Benutzer, Gruppen und Rollen. Beispielsweise zeigt die **`UserDetailList`**-Eigenschaft Details zu den Benutzern an. Ähnliche Informationen sind in den **`RoleDetailList`**- und **`GroupDetailList`**-Eigenschaften verfügbar.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Ausgabe:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Ausgabe:

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
GroupList       : {Administrators}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE1
UserName        : Administrator
UserPolicyList  : {}

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
GroupList       : {Developers}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE2
UserName        : bab
UserPolicyList  : {}

Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
GroupList       : {Backup}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE3
UserName        : BackupToS3
UserPolicyList  : {BackupServicePermissionsToS3Buckets}
```

- Einzelheiten zur API finden Sie unter GetAccountAuthorizationDetails AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Autorisierungsdetails zu den Identitäten im AWS Konto abgerufen und die Elementliste des zurückgegebenen Objekts angezeigt, einschließlich Benutzer, Gruppen und Rollen. Beispielsweise zeigt die **`UserDetailList`**-Eigenschaft Details zu den Benutzern an. Ähnliche Informationen sind in den **`RoleDetailList`**- und **`GroupDetailList`**-Eigenschaften verfügbar.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Ausgabe:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Ausgabe:

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/BackupToS3
CreateDate     : 1/27/2015 10:15:08 AM
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- Einzelheiten zur API finden Sie unter GetAccountAuthorizationDetails AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- Einzelheiten zur API finden Sie GetAccountAuthorizationDetailsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_iam->getaccountauthdetails( ).
       MESSAGE 'Retrieved account authorization details.' TYPE 'I'.
     CATCH /aws1/cx_iamservicefailureex.
       MESSAGE 'Service failure when getting account authorization details.'
TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetAccountAuthorizationDetailsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetAccountPasswordPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAccountPasswordPolicy` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
```

```
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So zeigen Sie die Passwortrichtlinie für das aktuelle Konto an

Mit dem folgenden `get-account-password-policy`-Befehl werden Details zur Passwortrichtlinie für das aktuelle Konto angezeigt.

```
aws iam get-account-password-policy
```

Ausgabe:

```
{
    "PasswordPolicy": {
        "AllowUsersToChangePassword": false,
        "RequireLowercaseCharacters": false,
        "RequireUppercaseCharacters": false,
        "MinimumPasswordLength": 8,
        "RequireNumbers": true,
        "RequireSymbols": true
    }
}
```

Wenn keine Passwortwortrichtlinie für das Konto definiert ist, gibt der Befehl einen `NoSuchEntity`-Fehler zurück.

Weitere Informationen finden Sie unter Festlegen einer Kontopasswortrichtlinie für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
```

```
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Passwort-Richtlinie des Kontos ab.

```
import {
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zur Passwortrichtlinie für das aktuelle Konto zurückgegeben. Wenn keine Passwortrichtlinie für das Konto definiert ist, gibt der Befehl einen **`NoSuchEntity`**-Fehler zurück.

```
Get-IAMAccountPasswordPolicy
```

Ausgabe:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
```

```
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Einzelheiten zur API finden Sie unter GetAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zur Passwortrichtlinie für das aktuelle Konto zurückgegeben. Wenn keine Passwortrichtlinie für das Konto definiert ist, gibt der Befehl einen **`NoSuchEntity`**-Fehler zurück.

```
Get-IAMAccountPasswordPolicy
```

Ausgabe:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- Einzelheiten zur API finden Sie unter GetAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
 {pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
 {pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
 {pw_policy.require_lowercase_characters}"
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
 {pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
 SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getaccountpasswordpolicy( ).
        MESSAGE 'Retrieved account password policy.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'No password policy exists.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting password policy.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetAccountPasswordPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetAccountSummary`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAccountSummary` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

CLI

AWS CLI

So rufen Sie Informationen über die Nutzung von IAM-Entitäten und IAM-Kontingenten auf dem aktuellen Konto ab

Der folgende `get-account-summary`-Befehl gibt Informationen zur aktuellen IAM-Entitätsnutzung und zu den aktuellen IAM-Entitätskontingenten im Konto zurück.

```
aws iam get-account-summary
```

Ausgabe:

```
{
    "SummaryMap": {
        "UsersQuota": 5000,
        "GroupsQuota": 100,
        "InstanceProfiles": 6,
        "SigningCertificatesPerUserQuota": 2,
        "AccountAccessKeysPresent": 0,
        "RolesQuota": 250,
        "RolePolicySizeQuota": 10240,
        "AccountSigningCertificatesPresent": 0,
        "Users": 27,
        "ServerCertificatesQuota": 20,
```

```
        "ServerCertificates": 0,
        "AssumeRolePolicySizeQuota": 2048,
        "Groups": 7,
        "MFADevicesInUse": 1,
        "Roles": 3,
        "AccountMFAEnabled": 1,
        "MFADevices": 3,
        "GroupsPerUserQuota": 10,
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 100,
        "AccessKeysPerUserQuota": 2,
        "Providers": 0,
        "UserPolicySizeQuota": 2048
    }
}
```

Weitere Informationen zu Entitätsbeschränkungen finden Sie unter IAM- und AWS STS-Kontingente im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetAccountSummaryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über die aktuelle Nutzung der IAM-Entität und die aktuellen Kontingente der IAM-Entität im AWS-Konto zurückgegeben.

```
Get-IAMAccountSummary
```

Ausgabe:

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
```

```
RolePolicySizeQuota                          10240
UserPolicySizeQuota                          2048
GroupsPerUserQuota                           10
AssumeRolePolicySizeQuota                    2048
AttachedPoliciesPerGroupQuota                2
Roles                                        9
VersionsPerPolicyQuota                       5
GroupsQuota                                  100
PolicySizeQuota                              5120
Policies                                     5
RolesQuota                                   250
ServerCertificates                           0
AttachedPoliciesPerRoleQuota                 2
MFADevicesInUse                              2
PoliciesQuota                                1000
AccountMFAEnabled                            1
Providers                                    2
InstanceProfilesQuota                        100
MFADevices                                   4
AccessKeysPerUserQuota                       2
AttachedPoliciesPerUserQuota                 2
SigningCertificatesPerUserQuota              2
PolicyVersionsInUse                          4
InstanceProfiles                             1
...
```

- Einzelheiten zur API finden Sie unter GetAccountSummary AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über die aktuelle Nutzung der IAM-Entität und die aktuellen Kontingente der IAM-Entität im AWS-Konto zurückgegeben.

```
Get-IAMAccountSummary
```

Ausgabe:

```

Key                                          Value
Users                                        7
GroupPolicySizeQuota                         5120
PolicyVersionsInUseQuota                     10000
```

```
ServerCertificatesQuota                     20
AccountSigningCertificatesPresent           0
AccountAccessKeysPresent                    0
Groups                                      3
UsersQuota                                  5000
RolePolicySizeQuota                         10240
UserPolicySizeQuota                         2048
GroupsPerUserQuota                          10
AssumeRolePolicySizeQuota                   2048
AttachedPoliciesPerGroupQuota               2
Roles                                       9
VersionsPerPolicyQuota                      5
GroupsQuota                                 100
PolicySizeQuota                             5120
Policies                                    5
RolesQuota                                  250
ServerCertificates                          0
AttachedPoliciesPerRoleQuota                2
MFADevicesInUse                             2
PoliciesQuota                               1000
AccountMFAEnabled                           1
Providers                                   2
InstanceProfilesQuota                       100
MFADevices                                  4
AccessKeysPerUserQuota                      2
AttachedPoliciesPerUserQuota                2
SigningCertificatesPerUserQuota             2
PolicyVersionsInUse                         4
InstanceProfiles                            1
...
```

- Einzelheiten zur API finden Sie unter GetAccountSummary AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- Einzelheiten zur API finden Sie GetAccountSummaryin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getaccountsummary( ).
        MESSAGE 'Retrieved account summary.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting account summary.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetAccountSummaryin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GetContextKeysForCustomPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetContextKeysForCustomPolicy` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie die Kontextschlüssel auf, auf die von einer oder mehreren benutzerdefinierten JSON-Richtlinien verwiesen wird, die als Parameter in der Befehlszeile bereitgestellt werden

Der folgende Befehl `get-context-keys-for-custom-policy` analysiert jede bereitgestellte Richtlinie und listet die von diesen Richtlinien verwendeten Kontextschlüssel auf. Verwenden Sie diesen Befehl, um zu ermitteln, welche Kontextschlüsselwerte Sie angeben müssen, um die Richtlinien-Simulatorbefehle `simulate-custom-policy` und `simulate-custom-policy` erfolgreich verwenden zu können. Sie können auch die Liste der Kontextschlüssel abrufen, die von allen Richtlinien verwendet werden, die einem IAM-Benutzer oder einer IAM-Rolle zugeordnet sind, indem Sie den Befehl `get-context-keys-for-custom-policy` verwenden. Parameterwerte, die mit `file://` beginnen, weisen den Befehl an, die Datei zu lesen und deren Inhalt als Wert für den Parameter anstelle des Dateinamens selbst zu verwenden.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

Ausgabe:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

Beispiel 2: So listen Sie die Kontextschlüssel auf, auf die von einer oder mehreren benutzerdefinierten JSON-Richtlinien verwiesen wird, die als Dateieingabe bereitgestellt werden

Der folgende Befehl `get-context-keys-for-custom-policy` ist derselbe wie im vorherigen Beispiel, mit der Ausnahme, dass die Richtlinien in einer Datei und nicht als Parameter bereitgestellt werden. Da der Befehl eine JSON-Liste mit Zeichenfolgen und keine Liste mit JSON-Strukturen erwartet, muss die Datei wie folgt strukturiert sein. Sie können sie jedoch zu einer einzigen Datei zusammenfassen.

```
[
    "Policy1",
    "Policy2"
]
```

Eine Datei, die die Richtlinie aus dem vorherigen Beispiel enthält, muss beispielsweise wie folgt aussehen. Sie müssen jedes eingebettete Anführungszeichen in der Zeichenfolge der Richtlinie durch Voranstellen eines umgekehrten Schrägstrichs " ersetzen.

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow
\", \"Action\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-
west-2:128716708097:table/${aws:username}\", \"Condition\": {\"DateGreaterThan\":
 {\"aws:CurrentTime\": \"2015-08-16T12:00:00Z\"}}}}" ]
```

Diese Datei kann dann dem folgenden Befehl übergeben werden.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list file://policyfile.json
```

Ausgabe:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

Weitere Informationen finden Sie unter Verwenden des IAM-Richtliniensimulators (AWS CLI und AWS API) im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetContextKeysForCustomPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die im bereitgestellten Richtlinien-JSON vorhanden sind. Um mehrere Richtlinien bereitzustellen, können Sie eine durch Kommas getrennte Werteliste bereitstellen.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Einzelheiten zur API finden Sie unter GetContextKeysForCustomPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die im bereitgestellten Richtlinien-JSON vorhanden sind. Um mehrere Richtlinien bereitzustellen, können Sie eine durch Kommas getrennte Werteliste bereitstellen.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
```

```
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- Einzelheiten zur API finden Sie unter GetContextKeysForCustomPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetContextKeysForPrincipalPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetContextKeysForPrincipalPolicy` verwendet wird.

CLI

AWS CLI

So listen Sie die Kontextschlüssel auf, auf die alle mit einem IAM-Prinzipal verknüpften Richtlinien verweisen

Mit dem folgenden Befehl `get-context-keys-for-principal-policy` werden alle Richtlinien abgerufen, die dem Benutzer `saanvi` und allen Gruppen, denen er angehört, zugeordnet sind. Anschließend werden alle Richtlinien analysiert und die von diesen Richtlinien verwendeten Kontextschlüssel aufgelistet. Verwenden Sie diesen Befehl, um zu ermitteln, welche Kontextschlüsselwerte Sie angeben müssen, um die Befehle `simulate-custom-policy` und `simulate-principal-policy` erfolgreich zu verwenden. Mit dem Befehl `get-context-keys-for-custom-policy` können Sie auch die Liste der Kontextschlüssel abrufen, die von einer beliebigen JSON-Richtlinie verwendet werden.

```
aws iam get-context-keys-for-principal-policy \
   --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

Ausgabe:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
```

```
    ]
}
```

Weitere Informationen finden Sie unter Verwenden des IAM-Richtliniensimulators (AWS CLI und AWS API) im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetContextKeysForPrincipalPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die im bereitgestellten Richtlinien-JSON und in den an die IAM-Entität angehängten Richtlinien (user/role usw.) vorhanden sind. Für — PolicyInputList Sie können eine Liste mit mehreren Werten als durch Kommas getrennte Werte angeben.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Einzelheiten zur API finden Sie unter GetContextKeysForPrincipalPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die im bereitgestellten Richtlinien-JSON und in den an die IAM-Entität angehängten Richtlinien (user/role usw.) vorhanden sind. Für — PolicyInputList Sie können eine Liste mit mehreren Werten als durch Kommas getrennte Werte angeben.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

```
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- Einzelheiten zur API finden Sie unter GetContextKeysForPrincipalPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetCredentialReport`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetCredentialReport` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

CLI

AWS CLI

So rufen Sie einen Bericht zu Anmeldeinformationen ab

In diesem Beispiel wird der zurückgegebene Bericht geöffnet und als Array von Textzeilen an die Pipeline ausgegeben.

```
aws iam get-credential-report
```

Ausgabe:

```
{
    "GeneratedTime":  "2015-06-17T19:11:50Z",
    "ReportFormat": "text/csv"
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Abrufen von Berichten zu Anmeldeinformationen für Ihr AWS Konto.

- Einzelheiten zur API finden Sie unter GetCredentialReport AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Example 1: In diesem Beispiel wird der zurückgegebene Bericht geöffnet und als Array von Textzeilen an die Pipeline ausgegeben. Die erste Zeile ist die Kopfzeile mit durch Kommas getrennten Spaltennamen. Jede nachfolgende Zeile ist die Detailzeile für einen Benutzer, wobei jedes Feld durch Kommas getrennt ist. Bevor Sie den Bericht anzeigen können, müssen Sie ihn mit dem **`Request-IAMCredentialReport`** Cmdlet generieren. Um den Bericht als einzelne Zeichenfolge abzurufen, verwenden Sie **`-Raw`** anstelle von **`-AsTextArray`**. Der Alias **`-SplitLines`** wird auch für den **`-AsTextArray`** Switch akzeptiert. Die vollständige Liste der Spalten in der Ausgabe finden Sie in der Service-API-Referenz. Beachten Sie, dass Sie, wenn Sie **`-AsTextArray`** oder **`-SplitLines`** nicht verwenden, Sie den Text mithilfe der Klasse .NET **`StreamReader`** aus der **`.Content`**-Eigenschaft extrahieren müssen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Ausgabe:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Einzelheiten zur API finden Sie unter GetCredentialReport AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Example 1: In diesem Beispiel wird der zurückgegebene Bericht geöffnet und als Array von Textzeilen an die Pipeline ausgegeben. Die erste Zeile ist die Kopfzeile mit durch Kommas getrennten Spaltennamen. Jede nachfolgende Zeile ist die Detailzeile für einen Benutzer, wobei jedes Feld durch Kommas getrennt ist. Bevor Sie den Bericht anzeigen können, müssen Sie ihn mit dem **`Request-IAMCredentialReport`** Cmdlet generieren. Um den Bericht als einzelne Zeichenfolge abzurufen, verwenden Sie **`-Raw`** anstelle von **`-AsTextArray`**. Der Alias **`-SplitLines`** wird auch für den **`-AsTextArray`** Switch akzeptiert. Die vollständige Liste der Spalten in der Ausgabe finden Sie in der Service-API-Referenz. Beachten Sie, dass Sie, wenn Sie **`-AsTextArray`** oder **`-SplitLines`** nicht verwenden, Sie den Text mithilfe der Klasse .NET **`StreamReader`** aus der **`.Content`**-Eigenschaft extrahieren müssen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Ausgabe:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- Einzelheiten zur API finden Sie unter GetCredentialReport AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- Einzelheiten zur API finden Sie GetCredentialReportin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getcredentialreport( ).
        MESSAGE 'Retrieved credential report.' TYPE 'I'.
      CATCH /aws1/cx_iamcredrptnotpresen00.
        MESSAGE 'Credential report not present.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptexpiredex.
        MESSAGE 'Credential report expired.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptnotreadyex.
        MESSAGE 'Credential report not ready.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting credential report.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetCredentialReportin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GetGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetGroup` verwendet wird.

CLI

AWS CLI

So rufen Sie eine IAM-Gruppe ab

In diesem Beispiel werden Details über die IAM-Gruppe `Admins` zurückgegeben.

```
aws iam get-group \
    --group-name Admins
```

Ausgabe:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-06-16T19:41:48Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    },
```

```
    "Users": []
}
```

Weitere Informationen finden Sie unter IAM-Identitäten (Benutzer, Benutzergruppen und Rollen) im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zur IAM-Gruppe **`Testers`** zurückgegeben, einschließlich einer Sammlung aller IAM-Benutzer, die zur Gruppe gehören.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Ausgabe:

```
Group                                     IsTruncated           Marker
     Users
-----                                     -----------           ------
     -----
Amazon.IdentityManagement.Model.Group     False
     {Theresa, David}
```

```
$results.Group
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Einzelheiten zur API finden Sie unter GetGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zur IAM-Gruppe **`Testers`** zurückgegeben, einschließlich einer Sammlung aller IAM-Benutzer, die zur Gruppe gehören.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Ausgabe:

```
Group                                     IsTruncated           Marker
     Users
-----                                     -----------           ------
     -----
Amazon.IdentityManagement.Model.Group     False
     {Theresa, David}
```

```
$results.Group
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
```

```
Path       : /
```

```
$results.Users
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- Einzelheiten zur API finden Sie unter GetGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetGroupPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetGroupPolicy` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zu einer Richtlinie, die einer IAM-Gruppe zugeordnet ist

Mit dem folgenden Befehl `get-group-policy` werden Informationen über die angegebene Richtlinie abgerufen, die der genannten Gruppe `Test-Group` zugeordnet ist.

```
aws iam get-group-policy \
    --group-name Test-Group \
    --policy-name S3-ReadOnly-Policy
```

Ausgabe:

```
{
    "GroupName": "Test-Group",
    "PolicyDocument": {
        "Statement": [
            {
                "Action": [
                    "s3:Get*",
                    "s3:List*"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    },
    "PolicyName": "S3-ReadOnly-Policy"
}
```

Weitere Informationen finden Sie unter Verwalten von IAM-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetGroupPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zur eingebetteten Inline-Richtlinie mit dem Namen **`PowerUserAccess-Testers`** für die Gruppe **`Testers`** zurückgegeben. Die **`PolicyDocument`**-Eigenschaft ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

Ausgabe:

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
```

```
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zur eingebetteten Inline-Richtlinie mit dem Namen **`PowerUserAccess-Testers`** für die Gruppe **`Testers`** zurückgegeben. Die **`PolicyDocument`**-Eigenschaft ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

Ausgabe:

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetInstanceProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetInstanceProfile` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit verwaltetem Streaming

CLI

AWS CLI

So rufen Sie Informationen zu einem Instance-Profil ab

Der folgende Befehl `get-instance-profile` ruft Informationen über das Instance-Profil mit dem Namen `ExampleInstanceProfile` ab.

```
aws iam get-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

Ausgabe:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
        "Roles": [
            {
                "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                "RoleId": "AIDGPMS9RO4H3FEXAMPLE",
                "CreateDate": "2013-01-09T06:33:26Z",
                "RoleName": "Test-Role",
                "Path": "/",
                "Arn": "arn:aws:iam::336924118301:role/Test-Role"
            }
        ],
        "CreateDate": "2013-06-12T23:52:02Z",
        "InstanceProfileName": "ExampleInstanceProfile",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
    }
}
```

Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetInstanceProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details des genannten Instanzprofils zurückgegeben**`ec2instancerole`**, das im aktuellen AWS Konto definiert ist.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
```

```
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter GetInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details des genannten Instanzprofils zurückgegeben**`ec2instancerole`**, das im aktuellen AWS Konto definiert ist.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter GetInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetLoginProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetLoginProfile` verwendet wird.

CLI

AWS CLI

So rufen Sie Passwortinformationen für einen IAM-Benutzer ab

Mit dem folgenden Befehl `get-login-profile` werden Informationen über das Passwort für den IAM-Benutzer mit dem Namen Bob abgerufen.

```
aws iam get-login-profile \
    --user-name Bob
```

Ausgabe:

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2012-09-21T23:03:39Z"
    }
}
```

Der Befehl `get-login-profile` kann verwendet werden, um zu überprüfen, ob ein IAM-Benutzer über ein Passwort verfügt. Der Befehl gibt einen `NoSuchEntity`-Fehler zurück, wenn für den Benutzer kein Passwort definiert ist.

Mit diesem Befehl können Sie kein Passwort anzeigen. Bei Verlust des Passwortes besteht die Möglichkeit, das Passwort (`update-login-profile`) für den Benutzer zurückzusetzen. Alternativ können Sie das Anmeldeprofil (`delete-login-profile`) für den Benutzer löschen und anschließend ein neues erstellen (`create-login-profile`).

Weitere Informationen finden Sie unter Verwalten von Passwörtern für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetLoginProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Erstellungsdatum des Passworts zurückgegeben und ob für den IAM-Benutzer **`David`** ein Passwortzurücksetzung erforderlich ist.

```
Get-IAMLoginProfile -UserName David
```

Ausgabe:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- Einzelheiten zur API finden Sie unter GetLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Erstellungsdatum des Passworts zurückgegeben und ob für den IAM-Benutzer **`David`** ein Passwortzurücksetzung erforderlich ist.

```
Get-IAMLoginProfile -UserName David
```

Ausgabe:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- Einzelheiten zur API finden Sie unter GetLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetOpenIdConnectProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetOpenIdConnectProvider` verwendet wird.

CLI

AWS CLI

So geben Sie Informationen zum angegebenen OpenID Connect-Anbieter zurück

In diesem Beispiel werden Details zum OpenID Connect-Anbieter zurückgegeben, dessen ARN `arn:aws:iam::123456789012:oidc-provider/server.example.com` lautet.

```
aws iam get-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

Ausgabe:

```
{
    "Url": "server.example.com"
        "CreateDate": "2015-06-16T19:41:48Z",
        "ThumbprintList": [
        "12345abcdefghijk67890lmnopqrst987example"
```

```
        ],
        "ClientIDList": [
        "example-application-ID"
        ]
}
```

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetOpenIdConnectProviderin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zum OpenID-Connect-Anbieter zurückgegeben, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`** lautet. Die **`ClientIDList`**-Eigenschaft ist eine Sammlung, die alle für diesen Anbieter definierten Client-IDs enthält.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Ausgabe:

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Einzelheiten zur API finden Sie unter GetOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zum OpenID-Connect-Anbieter zurückgegeben, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`** lautet. Die **`ClientIDList`**-Eigenschaft ist eine Sammlung, die alle für diesen Anbieter definierten Client-IDs enthält.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Ausgabe:

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- Einzelheiten zur API finden Sie unter GetOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetPolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetPolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit verwaltetem Streaming
- Arbeiten mit der IAM-Policy-Builder-API

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
```

```
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                            const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
```

```
                    policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                    << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen über die angegebene verwaltete Richtlinie ab

In diesem Beispiel werden Details über die verwaltete Richtlinie mit dem ARN `arn:aws:iam::123456789012:policy/MySamplePolicy` zurückgegeben.

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
{
    "Policy": {
        "PolicyName": "MySamplePolicy",
        "CreateDate": "2015-06-17T19:23;32Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
        "UpdateDate": "2015-06-17T19:23:32Z"
    }
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetPolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
```

```
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Richtlinie ab.

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- Einzelheiten zur API finden Sie GetPolicyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zur verwalteten Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Einzelheiten zur API finden Sie unter GetPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zur verwalteten Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
```

```
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- Einzelheiten zur API finden Sie unter GetPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- Einzelheiten zur API finden Sie GetPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Fetches an IAM policy by its ARN
 # @param policy_arn [String] the ARN of the IAM policy to retrieve
 # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
 def get_policy(policy_arn)
   response = @iam_client.get_policy(policy_arn: policy_arn)
   policy = response.policy
   @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
   policy
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
   raise
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
   raise
 end
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getpolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: getPolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie GetPolicyin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetPolicyVersion`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetPolicyVersion` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten von Richtlinien
- Arbeiten mit der IAM-Policy-Builder-API

CLI

AWS CLI

So rufen Sie Informationen über die angegebene Version der angegebenen verwalteten Richtlinie ab

In diesem Beispiel wird das Richtliniendokument für die Version v2 der Richtlinie zurückgegeben, deren ARN `arn:aws:iam::123456789012:policy/MyManagedPolicy` lautet.

```
aws iam get-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Ausgabe:

```
{
    "PolicyVersion": {
        "Document": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "iam:*",
                    "Resource": "*"
```

```
                }
            ]
        },
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2023-04-11T00:22:54+00:00"
    }
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetPolicyVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt das Richtliniendokument für die **v2**-Version der Richtlinie zurück, deren ARN **arn:aws:iam::123456789012:policy/MyManagedPolicy** lautet. Das Richtliniendokument in der **Document**-Eigenschaft ist URL-codiert und wird in diesem Beispiel mit der .NET-Methode **UrlDecode** decodiert.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Ausgabe:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
```

```
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- Einzelheiten zur API finden Sie unter GetPolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt das Richtliniendokument für die **v2**-Version der Richtlinie zurück, deren ARN **arn:aws:iam::123456789012:policy/MyManagedPolicy** lautet. Das Richtliniendokument in der **Document**-Eigenschaft ist URL-codiert und wird in diesem Beispiel mit der .NET-Methode **UrlDecode** decodiert.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Ausgabe:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": [
```

```
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- Einzelheiten zur API finden Sie unter GetPolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- Einzelheiten zur API finden Sie GetPolicyVersionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->getpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Retrieved policy version information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy or version does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iaminvalidinputex.
    MESSAGE 'Invalid input provided.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetPolicyVersionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetRole`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetRole` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Konfigurieren Sie die Konnektivität des Containerdienstes

- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Einen Containerdienst für VM-Instanzen erstellen
- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });

    return response.Role;
}
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen über eine IAM-Rolle ab

Mit dem folgenden `get-role`-Befehl werden Informationen über die Rolle mit dem Namen `Test-Role` abgerufen.

```
aws iam get-role \
    --role-name Test-Role
```

Ausgabe:

```
{
    "Role": {
        "Description": "Test Role",
        "AssumeRolePolicyDocument":"<URL-encoded-JSON>",
        "MaxSessionDuration": 3600,
        "RoleId": "AROA1234567890EXAMPLE",
        "CreateDate": "2019-11-13T16:45:56Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "us-east-1",
            "LastUsedDate": "2019-11-13T17:14:00Z"
        },
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

Der Befehl zeigt die Vertrauensrichtlinie an, die der Rolle zugeordnet ist. Verwenden Sie den `list-role-policies`-Befehl, um die einer Rolle zugeordneten Berechtigungsrichtlinien aufzulisten.

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetRolein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
```

```
}
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Rolle ab.

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt die Details von **`lamda_exec_role`** zurück. Es enthält das Vertrauensrichtlinien-Dokument, das angibt, wer diese Rolle übernehmen kann. Das Richtliniendokument ist URL-codiert und kann mit der .NET-Methode **`UrlDecode`**decodiert werden. In diesem Beispiel wurden aus der ursprünglichen Richtlinie sämtliche Leerzeichen entfernt, bevor sie in die Richtlinie hochgeladen wurden. Um die Berechtigungsrichtliniendokumente anzuzeigen, die bestimmen, was jemand, der die Rolle übernimmt, tun kann, verwenden Sie **`Get-IAMRolePolicy`** für Inline-Richtlinien und **`Get-IAMPolicyVersion`** für angefügte verwaltete Richtlinien.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Ausgabe:

```
Arn                      : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
RoleName                 : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Einzelheiten zur API finden Sie unter GetRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt die Details von **`lamda_exec_role`** zurück. Es enthält das Vertrauensrichtlinien-Dokument, das angibt, wer diese Rolle übernehmen kann. Das Richtliniendokument ist URL-codiert und kann mit der .NET-Methode **`UrlDecode`**decodiert werden. In diesem Beispiel wurden aus der ursprünglichen Richtlinie sämtliche Leerzeichen entfernt, bevor sie in die Richtlinie hochgeladen wurden. Um die Berechtigungsrichtliniendokumente anzuzeigen, die bestimmen, was jemand, der die Rolle übernimmt, tun kann, verwenden Sie **`Get-IAMRolePolicy`** für Inline-Richtlinien und **`Get-IAMPolicyVersion`** für angefügte verwaltete Richtlinien.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Ausgabe:

```
Arn                      : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
RoleName                 : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- Einzelheiten zur API finden Sie unter GetRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_role(role_name):
    """
    Gets a role by name.
```

```
    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- Einzelheiten zur API finden Sie GetRolein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                                role_name: name
                              }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
```

```
    role
  end
```

- Einzelheiten zur API finden Sie GetRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- Einzelheiten zur API finden Sie GetRolein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_iam->getrole( iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved role information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetRolein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: getRole:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie GetRolein der API-Referenz zum AWS SDK für Swift.

Verwendung von **`GetRolePolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetRolePolicy` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zu einer Richtlinie, die einer IAM-Rolle zugeordnet ist

Mit dem folgenden Befehl `get-role-policy` werden Informationen über die angegebene Richtlinie abgerufen, die der Rolle mit dem Namen `Test-Role` zugeordnet ist.

```
aws iam get-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

Ausgabe:

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
      "Statement": [
          {
              "Action": [
                  "s3:ListBucket",
                  "s3:Put*",
                  "s3:Get*",
                  "s3:*MultipartUpload*"
              ],
              "Resource": "*",
              "Effect": "Allow",
              "Sid": "1"
          }
      ]
  }
  "PolicyName": "ExamplePolicy"
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetRolePolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Berechtigungs-Richtliniendokument für die Richtlinie mit dem Namen **`oneClick_lambda_exec_role_policy`** zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet ist. Das resultierende Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Ausgabe:

```
PolicyDocument                                           PolicyName
              UserName
--------------                                           ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Ausgabe:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Berechtigungs-Richtliniendokument für die Richtlinie mit dem Namen **`oneClick_lambda_exec_role_policy`** zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet ist. Das resultierende Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Ausgabe:

```
PolicyDocument                                          PolicyName
              UserName
--------------                                          ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Ausgabe:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetSamlProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetSamlProvider` verwendet wird.

CLI

AWS CLI

So rufen Sie das Metadokument des SAML-Anbieters ab

In diesem Beispiel werden die Details zum SAML 2.0-Anbieter abgerufen, dessen ARM `arn:aws:iam::123456789012:saml-provider/SAMLADFS` lautet. Die Antwort enthält das Metadatendokument, das Sie vom Identitätsanbieter zur Erstellung der AWS SAML-Provider-Entität erhalten haben, sowie die Erstellungs- und Ablaufdaten.

```
aws iam get-saml-provider \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

Ausgabe:

```
{
    "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",
    "CreateDate": "2017-03-06T22:29:46+00:00",
    "ValidUntil": "2117-03-06T22:29:46.433000+00:00",
    "Tags": [
        {
            "Key": "DeptID",
            "Value": "123456"
        },
        {
            "Key": "Department",
            "Value": "Accounting"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-SAML-Identitätsanbietern im AWS - IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetSamlProviderin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details zum SAML 2.0-Anbieter abgerufen, dessen ARM arn:aws:iam: :123456789012:saml- lautet. provider/SAMLADFS Die Antwort enthält das Metadatendokument, das Sie vom Identitätsanbieter zur Erstellung der SAML-Provider-Entität erhalten haben, sowie die Erstellungs- und Ablaufdaten. AWS

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Ausgabe:

```
CreateDate                 SAMLMetadataDocument
        ValidUntil
----------                 --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- Einzelheiten zur API finden Sie unter GetSamlProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details über den SAML 2.0-Anbieter abgerufen, dessen ARM arn:aws:iam: :123456789012:saml- lautet. provider/SAMLADFS Die Antwort enthält das Metadatendokument, das Sie vom Identitätsanbieter zur Erstellung der SAML-Provider-Entität erhalten haben, sowie die Erstellungs- und Ablaufdaten. AWS

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Ausgabe:

```
CreateDate                 SAMLMetadataDocument
        ValidUntil
----------                 --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- Einzelheiten zur API finden Sie unter GetSamlProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetServerCertificate`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetServerCertificate` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                 << std::endl << "Body: " << certificate.GetCertificateBody() <<
                 std::endl << "Chain: " << certificate.GetCertificateChain() <<
                 std::endl;
    }

    return result;
}
```

- Einzelheiten zur API finden Sie GetServerCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um Details zu einem Serverzertifikat in Ihrem AWS Konto zu erhalten

Mit dem folgenden `get-server-certificate` Befehl werden alle Details zum angegebenen Serverzertifikat in Ihrem AWS Konto abgerufen.

```
aws iam get-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

Ausgabe:


```
{
    "ServerCertificate": {
        "ServerCertificateMetadata": {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        "CertificateBody": "-----BEGIN CERTIFICATE-----
            MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
            VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
            b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
            BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
            MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
            VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
            b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
            rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
            Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
            nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
            FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
            NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
        "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6md
            7oRw0uXOjANBgkqhkiG9w0BAqQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgT
            AldBMRAwDgYDVQQHEwdTZWF0drGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAs
            TC0lBTSBDb25zb2xlMRIwEAYDVsQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQ
            jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
            MCVVMxCzAJBgNVBAgTAldBMRAwDgsYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
            WF6b24xFDASBgNVBAsTC0lBTSBDb2d5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
            HzAdBgkqhkiG9w0BCQEWEG5vb25lQGfFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
            BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
```

```
            k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
            ITxOUSQv7c7ugFFDzQGBzZswY6786m86gjpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
            AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUhVVxYUntneD9+h8Mg9q6q+auN
            KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FlkbFFBjvSfpJIlJ00zbhNYS5f6Guo
            EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
            3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
    }
}
```

Verwenden Sie den `list-server-certificates` Befehl, um die in Ihrem AWS Konto verfügbaren Serverzertifikate aufzulisten.

Weitere Informationen finden Sie unter Verwaltung von Serverzertifikaten in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetServerCertificatein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Serverzertifikat ab.

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
```

```
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetServerCertificatein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetServerCertificatein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zum Serverzertifikat mit dem Namen **`MyServerCertificate`** abgerufen. Sie finden die Zertifikatsdetails in den Eigenschaften **`CertificateBody`** und **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Ausgabe:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
```

```
 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter GetServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zum Serverzertifikat mit dem Namen **`MyServerCertificate`** abgerufen. Sie finden die Zertifikatsdetails in den Eigenschaften **`CertificateBody`** und **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Ausgabe:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
```

```
 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter GetServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetServiceLastAccessedDetails`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetServiceLastAccessedDetails` verwendet wird.

CLI

AWS CLI

So rufen Sie einen Service-Zugriffsbericht ab

Im folgenden Beispiel `get-service-last-accessed-details` wird ein zuvor erstellter Bericht abgerufen, der die Services auflistet, auf die IAM-Entitäten zugreifen. Verwenden Sie den `generate-service-last-accessed-details`-Befehl, um einen Bericht zu generieren.

```
aws iam get-service-last-accessed-details \
    --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

Ausgabe:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:50:35.929Z",
    "ServicesLastAccessed": [
        ...
        {
            "ServiceName": "AWS Lambda",
            "LastAuthenticated": "2019-09-30T23:02:00Z",
            "ServiceNamespace": "lambda",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
            "TotalAuthenticatedEntities": 6
        },
    ]
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Verfeinerung von Berechtigungen bei der AWS Verwendung von Informationen, auf die zuletzt zugegriffen wurde.

- Einzelheiten zur API finden Sie GetServiceLastAccessedDetailsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zum Service bereitgestellt, auf den die im Anforderungsaufruf zugeordnete IAM-Entität (Benutzer, Gruppe, Rolle oder Richtlinie) zuletzt zugegriffen hat.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Ausgabe:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetails AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zum Service bereitgestellt, auf den die im Anforderungsaufruf zugeordnete IAM-Entität (Benutzer, Gruppe, Rolle oder Richtlinie) zuletzt zugegriffen hat.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Ausgabe:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetails AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetServiceLastAccessedDetailsWithEntities`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetServiceLastAccessedDetailsWithEntities` verwendet wird.

CLI

AWS CLI

So rufen Sie einen Service-Zugriffsbericht mit Details zu einem Service ab

Im folgenden Beispiel `get-service-last-accessed-details-with-entities` wird ein Bericht abgerufen, der Details zu IAM-Benutzern und anderen Entitäten enthält, die auf den angegebenen Service zugegriffen haben. Verwenden Sie den `generate-service-last-accessed-details`-Befehl, um einen Bericht zu generieren. Verwenden Sie `get-service-last-accessed-details`, um eine Liste der Services abzurufen, auf die über Namespaces zugegriffen wird.

```
aws iam get-service-last-accessed-details-with-entities \
    --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \
    --service-namespace lambda
```

Ausgabe:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:55:41.756Z",
    "JobCompletionDate": "2019-10-01T03:55:42.533Z",
    "EntityDetailsList": [
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/admin",
                "Name": "admin",
                "Type": "USER",
                "Id": "AIDAIO2XMPLENQEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-30T23:02:00Z"
        },
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/developer",
                "Name": "developer",
                "Type": "USER",
                "Id": "AIDAIBEYXMPL2YEXAMPLE",
                "Path": "/"
            },
```

```
            "LastAuthenticated": "2019-09-16T19:34:00Z"
        }
    ]
}
```

Weitere Informationen finden Sie im AWS IAM-Benutzerhandbuch unter Verfeinerung von Berechtigungen bei der AWS Verwendung von Informationen, auf die zuletzt zugegriffen wurde.

- Einzelheiten zur API finden Sie GetServiceLastAccessedDetailsWithEntitiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel stellt den Zeitstempel des letzten Zugriffs für den Service in der Anfrage durch die entsprechende IAM-Entität bereit.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Ausgabe:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Ausgabe:

```
EntityInfo                                LastAuthenticated
----------                                -----------------
```

```
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Ausgabe:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetailsWithEntities AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel stellt den Zeitstempel des letzten Zugriffs für den Service in der Anfrage durch die entsprechende IAM-Entität bereit.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Ausgabe:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Ausgabe:

```
EntityInfo                                    LastAuthenticated
```

```
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Ausgabe:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- Einzelheiten zur API finden Sie unter GetServiceLastAccessedDetailsWithEntities AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetServiceLinkedRoleDeletionStatus`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetServiceLinkedRoleDeletionStatus` verwendet wird.

CLI

AWS CLI

So überprüfen Sie den Status einer Anfrage zum Löschen einer serviceverknüpften Rolle

Im folgenden `get-service-linked-role-deletion-status`-Beispiel wird der Status einer früheren Anfrage zum Löschen einer serviceverknüpften Rolle angezeigt. Der Löschvorgang erfolgt asynchron. Wenn Sie die Anfrage stellen, erhalten Sie einen `DeletionTaskId`-Wert, den Sie als Parameter für diesen Befehl angeben.

```
aws iam get-service-linked-role-deletion-status \
    --deletion-task-id task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

Ausgabe:

```
{
```

```
"Status": "SUCCEEDED"
}
```

Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im AWS - IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetServiceLinkedRoleDeletionStatusin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetServiceLinkedRoleDeletionStatusin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetUser`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetUser` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }
```

- Einzelheiten zur API finden Sie GetUserin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi
```

```
    return 1 # 1 in Bash script means false.
  fi
}
```

- Einzelheiten zur API finden Sie GetUserin der AWS CLI Befehlsreferenz.

CLI

AWS CLI

So rufen Sie Informationen über einen IAM-Benutzer ab

Mit dem folgenden `get-user`-Befehl werden Informationen über den IAM-Benutzer mit dem Namen `Paulo` abgerufen.

```
aws iam get-user \
    --user-name Paulo
```

Ausgabe:

```
{
    "User": {
        "UserName": "Paulo",
        "Path": "/",
        "CreateDate": "2019-09-21T23:03:13Z",
        "UserId": "AIDA123456789EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Paulo"
    }
}
```

Weitere Informationen finden Sie unter Verwalten von IAM-Benutzern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetUserin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
```

```
    case *types.NoSuchEntityException:
     log.Printf("User %v does not exist.\n", userName)
     err = nil
    default:
     log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
    }
   }
  } else {
   user = result.User
  }
  return user, err
}
```

- Einzelheiten zur API finden Sie GetUserin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zum Benutzer mit dem Namen **`David`** abgerufen.

```
Get-IAMUser -UserName David
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Beispiel 2: In diesem Beispiel werden Details zum aktuell angemeldeten IAM-Benutzer abgerufen.

```
Get-IAMUser
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- Einzelheiten zur API finden Sie unter GetUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zum Benutzer mit dem Namen **`David`** abgerufen.

```
Get-IAMUser -UserName David
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

Beispiel 2: In diesem Beispiel werden Details zum aktuell angemeldeten IAM-Benutzer abgerufen.

```
Get-IAMUser
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- Einzelheiten zur API finden Sie unter GetUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- Einzelheiten zur API finden Sie GetUserin der AWS SDK für Ruby API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
```

```
import AWSS3


    public func getUser(name: String? = nil) async throws -> IAMClientTypes.User
 {
        let input = GetUserInput(
            userName: name
        )
        do {
            let output = try await iamClient.getUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            return user
        } catch {
            print("ERROR: getUser:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie GetUserin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`GetUserPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetUserPolicy` verwendet wird.

CLI

AWS CLI

So listen Sie Richtliniendetails für einen IAM-Benutzer auf

Der folgende Befehl `get-user-policy` listet die Details der angegebenen Richtlinie auf, die dem IAM-Benutzer mit dem Namen Bob angefügt ist.

```
aws iam get-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

Ausgabe:

```
{
    "UserName": "Bob",
    "PolicyName": "ExamplePolicy",
    "PolicyDocument": {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Action": "*",
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    }
}
```

Verwenden Sie den `list-user-policies`-Befehl, um eine Liste der Richtlinien für einen IAM-Benutzer abzurufen.

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetUserPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details der Inline-Richtlinie mit dem Namen **`Davids_IAM_Admin_Policy`** abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet ist. Das Richtliniendokument ist URL-codiert.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Ausgabe:

```
PolicyDocument                                          PolicyName
       UserName
--------------                                          ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details der Inline-Richtlinie mit dem Namen **`Davids_IAM_Admin_Policy`** abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet ist. Das Richtliniendokument ist URL-codiert.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Ausgabe:

```
PolicyDocument                                          PolicyName
       UserName
--------------                                          ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter GetUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListAccessKeys`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAccessKeys` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffsschlüsseln

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
 the specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
```

```
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie die Zugriffsschlüssel-IDs für einen IAM-Benutzer auf

Der folgende `list-access-keys`-Befehl listet die Zugriffsschlüssel-IDs für den IAM-Benutzer mit dem Namen Bob auf.

```
aws iam list-access-keys \
    --user-name Bob
```

Ausgabe:

```
{
    "AccessKeyMetadata": [
        {
            "UserName": "Bob",
            "Status": "Active",
            "CreateDate": "2013-06-04T18:17:34Z",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
        },
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CreateDate": "2013-06-06T20:42:26Z",
            "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
        }
    ]
}
```

Sie können die geheimen Zugriffsschlüssel für IAM-Benutzer nicht auflisten. Bei Verlust der geheimen Zugangsschlüssel müssen Sie mit dem `create-access-keys`-Befehl neue Zugangsschlüssel erstellen.

Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
```

```
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();
```

```
                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Zugriffsschlüssel auf.

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
 undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
```

```
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
```

```
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Mit diesem Befehl werden die Zugriffsschlüssel für den IAM-Benutzer namens **Bob** aufgelistet. Beachten Sie, dass Sie die geheimen Zugriffsschlüssel für IAM-Benutzer

nicht auflisten können. Bei Verlust der geheimen Zugriffsschlüssel müssen mit dem **New-IAMAccessKey** Cmdlet neue Zugriffsschlüssel erstellt werden.

```
Get-IAMAccessKey -UserName "Bob"
```

Ausgabe:

```
AccessKeyId                CreateDate                  Status
 UserName
-----------                ----------                  ------
 --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- Einzelheiten zur API finden Sie unter ListAccessKeys AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Mit diesem Befehl werden die Zugriffsschlüssel für den IAM-Benutzer namens **Bob** aufgelistet. Beachten Sie, dass Sie die geheimen Zugriffsschlüssel für IAM-Benutzer nicht auflisten können. Bei Verlust der geheimen Zugriffsschlüssel müssen mit dem **New-IAMAccessKey** Cmdlet neue Zugriffsschlüssel erstellt werden.

```
Get-IAMAccessKey -UserName "Bob"
```

Ausgabe:

```
AccessKeyId                CreateDate                  Status
 UserName
-----------                ----------                  ------
 --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- Einzelheiten zur API finden Sie unter ListAccessKeys AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- Einzelheiten zur API finden Sie ListAccessKeysin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
 more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
```

```
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listaccesskeys(
      iv_username = iv_user_name ).
    MESSAGE 'Retrieved access key list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListAccessKeysin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAccountAliases`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAccountAliases` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten Ihrer Konten

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie Konto-Aliasse auf

Der folgende `list-account-aliases`-Befehl listet die Aliasse für das aktuelle Konto auf.

```
aws iam list-account-aliases
```

Ausgabe:

```
{
    "AccountAliases": [
    "mycompany"
    ]
}
```

Weitere Informationen finden Sie unter Ihre AWS Konto-ID und deren Alias im AWS IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter ListAccountAliases AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Konto-Aliase auf.

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
```

```
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Dieser Befehl gibt den Kontoalias für das AWS-Konto zurück.

```
Get-IAMAccountAlias
```

Ausgabe:

```
ExampleCo
```

- Einzelheiten zur API finden Sie unter ListAccountAliases AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Kontoalias für das AWS-Konto zurück.

```
Get-IAMAccountAlias
```

Ausgabe:

```
ExampleCo
```

- Einzelheiten zur API finden Sie unter ListAccountAliases AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- Einzelheiten zur API finden Sie ListAccountAliasesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
```

```
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listaccountaliases( ).
        MESSAGE 'Retrieved account alias list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing account aliases.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`ListAttachedGroupPolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListAttachedGroupPolicies` verwendet wird.

CLI

AWS CLI

So listen Sie alle verwalteten Richtlinien auf, die der angegebenen Gruppe zugeordnet sind

In diesem Beispiel werden die Namen und ARNs der verwalteten Richtlinien zurückgegeben, die der `Admins` im Konto genannten IAM-Gruppe zugeordnet sind. AWS

```
aws iam list-attached-group-policies \
    --group-name Admins
```

Ausgabe:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter ListAttachedGroupPolicies AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der **`Admins`** im Konto genannten IAM-Gruppe zugeordnet sind. AWS Verwenden Sie den Befehl **`Get-IAMGroupPolicyList`**, um die Liste der in der Gruppe eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Ausgabe:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- Einzelheiten zur API finden Sie unter ListAttachedGroupPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der **`Admins`** im Konto genannten IAM-Gruppe zugeordnet sind. AWS Verwenden Sie den Befehl **`Get-IAMGroupPolicyList`**, um die Liste der in der Gruppe eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Ausgabe:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- Einzelheiten zur API finden Sie unter ListAttachedGroupPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListAttachedRolePolicies`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAttachedRolePolicies` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// List the IAM role policies that are attached to an IAM role.
   /// </summary>
   /// <param name="roleName">The IAM role to list IAM policies for.</param>
   /// <returns>A list of the IAM policies attached to the IAM role.</returns>
   public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
   {
       var attachedPolicies = new List<AttachedPolicyType>();
       var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

       await foreach (var response in attachedRolePoliciesPaginator.Responses)
       {
           attachedPolicies.AddRange(response.AttachedPolicies);
       }

       return attachedPolicies;
   }
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie alle verwalteten Richtlinien auf, die der angegebenen Rolle angefügt sind

Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der `SecurityAuditRole` im Konto genannten IAM-Rolle zugeordnet sind. AWS

```
aws iam list-attached-role-policies \
    --role-name SecurityAuditRole
```

Ausgabe:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Richtlinien auf, die an eine Rolle angefügt sind.

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
```

```
        }),
      );
    } else {
      break;
    }
  }
}
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker
 = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der IAM-Rolle mit dem Namen **SecurityAuditRole** im AWS -Konto zugeordnet sind. Verwenden Sie den **Get-IAMRolePolicyList**-Befehl, um die Liste der in die Rolle eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- Einzelheiten zur API finden Sie unter ListAttachedRolePolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der IAM-Rolle mit dem Namen **SecurityAuditRole** im AWS -Konto zugeordnet sind. Verwenden Sie den **Get-IAMRolePolicyList**-Befehl, um die Liste der in die Rolle eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- Einzelheiten zur API finden Sie unter ListAttachedRolePolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
 SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
```

```
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listattachedrolepolicies(
          iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved attached policy list for role.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3



    /// Returns a list of AWS Identity and Access Management (IAM) policies
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
 [IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []

        // Use "Paginated" to get all the attached role polices.
        // This lets the SDK handle the 'isTruncated' in
 "ListAttachedRolePoliciesOutput".
        let input = ListAttachedRolePoliciesInput(
            roleName: role
        )
        let output = client.listAttachedRolePoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let attachedPolicies = page.attachedPolicies else {
                    print("Error: no attached policies returned.")
                    continue
                }
                for attachedPolicy in attachedPolicies {
                    policyList.append(attachedPolicy)
```

```
                }
            }
        } catch {
            print("ERROR: listAttachedRolePolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Einzelheiten zur API finden Sie ListAttachedRolePoliciesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ListAttachedUserPolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListAttachedUserPolicies` verwendet wird.

CLI

AWS CLI

So listen Sie alle verwalteten Richtlinien auf, die dem angegebenen Benutzer zugeordnet sind

Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien für den Bob im Konto genannten IAM-Benutzer zurück. AWS

```
aws iam list-attached-user-policies \
    --user-name Bob
```

Ausgabe:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
```

```
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListAttachedUserPoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien für den **Bob** im Konto genannten IAM-Benutzer zurück. AWS Verwenden Sie den **Get-IAMUserPolicyList**-Befehl, um die Liste der im IAM-Benutzer eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/TesterPolicy                     TesterPolicy
```

- Einzelheiten zur API finden Sie unter ListAttachedUserPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien für den **Bob** im Konto genannten IAM-Benutzer zurück. AWS Verwenden Sie den **Get-IAMUserPolicyList**-Befehl, um die Liste der im IAM-Benutzer eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Ausgabe:

```
PolicyArn                                                   PolicyName
---------                                                   ----------
arn:aws:iam::aws:policy/TesterPolicy                        TesterPolicy
```

- Einzelheiten zur API finden Sie unter ListAttachedUserPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListEntitiesForPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListEntitiesForPolicy` verwendet wird.

CLI

AWS CLI

So listen Sie alle Benutzer, Gruppen und Rollen auf, denen die angegebene verwaltete Richtlinie zugeordnet ist

Dieses Beispiel gibt eine Liste mit IAM-Gruppen, Rollen und Benutzern zurück, denen die Richtlinie `arn:aws:iam::123456789012:policy/TestPolicy` zugeordnet ist.

```
aws iam list-entities-for-policy \
    --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

Ausgabe:

```
{
    "PolicyGroups": [
        {
            "GroupName": "Admins",
            "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyUsers": [
        {
            "UserName": "Alice",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyRoles": [
        {
```

```
            "RoleName": "DevRole",
            "RoleId": "AROADBQP57FF2AEXAMPLE"
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListEntitiesForPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt eine Liste der IAM-Gruppen, Rollen und Benutzer zurück, denen die Richtlinie **`arn:aws:iam::123456789012:policy/TestPolicy`** zugeordnet ist.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Ausgabe:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Einzelheiten zur API finden Sie unter ListEntitiesForPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt eine Liste der IAM-Gruppen, Rollen und Benutzer zurück, denen die Richtlinie **`arn:aws:iam::123456789012:policy/TestPolicy`** zugeordnet ist.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Ausgabe:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- Einzelheiten zur API finden Sie unter ListEntitiesForPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListGroupPolicies`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListGroupPolicies` verwendet wird.

CLI

AWS CLI

So listen Sie alle Inline-Richtlinien auf, die der angegebenen Gruppe zugeordnet sind

Der folgende `list-group-policies`-Befehl listet die Namen der Inline-Richtlinien auf, die der IAM-Gruppe mit dem Namen `Admins` im aktuellen Konto zugeordnet sind.

```
aws iam list-group-policies \
    --group-name Admins
```

Ausgabe:

```
{
    "PolicyNames": [
        "AdminRoot",
        "ExamplePolicy"
    ]
}
```

Weitere Informationen finden Sie unter Verwalten von IAM-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListGroupPoliciesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste der Inline-Richtlinien zurückgegeben, die in der Gruppe **`Testers`** eingebettet sind. Verwenden Sie den Befehl **`Get-IAMAttachedGroupPolicyList`**, um die verwalteten Richtlinien abzurufen, die der Gruppe zugeordnet sind.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Ausgabe:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Einzelheiten zur API finden Sie unter ListGroupPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste der Inline-Richtlinien zurückgegeben, die in der Gruppe **`Testers`** eingebettet sind. Verwenden Sie den Befehl **`Get-IAMAttachedGroupPolicyList`**, um die verwalteten Richtlinien abzurufen, die der Gruppe zugeordnet sind.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Ausgabe:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- Einzelheiten zur API finden Sie unter ListGroupPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListGroups`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListGroups` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die IAM-Gruppen für das aktuelle Konto auf

Der folgende `list-groups`-Befehl listet die IAM-Gruppen im aktuellen Konto auf.

```
aws iam list-groups
```

Ausgabe:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-06-04T20:27:27.972Z",
            "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "GroupName": "Admins"
        },
        {
            "Path": "/",
            "CreateDate": "2013-04-16T20:30:42Z",
            "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
            "GroupName": "S3-Admins"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwaltung von IAM-Benutzergruppen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListGroupsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
 actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
 ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(ctx, &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Gruppen auf.

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt eine Sammlung aller in der aktuellen AWS-Konto Version definierten IAM-Gruppen zurück.

```
Get-IAMGroupList
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Einzelheiten zur API finden Sie unter ListGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt eine Sammlung aller in der aktuellen Version definierten IAM-Gruppen zurück. AWS-Konto

```
Get-IAMGroupList
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
```

```
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- Einzelheiten zur API finden Sie unter ListGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- Einzelheiten zur API finden Sie ListGroupsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Einzelheiten zur API finden Sie ListGroupsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listgroups( ).
        MESSAGE 'Retrieved group list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing groups.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListGroupsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listGroups() async throws -> [String] {
        var groupList: [String] = []

        // Use "Paginated" to get all the groups.
        // This lets the SDK handle the 'isTruncated' property in
 "ListGroupsOutput".
        let input = ListGroupsInput()

        let pages = client.listGroupsPaginated(input: input)
        do {
            for try await page in pages {
                guard let groups = page.groups else {
                    print("Error: no groups returned.")
                    continue
                }
```

```
                for group in groups {
                    if let name = group.groupName {
                        groupList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listGroups:", dump(error))
            throw error
        }
        return groupList
    }
```

- Einzelheiten zur API finden Sie ListGroupsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ListGroupsForUser`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListGroupsForUser` verwendet wird.

CLI

AWS CLI

So listen Sie die Gruppen auf, zu denen ein IAM-Benutzer gehört

Der folgende Befehl `list-groups-for-user` zeigt die Gruppen an, zu denen der IAM-Benutzer mit dem Namen Bob angehört.

```
aws iam list-groups-for-user \
    --user-name Bob
```

Ausgabe:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:18:08Z",
            "GroupId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admin",
            "GroupName": "Admin"
```

```
        },
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:37:28Z",
            "GroupId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/s3-Users",
            "GroupName": "s3-Users"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwaltung von IAM-Benutzergruppen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListGroupsForUserin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der IAM-Gruppen zurückgegeben, zu denen der IAM-Benutzer **`David`** angehört.

```
Get-IAMGroupForUser -UserName David
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
```

```
GroupName  : Developers
Path       : /
```

- Einzelheiten zur API finden Sie unter ListGroupsForUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der IAM-Gruppen zurückgegeben, zu denen der IAM-Benutzer **`David`** angehört.

```
Get-IAMGroupForUser -UserName David
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- Einzelheiten zur API finden Sie unter ListGroupsForUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListInstanceProfiles`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListInstanceProfiles` verwendet wird.

CLI

AWS CLI

So listen Sie die Instance-Profile für das Konto auf

Im folgenden Befehl `list-instance-profiles` werden die Instance-Profile aufgelistet, die dem aktuellen Konto zugeordnet sind.

```
aws iam list-instance-profiles
```

Ausgabe:

```
{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "example-dev-role",
            "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
            "CreateDate": "2023-09-21T18:17:41+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-dev-role",
                    "RoleId": "AROAJ52OTH4H7LEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
                    "CreateDate": "2023-09-21T18:17:40+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
```

```
        {
            "Path": "/",
            "InstanceProfileName": "example-s3-role",
            "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
            "CreateDate": "2023-09-21T18:18:50+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-s3-role",
                    "RoleId": "AROAINUBC5O7XLEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
                    "CreateDate": "2023-09-21T18:18:49+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListInstanceProfilesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt eine Sammlung der Instanzprofile zurück, die in der aktuellen Version definiert sind AWS-Konto.

```
Get-IAMInstanceProfileList
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfiles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt eine Sammlung der in der aktuellen AWS-Konto Version definierten Instanzprofile zurück.

```
Get-IAMInstanceProfileList
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfiles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListInstanceProfilesForRole`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListInstanceProfilesForRole` verwendet wird.

CLI

AWS CLI

So listen Sie die Instance-Profile für eine IAM-Rolle auf

Mit dem folgenden Befehl `list-instance-profiles-for-role` werden die Instance-Profile aufgelistet, die der Rolle `Test-Role` zugeordnet sind.

```
aws iam list-instance-profiles-for-role \
    --role-name Test-Role
```

Ausgabe:

```
{
    "InstanceProfiles": [
        {
            "InstanceProfileId": "AIDGPMS9RO4H3FEXAMPLE",
            "Roles": [
                {
                    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
                    "CreateDate": "2013-06-07T20:42:15Z",
                    "RoleName": "Test-Role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:role/Test-Role"
                }
            ],
            "CreateDate": "2013-06-07T21:05:24Z",
            "InstanceProfileName": "ExampleInstanceProfile",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListInstanceProfilesForRolein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details des Instance-Profils zurückgegeben, das der Rolle **`ec2instancerole`** zugeordnet ist.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Ausgabe:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfilesForRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details des Instance-Profils zurückgegeben, das der Rolle **`ec2instancerole`** zugeordnet ist.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Ausgabe:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- Einzelheiten zur API finden Sie unter ListInstanceProfilesForRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListMfaDevices`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListMfaDevices` verwendet wird.

CLI

AWS CLI

So listen Sie alle MFA-Geräte für einen angegebenen Benutzer auf

In diesem Beispiel werden Details zum MFA-Gerät zurückgegeben, das dem IAM-Benutzer Bob zugewiesen ist.

```
aws iam list-mfa-devices \
    --user-name Bob
```

Ausgabe:

```
{
    "MFADevices": [
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
            "EnableDate": "2019-10-28T20:37:09+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "GAKT12345678",
            "EnableDate": "2023-02-18T21:44:42+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
            "EnableDate": "2023-09-19T02:25:35+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
            "EnableDate": "2023-09-19T01:49:18+00:00"
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListMfaDevicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zum MFA-Gerät zurückgegeben, das dem IAM-Benutzer **`David`** zugewiesen ist. In diesem Beispiel können Sie erkennen, dass es sich um ein virtuelles Gerät handelt, da die **`SerialNumber`** eine ARN und nicht die tatsächliche Seriennummer eines physischen Geräts ist.

```
Get-IAMMFADevice -UserName David
```

Ausgabe:

```
EnableDate                  SerialNumber                          UserName
----------                  ------------                          --------
4/8/2015 9:41:10 AM         arn:aws:iam::123456789012:mfa/David    David
```

- Einzelheiten zur API finden Sie unter ListMfaDevices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zum MFA-Gerät zurückgegeben, das dem IAM-Benutzer **`David`** zugewiesen ist. In diesem Beispiel können Sie erkennen, dass es sich um ein virtuelles Gerät handelt, da die **`SerialNumber`** eine ARN und nicht die tatsächliche Seriennummer eines physischen Geräts ist.

```
Get-IAMMFADevice -UserName David
```

Ausgabe:

```
EnableDate                  SerialNumber                          UserName
----------                  ------------                          --------
4/8/2015 9:41:10 AM         arn:aws:iam::123456789012:mfa/David    David
```

- Einzelheiten zur API finden Sie unter ListMfaDevices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListOpenIdConnectProviders`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListOpenIdConnectProviders` verwendet wird.

CLI

AWS CLI

Um Informationen über die OpenID Connect-Anbieter im AWS Konto aufzulisten

Dieses Beispiel gibt eine Liste der ARNS aller OpenID Connect-Anbieter zurück, die im AWS Girokonto definiert sind.

```
aws iam list-open-id-connect-providers
```

Ausgabe:

```
{
    "OpenIDConnectProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListOpenIdConnectProvidersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von ARNs aller OpenID Connect-Anbieter zurückgegeben, die im aktuellen AWS-Konto definiert sind.

```
Get-IAMOpenIDConnectProviderList
```

Ausgabe:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Einzelheiten zur API finden Sie unter ListOpenIdConnectProviders AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von ARNs aller OpenID Connect-Anbieter zurückgegeben, die im aktuellen AWS-Konto definiert sind.

```
Get-IAMOpenIDConnectProviderList
```

Ausgabe:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- Einzelheiten zur API finden Sie unter ListOpenIdConnectProviders AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListPolicies`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListPolicies` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Richtlinien

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
```

```
    return true;
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um verwaltete Richtlinien aufzulisten, die für Ihr AWS Konto verfügbar sind

In diesem Beispiel wird eine Sammlung der ersten beiden verwalteten Richtlinien zurückgegeben, die im aktuellen AWS Konto verfügbar sind.

```
aws iam list-policies \
    --max-items 3
```

Ausgabe:

```
{
    "Policies": [
        {
            "PolicyName": "AWSCloudTrailAccessPolicy",
            "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2019-09-04T17:43:42+00:00",
            "UpdateDate": "2019-09-04T17:43:42+00:00"
        },
        {
            "PolicyName": "AdministratorAccess",
            "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
            "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 6,
            "PermissionsBoundaryUsageCount": 0,
```

```
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:46+00:00",
            "UpdateDate": "2015-02-06T18:39:46+00:00"
        },
        {
            "PolicyName": "PowerUserAccess",
            "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",
            "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",
            "Path": "/",
            "DefaultVersionId": "v5",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:47+00:00",
            "UpdateDate": "2023-07-06T22:04:00+00:00"
        }
    ],
    "NextToken": "EXAMPLErZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiA4fQ=="
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListPoliciesunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet Sie die Richtlinien auf.

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services
 account.
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Sammlung der ersten drei verwalteten Richtlinien zurückgegeben, die im aktuellen AWS Konto verfügbar sind. Da nicht angegeben **`-scope`** ist, werden standardmäßig sowohl AWS verwaltete als auch vom **`all`** Kunden verwaltete Richtlinien verwendet und diese sind auch enthalten.

```
Get-IAMPolicyList -MaxItem 3
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
```

```
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Beispiel 2: In diesem Beispiel wird eine Sammlung der ersten beiden vom Kunden verwalteten Policen zurückgegeben, die im AWS Girokonto verfügbar sind. Es wird verwendet **`-Scope local`**, um die Ausgabe nur auf vom Kunden verwaltete Richtlinien zu beschränken.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Einzelheiten zur API finden Sie unter ListPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung der ersten drei verwalteten Richtlinien zurückgegeben, die im aktuellen AWS Konto verfügbar sind. Da nicht angegeben **`-scope`** ist, werden standardmäßig sowohl AWS verwaltete als auch vom **`all`** Kunden verwaltete Richtlinien verwendet und diese sind auch enthalten.

```
Get-IAMPolicyList -MaxItem 3
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Beispiel 2: In diesem Beispiel wird eine Sammlung der ersten beiden vom Kunden verwalteten Policen zurückgegeben, die im AWS Girokonto verfügbar sind. Es wird verwendet **`-Scope local`**, um die Ausgabe nur auf vom Kunden verwaltete Richtlinien zu beschränken.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- Einzelheiten zur API finden Sie unter ListPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(scope):
```

```
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- Einzelheiten zur API finden Sie ListPoliciesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
```

```
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listpolicies( iv_scope = iv_scope ).
        MESSAGE 'Retrieved policy list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policies.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListPoliciesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []

        // Use "Paginated" to get all the policies.
```

```
        // This lets the SDK handle the 'isTruncated' in "ListPoliciesOutput".
        let input = ListPoliciesInput()
        let output = client.listPoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let policies = page.policies else {
                    print("Error: no policies returned.")
                    continue
                }

                for policy in policies {
                    guard let name = policy.policyName,
                          let id = policy.policyId,
                          let arn = policy.arn
                    else {
                        throw ServiceHandlerError.noSuchPolicy
                    }
                    policyList.append(MyPolicyRecord(name: name, id: id, arn:
arn))
                }
            }
        } catch {
            print("ERROR: listPolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- Einzelheiten zur API finden Sie ListPoliciesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListPolicyVersions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListPolicyVersions` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten von Richtlinien
- Zurücksetzen einer Richtlinienversion

CLI

AWS CLI

So listen Sie Informationen zu den Versionen der angegebenen verwalteten Richtlinie auf

In diesem Beispiel wird die Liste der verfügbaren Versionen der Richtlinie zurückgegeben, deren ARN `arn:aws:iam::123456789012:policy/MySamplePolicy` lautet.

```
aws iam list-policy-versions \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
{
    "IsTruncated": false,
    "Versions": [
        {
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2015-06-02T23:19:44Z"
        },
        {
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2015-06-02T22:30:47Z"
        }
    ]
}
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListPolicyVersionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der verfügbaren Versionen der Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MyManagedPolicy`**

lautet. Um das Richtliniendokument für eine bestimmte Version abzurufen, verwenden Sie den Befehl **`Get-IAMPolicyVersion`** und geben Sie die **`VersionId`** der gewünschten Version an.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

Ausgabe:

```
CreateDate                    Document                  IsDefaultVersion
       VersionId
----------                    --------                  ----------------
       ---------
2/12/2015 9:39:53 AM                                    True
       v2
2/12/2015 9:39:09 AM                                    False
       v1
```

- Einzelheiten zur API finden Sie unter ListPolicyVersions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der verfügbaren Versionen der Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MyManagedPolicy`** lautet. Um das Richtliniendokument für eine bestimmte Version abzurufen, verwenden Sie den Befehl **`Get-IAMPolicyVersion`** und geben Sie die **`VersionId`** der gewünschten Version an.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

Ausgabe:

```
CreateDate                    Document                  IsDefaultVersion
       VersionId
----------                    --------                  ----------------
       ---------
2/12/2015 9:39:53 AM                                    True
       v2
```

```
2/12/2015 9:39:09 AM                                False
       v1
```

- Einzelheiten zur API finden Sie unter ListPolicyVersions AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListPolicyVersionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListRolePolicies`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListRolePolicies` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Geräteschutz für das Internet der Dinge

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die einer IAM-Rolle angefügten Richtlinien auf

Mit dem folgenden `list-role-policies`-Befehl werden die Namen der Berechtigungsrichtlinien für die angegebene IAM-Rolle aufgelistet.

```
aws iam list-role-policies \
    --role-name Test-Role
```

Ausgabe:

```
{
    "PolicyNames": [
        "ExamplePolicy"
    ]
}
```

Verwenden Sie den `get-role`-Befehl, um die einer Rolle angefügten Vertrauensrichtlinie anzuzeigen. Verwenden Sie den `get-role-policy`-Befehl, um die Details einer Berechtigungsrichtlinie anzuzeigen.

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet Sie die Richtlinien auf.

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient-
>listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der Namen von Inline-Richtlinien zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet sind. Um die Details einer Inline-Richtlinie anzuzeigen, verwenden Sie den Befehl **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Ausgabe:

```
oneClick_lambda_exec_role_policy
```

- Einzelheiten zur API finden Sie unter ListRolePolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Namen von Inline-Richtlinien zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet sind. Um die Details einer Inline-Richtlinie anzuzeigen, verwenden Sie den Befehl **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Ausgabe:

```
oneClick_lambda_exec_role_policy
```

- Einzelheiten zur API finden Sie unter ListRolePolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- Einzelheiten zur API finden Sie ListRolePoliciesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listrolepolicies(
          iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved inline policy list for role.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []

        // Use "Paginated" to get all the role policies.
```

```
        // This lets the SDK handle the 'isTruncated' in
 "ListRolePoliciesOutput".
        let input = ListRolePoliciesInput(
            roleName: role
        )
        let pages = client.listRolePoliciesPaginated(input: input)

        do {
            for try await page in pages {
                guard let policies = page.policyNames else {
                    print("Error: no role policies returned.")
                    continue
                }

                for policy in policies {
                    policyList.append(policy)
                }
            }
        } catch {
            print("ERROR: listRolePolicies:", dump(error))
            throw error
        }
        return policyList
    }
```

- Einzelheiten zur API finden Sie ListRolePoliciesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ListRoleTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListRoleTags` verwendet wird.

CLI

AWS CLI

So listen Sie die einer Rolle zugeordneten Tags auf

Der folgende `list-role-tags`-Befehl ruft die Liste mit Tags ab, die der angegebenen Rolle zugeordnet sind.

```
aws iam list-role-tags \
```

```
    --role-name production-role
```

Ausgabe:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS - IAM-Benutzer.

- Einzelheiten zur API finden Sie ListRoleTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das der Rolle zugeordnete Tag abgerufen.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter ListRoleTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das der Rolle zugeordnete Tag abgerufen.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- Einzelheiten zur API finden Sie unter ListRoleTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListRoles`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListRoles` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die IAM-Rollen für das aktuelle Konto auf

Der folgende `list-roles`-Befehl listet die IAM-Rollen für das aktuelle Konto auf.

```
aws iam list-roles
```

Ausgabe:

```
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "ExampleRole",
            "RoleId": "AROAJ52OTH4H7LEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
            "CreateDate": "2017-09-12T19:23:36+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/example_path/",
            "RoleName": "ExampleRoleWithPath",
            "RoleId": "AROAI4QRP7UFT7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
            "CreateDate": "2023-09-21T20:29:38+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
```

```
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListRolesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
```

```
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Rollen auf.

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
```

```
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste aller IAM-Rollen im AWS-Konto abgerufen.

```
Get-IAMRoleList
```

- Einzelheiten zur API finden Sie unter ListRoles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste aller IAM-Rollen im AWS-Konto abgerufen.

```
Get-IAMRoleList
```

- Einzelheiten zur API finden Sie unter ListRoles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
```

```
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- Einzelheiten zur API finden Sie ListRolesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count

      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
```

```
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- Einzelheiten zur API finden Sie ListRolesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- Einzelheiten zur API finden Sie ListRolesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listroles( ).
        MESSAGE 'Retrieved role list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing roles.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListRolesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listRoles() async throws -> [String] {
        var roleList: [String] = []

        // Use "Paginated" to get all the roles.
        // This lets the SDK handle the 'isTruncated' in "ListRolesOutput".
        let input = ListRolesInput()
```

```
        let pages = client.listRolesPaginated(input: input)

        do {
            for try await page in pages {
                guard let roles = page.roles else {
                    print("Error: no roles returned.")
                    continue
                }

                for role in roles {
                    if let name = role.roleName {
                        roleList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listRoles:", dump(error))
            throw error
        }
        return roleList
    }
```

- Einzelheiten zur API finden Sie ListRolesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListSamlProviders`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListSAMLProviders` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
```

```
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
 ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für .NET - API-Referenz.

CLI

AWS CLI

Um die SAML-Anbieter im Konto aufzulisten AWS

In diesem Beispiel wird die Liste der SAML 2.0-Anbieter abgerufen, die im aktuellen Konto erstellt wurde. AWS

```
aws iam list-saml-providers
```

Ausgabe:

```
{
    "SAMLProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
            "ValidUntil": "2015-06-05T22:45:14Z",
            "CreateDate": "2015-06-05T22:45:14Z"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-SAML-Identitätsanbietern im AWS - IAM-Benutzerhandbuch.

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für Go - API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die SAML-Anbieter auf.

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der API-Referenz für AWS SDK für JavaScript .

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für PHP - API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der im aktuellen AWS-Konto erstellten SAML 2.0-Anbieter abgerufen. Es gibt den ARN, das Erstellungs- und das Ablaufdatum für jeden SAML-Anbieter zurück.

```
Get-IAMSAMLProviderList
```

Ausgabe:

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- API-Details finden Sie unter ListSAMLProviders in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der im aktuellen AWS-Konto erstellten SAML 2.0-Anbieter abgerufen. Es gibt den ARN, das Erstellungs- und das Ablaufdatum für jeden SAML-Anbieter zurück.

```
Get-IAMSAMLProviderList
```

Ausgabe:

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- API-Details finden Sie unter ListSAMLProviders in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
```

```
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der API-Referenz zum AWS -SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
```

```
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für Ruby - API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der API-Referenz zum AWS -SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listsamlproviders( ).
        MESSAGE 'Retrieved SAML provider list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing SAML providers.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter ListSamlProviders in der API-Referenz zum AWS SDK für SAP ABAP.

`ListServerCertificates`Verwenden Sie mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListServerCertificates` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
```

```
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
                      certificate.GetArn() << std::setw(14) <<

certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::setw(14) <<

certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
```

```
    return true;
}
```

- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um die Serverzertifikate in Ihrem AWS Konto aufzulisten

Der folgende `list-server-certificates` Befehl listet alle Serverzertifikate auf, die in Ihrem AWS Konto gespeichert sind und zur Verwendung verfügbar sind.

```
aws iam list-server-certificates
```

Ausgabe:

```
{
    "ServerCertificateMetadataList": [
        {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        {
            "Path": "/cloudfront/",
            "ServerCertificateName": "MyTestCert",
            "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
            "UploadDate": "2015-04-21T18:14:16+00:00",
            "Expiration": "2018-01-14T17:52:36+00:00"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwaltung von Serverzertifikaten in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Zertifikate auf.

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
    } else {
      break;
    }
```

```
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der Serverzertifikate abgerufen, die in das aktuelle AWS-Konto hochgeladen wurden.

```
Get-IAMServerCertificateList
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter ListServerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Serverzertifikate abgerufen, die in das aktuelle AWS-Konto hochgeladen wurden.

```
Get-IAMServerCertificateList
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter ListServerCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
```

```
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für Ruby API-Referenz.

Verwendung von **`ListSigningCertificates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListSigningCertificates` verwendet wird.

CLI

AWS CLI

So listen Sie die Signaturzertifikate für einen IAM-Benutzer auf

Mit dem folgenden `list-signing-certificates`-Befehl werden die Signaturzertifikate für den IAM-Benutzer mit dem Namen Bob aufgelistet.

```
aws iam list-signing-certificates \
    --user-name Bob
```

Ausgabe:

```
{
    "Certificates": [
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-
body>-----END CERTIFICATE-----",
            "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
            "UploadDate": "2013-06-06T21:40:08Z"
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwaltung von Signaturzertifikaten im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie ListSigningCertificatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details über das Signaturzertifikat abgerufen, das dem Benutzer mit dem Namen **Bob** zugeordnet ist.

```
Get-IAMSigningCertificate -UserName Bob
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Einzelheiten zur API finden Sie unter ListSigningCertificates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details über das Signaturzertifikat abgerufen, das dem Benutzer mit dem Namen **Bob** zugeordnet ist.

```
Get-IAMSigningCertificate -UserName Bob
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Einzelheiten zur API finden Sie unter ListSigningCertificates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListUserPolicies`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListUserPolicies` verwendet wird.

CLI

AWS CLI

So listen Sie Richtlinien für einen IAM-Benutzer auf

Der folgende `list-user-policies`-Befehl listet die Richtlinien auf, die dem IAM-Benutzer mit dem Namen Bob zugeordnet sind.

```
aws iam list-user-policies \
    --user-name Bob
```

Ausgabe:

```
{
    "PolicyNames": [
        "ExamplePolicy",
        "TestPolicy"
    ]
}
```

Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Einen IAM-Benutzer in Ihrem AWS Konto erstellen.AWS

- Einzelheiten zur API finden Sie unter ListUserPolicies AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- Einzelheiten zur API finden Sie ListUserPoliciesin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der Namen der Inline-Richtlinien abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet sind.

```
Get-IAMUserPolicyList -UserName David
```

Ausgabe:

```
Davids_IAM_Admin_Policy
```

- Einzelheiten zur API finden Sie unter ListUserPolicies AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Namen der Inline-Richtlinien abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet sind.

```
Get-IAMUserPolicyList -UserName David
```

Ausgabe:

```
Davids_IAM_Admin_Policy
```

- Einzelheiten zur API finden Sie unter ListUserPolicies AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListUserTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListUserTags` verwendet wird.

CLI

AWS CLI

So listen Sie die an einen Benutzer angefügten Tags auf

Der folgende `list-user-tags`-Befehl ruft die Liste der Tags ab, die dem angegebenen IAM-Benutzer zugeordnet sind.

```
aws iam list-user-tags \
    --user-name alice
```

Ausgabe:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS - IAM-Benutzer.

- Einzelheiten zur API finden Sie ListUserTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das dem Benutzer zugeordnete Tag abgerufen.

```
Get-IAMUserTagList -UserName joe
```

- Einzelheiten zur API finden Sie unter ListUserTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das dem Benutzer zugeordnete Tag abgerufen.

```
Get-IAMUserTagList -UserName joe
```

- Einzelheiten zur API finden Sie unter ListUserTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListUsers`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListUsers` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }
```

```
        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie IAM Benutzer auf

Der folgende `list-users`-Befehl listet die IAM-Benutzer im aktuellen Konto auf.

```
aws iam list-users
```

Ausgabe:

```
{
    "Users": [
        {
            "UserName": "Adele",
            "Path": "/",
            "CreateDate": "2013-03-07T05:14:48Z",
            "UserId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Adele"
        },
        {
            "UserName": "Bob",
            "Path": "/",
            "CreateDate": "2012-09-21T23:03:13Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Auflisten von IAM-Benutzern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListUsersin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
```

```
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
 ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
```

```
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Benutzer auf.

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);

  for (const { UserName, CreateDate } of response.Users) {
    console.log(`${UserName} created on: ${CreateDate}`);
  }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();
```

```
    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Sammlung von Benutzern in der aktuellen AWS-Konto Version abgerufen.

```
Get-IAMUserList
```

Ausgabe:

```
      Arn              : arn:aws:iam::123456789012:user/Administrator
      CreateDate       : 10/16/2014 9:03:09 AM
      PasswordLastUsed : 3/4/2015 12:12:33 PM
      Path             : /
      UserId           : 7K3GJEANSKZF2EXAMPLE1
      UserName         : Administrator

      Arn              : arn:aws:iam::123456789012:user/Bob
      CreateDate       : 4/6/2015 12:54:42 PM
      PasswordLastUsed : 1/1/0001 12:00:00 AM
      Path             : /
      UserId           : L3EWNONDOM3YUEXAMPLE2
```

```
    UserName         : bab

    Arn              : arn:aws:iam::123456789012:user/David
    CreateDate       : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path             : /
    UserId           : Y4FKWQCXTA52QEXAMPLE3
    UserName         : David
```

- Einzelheiten zur API finden Sie unter ListUsers AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung von Benutzern in der aktuellen Version abgerufen. AWS-Konto

```
Get-IAMUserList
```

Ausgabe:

```
    Arn              : arn:aws:iam::123456789012:user/Administrator
    CreateDate       : 10/16/2014 9:03:09 AM
    PasswordLastUsed : 3/4/2015 12:12:33 PM
    Path             : /
    UserId           : 7K3GJEANSKZF2EXAMPLE1
    UserName         : Administrator

    Arn              : arn:aws:iam::123456789012:user/Bob
    CreateDate       : 4/6/2015 12:54:42 PM
    PasswordLastUsed : 1/1/0001 12:00:00 AM
    Path             : /
    UserId           : L3EWNONDOM3YUEXAMPLE2
    UserName         : bab

    Arn              : arn:aws:iam::123456789012:user/David
    CreateDate       : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path             : /
    UserId           : Y4FKWQCXTA52QEXAMPLE3
    UserName         : David
```

- Einzelheiten zur API finden Sie unter ListUsers AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- Einzelheiten zur API finden Sie ListUsersin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_users(
```

```
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listusers( ).
        MESSAGE 'Retrieved user list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing users.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listUsers() async throws -> [MyUserRecord] {
        var userList: [MyUserRecord] = []

        // Use "Paginated" to get all the users.
        // This lets the SDK handle the 'isTruncated' in "ListUsersOutput".
        let input = ListUsersInput()
        let output = client.listUsersPaginated(input: input)

        do {
            for try await page in output {
                guard let users = page.users else {
                    continue
                }
                for user in users {
                    if let id = user.userId, let name = user.userName {
                        userList.append(MyUserRecord(id: id, name: name))
                    }
                }
            }
        }
        catch {
            print("ERROR: listUsers:", dump(error))
            throw error
        }
       return userList
    }
```

- Einzelheiten zur API finden Sie ListUsersin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ListVirtualMfaDevices`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListVirtualMfaDevices` verwendet wird.

CLI

AWS CLI

So listen Sie virtuelle MFA-Geräte auf

Der folgende `list-virtual-mfa-devices`-Befehl listet die virtuellen MFA-Geräte auf, die für das aktuelle Konto konfiguriert wurden.

```
aws iam list-virtual-mfa-devices
```

Ausgabe:

```
{
    "VirtualMFADevices": [
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
        },
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
        }
    ]
}
```

Weitere Informationen finden Sie unter Aktivieren eines virtuellen Geräts zur Multi-Faktor-Authentifizierung (MFA) im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListVirtualMfaDevicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Sammlung der virtuellen MFA-Geräte abgerufen, die Benutzern im AWS Konto zugewiesen sind. Bei jeder **`User`**-Eigenschaft handelt es sich um ein Objekt mit Angaben zum IAM-Benutzer, dem das Gerät zugewiesen ist.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Ausgabe:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Einzelheiten zur API finden Sie unter ListVirtualMfaDevices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Sammlung der virtuellen MFA-Geräte abgerufen, die Benutzern im AWS Konto zugewiesen sind. Bei jeder **`User`**-Eigenschaft handelt es sich um ein Objekt mit Angaben zum IAM-Benutzer, dem das Gerät zugewiesen ist.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Ausgabe:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
```

```
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- Einzelheiten zur API finden Sie unter ListVirtualMfaDevices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutGroupPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutGroupPolicy` verwendet wird.

CLI

AWS CLI

So fügen Sie eine Richtlinie zu einer Gruppe hinzu

Der folgende `put-group-policy`-Befehl fügt eine Richtlinie zur IAM-Gruppe mit dem Namen `Admins` hinzu.

```
aws iam put-group-policy \
    --group-name Admins \
    --policy-document file://AdminPolicy.json \
    --policy-name AdminRoot
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Die Richtlinie ist in der AdminPolicy.jsonDatei als JSON-Dokument definiert. (Der Dateiname und die Erweiterung sind nicht von Bedeutung.)

Weitere Informationen finden Sie unter Verwalten von IAM-Richtlinien im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutGroupPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`AppTesterPolicy`** erstellt und in die IAM-Gruppe **`AppTesters`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie wird in der Datei **`apptesterpolicy.json`** gespeichert. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Einzelheiten zur API finden Sie unter PutGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`AppTesterPolicy`** erstellt und in die IAM-Gruppe **`AppTesters`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie wird in der Datei **`apptesterpolicy.json`** gespeichert. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- Einzelheiten zur API finden Sie unter PutGroupPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutRolePermissionsBoundary`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutRolePermissionsBoundary` verwendet wird.

CLI

AWS CLI

Beispiel 1: So wenden Sie eine Berechtigungsgrenze basierend auf einer benutzerdefinierten Richtlinie auf eine IAM-Rolle an

Im folgenden `put-role-permissions-boundary`-Beispiel wird die benutzerdefinierte Richtlinie mit dem Namen `intern-boundary` als Berechtigungsgrenze für die angegebene IAM-Rolle angewendet.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --role-name lambda-application-role
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So wenden Sie eine auf einer AWS verwalteten Richtlinie basierende Berechtigungsgrenze auf eine IAM-Rolle an

Im folgenden `put-role-permissions-boundary` Beispiel AWS wird die verwaltete `PowerUserAccess` Richtlinie als Berechtigungsgrenze für die angegebene IAM-Rolle angewendet.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --role-name x-account-admin
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter PutRolePermissionsBoundary AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für eine IAM-Rolle festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutRolePermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für eine IAM-Rolle festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutRolePermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutRolePolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRolePolicy` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
```

```
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie PutRolePolicyin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
```

```
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie PutRolePolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie einer IAM-Rolle eine Berechtigungsrichtlinie hinzu

Der folgende `put-role-policy`-Befehl fügt der Rolle mit dem Namen `Test-Role` eine Berechtigungsrichtlinie hinzu.

```
aws iam put-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Die Richtlinie ist in der AdminPolicy.jsonDatei als JSON-Dokument definiert. (Der Dateiname und die Erweiterung sind nicht von Bedeutung.)

Verwenden Sie den `update-assume-role-policy`-Befehl, um einer Rolle eine Vertrauensrichtlinie anzufügen.

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutRolePolicyin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::amzn-s3-demo-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
```

```
});

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie PutRolePolicyin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`FedTesterRolePolicy`** erstellt und in die IAM-Rolle **`FedTesterRole`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`FedTesterPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`FedTesterRolePolicy`** erstellt und in die IAM-Rolle **`FedTesterRole`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`FedTesterPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutUserPermissionsBoundary`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutUserPermissionsBoundary` verwendet wird.

CLI

AWS CLI

Beispiel 1: So wenden Sie eine Berechtigungsgrenze basierend auf einer benutzerdefinierten Richtlinie auf einen IAM-Benutzer an

Im folgenden `put-user-permissions-boundary`-Beispiel wird eine benutzerdefinierte Richtlinie mit dem Namen `intern-boundary` als Berechtigungsgrenze für den angegebenen IAM-Benutzer angewendet.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --user-name intern
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So wenden Sie eine auf einer AWS verwalteten Richtlinie basierende Berechtigungsgrenze auf einen IAM-Benutzer an

Im folgenden `put-user-permissions-boundary` Beispiel AWS wird die verwaltete Richtlinie angewendet, die `PowerUserAccess` als Berechtigungsgrenze für den angegebenen IAM-Benutzer bezeichnet wird.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --user-name developer
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Informationen finden Sie im Abschnitt Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter PutUserPermissionsBoundary AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für den Benutzer festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutUserPermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für den Benutzer festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- Einzelheiten zur API finden Sie unter PutUserPermissionsBoundary AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutUserPolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutUserPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So fügen Sie einem IAM-Benutzer eine Richtlinie an

Der folgende `put-user-policy`-Befehl fügt dem IAM-Benutzer mit dem Namen `Bob` eine Richtlinie an.

```
aws iam put-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Die Richtlinie ist in der AdminPolicy.jsonDatei als JSON-Dokument definiert. (Der Dateiname und die Erweiterung sind nicht von Bedeutung.)

Informationen finden Sie im Abschnitt Hinzufügen und Entfernen von IAM-Identitätsberechtigungen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutUserPolicyin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
```

```
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie PutUserPolicyin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`EC2AccessPolicy`** erstellt und in den IAM-Benutzer **`Bob`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`EC2AccessPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`EC2AccessPolicy`** erstellt und in den IAM-Benutzer **`Bob`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`EC2AccessPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- Einzelheiten zur API finden Sie unter PutUserPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates an inline policy for a specified user.
  # @param username [String] The name of the IAM user.
  # @param policy_name [String] The name of the policy to create.
  # @param policy_document [String] The JSON policy document.
  # @return [Boolean]
  def create_user_policy(username, policy_name, policy_document)
    @iam_client.put_user_policy({
                                  user_name: username,
                                  policy_name: policy_name,
                                  policy_document: policy_document
                                })
    @logger.info("Policy #{policy_name} created for user #{username}.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't create policy #{policy_name} for user #{username}.
 Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- Einzelheiten zur API finden Sie PutUserPolicyin der AWS SDK für Ruby API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    func putUserPolicy(policyDocument: String, policyName: String, user:
 IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            print("ERROR: putUserPolicy:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie PutUserPolicyin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`RemoveClientIdFromOpenIdConnectProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveClientIdFromOpenIdConnectProvider` verwendet wird.

CLI

AWS CLI

So entfernen Sie die angegebene Client-ID aus der Liste der für den angegebenen Anbieter von IAM OpenID Connect registrierten Client-IDs

In diesem Beispiel wird die Client-ID `My-TestApp-3` aus der Liste der Client-IDs entfernt, die dem IAM-OIDC-Anbieter zugeordnet sind, dessen ARN `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` lautet.

```
aws iam remove-client-id-from-open-id-connect-provider
    --client-id My-TestApp-3 \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie RemoveClientIdFromOpenIdConnectProviderin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Client-ID **`My-TestApp-3`** aus der Liste der Client-IDs entfernt, die dem IAM-OIDC-Anbieter zugeordnet sind, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter RemoveClientIdFromOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Client-ID **`My-TestApp-3`** aus der Liste der Client-IDs entfernt, die dem IAM-OIDC-Anbieter zugeordnet sind, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- Einzelheiten zur API finden Sie unter RemoveClientIdFromOpenIdConnectProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveRoleFromInstanceProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveRoleFromInstanceProfile` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit verwaltetem Streaming
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

CLI

AWS CLI

So entfernen Sie eine Rolle aus einem Instance-Profil

Mit dem folgenden `remove-role-from-instance-profile`-Befehl wird die Rolle mit dem Namen `Test-Role` aus dem Instance-Profil mit dem Namen `ExampleInstanceProfile` entfernt.

```
aws iam remove-role-from-instance-profile \
    --instance-profile-name ExampleInstanceProfile \
    --role-name Test-Role
```

Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie RemoveRoleFromInstanceProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **MyNewRole** aus dem EC2-Instance-Profil mit dem Namen **MyNewRole** gelöscht. Ein in der IAM-Konsole erstelltes Instance-Profil hat immer denselben Namen wie die Rolle, wie in diesem Beispiel. Wenn Sie sie in der API oder CLI erstellen, können sie unterschiedliche Namen haben.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- Einzelheiten zur API finden Sie unter RemoveRoleFromInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **MyNewRole** aus dem EC2-Instance-Profil mit dem Namen **MyNewRole** gelöscht. Ein in der IAM-Konsole erstelltes Instance-Profil hat immer denselben Namen wie die Rolle, wie in diesem Beispiel. Wenn Sie sie in der API oder CLI erstellen, können sie unterschiedliche Namen haben.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- Einzelheiten zur API finden Sie unter RemoveRoleFromInstanceProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **RemoveUserFromGroup** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveUserFromGroup` verwendet wird.

CLI

AWS CLI

So entfernen Sie einen Benutzer aus einer IAM-Gruppe

Mit dem folgenden `remove-user-from-group`-Befehl wird der Benutzer mit dem Namen `Bob` aus der IAM-Gruppe mit dem Namen `Admins` entfernt.

```
aws iam remove-user-from-group \
    --user-name Bob \
    --group-name Admins
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Hinzufügen und Entfernen von Benutzern in einer IAM-Benutzergruppe im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie RemoveUserFromGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** aus der Gruppe **`Testers`** entfernt.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Gruppen gesucht, in denen der IAM-Benutzer **`Theresa`** Mitglied ist, und dann wird **`Theresa`** aus diesen Gruppen entfernt.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

Beispiel 3: In diesem Beispiel wird eine alternative Möglichkeit zum Entfernen des IAM-Benutzers **`Bob`** aus der **`Testers`**-Gruppe gezeigt.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- Einzelheiten zur API finden Sie unter RemoveUserFromGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** aus der Gruppe **`Testers`** entfernt.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Gruppen gesucht, in denen der IAM-Benutzer **`Theresa`** Mitglied ist, und dann wird **`Theresa`** aus diesen Gruppen entfernt.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

Beispiel 3: In diesem Beispiel wird eine alternative Möglichkeit zum Entfernen des IAM-Benutzers **`Bob`** aus der **`Testers`**-Gruppe gezeigt.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- Einzelheiten zur API finden Sie unter RemoveUserFromGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ResyncMfaDevice`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ResyncMfaDevice` verwendet wird.

CLI

AWS CLI

So synchronisieren Sie ein MFA-Gerät

Im folgenden `resync-mfa-device`-Beispiel wird das MFA-Gerät, das dem IAM-Benutzer `Bob`zugeordnet ist und dessen ARN `arn:aws:iam::123456789012:mfa/BobsMFADevice` ist, mit einem Authentifizierungsprogramm synchronisiert, das die beiden Authentifizierungscodes bereitgestellt hat.

```
aws iam resync-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 987654
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ResyncMfaDevicein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das MFA-Gerät, das dem IAM-Benutzer **Bob** zugeordnet ist und dessen ARN **arn:aws:iam::123456789012:mfa/bob** lautet, mit einem Authentifizierungs-Programm synchronisiert, das die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Beispiel 2: In diesem Beispiel wird das IAM-MFA-Gerät, das dem IAM-Benutzer **Theresa** zugeordnet ist, mit einem physischen Gerät synchronisiert, das die Seriennummer **ABCD12345678** hat und die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Einzelheiten zur API finden Sie unter ResyncMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das MFA-Gerät, das dem IAM-Benutzer **Bob** zugeordnet ist und dessen ARN **arn:aws:iam::123456789012:mfa/bob** lautet, mit einem Authentifizierungs-Programm synchronisiert, das die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Beispiel 2: In diesem Beispiel wird das IAM-MFA-Gerät, das dem IAM-Benutzer **Theresa** zugeordnet ist, mit einem physischen Gerät synchronisiert, das die Seriennummer **ABCD12345678** hat und die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- Einzelheiten zur API finden Sie unter ResyncMfaDevice AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`SetDefaultPolicyVersion`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetDefaultPolicyVersion` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten von Richtlinien
- Zurücksetzen einer Richtlinienversion

CLI

AWS CLI

So legen Sie die angegebene Version der angegebenen Richtlinie als standardmäßige Version der Richtlinie fest.

In diesem Beispiel wird die `v2`-Version der Richtlinie, deren ARN `arn:aws:iam::123456789012:policy/MyPolicy` lautet, als standardmäßig aktive Version festgelegt.

```
aws iam set-default-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

Weitere Informationen finden Sie unter Richtlinien und Berechtigungen in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie SetDefaultPolicyVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die **v2**-Version der Richtlinie, deren ARN **arn:aws:iam::123456789012:policy/MyPolicy** lautet, als standardmäßige aktive Version festgelegt.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- Einzelheiten zur API finden Sie unter SetDefaultPolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die **v2**-Version der Richtlinie, deren ARN **arn:aws:iam::123456789012:policy/MyPolicy** lautet, als standardmäßige aktive Version festgelegt.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- Einzelheiten zur API finden Sie unter SetDefaultPolicyVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->setdefaultpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
```

```
        MESSAGE 'Default policy version set successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'Invalid input provided.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SetDefaultPolicyVersionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`TagRole`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TagRole` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Rolle ein Tag hinzu

Der folgende `tag-role`-Befehl fügt der angegebenen Rolle ein Tag mit einem Abteilungsnamen hinzu.

```
aws iam tag-role --role-name my-role \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS - IAM-Benutzer.

- Einzelheiten zur API finden Sie TagRolein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Rolle im Identity Management Service ein Tag hinzugefügt

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Rolle im Identity Management Service ein Tag hinzugefügt

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`TagUser`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TagUser` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Benutzer ein Tag hinzu

Der folgende `tag-user`-Befehl fügt dem angegebenen Benutzer ein Tag mit der zugehörigen Abteilung hinzu.

```
aws iam tag-user \
    --user-name alice \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS -IAM-Benutzer.

- Einzelheiten zur API finden Sie TagUserin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird dem Benutzer im Identity Management Service ein Tag hinzugefügt

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird dem Benutzer im Identity Management Service ein Tag hinzugefügt

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- Einzelheiten zur API finden Sie unter TagUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UntagRole`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UntagRole` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein Tag aus einer Rolle

Der folgende `untag-role`-Befehl entfernt alle Tags mit dem Schlüsselnamen „Abteilung“ aus der angegebenen Rolle.

```
aws iam untag-role \
    --role-name my-role \
    --tag-keys Department
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS - IAM-Benutzer.

- Einzelheiten zur API finden Sie UntagRolein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Tag aus der Rolle mit dem Namen "MyRoleName" mit dem Tag-Schlüssel „abac“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Tag aus der Rolle mit dem Namen "MyRoleName" mit dem Tag-Schlüssel „abac“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UntagUser`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UntagUser` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein Tag von einem Benutzer

Der folgende `untag-user`-Befehl entfernt alle Tags mit dem Schlüsselnamen „Abteilung“ vom angegebenen Benutzer.

```
aws iam untag-user \
    --user-name alice \
    --tag-keys Department
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Taggen von IAM-Ressourcen im Handbuch zu AWS - IAM-Benutzer.

- Einzelheiten zur API finden Sie UntagUserin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Tag vom Benutzer namens „joe“ mit den Tag-Schlüsseln „abac“ und „xyzw“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Tag vom Benutzer namens „joe“ mit den Tag-Schlüsseln „abac“ und „xyzw“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- Einzelheiten zur API finden Sie unter UntagUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateAccessKey`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAccessKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffsschlüsseln

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iam_update_access_key
#
# This function can activate or deactivate an IAM access key for the specified
 IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to update.
#       -a            -- Activate the selected access key.
#       -d            -- Deactivate the selected access key.
#
# Example:
#       # To deactivate the selected access key for IAM user Bob
#       iam_update_access_key -u Bob -k AKIAIOSFODNN7EXAMPLE -d
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_update_access_key() {
  local user_name access_key status response
  local option OPTARG # Required to use getopts command in a function.
  local activate_flag=false deactivate_flag=false

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_update_access_key"
    echo "Updates the status of an AWS Identity and Access Management (IAM)
 access key for the specified IAM user"
```

```
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to update."
    echo "  -a              Activate the access key."
    echo "  -d              Deactivate the access key."
    echo ""
  }

  # Retrieve the calling parameters.
    while getopts "u:k:adh" option; do
      case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        a) activate_flag=true ;;
        d) deactivate_flag=true ;;
        h)
          usage
          return 0
          ;;
        \?)
          echo "Invalid parameter"
          usage
          return 1
          ;;
      esac
    done
    export OPTIND=1

   # Validate input parameters
    if [[ -z "$user_name" ]]; then
      errecho "ERROR: You must provide a username with the -u parameter."
      usage
      return 1
    fi

    if [[ -z "$access_key" ]]; then
      errecho "ERROR: You must provide an access key with the -k parameter."
      usage
      return 1
    fi

    # Ensure that only -a or -d is specified
    if [[ "$activate_flag" == true && "$deactivate_flag" == true ]]; then
      errecho "ERROR: You cannot specify both -a (activate) and -d (deactivate)
at the same time."
```

```
      usage
      return 1
    fi

    # If neither -a nor -d is provided, return an error
    if [[ "$activate_flag" == false && "$deactivate_flag" == false ]]; then
      errecho "ERROR: You must specify either -a (activate) or -d (deactivate)."
      usage
      return 1
    fi

    # Determine the status based on the flag
    if [[ "$activate_flag" == true ]]; then
      status="Active"
    elif [[ "$deactivate_flag" == true ]]; then
      status="Inactive"
    fi

    iecho "Parameters:\n"
    iecho "    Username:   $user_name"
    iecho "    Access key: $access_key"
    iecho "    New status: $status"
    iecho ""

    # Update the access key status
    response=$(aws iam update-access-key \
      --user-name "$user_name" \
      --access-key-id "$access_key" \
      --status "$status" 2>&1)

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log $error_code
      errecho "ERROR: AWS reports update-access-key operation failed.\n$response"
      return 1
    fi

    iecho "update-access-key response: $response"
    iecho

    return 0
}
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktivieren oder deaktivieren Sie einen Zugriffsschlüssel für einen IAM-Benutzer

Mit dem folgenden `update-access-key`-Befehl wird der angegebene Zugriffsschlüssel (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel) für den IAM-Benutzer mit dem Namen `Bob` deaktiviert.

```
aws iam update-access-key \
    --access-key-id AKIAIOSFODNN7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Die Deaktivierung des Schlüssels bedeutet, dass er nicht für den programmatischen Zugriff auf verwendet werden kann. AWS Der Schlüssel ist jedoch weiterhin verfügbar und kann erneut aktiviert werden.

Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.
\s
                    accessId - The access key ID of the secret access key you
 want to update.\s
                    status - The status you want to assign to the secret access
 key.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
```

```
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s
 to" +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie den Zugriffsschlüssel.

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status des Zugriffsschlüssels **`AKIAIOSFODNN7EXAMPLE`** für den IAM-Benutzer mit dem Namen **`Bob`** in **`Inactive`** geändert.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- Einzelheiten zur API finden Sie unter UpdateAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status des Zugriffsschlüssels **`AKIAIOSFODNN7EXAMPLE`** für den IAM-Benutzer mit dem Namen **`Bob`** in **`Inactive`** geändert.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- Einzelheiten zur API finden Sie unter UpdateAccessKey AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
```

```
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
key_id
        )
        raise
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->updateaccesskey(
          iv_accesskeyid = iv_access_key_id
          iv_status = iv_status
          iv_username = iv_user_name ).
        MESSAGE 'Access key updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Access key or user does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`UpdateAccountPasswordPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAccountPasswordPolicy` verwendet wird.

CLI

AWS CLI

So legen Sie die aktuelle Passwortrichtlinie für Konten fest oder ändern diese

Der folgende `update-account-password-policy`-Befehl legt die Passwortrichtlinie so fest, dass eine Mindestlänge von acht Zeichen und eine oder mehrere Zahlen im Passwort erforderlich sind.

```
aws iam update-account-password-policy \
    --minimum-password-length 8 \
    --require-numbers
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Änderungen an der Passwortrichtlinie eines Kontos wirken sich auf alle neuen Passwörter aus, die für IAM-Benutzer im Konto erstellt werden. Änderungen der Passwortrichtlinie wirken sich nicht auf bestehende Passwörter aus.

Weitere Informationen finden Sie unter Festlegen einer Kontopasswortrichtlinie für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateAccountPasswordPolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Passwortrichtlinie für das Konto mit den angegebenen Einstellungen aktualisiert. Beachten Sie, dass alle Parameter, die nicht im Befehl enthalten sind, nicht unverändert bleiben. Stattdessen werden diese auf die Standardwerte zurückgesetzt.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
```

```
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Einzelheiten zur API finden Sie unter UpdateAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Passwortrichtlinie für das Konto mit den angegebenen Einstellungen aktualisiert. Beachten Sie, dass alle Parameter, die nicht im Befehl enthalten sind, nicht unverändert bleiben. Stattdessen werden diese auf die Standardwerte zurückgesetzt.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- Einzelheiten zur API finden Sie unter UpdateAccountPasswordPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateAssumeRolePolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAssumeRolePolicy` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie die Vertrauensrichtlinie für eine IAM-Rolle

Der folgende `update-assume-role-policy`-Befehl aktualisiert die Vertrauensrichtlinie für die Rolle mit dem Namen `Test-Role`.

```
aws iam update-assume-role-policy \
    --role-name Test-Role \
    --policy-document file://Test-Role-Trust-Policy.json
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Die Vertrauensrichtlinie ist in der Test-Role-Trust-Policy.jsonDatei als JSON-Dokument definiert. (Der Dateiname und die Erweiterung sind nicht von Bedeutung.) Die Vertrauensrichtlinie muss einen Prinzipal angeben.

Um die Berechtigungsrichtlinie für eine Rolle zu aktualisieren, verwenden Sie den `put-role-policy`-Befehl.

Weitere Informationen finden Sie unter Erstellen von IAM-Rollen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateAssumeRolePolicyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die IAM-Rolle mit dem Namen **`ClientRole`** mit einer neuen Vertrauensrichtlinie aktualisiert, deren Inhalt aus der Datei **`ClientRolePolicy.json`** stammt. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Einzelheiten zur API finden Sie unter UpdateAssumeRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die IAM-Rolle mit dem Namen **`ClientRole`** mit einer neuen Vertrauensrichtlinie aktualisiert, deren Inhalt aus der Datei **`ClientRolePolicy.json`** stammt. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- Einzelheiten zur API finden Sie unter UpdateAssumeRolePolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **UpdateGroup** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateGroup` verwendet wird.

CLI

AWS CLI

So benennen Sie eine IAM-Gruppe um

Mit dem folgenden `update-group`-Befehl wird der Name der IAM-Gruppe `Test` in `Test-1` verändert.

```
aws iam update-group \
    --group-name Test \
    --new-group-name Test-1
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Umbenennen einer IAM-Benutzergruppe im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe **`Testers`** in **`AppTesters`** umbenannt.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Beispiel 2: In diesem Beispiel wird der Pfad der IAM-Gruppe **`AppTesters`** in **`/Org1/Org2/`** geändert. Dadurch wird die ARN für die Gruppe in **`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`** geändert.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe **`Testers`** in **`AppTesters`** umbenannt.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Beispiel 2: In diesem Beispiel wird der Pfad der IAM-Gruppe **`AppTesters`** in **`/Org1/Org2/`** geändert. Dadurch wird die ARN für die Gruppe in **`arn:aws:iam::123456789012:group/Org1/Org2/AppTesters`** geändert.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateLoginProfile`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateLoginProfile` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie das Passwort für einen IAM-Benutzer

Der folgende `update-login-profile`-Befehl erstellt ein neues Passwort für den IAM-Benutzer mit dem Namen `Bob`.

```
aws iam update-login-profile \
    --user-name Bob \
    --password <password>
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Um eine Passwortrichtlinie für das Konto festzulegen, verwenden Sie den `update-account-password-policy`-Befehl. Wenn das neue Passwort gegen die Passwortrichtlinie des Kontos verstößt, gibt der Befehl einen `PasswordPolicyViolation`-Fehler zurück.

Sofern die Passwortrichtlinie des Kontos dies zulässt, können IAM-Benutzer mit dem `change-password`-Befehl ihre eigenen Passwörter ändern.

Speichern Sie das Passwort an einem sicheren Ort. Bei Verlust des Passwortes ist eine Wiederherstellung nicht möglich und Sie müssen mit dem Befehl `create-login-profile` ein neues Passwort erstellen.

Weitere Informationen finden Sie unter Verwalten von Passwörtern für IAM-Benutzer im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateLoginProfilein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues temporäres Passwort für den IAM-Benutzer **`Bob`** festgelegt und der Benutzer muss das Passwort bei der nächsten Anmeldung ändern.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- Einzelheiten zur API finden Sie unter UpdateLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues temporäres Passwort für den IAM-Benutzer **`Bob`** festgelegt und der Benutzer muss das Passwort bei der nächsten Anmeldung ändern.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- Einzelheiten zur API finden Sie unter UpdateLoginProfile AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateOpenIdConnectProviderThumbprint`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateOpenIdConnectProviderThumbprint` verwendet wird.

CLI

AWS CLI

So ersetzen Sie die vorhandene Liste der Serverzertifikat-Fingerabdrücke durch eine neue Liste

In diesem Beispiel wird die Liste der Zertifikat-Fingerabdrücke für den OIDC-Anbieter aktualisiert, dessen ARN `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` lautet, um einen neuen Fingerabdruck zu verwenden.

```
aws iam update-open-id-connect-provider-thumbprint \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com \
    --thumbprint-list 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

Wenn die Discovery- und JWKS-Endpoints Ihres OIDC-Anbieters unterschiedliche Zertifikate verwenden, fügen Sie dem Wert Fingerabdrücke für beide Endpunkte hinzu. `--thumbprint-list`

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von OpenID Connect (OIDC)-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. UpdateOpenIdConnectProviderThumbprintAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Zertifikat-Fingerabdruckliste für den OIDC-Anbieter, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet, aktualisiert, um einen neuen Fingerabdruck zu verwenden. Der OIDC-Anbieter gibt den neuen Wert frei, wenn sich das dem Anbieter zugeordnete Zertifikat ändert.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Einzelheiten zur API finden Sie unter UpdateOpenIdConnectProviderThumbprint AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Zertifikat-Fingerabdruckliste für den OIDC-Anbieter, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet, aktualisiert, um einen neuen Fingerabdruck zu verwenden. Der OIDC-Anbieter gibt den neuen Wert frei, wenn sich das dem Anbieter zugeordnete Zertifikat ändert.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- Einzelheiten zur API finden Sie unter UpdateOpenIdConnectProviderThumbprint AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateRole`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateRole` verwendet wird.

CLI

AWS CLI

So ändern Sie die Beschreibung oder Sitzungsdauer einer IAM-Rolle

Mit dem folgenden `update-role`-Befehl wird die Beschreibung der IAM-Rolle `production-role` in `Main production role` geändert und die maximale Sitzungsdauer auf 12 Stunden festgelegt.

```
aws iam update-role \
    --role-name production-role \
    --description 'Main production role' \
    --max-session-duration 43200
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateRolein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Rollenbeschreibung und der maximale Sitzungsdauerwert (in Sekunden) aktualisiert, für den die Sitzung einer Rolle angefordert werden kann.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Einzelheiten zur API finden Sie unter UpdateRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Rollenbeschreibung und der maximale Sitzungsdauerwert (in Sekunden) aktualisiert, für den die Sitzung einer Rolle angefordert werden kann.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- Einzelheiten zur API finden Sie unter UpdateRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateRoleDescription`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateRoleDescription` verwendet wird.

CLI

AWS CLI

So ändern Sie die Beschreibung einer IAM-Rolle

Der folgende `update-role`-Befehl ändert die Beschreibung der IAM-Rolle `production-role` in `Main production role`.

```
aws iam update-role-description \
```

```
    --role-name production-role \
    --description 'Main production role'
```

Ausgabe:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "production-role",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/production-role",
        "CreateDate": "2017-12-06T17:16:37+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        },
        "Description": "Main production role"
    }
}
```

Weitere Informationen finden Sie unter Ändern einer Rolle im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateRoleDescriptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Beschreibung einer IAM-Rolle in Ihrem Konto aktualisiert.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Einzelheiten zur API finden Sie unter UpdateRoleDescription AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung einer IAM-Rolle in Ihrem Konto aktualisiert.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- Einzelheiten zur API finden Sie unter UpdateRoleDescription AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateSamlProvider`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateSamlProvider` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie das Metadatendokument für einen vorhandenen SAML-Anbieter

In diesem Beispiel wird der SAML-Anbieter in IAM, dessen ARN `arn:aws:iam::123456789012:saml-provider/SAMLADFS` lautet, mit einem neuen SAML-Metadatendokument aus der Datei `SAMLMetaData.xml` aktualisiert.

```
aws iam update-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

Ausgabe:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"
}
```

Weitere Informationen finden Sie unter Erstellen von IAM-SAML-Identitätsanbietern im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateSamlProviderin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der SAML-Provider in IAM, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** lautet, mit einem neuen SAML-Metadatendokument aus der Datei **`SAMLMetaData.xml`** aktualisiert. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Einzelheiten zur API finden Sie unter UpdateSamlProvider AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der SAML-Provider in IAM, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** lautet, mit einem neuen SAML-Metadatendokument aus der Datei **`SAMLMetaData.xml`** aktualisiert. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- Einzelheiten zur API finden Sie unter UpdateSamlProvider AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateServerCertificate`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateServerCertificate` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
 &currentCertificateName,
                                           const Aws::String &newCertificateName,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
 ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }
```

```
    return result;
}
```

- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So ändern Sie den Pfad oder Namen eines Serverzertifikats in Ihrem AWS Konto

Mit dem folgenden `update-server-certificate`-Befehl wird der Name des Zertifikats von `myServerCertificate` in `myUpdatedServerCertificate` geändert. Außerdem wird der Pfad geändert, `/cloudfront/` sodass der CloudFront Amazon-Service darauf zugreifen kann. Mit diesem Befehl wird keine Ausgabe zurückgegeben. Sie können die Ergebnisse der Aktualisierung anzeigen, indem Sie den `list-server-certificates`-Befehl ausführen.

```
aws-iam update-server-certificate \
    --server-certificate-name myServerCertificate \
    --new-server-certificate-name myUpdatedServerCertificate \
    --new-path /cloudfront/
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwaltung von Serverzertifikaten in IAM im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie ein Serverzertifikat.

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });
```

```
var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Zertifikat mit dem Namen **`MyServerCertificate`** in **`MyRenamedServerCertificate`** umbenannt.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Beispiel 2: In diesem Beispiel wird das angegebene Zertifikat **`MyServerCertificate`** in den Pfad/Org1/Org2/verschoben. Dadurch wird die ARN für die Ressource zu **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`** geändert.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat mit dem Namen **`MyServerCertificate`** in **`MyRenamedServerCertificate`** umbenannt.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Beispiel 2: In diesem Beispiel wird das angegebene Zertifikat in **`MyServerCertificate`** den Pfad//Org1/Org2verschoben. Dadurch wird die ARN für die Ressource zu **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`** geändert.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
```

```
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
```

```
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für Ruby API-Referenz.

Verwendung von **`UpdateSigningCertificate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateSigningCertificate` verwendet wird.

CLI

AWS CLI

So aktivieren oder deaktivieren Sie ein Signaturzertifikat für einen IAM-Benutzer

Der folgende `update-signing-certificate`-Befehl deaktiviert das angegebene Signaturzertifikat für den IAM-Benutzer mit dem Namen Bob.

```
aws iam update-signing-certificate \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

Verwenden Sie den `list-signing-certificates`-Befehl, um die ID für ein Signaturzertifikat abzurufen.

Weitere Informationen finden Sie unter Verwaltung von Signaturzertifikaten im Benutzerhandbuch für Amazon EC2.

- Einzelheiten zur API finden Sie UpdateSigningCertificatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Zertifikat aktualisiert, das dem IAM-Benutzer mit dem Namen **Bob** und der Zertifikats-ID **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** zugeordnet ist, um es als inaktiv zu markieren.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Einzelheiten zur API finden Sie unter UpdateSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Zertifikat aktualisiert, das dem IAM-Benutzer mit dem Namen **Bob** und der Zertifikats-ID **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** zugeordnet ist, um es als inaktiv zu markieren.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- Einzelheiten zur API finden Sie unter UpdateSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateUser`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateUser` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName
 <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So ändern Sie den Namen eines IAM-Benutzers

Mit dem folgenden `update-user`-Befehl wird der Name des IAM-Benutzers `Bob` in `Robert` geändert.

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Umbenennen einer IAM-Benutzergruppe im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateUserin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String
 newName) {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
 newName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie den Benutzer.

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- Einzelheiten zur API finden Sie UpdateUserin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** in **`Robert`** umbenannt.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Beispiel 2: In diesem Beispiel wird der Pfad des IAM-Benutzers **`Bob`** in **`/Org1/Org2/`** geändert, wodurch der ARN für den Benutzer effektiv zu **`arn:aws:iam::123456789012:user/Org1/Org2/bob`** geändert wird.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateUser AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** in **`Robert`** umbenannt.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Beispiel 2: In diesem Beispiel wird der Pfad des IAM-Benutzers **`Bob`** in **`/Org1/Org2/`** geändert, wodurch der ARN für den Benutzer effektiv zu **`arn:aws:iam::123456789012:user/Org1/Org2/bob`** geändert wird.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- Einzelheiten zur API finden Sie unter UpdateUser AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
```

```
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- Einzelheiten zur API finden Sie UpdateUserin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
  false
end
```

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->updateuser(
      iv_username = iv_user_name
      iv_newusername = iv_new_user_name ).
    MESSAGE 'User updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'New user name already exists.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateUserin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UploadServerCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UploadServerCertificate` verwendet wird.

CLI

AWS CLI

Um ein Serverzertifikat auf Ihr AWS Konto hochzuladen

Mit dem folgenden Befehl upload-server-certificate wird ein Serverzertifikat auf Ihr Konto hochgeladen. AWS In diesem Beispiel befindet sich das Zertifikat in der Datei `public_key_cert_file.pem`, der zugehörige private Schlüssel in der Datei `my_private_key.pem` und die von der Zertifizierungsstelle (CA) bereitgestellte Zertifikatskette befindet sich in der `my_certificate_chain_file.pem`-Datei. Wenn der Upload der Datei abgeschlossen ist, ist sie unter dem Namen my verfügbar. ServerCertificate

Parameter, die mit `file://` beginnen, weisen den Befehl an, den Inhalt der Datei zu lesen und diesen als Parameterwert anstelle des Dateinamens selbst zu verwenden.

```
aws iam upload-server-certificate \
    --server-certificate-name myServerCertificate \
    --certificate-body file://public_key_cert_file.pem \
    --private-key file://my_private_key.pem \
    --certificate-chain file://my_certificate_chain_file.pem
```

Ausgabe:

```
{
    "ServerCertificateMetadata": {
        "Path": "/",
        "ServerCertificateName": "myServerCertificate",
        "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
        "Arn": "arn:aws:iam::1234567989012:server-certificate/
myServerCertificate",
        "UploadDate": "2019-04-22T21:13:44+00:00",
        "Expiration": "2019-10-15T22:23:16+00:00"
    }
}
```

Weitere Informationen finden Sie unter Erstellen, Hochladen und Löschen von Serverzertifikaten im Handbuch zur Verwendung von IAM.

- Einzelheiten zur API finden Sie UploadServerCertificatein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
```

```
import * as path from "node:path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
 or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });
```

```
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie UploadServerCertificatein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues Serverzertifikat in das IAM-Konto hochgeladen. Die Dateien, die den Text des Zertifikats, den privaten Schlüssel und (optional) die Zertifikatskette enthalten, müssen alle PEM-codiert sein. Beachten Sie, dass die Parameter den tatsächlichen Inhalt der Dateien und nicht die Dateinamen erfordern. Um den Dateiinhalt erfolgreich zu verarbeiten, müssen Sie den **-Raw**-Switch-Parameter verwenden.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter UploadServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Serverzertifikat in das IAM-Konto hochgeladen. Die Dateien, die den Text des Zertifikats, den privaten Schlüssel und (optional) die Zertifikatskette enthalten, müssen alle PEM-codiert sein. Beachten Sie, dass die Parameter den tatsächlichen Inhalt der Dateien und nicht die Dateinamen erfordern. Um den Dateiinhalt erfolgreich zu verarbeiten, müssen Sie den **-Raw**-Switch-Parameter verwenden.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- Einzelheiten zur API finden Sie unter UploadServerCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UploadSigningCertificate`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UploadSigningCertificate` verwendet wird.

CLI

AWS CLI

So laden Sie ein Signaturzertifikat für einen IAM-Benutzer hoch

Mit dem folgenden `upload-signing-certificate`-Befehl wird ein Signaturzertifikat für den IAM-Benutzer mit dem Namen Bob hochgeladen.

```
aws iam upload-signing-certificate \
    --user-name Bob \
    --certificate-body file://certificate.pem
```

Ausgabe:

```
{
    "Certificate": {
        "UserName": "Bob",
        "Status": "Active",
        "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
 CERTIFICATE-----",
```

```
        "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
        "UploadDate": "2013-06-06T21:40:08.121Z"
    }
}
```

Das Zertifikat befindet sich in einer Datei mit dem Namen certificate.pem im PEM-Format.

Weitere Informationen finden Sie unter „Erstellen und Hochladen eines Benutzersignaturzertifikats“ im Handbuch Verwendung von IAM.

- Einzelheiten zur API finden Sie UploadSigningCertificatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues X.509 Signaturzertifikat hochgeladen und es dem IAM-Benutzer mit dem Namen zugeordnet. **`Bob`** Die Datei mit dem Zertifikatstext ist PEM-codiert. Der **`CertificateBody`**-Parameter erfordert den tatsächlichen Inhalt der Zertifikatsdatei und nicht den Dateinamen. Sie müssen den **`-Raw`**-Switch-Parameter verwenden, um die Datei erfolgreich zu verarbeiten.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
```

```
                    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                    rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Einzelheiten zur API finden Sie unter UploadSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues X.509 Signaturzertifikat hochgeladen und es dem IAM-Benutzer mit dem Namen zugeordnet. **`Bob`** Die Datei mit dem Zertifikatstext ist PEM-codiert. Der **`CertificateBody`**-Parameter erfordert den tatsächlichen Inhalt der Zertifikatsdatei und nicht den Dateinamen. Sie müssen den **`-Raw`**-Switch-Parameter verwenden, um die Datei erfolgreich zu verarbeiten.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
```

```
 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- Einzelheiten zur API finden Sie unter UploadSigningCertificate AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von IAM AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in IAM mit SDKs implementieren. AWS Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von IAM oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen Sie IAM-Benutzer mit Schreibzugriff und Lese-/Schreibzugriff mit einem AWS SDK
- Einen Container-Service für VM-Instanzen erstellen
- Einen verwalteten Monitoring-Workspace erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern
- Erste Schritte mit der Workflow-Orchestrierung
- Verwalten Sie IAM-Zugriffsschlüssel mit einem AWS SDK
- Verwalten Sie IAM-Richtlinien mit einem AWS SDK
- Verwalten Sie IAM-Rollen mit einem AWS SDK
- Verwalten Sie Ihr IAM-Konto mit einem AWS SDK
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher
- Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen
- Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation
- Richtlinie zur Aktivierung der Automatisierung für Ihr Konto
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten

- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation
- Führen Sie ein Rollback einer IAM-Richtlinienversion mithilfe eines AWS SDK
- Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch
- Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2
- Systemmanagement einrichten
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2
- Arbeiten Sie mit der IAM Policy Builder-API unter Verwendung eines AWS SDK

Erstellen und verwalten Sie einen ausfallsicheren Service mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
```

```
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
```

```
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
```

```
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
```

```
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");
```

```
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
```

```
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
```

```
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");
```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
```

```
        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
```

```
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
```

```
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
```

```
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
               });
           return launchTemplateResponse.LaunchTemplate;
       }
       catch (AmazonEC2Exception ec2Exception)
       {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
```

```
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
```

```
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
 List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
```

```
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
```

```
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
```

```
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
```

```
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
```

```
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
```

```
   /// This can be done by allowing ingress from this computer's IP address.
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                       "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
```

```
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
```

```
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
```

```
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
```

```
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });
```

```
                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
```

```
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
```

```
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
```

```
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
```

```
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);
```

```
            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =
```

```
 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
```

```
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
```

```
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";
```

```
    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
 service.
                To keep things tidy and to avoid unwanted charges on your
 account, we can clean up all AWS resources
```

```
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
```

```
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
```

```
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
```

```
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
```

```
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
```

```
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
```

```
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
```

```
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
```

```
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
       autoScaler.terminateInstance(badInstanceId);

       System.out.println("""
               Even while the instance is terminating and the new instance is
starting, sending a GET
               request to the web service continues to get a successful
recommendation response because
               the load balancer routes requests to the healthy instances. After
the replacement instance
               starts and reports as healthy, it is included in the load
balancing rotation.
               Note that terminating and replacing an instance typically takes
several minutes, during which time you
               can see the changing health check status until the new instance
is running and healthy.
               """);

       demoChoices(loadBalancer);
       System.out.println(
               "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       demoChoices(loadBalancer);
       paramHelper.reset();
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
```

```
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());
```

```
                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
 balancer targets:\n");
                        List<TargetHealthDescription> health =
 loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
```

```
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
```

```
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
```

```
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();
```

```
            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
```

```
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
```

```
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }
```

```
    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

```
   // Verify this computer can successfully send a GET request to the load
balancer
   // endpoint.
   public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
       boolean success = false;
       int retries = 3;
       CloseableHttpClient httpClient = HttpClients.createDefault();

       // Create an HTTP GET request to the ELB.
       HttpGet httpGet = new HttpGet("http://" + elbDnsName);
       try {
           while ((!success) && (retries > 0)) {
               // Execute the request and get the response.
               HttpResponse response = httpClient.execute(httpGet);
               int statusCode = response.getStatusLine().getStatusCode();
               System.out.println("HTTP Status Code: " + statusCode);
               if (statusCode == 200) {
                   success = true;
               } else {
                   retries--;
                   System.out.println("Got connection error from load balancer
endpoint, retrying...");
                   TimeUnit.SECONDS.sleep(15);
               }
           }

       } catch (org.apache.http.conn.HttpHostConnectException e) {
           System.out.println(e.getMessage());
       }

       System.out.println("Status.." + success);
       return success;
   }

   /*
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
```

```
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
```

```
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;
```

```
        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
```

```
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();
```

```
            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();
```

```
            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();
```

```
        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
```

```
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
```

```
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
```

```
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
      type: "confirm",
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        }),
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
```

```
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );
```

```
    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
```

```
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
```

```
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
```

```
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
```

```
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
```

```
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
```

```
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
```

```
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
```

```
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
```

```
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
```

```
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
```

```
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
```

```
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
```

```
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
```

```
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
```

```
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
```

```
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
        throw new Error("Instance not found.");
      }
    });

    await ssmClient.send(
      new SendCommandCommand({
```

```
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
```

```
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
```

```
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
```

```
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
```

```
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
```

```
import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
```

```
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
```

```
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
```

```
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
```

```
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
```

```
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```

```
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
```

```
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
```

```
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
```

```
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.

- AttachLoadBalancerTargetGroups
- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
```

```
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")
```

```
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
```

```
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
```

```
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
```

```
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
```

```
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
```

```
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
```

```
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
    for action in actions:
        if action == "deploy":
```

```
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
```

```
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
```

```
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
        instance. The role has attached policies that specify the AWS permissions
 granted to
        clients that run on the instance.
```

```
        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
```

```
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
```

```
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
```

```
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
```

```
            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
```

```
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
```

```
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
```

```
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
```

```
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
```

```
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
```

```
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")
```

```
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
```

```
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
```

```
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
```

```
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
```

```
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
```

```
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
 found. "
                    "Ensure the load balancer exists, is in the correct AWS
 region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
```

```
        Initializes the RecommendationService class with the necessary
parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
```

```
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
 a
        dependency fails, and shallow health checks.
```

```
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen

- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)
```

```
# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
```

```
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
```

```
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
```

```
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
```

```
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
```

```
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy

- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
```

```
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error
```

```
    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1
```

```
    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
```

```
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
```

```
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
'$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
```

```
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
 security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"
```

```
    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"
```

```
# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json
```

```
    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
```

```
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
```

```
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
```

```
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"
```

```
CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
```

```
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetRole
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService
  - Wait

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Eine Lambda-Funktion erstellen und bereitstellen

- Erstellen Sie eine REST-API
- Lambda-Proxyintegration konfigurieren
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."
```

```
# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
```

```
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
```

```
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
```

```
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}
```

```
echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2
```

```
# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Erstellen Sie IAM-Benutzer mit Schreibzugriff und Lese-/Schreibzugriff mit einem AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie Benutzer erstellen und ihnen Richtlinien zuweisen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie zwei IAM-Benutzer.
- Fügen Sie eine Richtlinie zu einem Benutzer hinzu und bringen Sie Objekte in einem Amazon S3-Bucket unter.
- Fügen Sie eine Richtlinie für den zweiten Benutzer hinzu, mit der dieser Objekte aus dem Bucket abrufen kann.
- Erhalten Sie unterschiedliche Berechtigungen für den Bucket basierend auf Anmeldeinformationen des Benutzers.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
import logging
import time

import boto3
```

```
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
```

```
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

Erstellen Sie Funktionen, die IAM-Richtlinien-Aktionen umschließen.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.
```

```
    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
```

```
        raise
```

Erstellen Sie Funktionen, die IAM-Zugriffsschlüssel-Aktionen umschließen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
```

```
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Benutzer mit unterschiedlichen Richtlinien zu erstellen, und verwenden Sie deren Anmeldeinformationen, um auf einen Amazon-S3-Bucket zuzugreifen.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
 an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
```

```
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and
{user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")
```

```
    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
```

```
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
```

```
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachUserPolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteUser
  - DetachUserPolicy
  - ListUsers
  - UpdateUser

Einen Container-Service für VM-Instanzen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Einen Dienst erstellen und überwachen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
```

```
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
```

```
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
```

```
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
```

```
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"
```

```
    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"
```

```
    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"
```

```
    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
```

```
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
```

```
    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
```

```
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"
```

```
    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
```

```
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition

- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Einen verwalteten Monitoring-Workspace erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Ihren Workspace
- Erstellen Sie einen Grafana-Arbeitsbereich
- Konfigurieren Sie die Authentifizierung
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
```

```
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
```

```
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
```

```
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
```

```
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"
```

```
    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"
```

```
# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
```

```
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
```

```
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
```

```
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Eine IAM-Rolle für Lambda erstellen
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES
```

```
##############################################################################
# Helper functions
##############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
```

```
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
```

```
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
```

```
##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
```

```
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

###############################################################################
# Step 1: Create IAM execution role
###############################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10
```

```
##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi
```

```
echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
```

```
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "============================================"
echo "Step 4: View CloudWatch Logs"
echo "============================================"
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
```

```
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy

- CreateFunction
- CreateRole
- DeleteFunction
- DeleteLogGroup
- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace
- Benutze Redshift CreateWorkgroup
- Benutze Redshift DeleteNamespace
- Benutze iam CreateRole

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
```

```
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
```

```
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
```

```
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
```

```
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "==========================================="
  echo "CLEANUP CONFIRMATION"
  echo "==========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
```

```
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
```

```
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")
```

```
# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
```

```
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
```

```
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
```

```
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
```

```
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
```

```
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
```

```
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi
```

```
# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateNamespace

- CreateRole
- CreateSecret
- CreateWorkgroup
- DeleteNamespace
- DeleteRole
- DeleteRolePolicy
- DeleteSecret
- DeleteWorkgroup
- GetNamespace
- GetRole
- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Eine IAM-Rolle für Config erstellen
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
```

```
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
```

```
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

```
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Erste Schritte mit dem Geräteschutz für das Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- IoT Device Defender Defender-Audit-Checks aktivieren
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Abhilfemaßnahme
- Wenden Sie Maßnahmen zur Risikominderung auf die Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- IoT-Protokollierung aktivieren

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "==================================================="
```

```
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
```

```
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true
```

```
        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
```

```
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
```

```
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
```

```
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "======================================================"
echo "Step 1: Creating required IAM roles"
echo "======================================================"

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
```

```
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)
```

```
if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "==================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "==================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi
```

```
echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
echo "Step 3: Running an on-demand audit"
echo "==================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
```

```
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
```

```
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "==================================================="
echo "Step 4: Creating a mitigation action"
echo "==================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
```

```
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
```

```
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="
```

```
# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)
```

```
    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
```

```
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
```

```
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
```

```
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
```

```
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics

- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
```

```
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2
```

```
    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi
```

```
    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
```

```
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
```

```
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
```

```
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
```

```
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
```

```
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)
```

```
if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"
```

```
# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
```

```
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
```

```
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
```

```
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
```

```
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
```

```
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."
```

```
# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
```

```
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
```

```
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"
```

```
# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
```

```
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
```

```
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
```

```
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein

- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
```

```
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"
```

```
    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
```

```
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
```

```
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
"Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
"$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi
```

```
    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
 "instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
 profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}
```

```
# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done
```

```
    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
```

```
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi
```

```
# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

```
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
```

```
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
```

```
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"
```

```
# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"
```

```
# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
```

```
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
```

```
echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)
```

```
if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
```

```
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
```

```
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
```

```
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi
```

```
echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
```

```
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
```

```
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
```

```
echo "==============================================="
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair

- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Redshift-Cluster
- Erstellen Sie eine IAM-Rolle für den S3-Zugriff
- Erstellen Sie Tabellen und laden Sie Daten

- Führen Sie Beispielabfragen aus
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."
```

```
    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
```

```
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi
```

```
# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
```

```
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
```

```
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
```

```
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""
```

```
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Erste Schritte mit der Workflow-Orchestrierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Eine IAM-Rolle für Step Functions erstellen
- Erstellen Sie Ihre erste Zustandsmaschine
- Starten Sie die Ausführung Ihrer Zustandsmaschine
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS
 Step Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT
```

```
echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be
 less reliable."
    echo "Consider installing jq for better error handling: brew install jq
 (macOS) or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
```

```
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}
```

```
# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] ||
 [ "$wait_time" -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
```

```
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi
```

```
    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/
null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed
 to delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
```

```
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
```

```
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
```

```
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
```

```
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
```

```
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs
 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
```

```
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
```

```
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
```

```
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
```

```
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")
```

```
    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
```

```
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
```

```
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment
 analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"
```

```
    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in
 $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
```

```
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

Verwalten Sie IAM-Zugriffsschlüssel mit einem AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie Zugriffsschlüssel verwalten.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen und Listen Sie Zugriffsschlüssel auf.
- Finden Sie heraus, wann und wie ein Zugriffsschlüssel zuletzt verwendet wurde.
- Zugriffsschlüssel aktualisieren und löschen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Zugriffsschlüssel-Aktionen umschließen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
```

```
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
```

```
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response



def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise



def delete_key(user_name, key_id):
    """
```

```
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Zugriffsschlüssel-Aktionen für den aktuellen Benutzer auszuführen.

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
```

```
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access
{last_use['ServiceName']} "
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

Verwalten Sie IAM-Richtlinien mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen und Auflisten von Richtlinien.
- Erstellen und Abrufen von Richtlinienversionen.

- Zurücksetzen einer Richtlinie auf eine frühere Version.
- Löschen von Richtlinien.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Richtlinien-Aktionen umschließen.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
```

```
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies



def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
```

```
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
```

```
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
```

```
        logger.info("Set %s as the default version.",
 rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
 so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Richtlinien zu erstellen, Versionen zu aktualisieren und Informationen über sie zu erhalten.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::amzn-s3-demo-bucket"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy
  - DeletePolicyVersion
  - GetPolicyVersion
  - ListPolicies
  - ListPolicyVersions
  - SetDefaultPolicyVersion

Verwalten Sie IAM-Rollen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle.
- Anfügen und Trennen von Richtlinien für eine Rolle.
- Löschen Sie eine Rolle.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Rollen-Aktionen umschließen.

```
import json
```

```
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
```

```
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
```

```
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um eine Rolle zu erstellen, und fügen Sie dann eine Richtlinie hinzu und trennen Sie sie.

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
 "batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachRolePolicy
  - CreateRole
  - DeleteRole
  - DetachRolePolicy

Verwalten Sie Ihr IAM-Konto mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie den Konto-Alias ab und aktualisieren Sie ihn.
- Generieren Sie einen Bericht über Benutzer und Anmeldeinformationen.
- Rufen Sie eine Zusammenfassung der Kontonutzung ab.
- Rufen Sie Details zu allen Benutzern, Gruppen, Rollen und Richtlinien in Ihrem Konto ab, einschließlich deren Beziehungen untereinander.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Kontoaktionen umschließen.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")
```

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
```

```
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
```

```
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
```

```
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

Rufen Sie Wrapper-Funktionen auf, um den Konto-Alias zu ändern und Berichte über das Konto zu erhalten.

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
```

```
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
 cred_report.decode("utf-8").split("\n")
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)?
 ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
```

```
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print("-" * 88)
    print("Thanks for watching.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen

- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisieren Sie das Geheimnis
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
```

```
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
```

```
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""
```

```
# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
```

```
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
```

```
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
```

```
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
```

```
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "========================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.

- AttachRolePolicy
- CreateRole
- CreateSecret
- DeleteRole
- DeleteSecret
- DetachRolePolicy
- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie ermöglicht es der AWS Compute Optimizer Automatisierungsfunktion, empfohlene Aktionen anzuwenden

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "aco-automation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie die Automatisierung in Ihrer gesamten Organisation aktivieren

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:AssociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:DisassociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:ListAccounts",
            "Resource": "*"
        }
    ]
```

```
}
```

Richtlinie zur Aktivierung der Automatisierung für Ihr Konto

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie die Automatisierung für Ihr Konto aktivieren

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie gewährt einem Verwaltungskonto einer Organisation vollen Zugriff auf Compute Optimizer Automation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
               "ec2:DescribeVolumes",
               "organizations:ListAccounts",
               "organizations:DescribeOrganization",
               "organizations:DescribeAccount",
               "organizations:EnableAWSServiceAccess",
               "organizations:ListDelegatedAdministrators",
               "organizations:RegisterDelegatedAdministrator",
               "organizations:DeregisterDelegatedAdministrator"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie eigenständigen Konten vollen Zugriff auf Compute Optimizer Automation gewähren AWS

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
            "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie gewährt einem Verwaltungskonto einer Organisation schreibgeschützten Zugriff auf Compute Optimizer Automation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAccounts",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
               "aco-automation:ListRecommendedActions",
               "aco-automation:ListRecommendedActionSummaries",
               "aco-automation:ListTagsForResource",
               "ec2:DescribeVolumes"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie gewährt eigenständigen Konten schreibgeschützten Zugriff auf Compute Optimizer Automation. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
               "aco-automation:ListRecommendedActions",
               "aco-automation:ListRecommendedActionSummaries",
               "aco-automation:ListTagsForResource",
               "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation

Das folgende Codebeispiel zeigt, wie diese berechtigungsbasierte Richtlinie dienstbezogene Rollenberechtigungen für Compute Optimization Automation gewährt

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        }
    ]
}
```

Führen Sie ein Rollback einer IAM-Richtlinienversion mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie die Liste der Richtlinienversionen nach Datum sortiert ab.
- Suchen Sie die Standard-Richtlinienversion.
- Machen Sie die vorherige Version der Richtlinie zur Standardversion.
- Löschen Sie die alte Standardversion.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
```

```
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Experimentvorlage
- Führen Sie das Experiment durch

- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
```

```
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi
```

```
    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
```

```
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
```

```
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
```

```
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60
```

```
# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

```
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
```

```
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail
```

```
    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
```

```
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="
```

```
# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
```

```
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances
  - Wait

Richten Sie die Attribute-Based Zugriffskontrolle für DynamoDB ein mit AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie Attribute-Based Access Control (ABAC) für DynamoDB implementiert wird.

- Erstellen Sie eine IAM-Richtlinie für ABAC.
- Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.
- Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine IAM-Richtlinie für ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
```

```
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
```

```
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreatePolicy
  - CreateTable
  - ListTables

Systemmanagement einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- IAM-Berechtigungen für Systems Manager erstellen
- Erstellen einer IAM-Rolle für Systems Manager
- Systems Manager konfigurieren
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
```

```
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done
```

```
    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"
```

```
    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
```

```
        "Sid": "SsmReadOnly",
        "Effect": "Allow",
        "Action": [
          "ssm:DescribeAutomationExecutions",
          "ssm:GetAutomationExecution",
          "ssm:ListAssociations",
          "ssm:DescribeAssociation",
          "ssm:ListDocuments",
          "ssm:ListResourceDataSync",
          "ssm:DescribePatchBaselines",
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
```

```
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
```

```
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
```

```
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
```

```
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
```

```
         "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
         "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
         "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
         "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
         "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
       ]
     }
   ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
```

```
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
```

```
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
```

```
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
```

```
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
```

```
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Lambda-Funktionen für die Überwachung erstellen
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
```

```
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi
```

```
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
```

```
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json
```

```
# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
```

```
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
```

```
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
```

```
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
```

```
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Arbeiten Sie mit DynamoDB Streams und verwenden Sie Time-to-Live AWS Command Line Interface v2

Das folgende Codebeispiel zeigt, wie DynamoDB Streams und Time-to-Live -Funktionen verwaltet werden.

- Erstellen Sie eine Tabelle mit aktiviertem Streams.
- Beschreiben Sie Streams.
- Erstellen Sie eine Lambda-Funktion für die Verarbeitung von Streams.

- Aktivieren Sie TTL für eine Tabelle.
- Fügen Sie Elemente mit TTL-Attributen hinzu.
- Beschreiben Sie TTL-Einstellungen.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit aktiviertem Streams.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Beschreiben Sie Streams.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Erstellen Sie eine Lambda-Funktion für Streams.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Aktivieren Sie TTL für eine Tabelle.

```
# Create a table for TTL demonstration
```

```
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Fügen Sie Elemente mit TTL-Attributen hinzu.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Beschreiben Sie TTL-Einstellungen.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

Arbeiten Sie mit der IAM Policy Builder-API unter Verwendung eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Richtlinien mithilfe der objektorientierten API.
- Verwenden Sie die IAM-Policy-Builder-API mit dem IAM-Service.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In den Beispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Erstellen Sie eine zeitbasierte Richtlinie.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_GREATER_THAN)

 .key("aws:CurrentTime")

 .value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_LESS_THAN)

 .key("aws:CurrentTime")

 .value("2020-06-30T23:59:59Z")))
```

```
                                .build();

                // Use an IamPolicyWriter to write out the JSON string to a more
 readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

Erstellen Sie eine Richtlinie mit mehreren Bedingungen.

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")

 .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

 .addAction("dynamodb:BatchWriteItem")

 .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))

 .key("dynamodb:Select")
```

```
 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Verwenden Sie Prinzipale in einer Richtlinie.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Gewähren Sie kontoübergreifenden -Zugriff.

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)

 .addPrincipal(IamPrincipalType.AWS, "111122223333")
                                                .addAction("s3:PutObject")
```

```
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                .key("s3:x-amz-
acl")
                                                                .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Erstellen und laden Sie eine `IamPolicy` hoch.

```
        public String createAndUploadPolicyExample(IamClient iam, String
 accountID, String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:PutItem")

 .addResource("arn:aws:dynamodb:us-east-1:" + accountID
                                                                + ":table/
exampleTableName")
                                                .build())
                                .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

Downloaden und arbeiten Sie mit einer `IamPolicy`.

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
 accountID, String policyName,
```

```
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

                String policyVersion =
getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned
from IAM.
                String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

Codebeispiele für Amazon Inspector mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Inspector mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Amazon Inspector-Benutzerhandbuch — Weitere Informationen zu Amazon Inspector.
- Amazon Inspector API-Referenz — Details zu allen verfügbaren Amazon Inspector Inspector-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Inspector AWS SDKs
  - Hallo
  - Lernen Sie die Grundlagen von Amazon Inspector mit einem AWS SDK
  - Aktionen für Amazon Inspector mit AWS SDKs

- Verwenden Sie BatchGetAccountStatusmit einem AWS SDK
- Verwenden Sie BatchGetFindingDetailsmit einem AWS SDK
- Verwenden Sie CreateFiltermit einem AWS SDK
- Verwenden Sie DeleteFiltermit einem AWS SDK
- Verwenden Sie Disable mit einem AWS SDK
- Verwenden Sie Enable mit einem AWS SDK
- Verwenden Sie ListCoveragemit einem AWS SDK
- Verwenden Sie ListCoverageStatisticsmit einem AWS SDK
- Verwenden Sie ListFiltersmit einem AWS SDK
- Verwenden Sie ListFindingsmit einem AWS SDK oder CLI
- Verwenden Sie ListUsageTotalsmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Inspector AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Inspector mit AWS SDKs verwenden können.

Beispiele

- Hallo
- Lernen Sie die Grundlagen von Amazon Inspector mit einem AWS SDK
- Aktionen für Amazon Inspector mit AWS SDKs
  - Verwenden Sie BatchGetAccountStatusmit einem AWS SDK
  - Verwenden Sie BatchGetFindingDetailsmit einem AWS SDK
  - Verwenden Sie CreateFiltermit einem AWS SDK
  - Verwenden Sie DeleteFiltermit einem AWS SDK
  - Verwenden Sie Disable mit einem AWS SDK
  - Verwenden Sie Enable mit einem AWS SDK
  - Verwenden Sie ListCoveragemit einem AWS SDK
  - Verwenden Sie ListCoverageStatisticsmit einem AWS SDK
  - Verwenden Sie ListFiltersmit einem AWS SDK

- Verwenden Sie ListFindingsmit einem AWS SDK oder CLI
- Verwenden Sie ListUsageTotalsmit einem AWS SDK

Hallo

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient =
 Inspector2Client.builder().build()) {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");
```

```
        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }

    /**
     * Lists all member accounts associated with the current Inspector
administrator account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
e.awsErrorDetails().errorMessage());
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie ListMembersin der AWS SDK for Java 2.x API-Referenz.

Lernen Sie die Grundlagen von Amazon Inspector mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Inspector-Kontostatus.
- Stellen Sie sicher, dass Inspector aktiviert ist.
- Analysieren Sie die Sicherheitsergebnisse.
- Überprüfen Sie die Scanabdeckung.
- Erstellen Sie einen Ergebnisfilter.
- Listet vorhandene Filter auf.
- Prüfen Sie die Nutzung und die Kosten.
- Holen Sie sich Reichweitenstatistiken.
- Löschen Sie einen Filter.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die -Features gezeigt werden.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class InspectorScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorScenario.class);
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        InspectorActions inspectorActions = new InspectorActions();
        logger.info("Amazon Inspector Basics Scenario");

        logger.info("""
                    Amazon Inspector is a security assessment service provided by
 Amazon Web Services (AWS) that helps
                    improve the security and compliance of applications deployed
 on AWS. It automatically assesses
                    applications for vulnerabilities or deviations from best
 practices. By leveraging Amazon Inspector,
                    users can gain insights into the overall security state of
 their application and identify potential
                    security risks.

                    This service operates by conducting both network and host-
based assessments, allowing it to detect a
                    wide range of security issues, including those related to
 operating systems, network configurations,
                    and application dependencies.
                    """);

        waitForInputToContinue();

        try {
            runScenario(inspectorActions);

            logger.info("");
            logger.info("Scenario completed successfully!");
            logger.info("");
            logger.info("What you learned:");
```

```
            logger.info("  - How to check Inspector account status");
            logger.info("  - How to enable Inspector");
            logger.info("  - How to list and analyze findings");
            logger.info("  - How to check coverage information");
            logger.info("  - How to create and manage filters");
            logger.info("  - How to track usage and costs");
            logger.info("  - How to clean up resources");
            logger.info("");

        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }

    /**
     * Runs the Inspector scenario in a step-by-step sequence.
     *
     * All InspectorActions methods are asynchronous and return
CompletableFutures.
     * Each step ends with .join(). Any async exception thrown during .join()
will bubble up
     *
     */
    public static void runScenario(InspectorActions actions) {
        String filterArn = null;
        boolean inspectorEnabled = false;

        try {
            // Step 1
            logger.info(DASHES);
            logger.info("Step 1: Checking Inspector account status...");
            String status = actions.getAccountStatusAsync().join();
            logger.info(status);
            waitForInputToContinue();

            // Step 2
            logger.info(DASHES);
            logger.info("Step 2: Enabling Inspector...");
            String message = actions.enableInspectorAsync(null).join();
```

```
            logger.info(message);
            inspectorEnabled = true;  // track that Inspector was enabled
            waitForInputToContinue();

            // Step 3
            logger.info(DASHES);
            logger.info("Step 3: Listing LOW severity findings...");

            // Call the service method
            List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

            if (!allFindings.isEmpty()) {
                // Only proceed if there are findings
                String lastArn = allFindings.get(allFindings.size() - 1);
                logger.info("Look up details on: {}", lastArn);
                waitForInputToContinue();
                String details = actions.getFindingDetailsAsync(lastArn).join();
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName,
"Suppress low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
```

```
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();

            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(),
ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
```

```
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
 e.getMessage(), e);
                }
            }

            if (inspectorEnabled) {
                try {
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
 e.getMessage(), e);
                }
            }
        }
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

Eine Wrapper-Klasse für -SDK-Methoden.

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource
types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String>
accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
```

```
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
 target accounts.");
                        return summary.toString();
                    }
```

```
                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }


    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage
statistics: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
```

```
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");
```

```
        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
 getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !
response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
 usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage
 totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." :
 summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
 %s".formatted(ve.getMessage()),
                                ve
```

```
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
```

```
                                "Unexpected error getting account status:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID:
 ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !
= null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }
```

```
                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
```

```
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the
filter
     * @return a CompletableFuture that completes with the ARN of the created
filter
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
```

```
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
```

```
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead
 of a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
```

```
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
 paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
 getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
```

```
                summary.append("    Scan Type:
").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " +
cause.getMessage(), cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
```

```
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter:
" + cause, cause);
                    }
                    return null;
                });
   }

   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
                .findingArns(findingArn)
                .build();

       return getAsyncClient().batchGetFindingDetails(request)
                .thenApply(response -> {
                    if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                        return String.format("No details found for ARN: ",
findingArn);
```

```
                    }

                    StringBuilder sb = new StringBuilder();
                    response.findingDetails().forEach(detail -> {
                        sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }
```

```
                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and
resource types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
Inspector;
     *                   may be {@code null} or empty to target the current
account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to
validation errors,
     *                             service errors, or other exceptions
```

```
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String>
 accountIds) {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
```

```
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
 target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
 ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter

- Aktivieren
- ListCoverage
- ListCoverageStatistics
- ListFilters
- ListFindings
- ListUsageTotals

Aktionen für Amazon Inspector mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Inspector Inspector-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Inspector API-Referenz.

Beispiele

- Verwenden Sie BatchGetAccountStatusmit einem AWS SDK
- Verwenden Sie BatchGetFindingDetailsmit einem AWS SDK
- Verwenden Sie CreateFiltermit einem AWS SDK
- Verwenden Sie DeleteFiltermit einem AWS SDK
- Verwenden Sie Disable mit einem AWS SDK
- Verwenden Sie Enable mit einem AWS SDK
- Verwenden Sie ListCoveragemit einem AWS SDK
- Verwenden Sie ListCoverageStatisticsmit einem AWS SDK
- Verwenden Sie ListFiltersmit einem AWS SDK
- Verwenden Sie ListFindingsmit einem AWS SDK oder CLI
- Verwenden Sie ListUsageTotalsmit einem AWS SDK

Verwenden Sie **`BatchGetAccountStatus`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`BatchGetAccountStatus`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
```

```
                            throw new CompletionException(
                                    "Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID:
 ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !
= null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }
```

```
                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }
```

- Einzelheiten zur API finden Sie BatchGetAccountStatusin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`BatchGetFindingDetails`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`BatchGetFindingDetails`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
     *
     * @param findingArn The ARN of the finding to look up.
```

```
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                   .append("Risk Score:
").append(detail.riskScore()).append("\n");

                       // ExploitObserved timings
                       if (detail.exploitObserved() != null) {
                           sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                       .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                       }

                       // Reference URLs
                       if (detail.hasReferenceUrls()) {
                           sb.append("Reference URLs:\n");
                           detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                       }

                       // Tools
                       if (detail.hasTools()) {
                           sb.append("Tools:\n");
```

```
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
```

```
    }
```

- Einzelheiten zur API finden Sie BatchGetFindingDetailsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateFilter`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateFilter`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
    *
    * @param filterName  the name of the filter to create
    * @param description a descriptive string explaining the purpose of the
filter
    * @return a CompletableFuture that completes with the ARN of the created
filter
    * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
    */
   public CompletableFuture<String> createLowSeverityFilterAsync(
           String filterName,
           String description) {
```

```
        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
```

```
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- Einzelheiten zur API finden Sie CreateFilterin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteFilter`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteFilter`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
```

```
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter:
" + cause, cause);
                    }
                    return null;
                });
   }
```

- Einzelheiten zur API finden Sie DeleteFilterin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`Disable`** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`Disable`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously disables AWS Inspector for the specified accounts and
resource types.
```

```
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current
 account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to
 validation errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String>
 accountIds) {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
```

```
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:
\n");

                    if (response.accounts() == null ||
 response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
 target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
 ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Einzelheiten zur API finden Sie unter Deaktivieren in der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`Enable`** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`Enable`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Enables AWS Inspector for the provided account(s) and default resource
types.
    *
    * @param accountIds Optional list of AWS account IDs.
    */
   public CompletableFuture<String> enableInspectorAsync(List<String>
accountIds) {

       // The resource types to enable.
       List<ResourceScanType> resourceTypes = List.of(
               ResourceScanType.EC2,
               ResourceScanType.ECR,
               ResourceScanType.LAMBDA,
               ResourceScanType.LAMBDA_CODE
       );
```

```
        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
 %s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:
\n");
```

```
                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status:
").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- Einzelheiten zur API finden Sie unter Aktivieren in der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListCoverage`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListCoverage`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
 paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
 getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
```

```
                    summary.append("    Status:
").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " +
cause.getMessage(), cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- Einzelheiten zur API finden Sie ListCoveragein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListCoverageStatistics`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListCoverageStatistics`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves and prints the coverage statistics using a paginator.
    */
   public CompletableFuture<String> listCoverageStatisticsAsync() {
       ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
               .build();

       return getAsyncClient().listCoverageStatistics(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();

                       if (cause instanceof ValidationException) {
                           throw new CompletionException(
                                   "Validation error listing coverage
statistics: %s".formatted(cause.getMessage()),
                                   cause
                           );
                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }
```

```
                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }
```

- Einzelheiten zur API finden Sie ListCoverageStatisticsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListFilters`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListFilters`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
```

```
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
 %s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- Einzelheiten zur API finden Sie ListFiltersin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListFindings`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFindings` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So listen Sie Erkenntnisse auf

Der folgende `list-findings`-Befehl listet alle generierten Erkenntnisse auf:

```
aws inspector list-findings
```

Ausgabe:

```
{
        "findingArns": [
        "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-MKkpXXPE/finding/0-HwPnsDm4",
```

```
        "arn:aws:inspector:us-west-2:123456789012:target/0-0kFIPusq/
template/0-4r1V2mAw/run/0-v5D6fI3v/finding/0-tyvmqBLy"
      ]
}
```

Weitere Informationen finden Sie unter „Amazon-Inspector-Erkenntnisse“ im Handbuch für Amazon Inspector.

- Einzelheiten zur API finden Sie ListFindingsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
```

```
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead
 of a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
    }
```

- Einzelheiten zur API finden Sie ListFindingsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListUsageTotals`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListUsageTotals`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();
```

```
        return paginator.subscribe(response -> {
                    if (response.totals() != null && !
response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
 usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage
 totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." :
 summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
 cause);
                });
    }
```

- Einzelheiten zur API finden Sie ListUsageTotalsin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für AWS IoT verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS IoT mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS IoT Entwicklerhandbuch — Weitere Informationen zu AWS IoT.
- AWS IoT API-Referenz — Details zu allen verfügbaren AWS IoT Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS IoT verwenden AWS SDKs
  - Hallo AWS IoT
  - Lernen Sie die Grundlagen von AWS IoT mit einem AWS SDK
  - Aktionen für AWS IoT verwenden AWS SDKs
    - Verwenden Sie mit AttachThingPrincipaleinem AWS SDK oder CLI
    - Verwenden Sie CreateKeysAndCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie CreateThingmit einem AWS SDK oder CLI
    - Verwenden Sie CreateTopicRulemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteThingmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteTopicRulemit einem AWS SDK oder CLI

    - Verwenden Sie DescribeEndpointmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeThingmit einem AWS SDK oder CLI
    - Verwenden Sie DetachThingPrincipalmit einem AWS SDK oder CLI
    - Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI
    - Verwenden Sie ListThingsmit einem AWS SDK oder CLI
    - Verwenden Sie SearchIndexmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateIndexingConfigurationmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateThingmit einem AWS SDK oder CLI
- Szenarien für AWS IoT verwenden AWS SDKs
  - Erste Schritte mit dem Geräteschutz für das Internet der Dinge
  - Erste Schritte mit Internet of Things-Messaging

Grundlegende Beispiele für AWS IoT verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS IoT mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS IoT
- Lernen Sie die Grundlagen von AWS IoT mit einem AWS SDK
- Aktionen für AWS IoT verwenden AWS SDKs
  - Verwenden Sie mit AttachThingPrincipaleinem AWS SDK oder CLI
  - Verwenden Sie CreateKeysAndCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie CreateThingmit einem AWS SDK oder CLI
  - Verwenden Sie CreateTopicRulemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteThingmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTopicRulemit einem AWS SDK oder CLI
  - Verwenden Sie DescribeEndpointmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeThingmit einem AWS SDK oder CLI
  - Verwenden Sie DetachThingPrincipalmit einem AWS SDK oder CLI
  - Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI

- Verwenden Sie ListThingsmit einem AWS SDK oder CLI
- Verwenden Sie SearchIndexmit einem AWS SDK oder CLI
- Verwenden Sie UpdateIndexingConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie UpdateThingmit einem AWS SDK oder CLI

Hallo AWS IoT

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS IoT beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
```

```
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
 next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
 type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
```

```
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
 scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_iot.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>
```

```
/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
```

```
                            << listThingsOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK für C++ -API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
```

```
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der API-Referenz zum AWS -SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_iot():
```

```
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])

        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- API-Details finden Sie unter ListThings in AWS SDK for Python (Boto3) API-Referenz.

Lernen Sie die Grundlagen von AWS IoT mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstelle ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.

- Aktualisiere einen AWS IoT Schatten.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche ein AWS IoT Ding.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS IoT Funktionen demonstriert werden.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/
basics/iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
```

```
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
```

```
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));

        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected
 dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
```

```
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in
the AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN
value is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
```

```
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing
 the communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for
 {thingName}? (y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
```

```
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to
Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
```

```
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));

            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value
pairs, offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web
Services account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
```

```
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5
certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
```

```
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
                        humidity = 50
                    }
                }
            });

            await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
            Console.WriteLine("Thing Shadow updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 8: Write out the state information, in JSON format
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("8. Write out the state information, in JSON
format.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
            Console.WriteLine($"Received Shadow Data: {shadowData}");
            Console.WriteLine(new string('-', 80));

            // Step 9: Set up resources (SNS topic and IAM role) and create a
rule
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("9. Set up resources and create a rule");
            Console.WriteLine();

            // Deploy CloudFormation stack to create SNS topic and IAM role
            Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

            var deployStack = !IsInteractive || GetYesNoResponse("Would you like
to deploy the CloudFormation stack? (y/n) ");
            if (deployStack)
            {
                if (IsInteractive)
                {
```

```
                    Console.Write(
                        $"Enter stack resource file path (or press Enter for
default '{_stackResourcePath}'): ");
                    var userResourcePath = Console.ReadLine();
                    if (!string.IsNullOrEmpty(userResourcePath))
                        _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS
topic: {snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM
role: {roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for
default '{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation
outputs
```

```
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and
rule creation.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
```

```
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
 {searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
 {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the
 certificate for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
 {thingName}...");
```

```
                }

                if (deleteCert?.ToLower() == "y")
                {
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");

                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));
```

```
            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
iotWrapper.DeleteTopicRuleAsync(ruleName);

                    var stackCleanupSuccess = await
DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check
the logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
{_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
execution.");

            // Cleanup on error
```

```
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
 certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
 cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
 cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during
 CloudFormation stack cleanup.");
                }
            }
```

```
            throw;
        }
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string>
{ Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await
WaitForStackCompletion(response.StackId, cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created
successfully.");
                    return true;
                }
                else
                {
```

```
                    scenarioLogger.LogError($"CloudFormation stack creation
failed: {stackName}");
                    return false;
                }
            }
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}'
already exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
```

```
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack
creation to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
```

```
   private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
   {
       try
       {
           var describeStacksRequest = new DescribeStacksRequest
           {
               StackName = stackName
           };

           var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

           if (response.Stacks.Count > 0)
           {
               var outputs = new Dictionary<string, string>();
               foreach (var output in response.Stacks[0].Outputs)
               {
                   outputs[output.OutputKey] = output.OutputValue;
               }
               return outputs;
           }

           return null;
       }
       catch (Exception ex)
       {
           scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
           return null;
       }
   }

   /// <summary>
   /// Deletes the CloudFormation stack and waits for confirmation.
   /// </summary>
   /// <param name="stackName">The name of the CloudFormation stack.</param>
   /// <param name="cloudFormationClient">The CloudFormation client.</param>
   /// <param name="scenarioLogger">The logger.</param>
   /// <returns>True if the stack was deleted successfully.</returns>
   private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
   {
```

```
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };

            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being
deleted. This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
```

```
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to
be deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
```

```
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press
 Enter for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-
a-zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Eine Wrapper-Klasse für AWS IoT SDK-Methoden.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
```

```
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };

            var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }
```

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
```

```
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
```

```
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing
failed.</returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
{ex.Message}");
            return new List<Certificate>();
        }
    }

    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
```

```
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
```

```
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT
rules");
```

```
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
```

```
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });
```

```
            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
```

```
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
 (ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
 wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
 after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
```

```
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
```

```
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
```

```
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erschaffe AWS IoT etwas.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Generieren Sie ein Gerätezertifikat und fügen Sie es an.

```
    Aws::String certificateARN;
    Aws::String certificateID;
```

```
   if (askYesNoQuestion("Would you like to create a certificate for your thing?
 (y/n) ")) {
       Aws::String outputFolder;
       if (askYesNoQuestion(
               "Would you like to save the certificate and keys to file? (y/n)
 ")) {
           outputFolder = std::filesystem::current_path();
           outputFolder += "/device_keys_and_certificates";

           std::filesystem::create_directories(outputFolder);

           std::cout << "The certificate and keys will be saved to the folder: "
                     << outputFolder << std::endl;
       }

       if (!createKeysAndCertificate(outputFolder, certificateARN,
 certificateID,
                                     clientConfiguration)) {
           std::cerr << "Exiting because createKeysAndCertificate failed."
                     << std::endl;
           cleanup(thingName, "", "", "", "", false, clientConfiguration);
           return false;
       }

       std::cout << "\nNext, the certificate will be attached to the thing.\n"
                 << std::endl;
       if (!attachThingPrincipal(certificateARN, thingName,
 clientConfiguration)) {
           std::cerr << "Exiting because attachThingPrincipal failed." <<
 std::endl;
           cleanup(thingName, certificateARN, certificateID, "", "",
                   false,
                   clientConfiguration);
           return false;
       }
   }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
```

```
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
```

```
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Führe verschiedene Operationen an dem AWS IoT Ding durch.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
 std::endl;
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
 Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
 std::endl;

    askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
    Aws::String endpoint;
    if (!describeEndpoint(endpoint, clientConfiguration)) {
        std::cerr << "Exiting because getEndpoint failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
    printAsterisksLine();

    std::cout << "Now the certificates in your account will be listed." <<
 std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!listCertificates(clientConfiguration)) {
        std::cerr << "Exiting because listCertificates failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
    std::cout << "A thing shadow refers to a feature that enables you to create a
 virtual representation, or \"shadow,\"\n"
    << "of a physical device or thing. The thing shadow allows you to synchronize
 and control the state of a device between\n"
    << "the cloud and the device itself. and the AWS IoT service. For example,
 you can write and retrieve JSON data from a thing shadow." << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
   std::cout << "Now, the state information for the shadow will be retrieved.\n"
<< std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String shadowState;
   if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
       std::cerr << "Exiting because getThingShadow failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

   printAsterisksLine();

   std::cout << "A rule with now be added to to the thing.\n" << std::endl;
   std::cout << "Any user who has permission to create rules will be able to
access data processed by the rule." << std::endl;
   std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
   std::cout << "These resources will be created using a CloudFormation
template." << std::endl;
   std::cout << "Stack creation may take a few minutes." << std::endl;

   askQuestion("Press Enter to continue: ", alwaysTrueTest);
   Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
   if (outputs.empty()) {
       std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
   auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
   auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
   if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
       std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
       "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
<< std::endl;
       cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
               false,
```

```
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
```

```
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
```

```
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
```

```
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
```

```
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
```

```
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
```

```
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Bereinigen von Ressourcen.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                              askYesNoQuestion(
                                      "Delete the CloudFormation stack '" +
 stackName +
                                      "'? (y/n) "))) {
```

```
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName
 +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
```

```
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS IoT Funktionen demonstriert werden.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
```

```
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für AWS IoT SDK-Methoden.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
```

```
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
```

```
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
```

```
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
```

```
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
     * If the request is successful, it prints a confirmation message and
additional information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn)
{
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
```

```
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();
```

```
        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
```

```
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();
```

```
        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
```

```
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
 Internet of Things (IoT) Core service.
```

```
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
```

```
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
```

```
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
```

```
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
```

```
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
```

```
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
```

```
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }
```

```
    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
```

```
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
```

```
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
```

```
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings

- SearchIndex
- UpdateIndexingConfiguration
- UpdateThing

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine IoT-Wrapper-Klasse zur Verwaltung von Vorgängen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
```

```
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response


    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
```

```
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
```

```
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
```

```
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
```

```
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response =
 self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Führen Sie ein interaktives Szenario durch, das die IoT-Grundlagen demonstriert.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True
```

```
    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)

            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
```

```
                            for output in response["Stacks"][0]["Outputs"]}
                    return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.

        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the
AWS SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between
Internet-connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
```

```
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")

            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter
to continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")

            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")
```

```
            input("\nNext, we'll get the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")

            input("\nNext, we'll list certificates. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
```

```
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)

            input("\nNext, we'll search for things. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)

                if self.rule_name:
                    print(f"Deleting topic rule: {self.rule_name}")
                    self.iot_wrapper.delete_topic_rule(self.rule_name)

                self._cleanup_stack()
                print("Resources deleted successfully.")
            except ClientError as err:
                logger.error(
                    "Cleanup failed: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
        else:
            print("Resources will remain in your account.")
```

```
        print("\n" + "-" * 88)
        print("Thanks for using AWS IoT!")
        print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Aktionen für AWS IoT verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS IoT Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS IoT API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS IoT verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS IoT -API-Referenz.

Beispiele

- Verwenden Sie mit AttachThingPrincipaleinem AWS SDK oder CLI
- Verwenden Sie CreateKeysAndCertificatemit einem AWS SDK oder CLI
- Verwenden Sie CreateThingmit einem AWS SDK oder CLI
- Verwenden Sie CreateTopicRulemit einem AWS SDK oder CLI
- Verwenden Sie DeleteCertificatemit einem AWS SDK oder CLI
- Verwenden Sie DeleteThingmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTopicRulemit einem AWS SDK oder CLI
- Verwenden Sie DescribeEndpointmit einem AWS SDK oder CLI
- Verwenden Sie DescribeThingmit einem AWS SDK oder CLI
- Verwenden Sie DetachThingPrincipalmit einem AWS SDK oder CLI
- Verwenden Sie ListCertificatesmit einem AWS SDK oder CLI
- Verwenden Sie ListThingsmit einem AWS SDK oder CLI
- Verwenden Sie SearchIndexmit einem AWS SDK oder CLI
- Verwenden Sie UpdateIndexingConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie UpdateThingmit einem AWS SDK oder CLI

Verwenden Sie mit **`AttachThingPrincipal`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachThingPrincipal` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
```

```
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So hängen Sie Ihrem Objekt ein Zertifikat an

Im folgenden `attach-thing-principal` Beispiel wird ein Zertifikat an das MyTemperatureSensor Ding angehängt. Das Zertifikat wird durch einen ARN identifiziert. Sie finden den ARN für ein Zertifikat in der AWS IoT-Konsole.

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie AttachThingPrincipalin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing
successfully.");

               // Print additional information about the Thing.
               describeThing(thingName);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               } else {
                   System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                       attachResponse.sdkHttpResponse().statusCode());
               }
```

```
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       lo_iot->attachthingprincipal(
         iv_thingname = iv_thing_name
         iv_principal = iv_principal ).
       MESSAGE |Principal attached to IoT thing '{ iv_thing_name }'.| TYPE 'I'.
     CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
       MESSAGE |Resource not found when attaching principal to
'{ iv_thing_name }'.| TYPE 'I'.
       RAISE EXCEPTION lo_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
       MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_svc_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie AttachThingPrincipalin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateKeysAndCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateKeysAndCertificate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Creates a device certificate for AWS IoT.
   /// </summary>
   /// <returns>The certificate details including ARN and certificate PEM, or
null if creation failed.</returns>
   public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
   {
       try
       {
           var request = new CreateKeysAndCertificateRequest
           {
               SetAsActive = true
           };

           var response = await
_amazonIoT.CreateKeysAndCertificateAsync(request);
```

```
            _logger.LogInformation($"Created certificate with ARN
 {response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
 response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
 {ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
```

```
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();
```

```
            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um ein RSA-Schlüsselpaar zu erstellen und ein X.509 Zertifikat auszustellen

Im Folgenden `create-keys-and-certificate` wird ein 2048-Bit-RSA-Schlüsselpaar erstellt und ein X.509 Zertifikat unter Verwendung des ausgegebenen öffentlichen Schlüssels ausgestellt. Da dies das einzige Mal ist, dass AWS IoT den privaten Schlüssel für dieses Zertifikat bereitstellt, sollten Sie es an einem sicheren Ort aufbewahren.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Ausgabe:

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
```

```
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Weitere Informationen finden Sie unter Erstellen und Registrieren eines AWS IoT-Gerätezertifikats im AWS IoT Developer Guide.

- Einzelheiten zur API finden Sie CreateKeysAndCertificateunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT
certificate.
    * If the request is successful, it prints the certificate details and
returns the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();
```

```
                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
```

```
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response =
self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
```

```
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        oo_result = lo_iot->createkeysandcertificate( iv_setasactive =
abap_true ).
        MESSAGE |Certificate created: { oo_result->get_certificateid( ) }| TYPE
'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateThing`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateThing` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
```

```
    /// <returns>The ARN of the Thing created, or null if creation failed.</
returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
 {response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
 {ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie CreateThingin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateThingin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Objektdatensatz in der Registry

Das folgende `create-thing` Beispiel erstellt einen Eintrag für ein Gerät in der AWS IoT-Dingregistrierung.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Ausgabe:

```
{
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Beispiel 2: So definieren Sie ein Objekt, das einem Objekttyp zugeordnet ist

Im folgenden Beispiel für `create-thing` wird ein Objekt mit dem angegebenen Objekttyp und seinen Attributen erstellt.

```
aws iot create-thing \
    --thing-name "MyLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Ausgabe:

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry und Objekttypen im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie CreateThingunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " +
cause.getMessage());
               }
           }
       });

       future.join();
   }
```

- Einzelheiten zur API finden Sie CreateThingin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Einzelheiten zur API finden Sie CreateThingin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
```

```
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie CreateThingin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       oo_result = lo_iot->creatething(
         iv_thingname = iv_thing_name ).
       MESSAGE |IoT thing created: { oo_result->get_thingname( ) } ARN:
{ oo_result->get_thingarn( ) }| TYPE 'I'.
     CATCH /aws1/cx_iotresrcalrdyexistsex.
       MESSAGE |IoT thing '{ iv_thing_name }' already exists.| TYPE 'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateThingin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateTopicRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTopicRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</
param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
 snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
```

```
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CreateTopicRulein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTopicRulein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Regel, die eine Amazon-SNS-Warnung sendet

Im folgenden Beispiel für `create-topic-rule` wird eine Regel erstellt, die eine Amazon-SNS-Nachricht sendet, wenn die Werte für die Erdfeuchte in einem Geräteschatten niedrig sind.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

Für das Beispiel muss der folgende JSON-Code in einer Datei mit dem Namen `plant-rule.json` gespeichert werden:

```
{
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen einer AWS IoT-Regel im AWS IoT-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie CreateTopicRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
```

```
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie CreateTopicRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIoTRule(
```

```
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicRulein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
```

```
                        {"sns": {"targetArn": sns_action_arn, "roleArn":
role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateTopicRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Build the SNS action that will receive messages matching the rule.
    DATA lo_sns_action TYPE REF TO /aws1/cl_iotsnsaction.
    CREATE OBJECT lo_sns_action
      EXPORTING
```

```
            iv_targetarn = iv_sns_action_arn
            iv_rolearn   = iv_role_arn.

        DATA lo_action TYPE REF TO /aws1/cl_iotaction.
        CREATE OBJECT lo_action
          EXPORTING
            io_sns = lo_sns_action.

        DATA lt_actions TYPE /aws1/cl_iotaction=>tt_actionlist.
        APPEND lo_action TO lt_actions.

        " iv_topic = 'my/iot/topic' - The MQTT topic pattern to match
        DATA lo_payload TYPE REF TO /aws1/cl_iottopicrulepayload.
        CREATE OBJECT lo_payload
          EXPORTING
            iv_sql     = |SELECT * FROM '{ iv_topic }'|
            it_actions = lt_actions.

        lo_iot->createtopicrule(
          iv_rulename         = iv_rule_name
          io_topicrulepayload = lo_payload ).
        MESSAGE |IoT topic rule created: { iv_rule_name }| TYPE 'I'.
      CATCH /aws1/cx_iotresrcalrdyexistsex.
        MESSAGE |Topic rule '{ iv_rule_name }' already exists.| TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTopicRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteCertificate`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCertificate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie ein Gerätezertifikat

Im folgenden Beispiel für `delete-certificate` wird das Gerätezertifikat mit der angegebenen ID gelöscht.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie DeleteCertificatein der AWS IoT-API-Referenz.

- Einzelheiten zur API finden Sie DeleteCertificateunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteCertificatein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
```

```
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteCertificatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Certificates must be deactivated before they can be deleted.
    lo_iot->updatecertificate(
      iv_certificateid = iv_certificate_id
      iv_newstatus     = 'INACTIVE' ).
    lo_iot->deletecertificate( iv_certificateid = iv_certificate_id ).
    MESSAGE |Certificate deleted: { iv_certificate_id }| TYPE 'I'.
  CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |Certificate '{ iv_certificate_id }' not found.| TYPE 'I'.
    RAISE EXCEPTION lo_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
    MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_svc_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteCertificatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteThing`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteThing` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteThingin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteThingin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So zeigen Sie detaillierte Informationen zu einem Objekt an

Das folgende `delete-thing` Beispiel löscht eine Sache aus der AWS IoT-Registrierung für Ihr AWS Konto.

as iot delete-thing --thing-name“ FourthBulb

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. DeleteThingAWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes an IoT Thing asynchronously.
 *
 * @param thingName The name of the IoT Thing to delete.
 *
 * This method initiates an asynchronous request to delete an IoT Thing.
 * If the deletion is successful, it prints a confirmation message.
 * If an exception occurs, it prints the error message.
 */
public void deleteIoTThing(String thingName) {
    DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
        .thingName(thingName)
        .build();
```

```
        CompletableFuture<DeleteThingResponse> future =
 getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie DeleteThingin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
```

```
}
```

- Einzelheiten zur API finden Sie DeleteThingin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteThingin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletething( iv_thingname = iv_thing_name ).
    MESSAGE |IoT thing deleted: { iv_thing_name }| TYPE 'I'.
  CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |IoT thing '{ iv_thing_name }' not found.| TYPE 'I'.
    RAISE EXCEPTION lo_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
    MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_svc_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteThingin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteTopicRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTopicRule` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTopicRulein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine Regel

Im folgenden Beispiel für `delete-topic-rule` wird die angegebene Regel gelöscht.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen einer Regel im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie DeleteTopicRulein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
```

```
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteTopicRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        lo_iot->deletetopicrule( iv_rulename = iv_rule_name ).
        MESSAGE |IoT topic rule deleted: { iv_rule_name }| TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTopicRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeEndpoint`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEndpoint` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
```

```
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeEndpointin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe the endpoint specific to the AWS account making the call.
```

```
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeEndpointin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: Um Ihren aktuellen AWS Endpunkt zu ermitteln

Im folgenden `describe-endpoint` Beispiel wird der AWS Standardendpunkt abgerufen, auf den alle Befehle angewendet werden.

```
aws iot describe-endpoint
```

Ausgabe:

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Weitere Informationen finden Sie DescribeEndpointim AWS IoT Developer Guide.

Beispiel 2: So rufen Sie Ihren ATS-Endpunkt ab

Im folgenden Beispiel für `describe-endpoint` wird der Amazon Trust Services (ATS)-Endpunkt abgerufen.

```
aws iot describe-endpoint \
    --endpoint-type iot:Data-ATS
```

Ausgabe:

```
{
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Weitere Informationen finden Sie unter X.509 Zertifikate und AWS IoT im AWS IoT Developer Guide.

- Einzelheiten zur API finden Sie DescribeEndpointin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Einzelheiten zur API finden Sie DescribeEndpointin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Einzelheiten zur API finden Sie DescribeEndpointin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response =
 self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- Einzelheiten zur API finden Sie DescribeEndpointin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeEndpointin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   " iv_endpoint_type = 'iot:Data-ATS' - Endpoint type for data operations
   TRY.
       DATA(lo_result) = lo_iot->describeendpoint( iv_endpointtype =
iv_endpoint_type ).
       ov_endpoint_address = lo_result->get_endpointaddress( ).
       MESSAGE |Endpoint address: { ov_endpoint_address }| TYPE 'I'.
     CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
       MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
       RAISE EXCEPTION lo_throttle_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeEndpointin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeThing`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeThing` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DescribeThingin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So zeigen Sie detaillierte Informationen zu einem Objekt an

Im folgenden `describe-thing` Beispiel werden Informationen zu einer Sache (einem Gerät) angezeigt, die in der AWS IoT-Registrierung für Ihr AWS Konto definiert ist.

als iot describe-thing --thing-name“ MyLightBulb

Ausgabe:

```
{
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
    "version": 1
}
```

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. DescribeThingAWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
```

```
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie DescribeThingin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeThingin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DetachThingPrincipal`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetachThingPrincipal` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Internet of Things-Messaging

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</
param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
 certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
 Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
```

```
            _logger.LogError($"Cannot detach certificate - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's
 why: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
```

```
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um ein Ding certificate/principal von einem Ding zu trennen

Im folgenden Beispiel für `detach-thing-principal` wird ein Zertifikat, das einen Prinzipal darstellt, aus dem angegebenen Objekt entfernt.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie DetachThingPrincipalin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed
from " + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
```

```
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       lo_iot->detachthingprincipal(
         iv_thingname = iv_thing_name
         iv_principal = iv_principal ).
       MESSAGE |Principal detached from IoT thing '{ iv_thing_name }'.| TYPE
'I'.
     CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
       MESSAGE |Resource not found when detaching principal from
'{ iv_thing_name }'.| TYPE 'I'.
       RAISE EXCEPTION lo_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
       MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_svc_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DetachThingPrincipalin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListCertificates`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListCertificates` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Lists all certificates associated with the account.
   /// </summary>
   /// <returns>List of certificate information, or empty list if listing
failed.</returns>
   public async Task<List<Certificate>> ListCertificatesAsync()
   {
       try
       {
           var request = new ListCertificatesRequest();
           var response = await _amazonIoT.ListCertificatesAsync(request);

           _logger.LogInformation($"Retrieved {response.Certificates.Count}
certificates");
           return response.Certificates;
       }
```

```
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
```

```
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: Um die in Ihrem AWS Konto registrierten Zertifikate aufzulisten

Im folgenden Beispiel für `list-certificates` werden alle Zertifikate aufgelistet, die in Ihrem Konto registriert sind. Wenn die Anzahl das standardmäßige Paging-Limit von 25 überschreitet, können Sie den Antwortwert `nextMarker` aus diesem Befehl an den nächsten Befehl übergeben, um den nächsten Ergebnisstapel zu erhalten. Wiederholen Sie den Vorgang, bis `nextMarker` keinen Wert mehr zurückgibt.

```
aws iot list-certificates
```

Ausgabe:

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
```

```
            "creationDate": 1541457693.453
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
```

```
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
```

```
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_certificates(self):
        """
        Lists AWS IoT certificates.
```

```
        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListCertificatesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Collect all certificates by following the pagination marker.
    DATA lv_marker TYPE /aws1/iotmarker.
```

```
      DATA lv_count  TYPE i.

      DO.
        oo_result = lo_iot->listcertificates( iv_marker = lv_marker ).
        lv_count  = lv_count + lines( oo_result->get_certificates( ) ).
        lv_marker = oo_result->get_nextmarker( ).
        IF lv_marker IS INITIAL.
          EXIT.
        ENDIF.
      ENDDO.

      MESSAGE |Retrieved { lv_count } IoT certificates.| TYPE 'I'.
    CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
      MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
      RAISE EXCEPTION lo_throttle_ex.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
      MESSAGE lo_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- Einzelheiten zur API finden Sie ListCertificatesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListThings`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListThings` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the
next page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- Einzelheiten zur API finden Sie ListThingsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Objekte in der Registrierung auf

Das folgende `list-things` Beispiel listet die Dinge (Geräte) auf, die in der AWS IoT-Registrierung für Ihr AWS Konto definiert sind.

```
aws iot list-things
```

Ausgabe:

```
{
    "things": [
        {
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
```

```
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
    ]
}
```

Beispiel 2: So listen Sie die definierten Objekte auf, die ein bestimmtes Attribut haben

Im folgenden Beispiel für `list-things` wird eine Liste von Objekten angezeigt, die ein Attribut mit dem Namen `wattage` haben.

```
aws iot list-things \
    --attribute-name wattage
```

Ausgabe:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
```

```
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Weitere Informationen finden Sie unter Objektverwaltung mit der Registry im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie ListThingsunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
```

```
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie ListThingsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListThingsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    " Collect all things by following the pagination token.
    DATA lv_nexttoken TYPE /aws1/iotnexttoken.
    DATA lv_count     TYPE i.

    DO.
      oo_result    = lo_iot->listthings( iv_nexttoken = lv_nexttoken ).
      lv_count     = lv_count + lines( oo_result->get_things( ) ).
      lv_nexttoken = oo_result->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    MESSAGE |Retrieved { lv_count } IoT things.| TYPE 'I'.
  CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
    MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
    RAISE EXCEPTION lo_throttle_ex.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListThingsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SearchIndex`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchIndex` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count}
Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
```

```
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting
for the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
```

```
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready
(attempt {attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if
it's ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if
(configResponse.ThingIndexingConfiguration?.ThingIndexingMode ==
ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await
_amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
```

```
                    _logger.LogInformation("Index still not ready, continuing to
 wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
 (ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
 wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready
 after configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
 why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- Einzelheiten zur API finden Sie SearchIndexin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
```

```
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
```

```
                    << std::endl;
    }
    return true;
}
```

- Einzelheiten zur API finden Sie SearchIndexin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fragen Sie den Objektindex ab

Im folgenden Beispiel für `search-index` wird der `AWS_Things`-Index nach Objekten abgefragt, die den Typ `LightBulb` haben.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Ausgabe:

```
{
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
```

```
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten der Objektindizierung im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie SearchIndexin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie SearchIndexin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Einzelheiten zur API finden Sie SearchIndexin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- Einzelheiten zur API finden Sie SearchIndexin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   " iv_query_string = 'thingName:MyThing*' - Fleet indexing query string
   TRY.
       oo_result = lo_iot->searchindex( iv_querystring = iv_query_string ).
       MESSAGE |Found { lines( oo_result->get_things( ) ) } IoT things matching
query.| TYPE 'I'.
     CATCH /aws1/cx_iotindexnotreadyex INTO DATA(lo_idx_ex).
       MESSAGE 'Fleet indexing is not ready. Enable indexing with
UpdateIndexingConfiguration first.' TYPE 'I'.
       RAISE EXCEPTION lo_idx_ex.
     CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
       MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
       RAISE EXCEPTION lo_throttle_ex.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie SearchIndexin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateIndexingConfiguration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateIndexingConfiguration` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
```

```
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateIndexingConfigurationin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktivieren Sie die Indizierung von Objekten

Im folgenden `update-indexing-configuration` Beispiel wird die Dingindizierung aktiviert, sodass die Suche nach Registrierungsdaten, Shadow-Daten und dem Status der Ding-Konnektivität mithilfe des AWS_Things-Index unterstützt wird.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwalten der Objektindizierung im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie UpdateIndexingConfigurationin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- Einzelheiten zur API finden Sie UpdateIndexingConfigurationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       DATA lo_idx_conf TYPE REF TO /aws1/cl_iotthingindexingconf.
       CREATE OBJECT lo_idx_conf
         EXPORTING
           iv_thingindexingmode = 'REGISTRY'.
       lo_iot->updateindexingconfiguration(
         io_thingindexingconf = lo_idx_conf ).
       MESSAGE 'IoT thing indexing configuration updated to REGISTRY mode.' TYPE
'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateIndexingConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateThing`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateThing` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
 string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
```

```
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateThingin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
```

```
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateThingin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So weisen Sie einem Objekt einen Objekttyp zu

Das folgende `update-thing` Beispiel ordnet ein Ding in der AWS IoT-Registrierung einem Dingtyp zu. Wenn Sie die Zuordnung vornehmen, geben Sie Werte für die durch den Objekttyp definierten Attribute an.

```
aws iot update-thing \
    --thing-name "MyOtherLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Dieser Befehl erzeugt keine Ausgabe. Verwenden Sie den `describe-thing`-Befehl, um das Ergebnis anzuzeigen.

Weitere Informationen finden Sie unter Objekttypen im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie UpdateThingin der AWS CLI Befehlsreferenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Einzelheiten zur API finden Sie UpdateThingin der API-Referenz zum AWS SDK für Kotlin.

Szenarien für AWS IoT verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS IoT mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS IoT oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Erste Schritte mit Internet of Things-Messaging

Erste Schritte mit dem Geräteschutz für das Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- IoT Device Defender Defender-Audit-Checks aktivieren
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Abhilfemaßnahme
- Wenden Sie Maßnahmen zur Risikominderung auf die Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- IoT-Protokollierung aktivieren

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}
```

```
# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
```

```
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
```

```
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
```

```
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "======================================================"
echo "Step 1: Creating required IAM roles"
echo "======================================================"

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
```

```
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "==================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "==================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
```

```
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
echo "Step 3: Running an on-demand audit"
echo "==================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
```

```
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
```

```
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi
```

```
echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")
```

```
# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "==================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
```

```
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
```

```
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
```

```
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi
```

```
# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
```

```
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
```

```
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy

- DeleteTopic
- DescribeAccountAuditConfiguration
- DescribeAuditTask
- DetachRolePolicy
- GetLoggingOptions
- GetRole
- GetV2LoggingOptions
- ListAuditFindings
- ListAuditMitigationActionsTasks
- ListMitigationActions
- ListRolePolicies
- ListTopics
- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Erste Schritte mit Internet of Things-Messaging

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- IoT-Ressourcen erstellen
- Konfigurieren Ihres Geräts
- Ausführen der Beispielanwendungen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 $LOG_FILE
```

```
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID
 --new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a
 $LOG_FILE
```

```
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your
 region." | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-
document file://iot-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn'
 --output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?
```

```
# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active
 --certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile
 $CERTS_DIR/public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
```

```
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." |
 tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-
python-v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
```

```
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?
```

```
log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
```

```
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
```

```
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint
  - DetachPolicy
  - DetachThingPrincipal
  - GetCallerIdentity
  - UpdateCertificate

Codebeispiele für AWS IoT data verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS IoT data mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS IoT data Entwicklerhandbuch — Weitere Informationen zu AWS IoT data.
- AWS IoT data API-Referenz — Details zu allen verfügbaren AWS IoT data Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS IoT data verwenden AWS SDKs
  - Aktionen für AWS IoT data verwenden AWS SDKs
    - Verwenden Sie GetThingShadowmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateThingShadowmit einem AWS SDK oder CLI

Grundlegende Beispiele für AWS IoT data verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS IoT data mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS IoT data verwenden AWS SDKs
  - Verwenden Sie GetThingShadowmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateThingShadowmit einem AWS SDK oder CLI

Aktionen für AWS IoT data verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS IoT data Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS IoT data -API-Referenz.

Beispiele

- Verwenden Sie GetThingShadowmit einem AWS SDK oder CLI
- Verwenden Sie UpdateThingShadowmit einem AWS SDK oder CLI

Verwenden Sie **`GetThingShadow`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetThingShadow` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</
returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
 {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
 {ex.Message}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie GetThingShadowin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
```

```
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetThingShadowin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie ein Objektschattendokument ab

Im folgenden Beispiel für `get-thing-shadow` wird das Objektschattendokument für das angegebene IoT-Objekt abgerufen.

```
aws iot-data get-thing-shadow \
    --thing-name MyRPi \
    output.txt
```

Der Befehl erzeugt keine Ausgabe auf dem Display, aber im Folgenden wird der Inhalt von `output.txt` angezeigt:

```
{
  "state":{
    "reported":{
    "moisture":"low"
    }
  },
  "metadata":{
    "reported":{
      "moisture":{
        "timestamp":1560269319
      }
```

```
    }
  },
  "version":1,"timestamp":1560269405
}
```

Weitere Informationen finden Sie unter Datenfluss des Geräteschatten-Service im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie GetThingShadowin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
       future.whenComplete((getThingShadowResponse, ex) -> {
           if (getThingShadowResponse != null) {
               // Extracting payload from response.
               SdkBytes payload = getThingShadowResponse.payload();
```

```
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie GetThingShadowin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
```

```
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- Einzelheiten zur API finden Sie GetThingShadowin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def get_thing_shadow(self, thing_name):
        """
```

```
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response =
self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

- Einzelheiten zur API finden Sie GetThingShadowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_iop->getthingshadow( iv_thingname = iv_thing_name ).
```

```
        " Convert xstring payload to JSON string
        DATA(lv_payload) = lo_result->get_payload( ).
        ov_shadow = /aws1/cl_rt_util=>xstring_to_string( lv_payload ).
        MESSAGE 'Thing shadow retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_iopresourcenotfoundex.
        MESSAGE 'Thing shadow not found.' TYPE 'E'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
        DATA(lv_error) = |{ lo_exception->get_text( ) }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetThingShadowin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateThingShadow`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateThingShadow` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
    {
        try
        {
```

```
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };

            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateThingShadowin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update the shadow of an AWS IoT thing.
/*!
```

```
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateThingShadowin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So aktualisieren Sie einen Objektschatten

Im folgenden Beispiel für `update-thing-shadow` wird der aktuelle Zustand des Geräteschattens für das angegebene Objekt geändert und in der Datei `output.txt` gespeichert.

```
aws iot-data update-thing-shadow \
    --thing-name MyRPi \
    --payload "{"state":{"reported":{"moisture":"okay"}}}" \
    "output.txt"
```

Der Befehl erzeugt keine Ausgabe auf dem Display, aber im Folgenden wird der Inhalt von `output.txt` angezeigt:

```
{
    "state": {
        "reported": {
            "moisture": "okay"
        }
    },
    "metadata": {
        "reported": {
            "moisture": {
                "timestamp": 1560270036
            }
        }
    },
    "version": 2,
    "timestamp": 1560270036
}
```

Weitere Informationen finden Sie unter Datenfluss des Geräteschatten-Service im Entwicklerhandbuch für AWS IoT.

- Einzelheiten zur API finden Sie UpdateThingShadowin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Einzelheiten zur API finden Sie UpdateThingShadowin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- Einzelheiten zur API finden Sie UpdateThingShadowin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie UpdateThingShadowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Convert JSON string to xstring for payload
        DATA(lv_payload) = /aws1/
cl_rt_util=>string_to_xstring( iv_shadow_state ).

        lo_iop->updatethingshadow(
          iv_thingname = iv_thing_name
          iv_payload = lv_payload ).
        MESSAGE 'Thing shadow updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_iopresourcenotfoundex.
        MESSAGE 'Thing not found.' TYPE 'E'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
        DATA(lv_error) = |{ lo_exception->get_text( ) }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateThingShadowin der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für AWS IoT SiteWise verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS IoT SiteWise mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS IoT SiteWise Entwicklerhandbuch — Weitere Informationen zu AWS IoT SiteWise.
- AWS IoT SiteWise API-Referenz — Details zu allen verfügbaren AWS IoT SiteWise Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS IoT SiteWise verwenden AWS SDKs
  - Hallo AWS IoT SiteWise
  - Lernen Sie die Grundlagen von AWS IoT SiteWise mit einem AWS SDK
  - Aktionen für AWS IoT SiteWise verwenden AWS SDKs
    - Verwenden Sie BatchPutAssetPropertyValuemit einem AWS SDK oder CLI
    - Verwenden Sie CreateAssetmit einem AWS SDK oder CLI
    - Verwenden Sie CreateAssetModelmit einem AWS SDK oder CLI
    - Verwenden Sie CreateGatewaymit einem AWS SDK oder CLI
    - Verwenden Sie DeleteAssetmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteAssetModelmit einem AWS SDK oder CLI

- Verwenden Sie DeleteGatewaymit einem AWS SDK oder CLI
- Verwenden Sie DescribeAssetModelmit einem AWS SDK oder CLI
- Verwenden Sie DescribeGatewaymit einem AWS SDK oder CLI
- Verwenden Sie GetAssetPropertyValuemit einem AWS SDK oder CLI
- Verwenden Sie ListAssetModelPropertiesmit einem AWS SDK
- Verwenden Sie ListAssetModelsmit einem AWS SDK oder CLI

Grundlegende Beispiele für AWS IoT SiteWise verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS IoT SiteWise mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS IoT SiteWise
- Lernen Sie die Grundlagen von AWS IoT SiteWise mit einem AWS SDK
- Aktionen für AWS IoT SiteWise verwenden AWS SDKs
  - Verwenden Sie BatchPutAssetPropertyValuemit einem AWS SDK oder CLI
  - Verwenden Sie CreateAssetmit einem AWS SDK oder CLI
  - Verwenden Sie CreateAssetModelmit einem AWS SDK oder CLI
  - Verwenden Sie CreateGatewaymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteAssetmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteAssetModelmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteGatewaymit einem AWS SDK oder CLI
  - Verwenden Sie DescribeAssetModelmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeGatewaymit einem AWS SDK oder CLI
  - Verwenden Sie GetAssetPropertyValuemit einem AWS SDK oder CLI
  - Verwenden Sie ListAssetModelPropertiesmit einem AWS SDK
  - Verwenden Sie ListAssetModelsmit einem AWS SDK oder CLI

Hallo AWS IoT SiteWise

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS IoT SiteWise beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest =
 ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -
> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- Einzelheiten zur API finden Sie ListAssetModelsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateListAssetModels,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new IoTSiteWiseClient();
  const listAssetModelsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
    for await (const page of paginator) {
      listAssetModelsPaginated.push(...page.assetModelSummaries);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }
  for (const { name, creationDate } of listAssetModelsPaginated) {
    console.log(`${name} - ${creationDate}`);
  }
};

// Call function if run directly.
```

```
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie ListAssetModelsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_iot_sitewise(iot_sitewise_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise
    client and list the asset models in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This
 object wraps
                               the low-level AWS IoT SiteWise service API.
    """
    print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
    paginator = iot_sitewise_client.get_paginator("list_asset_models")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    asset_model_names: [str] = []
    for page in page_iterator:
        for asset_model in page["assetModelSummaries"]:
            asset_model_names.append(asset_model["name"])
```

```
    print(f"{len(asset_model_names)} asset model(s) retrieved.")
    for asset_model_name in asset_model_names:
        print(f"\t{asset_model_name}")


if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- Einzelheiten zur API finden Sie ListAssetModelsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->listassetmodels( ). " oo_result is returned for
 testing purposes. "
        DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
        MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list asset models.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListAssetModelsin der API-Referenz zum AWS SDK für SAP ABAP.

Lernen Sie die Grundlagen von AWS IoT SiteWise mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein AWS IoT SiteWise Asset-Modell.
- Erstellen Sie ein AWS IoT SiteWise Asset.
- Rufen Sie die Eigenschafts-ID-Werte ab.
- Daten an ein AWS IoT SiteWise Asset senden.
- Ruft den Wert der Eigenschaft AWS IoT SiteWise Asset ab.
- Erstellen Sie ein AWS IoT SiteWise Portal.
- Erstellen Sie ein AWS IoT SiteWise Gateway.
- Beschreiben Sie das AWS IoT SiteWise Gateway.
- Löschen Sie die AWS IoT SiteWise Assets.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem AWS IoT SiteWise Funktionen demonstriert werden.

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
```

```
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String gatewayName = "MyGateway1" ;
        String myThing =  "MyThing1" ;

        logger.info("""
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS)
that
            makes it easy to collect, store, organize, and monitor data from
industrial equipment and processes.
            It is designed to help industrial and manufacturing organizations
collect data from their equipment and
            processes, and use that data to make informed decisions about their
operations.

            One of the key features of AWS IoT SiteWise is its ability to connect
to a wide range of industrial
            equipment and systems, including programmable logic controllers
(PLCs), sensors, and other
            industrial devices. It can collect data from these devices and
organize it into a unified data model,
            making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
            visualizing the data, setting up alarms and alerts, and generating
reports.

            Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
            It can collect and store data from thousands of devices and process
millions of data points per second,
            making it suitable for large-scale industrial operations.
Additionally, AWS IoT SiteWise is designed
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
```

```
            runScenario(assetModelName, assetName, gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName,
String gatewayName, String myThing) throws Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment.
This model provides a structured and
             hierarchical representation of these assets, allowing users to
define the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
            """);
        waitForInputToContinue(scanner);
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
```

```
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " +
cause.getMessage(), cause);
                return;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure
and metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data
from
            multiple sources, normalize the data into a standard format, and
store it in a
            centralized location. This makes it easier to analyze and gain
insights from the data.
```

```
            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process,
and analyze industrial data
            from connected equipment and sensors. One of the key benefits of
reading an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
```

```
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT
SiteWise platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {},
Error code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
```

```
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}",
response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
```

```
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
```

```
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException)
{
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
```

```
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

Eine Wrapper-Klasse für AWS IoT SiteWise SDK-Methoden.

```
public class SitewiseActions {

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
```

```
    }


    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
            .build();

        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();
```

```
        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
```

```
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
```

```
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property
value: {}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
       ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
       return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
           .handle((response, throwable) -> {
               if (response != null) {
                   return response.assetModelPropertySummaries().stream()
                       .collect(Collectors
                           .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
               } else {
                   logger.error("Error occurred while fetching property IDs:
{}.", throwable.getCause().getMessage());
                   throw (CompletionException) throwable;
               }
           });
   }

   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
```

```
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
                }
            });
    }
```

```
   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
                           return assetModelSummary.id();
                       }
                   }
                   return null;
               });
   }


   /**
    * Creates a new IoT Sitewise gateway.
    *
```

```
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");

       CreateGatewayRequest createGatewayRequest =
CreateGatewayRequest.builder()
           .gatewayName(gatewayName)
           .gatewayPlatform(platform)
           .tags(tag)
           .build();

       return getAsyncClient().createGateway(createGatewayRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Error creating the gateway.");
                   throw (CompletionException) exception;
               }
               logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
```

```
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
```

```
     *         can attach callbacks, then handle the result or exception by
 calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
 method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
 By calling
     *         {@link CompletionException#getCause()}, the calling code can
 access the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
 gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
 method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway

- DeleteAsset
- DeleteAssetModel
- DeleteGateway
- DescribeAssetModel
- DescribeGateway
- GetAssetPropertyValue
- ListAssetModelProperties
- ListAssetModels

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
```

```
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
```

```
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
 that makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes. It is designed to help industrial and manufacturing
 organizations collect data from their equipment and processes, and use that data
 to make informed decisions about their operations.
One of the key features of AWS IoT SiteWise is its ability to connect to a
 wide range of industrial equipment and systems, including programmable logic
 controllers (PLCs), sensors, and other industrial devices. It can collect data
 from these devices and organize it into a unified data model, making it easier
 to analyze and gain insights from the data. AWS IoT SiteWise also provides tools
 for visualizing the data, setting up alarms and alerts, and generating reports.
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data. It can collect and store data from thousands of devices and
 process millions of data points per second, making it suitable for large-scale
 industrial operations. Additionally, AWS IoT SiteWise is designed to be secure
 and compliant, with features like role-based access controls, data encryption,
 and integration with other AWS services for additional security and compliance
 features.

Let's get started...`,
  { header: true },
);

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "This scenario uses AWS CloudFormation to create an IAM role that is required
 for this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
```

```
        `${__dirname}/../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-
template.yml`,
        "utf8",
      );
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
  `1. Create an AWS SiteWise Asset Model
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment, processes, and systems, that exist in an industrial environment.
 This model provides a structured and hierarchical representation of these
 assets, allowing users to define the relationships and properties of each asset.

This scenario creates two asset model properties: temperature and humidity.`,
);
```

```
const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
            {
              name: "Humidity",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
          ],
        }),
      );
      state.assetModel.assetModelId = assetModelResponse.assetModelId;
      console.log(
        `Asset Model successfully created. Asset Model ID:
 ${state.assetModel.assetModelId}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceAlreadyExistsException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} already exists.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);
```

```
const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
The IoT SiteWise model that we just created defines the structure and metadata
 for your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});

const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        }),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values
```

```
To send data to an asset, we need to get the property ID values. In this
 scenario, we access the temperature and humidity property ID values.`,
);

const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      for (const retrieveResponseKey in
 retrieveResponse.assetModelPropertySummaries) {
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Humidity"
        ) {
          state.propertyIds.Humidity =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Temperature"
        ) {
          state.propertyIds.Temperature =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
      }
      console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
      console.log(
        `The Temperature propertyId is ${state.propertyIds.Temperature}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem retrieving the properties: ${caught.message}`,
        );
        throw caught;
```

```
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
  "displaySendDataToIoTSiteWiseAsset",
  `4. Send data to an AWS IoT SiteWise Asset

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple
 sources, normalize the data into a standard format, and store it in a
 centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to
 the AWS IoT SiteWise asset.`,
);

const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(
  "sdkSendDataToIoTSiteWiseAsset",
  async (state) => {
    try {
      const sendResponse = await state.iotSiteWiseClient.send(
        new BatchPutAssetPropertyValueCommand({
          entries: [
            {
              entryId: "entry-3",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Humidity,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.humidity,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
            {
              entryId: "entry-4",
```

```
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Temperature,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.temperature,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
          ],
        }),
      );
      console.log("The data was sent successfully.");
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data from connected equipment and sensors. One of the key benefits of
 reading an IoT SiteWise property is the ability to gain valuable insights from
 your industrial data.`,
);

const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (/** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
```

```
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
        }),
      );
      console.log(
        `The property value for Temperature is
 ${temperatureResponse.propertyValue.value.doubleValue}`,
      );
      console.log(
        `The property value for Humidity is
 ${humidityResponse.propertyValue.value.doubleValue}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources, such as sensors, equipment, and control systems, into a centralized
 platform.`,
);

const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (/** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
```

```
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
        `Portal created successfully. Portal ID
 ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribePortal = new ScenarioOutput(
  "displayDescribePortal",
  `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`,
);

const sdkDescribePortal = new ScenarioAction(
  "sdkDescribePortal",
  async (/** @type {State} */ state) => {
    try {
      const describePortalResponse = await state.iotSiteWiseClient.send(
        new DescribePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
```

```
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
  "displayCreateIoTSiteWiseGateway",
  `8. Create an IoT SiteWise Gateway

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors,
 and the cloud-based IoT SiteWise service. It is responsible for securely
 collecting, processing, and transmitting data from various industrial assets
 to the IoT SiteWise platform, enabling real-time monitoring, analysis, and
 optimization of industrial operations.`,
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        }),
      );
      console.log(
        `Gateway creation completed successfully. ID is
 ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
```

```
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform:
 ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
```

```
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
  "displayConfirmDeleteResources",
  async (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
    return "The resources will not be deleted. Please delete them manually to
 avoid charges.";
  },
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    await wait(10); // Give the portal status time to catch up.
    try {
      await state.iotSiteWiseClient.send(
        new DeletePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(
        `Portal ${state.portal.portalName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
      } else {
        console.log(`When trying to delete the portal: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
```

```
      console.log(
        `Gateway ${state.gateway.gatewayName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
      } else {
        console.log(`When trying to delete the gateway: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetCommand({
          assetId: state.asset.assetId,
        }),
      );
      await wait(5); // Allow the delete to finish.
      console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
      } else {
        console.log(`When deleting the asset: ${caught.message}`);
      }
    }

    await wait(30); // Allow asset deletion to finish.
    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetModelCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      console.log(
        `Asset Model ${state.assetModel.assetModelName} was deleted
successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} was not found.`,
        );
      } else {
```

```
        console.log(`When deleting the asset model: ${caught.message}`);
      }
    }

    try {
      await state.cloudFormationClient.send(
        new DeleteStackCommand({
          StackName: stackName,
        }),
      );
      await waitUntilStackDeleteComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the
 resources manually.`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
```

```
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
```

```
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class IoTSitewiseGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to manage IoT physical assets
 using
    the AWS IoT SiteWise.
    """

    def __init__(
        self,
        iot_sitewise_wrapper: IoTSitewiseWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.iot_sitewise_wrapper = iot_sitewise_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack = None
        self.asset_model_id = None
        self.asset_id = None
        self.portal_id = None
        self.gateway_id = None

    def run(self) -> None:
        """
        Runs the scenario.
        """
        print(
            """
AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes.
It is designed to help industrial and manufacturing organizations collect data
 from their equipment and
```

```
processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial
equipment and systems, including programmable logic controllers (PLCs), sensors,
 and other
industrial devices. It can collect data from these devices and organize it into a
 unified data model,
making it easier to analyze and gain insights from the data. AWS IoT SiteWise
 also provides tools for
visualizing the data, setting up alarms and alerts, and generating reports.

Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data.
It can collect and store data from thousands of devices and process millions of
 data points per second,
making it suitable for large-scale industrial operations. Additionally, AWS IoT
 SiteWise is designed
to be secure and compliant, with features like role-based access controls, data
 encryption,
and integration with other AWS services for additional security and compliance
 features.

Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(f"")
        print(
            f"Use AWS CloudFormation to create an IAM role that is required for
 this scenario."
        )
        template_file = IoTSitewiseGettingStarted.get_template_as_string()

        self.stack = self.deploy_cloudformation_stack(
            "python-iot-sitewise-basics", template_file
        )
        outputs = self.stack.outputs
        iam_role = None

        for output in outputs:
            if output.get("OutputKey") == "SitewiseRoleArn":
                iam_role = output.get("OutputValue")
```

```
        if iam_role is None:
            error_string = f"Failed to retrieve iam_role from CloudFormation
 stack."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The ARN of the IAM role is {iam_role}")
        print_dashes()
        print_dashes()
        print(f"1. Create an AWS SiteWise Asset Model")
        print(
            """
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment,
processes, and systems, that exist in an industrial environment. This model
 provides a structured and
hierarchical representation of these assets, allowing users to define the
 relationships and values
of each asset.

This scenario creates two asset model values: temperature and humidity.
        """
        )
        press_enter_to_continue()
        asset_model_name = "MyAssetModel1"
        temperature_property_name = "temperature"
        humidity_property_name = "humidity"
        try:
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
```

```
            ]
            self.asset_model_id = self.iot_sitewise_wrapper.create_asset_model(
                asset_model_name, properties
            )
            print(
                f"Asset Model successfully created. Asset Model ID:
 {self.asset_model_id}. "
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                self.asset_model_id =
 self.get_model_id_for_model_name(asset_model_name)
                print(
                    f"Asset Model {asset_model_name} already exists. Asset Model
 ID: {self.asset_model_id}. "
                )
            else:
                raise

        press_enter_to_continue()
        print_dashes()
        print(f"2. Create an AWS IoT SiteWise Asset")
        print(
            """
The IoT SiteWise model that we just created defines the structure and metadata
 for your physical assets.
Now we create an asset from the asset model.

        """
        )
        press_enter_to_continue()

        self.asset_id = self.iot_sitewise_wrapper.create_asset(
            "MyAsset1", self.asset_model_id
        )

        print(f"Asset created with ID: {self.asset_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()
        print(f"3. Retrieve the property ID values")
        print(
            """
```

```
To send data to an asset, we need to get the property ID values. In this
 scenario, we access the
temperature and humidity property ID values.
        """
        )
        press_enter_to_continue()
        property_ids = self.iot_sitewise_wrapper.list_asset_model_properties(
            self.asset_model_id
        )
        humidity_property_id = None
        temperature_property_id = None
        for property_id in property_ids:
            if property_id.get("name") == humidity_property_name:
                humidity_property_id = property_id.get("id")
            elif property_id.get("name") == temperature_property_name:
                temperature_property_id = property_id.get("id")
        if humidity_property_id is None or temperature_property_id is None:
            error_string = f"Failed to retrieve property IDs from Asset Model."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The Humidity property Id is {humidity_property_id}")
        print(f"The Temperature property Id is {temperature_property_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"4. Send data to an AWS IoT SiteWise Asset")
        print(
            """
By sending data to an IoT SiteWise Asset, you can aggregate data from
multiple sources, normalize the data into a standard format, and store it in a
centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to
 the AWS IoT SiteWise asset.

        """
        )
        press_enter_to_continue()

        values = [
            {
```

```
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
        self.iot_sitewise_wrapper.batch_put_asset_property_value(self.asset_id,
 values)
        print(f"Data sent successfully.")

        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"5. Retrieve the value of the IoT SiteWise Asset property")
        print(
            """
IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data
from connected equipment and sensors. One of the key benefits of reading an IoT
 SiteWise property
is the ability to gain valuable insights from your industrial data.

        """
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, temperature_property_id
        )
        print(f"The property name is '{temperature_property_name}'.")

        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, humidity_property_id
```

```
        )
        print(f"The property name is '{humidity_property_name}'.")
        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"6. Create an IoT SiteWise Portal")
        print(
            """
An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources,
such as sensors, equipment, and control systems, into a centralized platform.
        """
        )

        press_enter_to_continue()
        contact_email = q.ask("Enter a contact email for the portal:",
 q.non_empty)
        print("Creating the portal. The portal may take a while to become
 active.")
        self.portal_id = self.iot_sitewise_wrapper.create_portal(
            "MyPortal1", iam_role, contact_email
        )
        print(f"Portal created successfully. Portal ID {self.portal_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"7. Describe the Portal")
        print(
            """
In this step, we get a description of the portal and display the portal URL.
        """
        )
        press_enter_to_continue()
        portal_description =
 self.iot_sitewise_wrapper.describe_portal(self.portal_id)
        print(f"Portal URL: {portal_description['portalStartUrl']}")
        press_enter_to_continue()
        print_dashes()
```

```
        print_dashes()

        print(f"8. Create an IoT SiteWise Gateway")
        press_enter_to_continue()
        self.gateway_id = self.iot_sitewise_wrapper.create_gateway(
            "MyGateway1", "MyThing1"
        )
        print(f"Gateway creation completed successfully. id is
{self.gateway_id}")
        print_dashes()
        print_dashes()
        print(f"9. Describe the IoT SiteWise Gateway")
        press_enter_to_continue()

        gateway_description = self.iot_sitewise_wrapper.describe_gateway(
            self.gateway_id
        )
        print(f"Gateway Name: {gateway_description['gatewayName']}")
        print(f"Gateway ARN: {gateway_description['gatewayArn']}")
        print(f"Gateway Platform:\n{gateway_description['gatewayPlatform']}")
        print(f"Gateway Creation Date: {gateway_description['gatewayArn']}")
        print_dashes()
        print_dashes()

        print(f"10. Delete the AWS IoT SiteWise Assets")
        if q.ask("Would you like to delete the IoT SiteWise Assets? (y/n)",
q.is_yesno):
            self.cleanup()
        else:
            print(f"The resources will not be deleted.")
        print_dashes()
        print_dashes()
        print(f"This concludes the AWS IoT SiteWise Scenario")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.gateway_id is not None:
            self.iot_sitewise_wrapper.delete_gateway(self.gateway_id)
            print(f"Deleted gateway with id {self.gateway_id}.")
            self.gateway_id = None
        if self.portal_id is not None:
```

```
            self.iot_sitewise_wrapper.delete_portal(self.portal_id)
            print(f"Deleted portal with id {self.portal_id}.")
            self.portal_id = None
        if self.asset_id is not None:
            self.iot_sitewise_wrapper.delete_asset(self.asset_id)
            print(f"Deleted asset with id {self.asset_id}.")
            self.iot_sitewise_wrapper.wait_asset_deleted(self.asset_id)
            self.asset_id = None
        if self.asset_model_id is not None:
            self.iot_sitewise_wrapper.delete_asset_model(self.asset_model_id)
            print(f"Deleted asset model with id {self.asset_model_id}.")
            self.asset_model_id = None
        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `SitewiseRoles-template.yaml` AWS
CloudFormation script and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")
```

```
        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
 CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
 resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take
 a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        template_file_path = os.path.join(script_dir, "SitewiseRoles-
template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def get_model_id_for_model_name(self, model_name: str) -> str:
        """
        Returns the model ID for the given model name.

        :param model_name: The name of the model.
        :return: The model ID.
        """
        model_id = None
        asset_models = self.iot_sitewise_wrapper.list_asset_models()
        for asset_model in asset_models:
            if asset_model["name"] == model_name:
```

```
                model_id = asset_model["id"]
                break
        return model_id
```

IoTSitewiseWrapper Klasse, die AWS IoT SiteWise Aktionen umschließt.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                              access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.
```

```
        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model
description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
```

```
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def list_asset_model_properties(self, asset_model_id: str) -> List[Dict[str,
Any]]:
        """
        Lists all AWS IoT SiteWise Asset Model Properties.

        :param asset_model_id: The ID of the asset model to list values for.
        :return: A list of dictionaries containing information about each asset
model property.
```

```
        """
        try:
            asset_model_properties = []
            paginator = self.iotsitewise_client.get_paginator(
                "list_asset_model_properties"
            )
            pages = paginator.paginate(assetModelId=asset_model_id)
            for page in pages:

 asset_model_properties.extend(page["assetModelPropertySummaries"])
            return asset_model_properties
        except ClientError as err:
            logger.error(
                "Error listing asset model values. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)

 self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
```

```
            raise


    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
```

```
                ],
            }
            entries.append(entry)
        return entries


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id,
property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def create_portal(
        self, portal_name: str, iam_role_arn: str, portal_contact_email: str
    ) -> str:
        """
        Creates an AWS IoT SiteWise Portal.

        :param portal_name: The name of the portal to create.
        :param iam_role_arn: The ARN of an IAM role.
        :param portal_contact_email: The contact email of the portal.
```

```
        :return: The ID of the created portal.
        """
        try:
            response = self.iotsitewise_client.create_portal(
                portalName=portal_name,
                roleArn=iam_role_arn,
                portalContactEmail=portal_contact_email,
            )
            portal_id = response["portalId"]
            waiter = self.iotsitewise_client.get_waiter("portal_active")
            waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
            return portal_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Portal %s already exists.", portal_name)
            else:
                logger.error(
                    "Error creating portal %s. Here's why %s",
                    portal_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_portal(self, portal_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to describe.
        :return: A dictionary containing information about the portal.
        """
        try:
            response =
self.iotsitewise_client.describe_portal(portalId=portal_id)
            return response
        except ClientError as err:
            logger.error(
                "Error describing portal %s. Here's why %s",
                portal_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
```

```
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_portal(self, portal_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to delete.
        """
        try:
            self.iotsitewise_client.delete_portal(portalId=portal_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Portal %s does not exist.", portal_id)
            else:
                logger.error(
                    "Error deleting portal %s. Here's why %s",
                    portal_id,
```

```
                    err.response["Error"]["Message"],
                )
            raise


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:

self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise


    def wait_asset_deleted(self, asset_id: str) -> None:
        """
        Waits for an AWS IoT SiteWise Asset to be deleted.
```

```
        :param asset_id: The ID of the asset to wait for.
        """
        try:
            waiter = self.iotsitewise_client.get_waiter("asset_not_exists")
            waiter.wait(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error waiting for asset %s to be deleted. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Aktionen für AWS IoT SiteWise verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS IoT SiteWise Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS IoT SiteWise -API-Referenz.

Beispiele

- Verwenden Sie BatchPutAssetPropertyValuemit einem AWS SDK oder CLI
- Verwenden Sie CreateAssetmit einem AWS SDK oder CLI
- Verwenden Sie CreateAssetModelmit einem AWS SDK oder CLI
- Verwenden Sie CreateGatewaymit einem AWS SDK oder CLI
- Verwenden Sie DeleteAssetmit einem AWS SDK oder CLI
- Verwenden Sie DeleteAssetModelmit einem AWS SDK oder CLI
- Verwenden Sie DeleteGatewaymit einem AWS SDK oder CLI
- Verwenden Sie DescribeAssetModelmit einem AWS SDK oder CLI
- Verwenden Sie DescribeGatewaymit einem AWS SDK oder CLI
- Verwenden Sie GetAssetPropertyValuemit einem AWS SDK oder CLI
- Verwenden Sie ListAssetModelPropertiesmit einem AWS SDK
- Verwenden Sie ListAssetModelsmit einem AWS SDK oder CLI

Verwenden Sie **`BatchPutAssetPropertyValue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `BatchPutAssetPropertyValue` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So senden Sie Daten an Komponenteneigenschaften

Im folgenden Beispiel für `batch-put-asset-property-value` werden Leistungs- und Temperaturdaten an die Eigenschaften der Komponente gesendet, die durch Eigenschaftsaliase identifiziert sind.

```
aws iotsitewise batch-put-asset-property-value \
    --cli-input-json file://batch-put-asset-property-value.json
```

Inhalt von `batch-put-asset-property-value.json`:

```
{
    "entries": [
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-power",
            "propertyAlias": "company-windfarm-3-turbine-7-power",
            "propertyValues": [
                {
                    "value": {
                        "doubleValue": 4.92
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    },
                    "quality": "GOOD"
                }
            ]
        },
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-temperature",
            "propertyAlias": "company-windfarm-3-turbine-7-temperature",
            "propertyValues": [
                {
                    "value": {
                        "integerValue": 38
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    }
                }
            ]
        }
    ]
}
```

Ausgabe:

```
{
```

```
    "errorEntries": []
}
```

Weitere Informationen finden Sie unter Daten mithilfe der AWS SiteWise IoT-API aufnehmen im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie BatchPutAssetPropertyValuein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
```

```
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();

        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    logger.error("An exception occurred: {}",
 exception.getCause().getMessage());
                }
            });
    }
```

- Einzelheiten zur API finden Sie BatchPutAssetPropertyValuein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  BatchPutAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Batch put asset property values.
 * @param {{ entries : array }}
 */
export const main = async ({ entries }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new BatchPutAssetPropertyValueCommand({
        entries: entries,
      }),
    );
    console.log("Asset properties batch put successfully.");
    return result;
  } catch (caught) {
```

```
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(`${caught.message}. A resource could not be found.`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie BatchPutAssetPropertyValuein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                              access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
```

```
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)

 self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Eine Hilfsfunktion zum Generieren des Parameters „entries“ aus einer Werteliste.

```
    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
```

```
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries
```

Hier ist ein Beispiel für eine Werteliste, die an die Hilfsfunktion übergeben werden kann.

```
        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- Einzelheiten zur API finden Sie BatchPutAssetPropertyValuein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ios->batchputassetpropertyvalue(
          it_entries = it_entries
        ).
        MESSAGE 'Data sent to IoT SiteWise asset successfully.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Asset does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie BatchPutAssetPropertyValuein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateAsset`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAsset` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie eine Komponente

Im folgenden Beispiel für `create-asset` wird aus dem Komponentenmodell einer Windturbine eine Windturbinenkomponente erstellt.

```
aws iotsitewise create-asset \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
    --asset-name "Wind Turbine 1"
```

Ausgabe:

```
{
    "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetStatus": {
        "state": "CREATING"
    }
}
```

Weitere Informationen finden Sie im AWS SiteWise IoT-Benutzerhandbuch unter Assets erstellen.

- Einzelheiten zur API finden Sie CreateAssetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an asset with the specified name and asset model Id.
    *
    * @param assetName    the name of the asset to create.
    * @param assetModelId the Id of the asset model to associate with the asset.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
           .assetDescription("Created using the AWS SDK for Java")
           .assetName(assetName)
           .build();

       return getAsyncClient().createAsset(createAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
```

```
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateAssetin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetCommand({
        assetName: assetName, // The name to give the Asset.
        assetModelId: assetModelId, // The ID of the asset model from which to
 create the asset.
      }),
    );
    console.log("Asset created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
```

```
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateAssetin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie CreateAssetin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->createasset(
      iv_assetname = iv_asset_name
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise asset created' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset model does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAssetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateAssetModel`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAssetModel` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie ein Komponentenmodell

Im folgenden Beispiel für `create-asset-model` wird ein Komponentenmodell erstellt, das eine Windturbine mit den folgenden Eigenschaften definiert:

Seriennummer — Die Seriennummer einer WindturbineErzeugter Strom — Der erzeugte Energiedatenstrom einer WindturbineTemperatur C — Der Temperaturdatenstrom einer Windturbine in CelsiusTemperature F — Die abgebildeten Temperaturdatenpunkte von Celsius bis Fahrenheit

```
aws iotsitewise create-asset-model \
    --cli-input-json file://create-wind-turbine-model.json
```

Inhalt von `create-wind-turbine-model.json`:

```
{
    "assetModelName": "Wind Turbine Model",
    "assetModelDescription": "Represents a wind turbine",
    "assetModelProperties": [
        {
            "name": "Serial Number",
            "dataType": "STRING",
            "type": {
                "attribute": {}
            }
        },
        {
            "name": "Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "measurement": {}
            }
        },
        {
            "name": "Temperature C",
            "dataType": "DOUBLE",
            "unit": "Celsius",
            "type": {
                "measurement": {}
            }
        },
        {
            "name": "Temperature F",
            "dataType": "DOUBLE",
            "unit": "Fahrenheit",
            "type": {
```

```
                "transform": {
                    "expression": "temp_c * 9 / 5 + 32",
                    "variables": [
                        {
                            "name": "temp_c",
                            "value": {
                                "propertyId": "Temperature C"
                            }
                        }
                    ]
                }
            }
        },
        {
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "Generated Power"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        }
    ]
}
```

Ausgabe:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
```

```
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelStatus": {
        "state": "CREATING"
    }
}
```

Weitere Informationen finden Sie unter Definieren von Asset-Modellen im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateAssetModelin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an asset model.
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
```

```
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
            .build();

        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();

        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateAssetModelin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset Model.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetModelName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetModelCommand({
        assetModelName: assetModelName, // The name to give the Asset Model.
      }),
    );
    console.log("Asset model created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the asset model.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateAssetModelin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model
description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise
```

Hier finden Sie ein Beispiel für eine Eigenschaftenliste, die an die Funktion übergeben wird.

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
```

```
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- Einzelheiten zur API finden Sie CreateAssetModelin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->createassetmodel(
      iv_assetmodelname = iv_asset_model_name
      iv_assetmodeldescription = 'This is a sample asset model description.'
      it_assetmodelproperties = it_properties
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise asset model created' TYPE 'I'.
  CATCH /aws1/cx_iosresrcalrdyexistsex.
    MESSAGE 'Asset model already exists.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAssetModelin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateGateway`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGateway` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Um ein Gateway zu erstellen

Das folgende `create-gateway` Beispiel erstellt ein Gateway, das auf AWS IoT Greengrass läuft.

```
aws iotsitewise create-gateway \
    --gateway-name ExampleCorpGateway \
    --gateway-platform greengrass={groupArn=arn:aws:greengrass:us-
west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE}
```

Ausgabe:

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

Weitere Informationen finden Sie unter Konfiguration eines Gateways im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateGatewayin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new IoT Sitewise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");
```

```
        CreateGatewayRequest createGatewayRequest =
 CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
 response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- Einzelheiten zur API finden Sie CreateGatewayin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create a Gateway.
 * @param {{  }}
```

```
 */
export const main = async ({ gatewayName }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateGatewayCommand({
        gatewayName: gatewayName, // The name to give the created Gateway.
      }),
    );
    console.log("Gateway created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the Gateway.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateGatewayin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
```

```
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
```

```
        else:
            logger.error(
                "Error creating gateway %s. Here's why %s",
                gateway_name,
                err.response["Error"]["Message"],
            )
        raise
```

- Einzelheiten zur API finden Sie CreateGatewayin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->creategateway(
      iv_gatewayname = iv_gateway_name
      io_gatewayplatform = NEW /aws1/cl_iosgatewayplatform(
        io_greengrassv2 = NEW /aws1/cl_iosgreengrassv2(
          iv_coredevicethingname = iv_core_device_thing_name
        )
      )
      it_tags = VALUE /aws1/cl_iostagmap_w=>tt_tagmap(
        (
          VALUE /aws1/cl_iostagmap_w=>ts_tagmap_maprow(
            key = 'Environment'
            value = NEW /aws1/cl_iostagmap_w( 'Production' )
          )
        )
      )
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise gateway created' TYPE 'I'.
```

```
      CATCH /aws1/cx_iosresrcalrdyexistsex.
        MESSAGE 'Gateway already exists.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateGatewayin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteAsset`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAsset` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So löschen Sie ein Asset

Im folgenden Beispiel für `delete-asset` wird eine Windturbinen-Komponente gelöscht.

```
aws iotsitewise delete-asset \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

Ausgabe:

```
{
    "assetStatus": {
        "state": "DELETING"
    }
}
```

Weitere Informationen finden Sie unter Löschen von Assets im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAssetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteAsset in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset.
 * @param {{ assetId : string }}
 */
export const main = async ({ assetId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetCommand({
        assetId: assetId, // The model id to delete.
      }),
    );
    console.log("Asset deleted successfully.");
    return { assetDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteAssetin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteAssetin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ios->deleteasset(
          iv_assetid = iv_asset_id
        ).
        MESSAGE 'IoT SiteWise asset deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to delete asset.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAsset in der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteAssetModel`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAssetModel` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So löschen Sie ein Komponentenmodell

Im folgenden Beispiel für `delete-asset-model` wird ein Windturbinen-Komponentenmodell gelöscht.

```
aws iotsitewise delete-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

Ausgabe:

```
{
    "assetModelStatus": {
        "state": "DELETING"
    }
}
```

Weitere Informationen finden Sie unter Löschen von Asset-Modellen im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAssetModel in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteAssetModelin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetModelCommand({
        assetModelId: assetModelId, // The model id to delete.
      }),
    );
    console.log("Asset model deleted successfully.");
    return { assetModelDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset model.`,
      );
    } else {
```

```
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteAssetModelin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:

self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteAssetModelin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ios->deleteassetmodel(
```

```
          iv_assetmodelid = iv_asset_model_id
        ).
        MESSAGE 'IoT SiteWise asset model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to delete asset model.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAssetModelin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteGateway`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteGateway` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Um ein Gateway zu löschen

Im folgenden Beispiel für `delete-gateway` wird ein Gateway gelöscht.

```
aws iotsitewise delete-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Daten mithilfe eines Gateways aufnehmen im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteGatewayin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }
```

- Einzelheiten zur API finden Sie DeleteGatewayin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway deleted successfully.");
    return { gatewayDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the
 Gateway Id.`,
      );
    } else {
      throw caught;
```

```
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteGatewayin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
```

```
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie DeleteGatewayin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ios->deletegateway(
```

```
          iv_gatewayid = iv_gateway_id
        ).
        MESSAGE 'IoT SiteWise gateway deleted.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteGatewayin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeAssetModel`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAssetModel` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie ein Komponentenmodell

Im folgenden Beispiel für `describe-asset-model` wird das Windpark-Komponentenmodell beschrieben.

```
aws iotsitewise describe-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

Ausgabe:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelName": "Wind Farm Model",
    "assetModelDescription": "Represents a wind farm that comprises many wind
 turbines",
```

```
    "assetModelProperties": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "a1b2c3d4-5678-90ab-
cdef-66666EXAMPLE",
                                "hierarchyId": "a1b2c3d4-5678-90ab-
cdef-77777EXAMPLE"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
            "name": "Region",
            "dataType": "STRING",
            "type": {
                "attribute": {
                    "defaultValue": " "
                }
            }
        }
    ],
    "assetModelHierarchies": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
            "name": "Wind Turbines",
            "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
```

```
        }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
        "state": "ACTIVE"
    }
}
```

Weitere Informationen finden Sie unter Beschreibung eines bestimmten Asset-Modells im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAssetModelin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
```

```
    public CompletableFuture<Map<String, String>> getPropertyIds(String
 assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
 ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
 AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
 {}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- Einzelheiten zur API finden Sie DescribeAssetModelin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DescribeAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset model.
```

```
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { assetModelDescription } = await client.send(
      new DescribeAssetModelCommand({
        assetModelId: assetModelId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset model information retrieved successfully.");
    return { assetModelDescription: assetModelDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeAssetModelin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DescribeGateway`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeGateway` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie ein Gateway

Im folgenden Beispiel für `describe-gateway` wird ein Gateway beschrieben.

```
aws iotsitewise describe-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

Ausgabe:

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayName": "ExampleCorpGateway",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayPlatform": {
        "greengrass": {
            "groupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/
groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE"
        }
    },
    "gatewayCapabilitySummaries": [
        {
            "capabilityNamespace": "iotsitewise:opcuacollector:1",
            "capabilitySyncStatus": "IN_SYNC"
        }
    ],
    "creationDate": 1588369971.457,
    "lastUpdateDate": 1588369971.457
}
```

Weitere Informationen finden Sie unter Daten mithilfe eines Gateways aufnehmen im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeGatewayin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }
```

- Einzelheiten zur API finden Sie DescribeGatewayin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DescribeGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { gatewayDescription } = await client.send(
      new DescribeGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway information retrieved successfully.");
    return { gatewayDescription: gatewayDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the
 Gateway Id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DescribeGatewayin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
 AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie DescribeGatewayin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->describegateway(
          iv_gatewayid = iv_gateway_id
        ). " oo_result is returned for testing purposes. "
```

```
        MESSAGE 'Retrieved gateway description.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeGatewayin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetAssetPropertyValue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetAssetPropertyValue` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie den aktuellen Wert einer Komponenteneigenschaft ab

Im folgenden Beispiel für `get-asset-property-value` wird die aktuelle Gesamtleistung einer Windturbinenkomponente abgerufen.

```
aws iotsitewise get-asset-property-value \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
    --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE
```

Ausgabe:

```
{
    "propertyValue": {
        "value": {
            "doubleValue": 6890.8677520453875
        },
        "timestamp": {
            "timeInSeconds": 1580853000,
            "offsetInNanos": 0
```

```
        },
        "quality": "GOOD"
    }
}
```

Weitere Informationen finden Sie unter Abfragen aktueller Objekteigenschaftswerte im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetAssetPropertyValue AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Fetches the value of an asset property.
    *
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
```

```
                .propertyId(propId)
                .assetId(assetId)
                .build();

        return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
                .handle((response, exception) -> {
                    if (exception != null) {
                        logger.error("Error occurred while fetching property
 value: {}.", exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    return response.propertyValue().value().doubleValue();
                });
    }
```

- Einzelheiten zur API finden Sie GetAssetPropertyValuein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset property value.
 * @param {{ entryId : string }}
 */
export const main = async ({ entryId }) => {
  const client = new IoTSiteWiseClient({});
```

```
  try {
    const result = await client.send(
      new GetAssetPropertyValueCommand({
        entryId: entryId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset property information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset property entry could not be found. Please
 check the entry id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie GetAssetPropertyValuein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.
```

```
        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                           access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id,
property_id
                )
            else:
                logger.error(
```

```
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie GetAssetPropertyValuein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->getassetpropertyvalue(
      iv_assetid = iv_asset_id
      iv_propertyid = iv_property_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'Retrieved asset property value.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset or property does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetAssetPropertyValuein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAssetModelProperties`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListAssetModelProperties`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->listassetmodelproperties(
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    DATA(lt_properties) = oo_result->get_assetmodelpropertysums( ).
    MESSAGE 'Retrieved list of asset model properties.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list asset model properties.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListAssetModelPropertiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAssetModels`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAssetModels` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So listen Sie alle Komponentenmodelle auf

Das folgende `list-asset-models` Beispiel listet alle Asset-Modelle auf, die in Ihrem AWS Konto in der aktuellen Region definiert sind.

```
aws iotsitewise list-asset-models
```

Ausgabe:

```
{
    "assetModelSummaries": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "name": "Wind Farm Model",
            "description": "Represents a wind farm that comprises many wind
 turbines",
            "creationDate": 1575671284.0,
            "lastUpdateDate": 1575671988.0,
            "status": {
                "state": "ACTIVE"
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "name": "Wind Turbine Model",
            "description": "Represents a wind turbine manufactured by Example
 Corp",
            "creationDate": 1575671207.0,
            "lastUpdateDate": 1575686273.0,
            "status": {
                "state": "ACTIVE"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Auflisten aller Asset-Modelle im AWS SiteWise IoT-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListAssetModelsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
```

```
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }
```

- Einzelheiten zur API finden Sie ListAssetModelsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ListAssetModelsCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ assetModelTypes : array }}
 */
export const main = async ({ assetModelTypes = [] }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new ListAssetModelsCommand({
        assetModelTypes: assetModelTypes, // The model types to list
      }),
    );
    console.log("Asset model types retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
```

```
      console.warn(
        `${caught.message}. There was a problem listing the asset model types.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie ListAssetModelsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
```

```
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListAssetModelsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_ios->listassetmodels( ). " oo_result is returned for
testing purposes. "
       DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
       MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list asset models.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ListAssetModelsin der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Amazon Keyspaces mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Keyspaces (für Apache Cassandra) mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Keyspaces – Weitere Informationen zu Amazon Keyspaces.
- API-Referenz für Amazon Keyspaces – Details zu allen verfügbaren Amazon-Keyspaces-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Keyspaces AWS SDKs
  - Hello Amazon Keyspaces
  - Lernen Sie die Grundlagen von Amazon Keyspaces mit einem AWS SDK
  - Aktionen für Amazon Keyspaces mit AWS SDKs
    - Verwenden Sie CreateKeyspacemit einem AWS SDK
    - Verwenden Sie CreateTablemit einem AWS SDK
    - Verwenden Sie DeleteKeyspacemit einem AWS SDK
    - Verwenden Sie DeleteTablemit einem AWS SDK
    - Verwenden Sie GetKeyspacemit einem AWS SDK
    - Verwenden Sie GetTablemit einem AWS SDK
    - Verwenden Sie ListKeyspacesmit einem AWS SDK
    - Verwenden Sie ListTablesmit einem AWS SDK
    - Verwenden Sie RestoreTablemit einem AWS SDK
    - Verwenden Sie UpdateTablemit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Keyspaces AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Keyspaces (für Apache Cassandra) mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Keyspaces
- Lernen Sie die Grundlagen von Amazon Keyspaces mit einem AWS SDK
- Aktionen für Amazon Keyspaces mit AWS SDKs
  - Verwenden Sie CreateKeyspacemit einem AWS SDK
  - Verwenden Sie CreateTablemit einem AWS SDK
  - Verwenden Sie DeleteKeyspacemit einem AWS SDK

- Verwenden Sie DeleteTablemit einem AWS SDK
- Verwenden Sie GetKeyspacemit einem AWS SDK
- Verwenden Sie GetTablemit einem AWS SDK
- Verwenden Sie ListKeyspacesmit einem AWS SDK
- Verwenden Sie ListTablesmit einem AWS SDK
- Verwenden Sie RestoreTablemit einem AWS SDK
- Verwenden Sie UpdateTablemit einem AWS SDK

Hello Amazon Keyspaces

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von Amazon Keyspaces beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace KeyspacesActions;

public class HelloKeyspaces
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache
 Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient =
 host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- Einzelheiten zur API finden Sie ListKeyspacesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;
```

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
 ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeyspacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeyspacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
 Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
 wraps
                             the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
        print(ks["keyspaceName"])
        print(f"\t{ks['resourceArn']}")


if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- Einzelheiten zur API finden Sie ListKeyspacesin AWS SDK for Python (Boto3) API Reference.

Lernen Sie die Grundlagen von Amazon Keyspaces mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die zeitpunktbezogene Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.
- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using System.Security.Cryptography.X509Certificates;
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;


namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
```

```
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonKeyspaces>()
            .AddTransient<KeyspacesWrapper>()
            .AddTransient<CassandraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<KeyspacesBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var keyspacesWrapper =
 host.Services.GetRequiredService<KeyspacesWrapper>();
        var uiMethods = new UiMethods();

        var keyspaceName = configuration["KeyspaceName"];
        var tableName = configuration["TableName"];

        bool success; // Used to track the results of some operations.

        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the keyspace.
```

```
        var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

        // Wait for the keyspace to be available. GetKeyspace results in a
        // resource not found error until it is ready for use.
        try
        {
            var getKeyspaceArn = "";
            Console.Write($"Created {keyspaceName}. Waiting for it to become
available. ");
            do
            {
                getKeyspaceArn = await
keyspacesWrapper.GetKeyspace(keyspaceName);
                Console.Write(". ");
            } while (getKeyspaceArn != keyspaceArn);
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine("Waiting for keyspace to be created.");
        }

        Console.WriteLine($"\nThe keyspace {keyspaceName} is ready for use.");

        uiMethods.PressEnter();

        // Create the table.
        // First define the schema.
        var allColumns = new List<ColumnDefinition>
        {
            new ColumnDefinition { Name = "title", Type = "text" },
            new ColumnDefinition { Name = "year", Type = "int" },
            new ColumnDefinition { Name = "release_date", Type = "timestamp" },
            new ColumnDefinition { Name = "plot", Type = "text" },
        };

        var partitionKeys = new List<PartitionKey>
        {
            new PartitionKey { Name = "year", },
            new PartitionKey { Name = "title" },
        };

        var tableSchema = new SchemaDefinition
        {
            AllColumns = allColumns,
```

```
            PartitionKeys = partitionKeys,
        };

        var tableArn = await keyspacesWrapper.CreateTable(keyspaceName,
tableSchema, tableName);

        // Wait for the table to be active.
        try
        {
            var resp = new GetTableResponse();
            Console.Write("Waiting for the new table to be active. ");
            do
            {
                try
                {
                    resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);
                    Console.Write(".");
                }
                catch (ResourceNotFoundException)
                {
                    Console.Write(".");
                }
            } while (resp.Status != TableStatus.ACTIVE);

            // Display the table's schema.
            Console.WriteLine($"\nTable {tableName} has been created in
{keyspaceName}");
            Console.WriteLine("Let's take a look at the schema.");
            uiMethods.DisplayTitle("All columns");
            resp.SchemaDefinition.AllColumns.ForEach(column =>
            {
                Console.WriteLine($"{column.Name,-40}\t{column.Type,-20}");
            });

            uiMethods.DisplayTitle("Cluster keys");
            resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
            {

Console.WriteLine($"{clusterKey.Name,-40}\t{clusterKey.OrderBy,-20}");
            });

            uiMethods.DisplayTitle("Partition keys");
            resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
```

```
            {
                Console.WriteLine($"{partitionKey.Name}");
            });

            uiMethods.PressEnter();
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        // Access Apache Cassandra using the Cassandra drive for C#.
        var cassandraWrapper =
host.Services.GetRequiredService<CassandraWrapper>();
        var movieFilePath = configuration["MovieFile"];

        Console.WriteLine("Let's add some movies to the table we created.");
        var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);

        uiMethods.PressEnter();

        Console.WriteLine("Added the following movies to the table:");
        var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
        uiMethods.DisplayTitle("All Movies");

        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            var plot = row.GetValue<string>("plot");
            var release_date = row.GetValue<DateTime>("release_date");
            Console.WriteLine($"{release_date}\t{title}\t{year}\n{plot}");
            Console.WriteLine(uiMethods.SepBar);
        }

        // Update the table schema
        uiMethods.DisplayTitle("Update table schema");
        Console.WriteLine("Now we will update the table to add a boolean field
called watched.");

        // First save the current time as a UTC Date so the original
        // table can be restored later.
        var timeChanged = DateTime.UtcNow;
```

```
        // Now update the schema.
        var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
tableName);
        uiMethods.PressEnter();

        Console.WriteLine("Now let's mark some of the movies as watched.");

        // Pick some files to mark as watched.
        var movieToWatch = rows[2].GetValue<string>("title");
        var watchedMovieYear = rows[2].GetValue<int>("year");
        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();
```

```
        Console.WriteLine("We can restore the table to its previous state but
that can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")
        {
            var restoredTableName = $"{tableName}_restored";
            var restoredTableArn = await keyspacesWrapper.RestoreTable(
                keyspaceName,
                tableName,
                restoredTableName,
                timeChanged);
            // Loop and call GetTable until the table is gone. Once it has been
            // deleted completely, GetTable will raise a
ResourceNotFoundException.
            bool wasRestored = false;

            try
            {
                do
                {
                    var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                    wasRestored = (resp.Status == TableStatus.ACTIVE);
                } while (!wasRestored);
            }
            catch (ResourceNotFoundException)
            {
                // If the restored table raised an error, it isn't
                // ready yet.
                Console.Write(".");
            }
        }

        uiMethods.DisplayTitle("Clean up resources.");

        // Delete the table.
        success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);
```

```
        Console.WriteLine($"Table {tableName} successfully deleted from
 {keyspaceName}.");
        Console.WriteLine("Waiting for the table to be removed completely. ");

        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a ResourceNotFoundException.
        bool wasDeleted = false;

        try
        {
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName,
 tableName);
            } while (!wasDeleted);
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
            Console.WriteLine($"{ex.Message} indicates that the table has been
 deleted.");
        }

        // Delete the keyspace.
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
 complete.");
    }
}
```

```
namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
```

```
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
```

```
    }


    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }
```

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }


    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }


    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
```

```
        response.Tables.ForEach(table =>
        {

 Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }


    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }


    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
```

```
        var newColumn = new ColumnDefinition { Name = "watched", Type =
 "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }

}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;

/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
 database.
///
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if
 available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/
repository/sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;
```

```
    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();

        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/
{_certFileName}");

        //var httpClient = new HttpClient();
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
        _amazoncert = new X509Certificate2($"{_localPathToFile}/
{_certFileName}");

        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
```

```
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>
    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport
 = 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
        {
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
 movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;

        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
 (title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
 '{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }

    /// <summary>
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
 {keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
 tableName, string title, int year)
    {
        var session = _cluster.Connect();
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
 WHERE title = $${title}$$ AND year = {year};";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }

    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
 tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT
 title, year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW
 FILTERING"));
```

```
            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   /*
    * Usage:
    * fileName - The name of the JSON file that contains movie data. (Get this
file
    * from the GitHub repo at resources/sample_file.)
    * keyspaceName - The name of the keyspace to create.
    */
   public static void main(String[] args) throws InterruptedException,
IOException {
       String fileName = "<Replace with the JSON file that contains movie
data>";
       String keyspaceName = "<Replace with the name of the keyspace to
create>";
       String titleUpdate = "The Family";
       int yearUpdate = 2013;
       String tableName = "Movie";
       String tableNameRestore = "MovieRestore";
       Region region = Region.US_EAST_1;
       KeyspacesClient keyClient = KeyspacesClient.builder()
               .region(region)
               .build();

       DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
       CqlSession session = CqlSession.builder()
               .withConfigLoader(loader)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the Amazon Keyspaces example scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Create a keyspace.");
       createKeySpace(keyClient, keyspaceName);
       System.out.println(DASHES);

       System.out.println(DASHES);
       Thread.sleep(5000);
       System.out.println("2. Check for keyspace existence.");
       checkKeyspaceExistence(keyClient, keyspaceName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
 enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
 the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
```

```
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state
using the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
ResourceNotFoundException is
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
```

```
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println("The table status is " + status);
```

```
                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
```

```
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
                + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year
= :k1;";
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", titleUpdate)
                .setInt("k1", yearUpdate)
                .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);
```

```
        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName)
{
        ResultSet resultSet = session.execute(
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title =
'The Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
```

```
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
```

```
                .name("title")
                .type("text")
                .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
                .name("year")
                .type("int")
                .build();

        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                .name("release_date")
                .type("timestamp")
                .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
                .name("plot")
                .type("text")
                .build();

        List<ColumnDefinition> colList = new ArrayList<>();
        colList.add(defTitle);
        colList.add(defYear);
        colList.add(defReleaseDate);
        colList.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
                .name("year")
                .build();

        PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
```

```
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
```

```
                    .build();

            GetKeyspaceResponse response =
 keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest =
 CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
 keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
 response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace

- GetTable
- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
```

```
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this
 file from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
```

```
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-
in-time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
```

```
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the
timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)
```

```
    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient
 ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
```

```
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
```

```
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
```

```
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\" WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0
 AND year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }
```

```
    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE
 title = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
```

```
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
```

```
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
```

```
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }
```

```
    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
```

```
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable

- UpdateTable

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper

    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspaces in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only
 letters, "
            "numbers and underscores: ",
            q.non_empty,
        )
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
```

```
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
            print(f"Created a new keyspace.\n\t{ks_arn}.")
        print("The first 10 keyspaces in your account are:\n")
        self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema
to hold
           movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
        print(f"Your table is {table['status']}. Its schema is:")
        pp(table["schemaDefinition"])
        print("\nThe tables in your keyspace are:\n")
        self.ks_wrapper.list_tables()

    @demo_func
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the
connection
```

```
        to the keyspace. This function downloads a default certificate or lets
you
        specify your own.
        """
        print("To connect to your keyspace, you must have a TLS certificate.")
        print("Checking for TLS certificate...")
        cert_path = os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        )
        if not os.path.exists(cert_path):
            cert_choice = q.ask(
                f"Press enter to download a certificate from
{QueryManager.CERT_URL} "
                f"or enter the full path to the certificate you want to use: "
            )
            if cert_choice:
                cert_path = cert_choice
            else:
                cert = requests.get(QueryManager.CERT_URL).text
                with open(cert_path, "w") as cert_file:
                    cert_file.write(cert)
        else:
            q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
        print(
            f"Certificate {cert_path} will be used to secure the connection to
your keyspace."
        )
        return cert_path

    @demo_func
    def query_table(self, qm, movie_file):
        """
        1. Adds movies to the table from a sample movie data file.
        2. Gets a list of movies from the table and lets you select one.
        3. Displays more information about the selected movie.
        """
        qm.add_movies(self.ks_wrapper.table_name, movie_file)
        movies = qm.get_movies(self.ks_wrapper.table_name)
        print(f"Added {len(movies)} movies to the table:")
        sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
        movie_choice = qm.get_movie(
            self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
        )
```

```
        print(movie_choice.title)
        print(f"\tReleased: {movie_choice.release_date}")
        print(f"\tPlot: {movie_choice.plot}")

    @demo_func
    def update_and_restore_table(self, qm):
        """
        1. Updates the table by adding a column to track watched movies.
        2. Marks some of the movies as watched.
        3. Gets the list of watched movies from the table.
        4. Restores to a movies_restored table at a previous point in time.
        5. Gets the list of movies from the restored table.
        """
        print("Let's add a column to record which movies you've watched.")
        pre_update_timestamp = datetime.utcnow()
        print(
            f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later."
        )
        self.ks_wrapper.update_table()
        print("Waiting for your table to update...")
        table = {"status": "UPDATING"}
        while table["status"] != "ACTIVE":
            wait(5)
            table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
        print("Column 'watched' added to table.")
        q.ask(
            "Let's mark some of the movies as watched. Press Enter when you're
ready.\n"
        )
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print("-" * 88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print("-" * 88)
        if q.ask(
            "Do you want to restore the table to the way it was before all of
these\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
```

```
            q.is_yesno,
        ):
            starting_table_name = self.ks_wrapper.table_name
            table_name_restored =
self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {"status": "RESTORING"}
            while table["status"] != "ACTIVE":
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(
                f"Restored {starting_table_name} to {table_name_restored} "
                f"at a point in time of {pre_update_timestamp}."
            )
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
                print(movie.title)

    def cleanup(self, cert_path):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.

        :param cert_path: The path of the TLS certificate used in the demo. If
the
                          certificate was downloaded during the demo, it is
removed.
        """
        if q.ask(
            f"Do you want to delete your {self.ks_wrapper.table_name} table and "
            f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
            q.is_yesno,
        ):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print(
```

```
                "Keyspace deleted. If you chose to restore your table during the
 "
                "demo, the original table is also deleted."
            )
            if cert_path == os.path.join(
                os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
            ) and os.path.exists(cert_path):
                os.remove(cert_path)
                print("Removed certificate that was downloaded for this demo.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
 %(message)s")

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is
 closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../../../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
        self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Keyspace- und Tabellenaktionen umschließt.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn


    def exists_keyspace(self, name):
```

```
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.
```

```
        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Definieren Sie eine Klasse, die eine TLS-Verbindung zu einem Keyspace herstellt, sich mit SigV4 authentifiziert und CQL-Abfragen an eine Tabelle im Keyspace sendet.

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4
 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you
 specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"

    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for
 TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
 credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
```

```
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.
{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

    def __exit__(self, *args):
        """
        Exits the cluster. This shuts down all existing session connections.
        """
        self.cluster.__exit__(*args)

    def add_movies(self, table_name, movie_file_path):
        """
        Gets movies from a JSON file and adds them to a table in the keyspace.

        :param table_name: The name of the table.
        :param movie_file_path: The path and file name of a JSON file that
 contains movie data.
        """
        with open(movie_file_path, "r") as movie_file:
            movies = json.loads(movie_file.read())
        stmt = self.session.prepare(
            f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
 (?, ?, ?, ?);"
        )
```

```
        for movie in movies[:20]:
            self.session.execute(
                stmt,
                parameters=[
                    movie["year"],
                    movie["title"],
                    date.fromisoformat(movie["info"]
["release_date"].partition("T")[0]),
                    movie["info"]["plot"],
                ],
            )

    def get_movies(self, table_name, watched=None):
        """
        Gets the title and year of the full list of movies from the table.

        :param table_name: The name of the movie table.
        :param watched: When specified, the returned list of movies is filtered
 to
                        either movies that have been watched or movies that have
 not
                        been watched. Otherwise, all movies are returned.
        :return: A list of movies in the table.
        """
        if watched is None:
            stmt = SimpleStatement(f"SELECT title, year from {table_name}")
            params = None
        else:
            stmt = SimpleStatement(
                f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
 FILTERING"
            )
            params = [watched]
        return self.session.execute(stmt, parameters=params).all()

    def get_movie(self, table_name, title, year):
        """
        Gets a single movie from the table, by title and year.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        :return: The requested movie.
        """
```

```
        return self.session.execute(
            SimpleStatement(
                f"SELECT * from {table_name} WHERE title = %s AND year = %s"
            ),
            parameters=[title, year],
        ).one()

    def watched_movie(self, table_name, title, year):
        """
        Updates a movie as having been watched.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        """
        self.session.execute(
            SimpleStatement(
                f"UPDATE {table_name} SET watched=true WHERE title = %s AND year
= %s"
            ),
            parameters=[title, year],
        )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Aktionen für Amazon Keyspaces mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Keyspaces-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Keyspaces (für Apache Cassandra).

Beispiele

- Verwenden Sie CreateKeyspacemit einem AWS SDK
- Verwenden Sie CreateTablemit einem AWS SDK
- Verwenden Sie DeleteKeyspacemit einem AWS SDK
- Verwenden Sie DeleteTablemit einem AWS SDK
- Verwenden Sie GetKeyspacemit einem AWS SDK
- Verwenden Sie GetTablemit einem AWS SDK
- Verwenden Sie ListKeyspacesmit einem AWS SDK
- Verwenden Sie ListTablesmit einem AWS SDK
- Verwenden Sie RestoreTablemit einem AWS SDK
- Verwenden Sie UpdateTablemit einem AWS SDK

Verwenden Sie **`CreateKeyspace`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateKeyspace` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }
```

- Einzelheiten zur API finden Sie CreateKeyspacein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
```

```
            CreateKeyspaceRequest keyspaceRequest =
 CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
 keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
 response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeyspacein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Einzelheiten zur API finden Sie CreateKeyspacein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
```

```
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn
```

- Einzelheiten zur API finden Sie CreateKeyspacein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->createkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysconflictexception.
        MESSAGE 'Keyspace already exists.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateKeyspacein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateTable` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Create a new Amazon Keyspaces table.
   /// </summary>
   /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
   /// <param name="schema">The schema for the new table.</param>
   /// <param name="tableName">The name of the new table.</param>
   /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
   public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
   {
       var request = new CreateTableRequest
       {
           KeyspaceName = keyspaceName,
           SchemaDefinition = schema,
           TableName = tableName,
           PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
       };
```

```
        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();

           ColumnDefinition defYear = ColumnDefinition.builder()
                   .name("year")
                   .type("int")
                   .build();

           ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                   .name("release_date")
                   .type("timestamp")
                   .build();

           ColumnDefinition defPlot = ColumnDefinition.builder()
                   .name("plot")
                   .type("text")
                   .build();
```

```
            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }
```

```
    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
```

```
    }
}
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
```

```
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]
```

- Einzelheiten zur API finden Sie CreateTablein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Define schema with columns
        DATA(lt_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'title' iv_type =
 'text' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'year' iv_type =
 'int' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'release_date' iv_type =
 'timestamp' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'plot' iv_type =
 'text' ) )
        ).

        " Define partition keys
        DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'year' ) )
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'title' ) )
        ).

        " Create schema definition
        DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
          it_allcolumns = lt_columns
          it_partitionkeys = lt_partition_keys ).

        " Enable point-in-time recovery
        DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
          iv_status = 'ENABLED' ).

        oo_result = lo_kys->createtable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          io_schemadefinition = lo_schema
          io_pointintimerecovery = lo_pitr ).
        MESSAGE 'Table created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTablein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteKeyspace`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteKeyspace` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete an existing keyspace.
/// </summary>
/// <param name="keyspaceName"></param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
        new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DeleteKeyspacein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           keyClient.deleteKeyspace(deleteKeyspaceRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DeleteKeyspacein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
```

```
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Einzelheiten zur API finden Sie DeleteKeyspacein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)
```

```
    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteKeyspacein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_kys->deletekeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteKeyspacein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteTable` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
       try {
           DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           keyClient.deleteTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None
```

```
    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteTablein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_kys->deletetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
```

```
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTablein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetKeyspace`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetKeyspace` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
```

```
    }
```

- Einzelheiten zur API finden Sie GetKeyspacein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
           String name = response.keyspaceName();
           System.out.println("The " + name + " KeySpace is ready");

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie GetKeyspacein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- Einzelheiten zur API finden Sie GetKeyspacein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""
```

```
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists
```

- Einzelheiten zur API finden Sie GetKeyspacein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->getkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Keyspace does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetKeyspacein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetTable` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

- Einzelheiten zur API finden Sie GetTablein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
           }

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie GetTablein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie GetTablein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
```

```
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Einzelheiten zur API finden Sie GetTablein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->gettable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Table does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetTablein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListKeyspaces`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListKeyspaces` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
 ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {
```

```
Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }
```

- Einzelheiten zur API finden Sie ListKeyspacesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListKeyspacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListKeyspacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
```

```
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListKeyspacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->listkeyspaces(
          iv_maxresults = iv_max_results ).
        MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListKeyspacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTables`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListTables` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName)
 {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
 keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
 content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_tables(self):
```

```
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListTablesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->listtables(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Tables listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTablesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RestoreTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `RestoreTable` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
```

```
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }
```

- Einzelheiten zur API finden Sie RestoreTablein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                   .restoreTimestamp(myTime)
                   .sourceTableName("Movie")
                   .targetKeyspaceName(keyspaceName)
                   .targetTableName("MovieRestore")
                   .sourceKeyspaceName(keyspaceName)
                   .build();

           RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
           System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- Einzelheiten zur API finden Sie RestoreTablein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- Einzelheiten zur API finden Sie RestoreTablein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
 time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
```

```
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name
```

- Einzelheiten zur API finden Sie RestoreTablein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->restoretable(
          iv_sourcekeyspacename = iv_source_keyspace_name
          iv_sourcetablename = iv_source_table_name
          iv_targetkeyspacename = iv_target_keyspace_name
          iv_targettablename = iv_target_table_name
          iv_restoretimestamp = iv_restore_timestamp ).
        MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RestoreTablein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateTable`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `UpdateTable` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type =
"boolean" };
        var request = new UpdateTableRequest
        {
```

```
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
    }
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- Einzelheiten zur API finden Sie UpdateTablein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie UpdateTablein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Add a new column to track watched movies
        DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'watched' iv_type =
 'boolean' ) )
        ).

        oo_result = lo_kys->updatetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          it_addcolumns = lt_add_columns ).
        MESSAGE 'Table updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateTablein der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Kinesis mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Kinesis mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Kinesis-Entwicklerhandbuch – Weitere Informationen zu Kinesis.
- Kinesis-API-Referenz – Details zu allen verfügbaren Kinesis-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Kinesis AWS SDKs
  - Erlernen Sie die Grundlagen von Kinesis mit einem AWS SDK
  - Aktionen für Kinesis mit AWS SDKs
    - Verwenden Sie AddTagsToStreammit einem AWS SDK oder CLI
    - Verwenden Sie CreateStreammit einem AWS SDK oder CLI
    - Verwenden Sie DeleteStreammit einem AWS SDK oder CLI
    - Verwenden Sie DeregisterStreamConsumermit einem AWS SDK oder CLI
    - Verwenden Sie DescribeStreammit einem AWS SDK oder CLI
    - Verwenden Sie GetRecordsmit einem AWS SDK oder CLI

    - Verwendung von GetShardIterator mit einer CLI
    - Verwenden Sie mit einem ListStreamConsumers AWS SDK
    - Verwenden Sie ListStreamsmit einem AWS SDK oder CLI
    - Verwenden Sie ListTagsForStreammit einem AWS SDK oder CLI
    - Verwenden Sie PutRecordmit einem AWS SDK oder CLI
    - Verwenden Sie PutRecordsmit einem AWS SDK oder CLI
    - Verwenden Sie RegisterStreamConsumermit einem AWS SDK oder CLI
- Serverless-Beispiele für Kinesis
  - Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser
  - Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Grundlegende Beispiele für die Verwendung von Kinesis AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Kinesis mit AWS SDKs verwendet werden.

Beispiele

- Erlernen Sie die Grundlagen von Kinesis mit einem AWS SDK
- Aktionen für Kinesis mit AWS SDKs
  - Verwenden Sie AddTagsToStreammit einem AWS SDK oder CLI
  - Verwenden Sie CreateStreammit einem AWS SDK oder CLI
  - Verwenden Sie DeleteStreammit einem AWS SDK oder CLI
  - Verwenden Sie DeregisterStreamConsumermit einem AWS SDK oder CLI
  - Verwenden Sie DescribeStreammit einem AWS SDK oder CLI
  - Verwenden Sie GetRecordsmit einem AWS SDK oder CLI
  - Verwendung von GetShardIterator mit einer CLI
  - Verwenden Sie mit einem ListStreamConsumers AWS SDK
  - Verwenden Sie ListStreamsmit einem AWS SDK oder CLI
  - Verwenden Sie ListTagsForStreammit einem AWS SDK oder CLI
  - Verwenden Sie PutRecordmit einem AWS SDK oder CLI
  - Verwenden Sie PutRecordsmit einem AWS SDK oder CLI
  - Verwenden Sie RegisterStreamConsumermit einem AWS SDK oder CLI

Erlernen Sie die Grundlagen von Kinesis mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Stream und fügen Sie einen Datensatz ein.
- Erstellen Sie einen Shard-Iterator.
- Lesen Sie den Datensatz und bereinigen Sie anschließend die Ressourcen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
   DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
   DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
   DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

   "Create stream."
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
   ENDTRY.

   "Wait for stream to becomes active."
```

```
    lo_stream_describe_result = lo_kns->describestream( iv_streamname =
 iv_stream_name ).
    lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
    WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
      IF sy-index = 30.
        EXIT.               "maximum 5 minutes"
      ENDIF.
      WAIT UP TO 10 SECONDS.
      lo_stream_describe_result = lo_kns->describestream( iv_streamname =
 iv_stream_name ).
      lo_stream_description = lo_stream_describe_result-
>get_streamdescription( ).
    ENDWHILE.

    "Create record."
    TRY.
        lo_record_result = lo_kns->putrecord(
            iv_streamname = iv_stream_name
            iv_data       = iv_data
            iv_partitionkey = iv_partition_key ).
        MESSAGE 'Record created.' TYPE 'I'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
 'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request
 quotas.' TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested
 data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

```
   "Create a shard iterator in order to read the record."
   TRY.
       lo_sharditerator = lo_kns->getsharditerator(
         iv_shardid = lo_record_result->get_shardid( )
         iv_sharditeratortype = iv_sharditeratortype
         iv_streamname = iv_stream_name ).
       MESSAGE 'Shard iterator created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.

   "Read the record."
   TRY.
       oo_result = lo_kns->getrecords(                    " oo_result is
returned for testing purposes. "
           iv_sharditerator   = lo_sharditerator->get_sharditerator( ) ).
       MESSAGE 'Shard iterator created.' TYPE 'I'.
     CATCH /aws1/cx_knsexpirediteratorex.
       MESSAGE 'Iterator expired.' TYPE 'E'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
```

```
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Delete stream."
    TRY.
        lo_kns->deletestream(
            iv_streamname = iv_stream_name ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CreateStream
  - DeleteStream
  - GetRecords
  - GetShardIterator
  - PutRecord

Aktionen für Kinesis mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Kinesis-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Kinesis.

Beispiele

- Verwenden Sie AddTagsToStreammit einem AWS SDK oder CLI
- Verwenden Sie CreateStreammit einem AWS SDK oder CLI
- Verwenden Sie DeleteStreammit einem AWS SDK oder CLI

- Verwenden Sie DeregisterStreamConsumermit einem AWS SDK oder CLI
- Verwenden Sie DescribeStreammit einem AWS SDK oder CLI
- Verwenden Sie GetRecordsmit einem AWS SDK oder CLI
- Verwendung von GetShardIterator mit einer CLI
- Verwenden Sie mit einem ListStreamConsumers AWS SDK
- Verwenden Sie ListStreamsmit einem AWS SDK oder CLI
- Verwenden Sie ListTagsForStreammit einem AWS SDK oder CLI
- Verwenden Sie PutRecordmit einem AWS SDK oder CLI
- Verwenden Sie PutRecordsmit einem AWS SDK oder CLI
- Verwenden Sie RegisterStreamConsumermit einem AWS SDK oder CLI

Verwenden Sie **`AddTagsToStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AddTagsToStream` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream.
/// </summary>
public class TagStream
{
    public static async Task Main()
```

```
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamName = "AmazonKinesisStream";
            var tags = new Dictionary<string, string>
            {
                { "Project", "Sample Kinesis Project" },
                { "Application", "Sample Kinesis App" },
            };

            var success = await ApplyTagsToStreamAsync(client, streamName, tags);

            if (success)
            {
                Console.WriteLine($"Taggs successfully added to {streamName}.");
            }
            else
            {
                Console.WriteLine("Tags were not added to the stream.");
            }
        }

        /// <summary>
        /// Applies the set of tags to the named Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client.</param>
        /// <param name="streamName">The name of the Kinesis stream to which
        /// the tags will be attached.</param>
        /// <param name="tags">A sictionary containing key/value pairs which
        /// will be used to create the Kinesis tags.</param>
        /// <returns>A Boolean value which represents the success or failure
        /// of AddTagsToStreamAsync.</returns>
        public static async Task<bool> ApplyTagsToStreamAsync(
            IAmazonKinesis client,
            string streamName,
            Dictionary<string, string> tags)
        {
            var request = new AddTagsToStreamRequest
            {
                StreamName = streamName,
                Tags = tags,
            };

            var response = await client.AddTagsToStreamAsync(request);
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Einzelheiten zur API finden Sie AddTagsToStreamin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie einem Datenstrom Tags hinzu

Im folgenden Beispiel für `add-tags-to-stream` wird dem angegebenen Stream ein Tag mit dem Schlüssel `samplekey` und dem Wert `example` zugewiesen.

```
aws kinesis add-tags-to-stream \
    --stream-name samplestream \
    --tags samplekey=example
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Markieren Ihrer Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie AddTagsToStreamin der AWS CLI Befehlsreferenz.

Verwenden Sie **`CreateStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateStream` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to create a new Amazon Kinesis stream.
/// </summary>
public class CreateStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        int shardCount = 1;

        var success = await CreateNewStreamAsync(client, streamName,
shardCount);
        if (success)
        {
            Console.WriteLine($"The stream, {streamName} successfully
created.");
        }
    }

    /// <summary>
    /// Creates a new Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client.</param>
    /// <param name="streamName">The name for the new stream.</param>
    /// <param name="shardCount">The number of shards the new stream will
```

```
        /// use. The throughput of the stream is a function of the number of
        /// shards; more shards are required for greater provisioned
        /// throughput.</param>
        /// <returns>A Boolean value indicating whether the stream was created.</
returns>
        public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis
 client, string streamName, int shardCount)
        {
            var request = new CreateStreamRequest
            {
                StreamName = streamName,
                ShardCount = shardCount,
            };

            var response = await client.CreateStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Einzelheiten zur API finden Sie CreateStreamin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Datenstrom

Im folgenden Beispiel für `create-stream` wird ein Datenstrom namens samplestream mit 3 Shards erstellt.

```
aws kinesis create-stream \
    --stream-name samplestream \
    --shard-count 3
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen eines Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie CreateStreamin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();
        createStream(kinesisClient, streamName);
        System.out.println("Done");
        kinesisClient.close();
    }

    public static void createStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            CreateStreamRequest streamReq = CreateStreamRequest.builder()
                    .streamName(streamName)
                    .shardCount(1)
                    .build();

            kinesisClient.createStream(streamReq);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateStreamin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erzeugt einen neuen Stream. Standardmäßig gibt dieses Cmdlet keine Ausgabe zurück, sodass der PassThru Schalter - hinzugefügt wird, um den Wert zurückzugeben, der dem StreamName Parameter - zur späteren Verwendung übergeben wurde.

```
$streamName = New-KINStream -StreamName "mystream" -ShardCount 1 -PassThru
```

- Einzelheiten zur API finden Sie unter CreateStream AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erzeugt einen neuen Stream.

```
New-KINStream -StreamName "mystream" -ShardCount 1
```

- Einzelheiten zur API finden Sie unter CreateStream AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def create(self, name, wait_until_exists=True):
        """
        Creates a stream.

        :param name: The name of the stream.
        :param wait_until_exists: When True, waits until the service reports that
                                  the stream exists, then queries for its
 metadata.
        """
```

```
        try:
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)
            self.name = name
            logger.info("Created stream %s.", name)
            if wait_until_exists:
                logger.info("Waiting until exists.")
                self.stream_exists_waiter.wait(StreamName=name)
                self.describe(name)
        except ClientError:
            logger.exception("Couldn't create stream %s.", name)
            raise
```

- Einzelheiten zur API finden Sie CreateStreamin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateStreamin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateStreamin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteStream` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to delete an Amazon Kinesis stream.
/// </summary>
public class DeleteStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        var success = await DeleteStreamAsync(client, streamName);

        if (success)
        {
            Console.WriteLine($"Stream, {streamName} successfully deleted.");
        }
        else
        {
            Console.WriteLine("Stream not deleted.");
        }
    }

    /// <summary>
    /// Deletes a Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client object.</param>
    /// <param name="streamName">The name of the string to delete.</param>
```

```
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client,
 string streamName)
        {
            // If EnforceConsumerDeletion is true, any consumers
            // of this stream will also be deleted. If it is set
            // to false and this stream has any consumers, the
            // call will fail with a ResourceInUseException.
            var request = new DeleteStreamRequest
            {
                StreamName = streamName,
                EnforceConsumerDeletion = true,
            };

            var response = await client.DeleteStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteStreamin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie einen Datenstrom

Im folgenden Beispiel für `delete-stream` wird der angegebene Datenstrom gelöscht.

```
aws kinesis delete-stream \
    --stream-name samplestream
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Löschen eines Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie DeleteStreamin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            kinesisClient.deleteStream(delStream);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteStreamin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht den angegebenen Stream. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Verwenden Sie den Schalter -Force, um die Bestätigungsaufforderung zu unterdrücken.

```
Remove-KINStream -StreamName "mystream"
```

- Einzelheiten zur API finden Sie unter DeleteStream AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht den angegebenen Stream. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Verwenden Sie den Schalter -Force, um die Bestätigungsaufforderung zu unterdrücken.

```
Remove-KINStream -StreamName "mystream"
```

- Einzelheiten zur API finden Sie unter DeleteStream AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def delete(self):
        """
        Deletes a stream.
        """
        try:
            self.kinesis_client.delete_stream(StreamName=self.name)
            self._clear()
```

```
            logger.info("Deleted stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete stream %s.", self.name)
            raise
```

- Einzelheiten zur API finden Sie DeleteStreamin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client.delete_stream().stream_name(stream).send().await?;

    println!("Deleted stream.");

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteStreamin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_kns->deletestream(
           iv_streamname = iv_stream_name ).
       MESSAGE 'Stream deleted.' TYPE 'I'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in
use.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteStreamin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeregisterStreamConsumer`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterStreamConsumer` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to deregister a consumer from an Amazon Kinesis stream.
/// </summary>
public class DeregisterConsumer
{
    public static async Task Main(string[] args)
```

```
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream";
            string consumerName = "CONSUMER_NAME";
            string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";

            var success = await DeregisterConsumerAsync(client, streamARN,
 consumerARN, consumerName);

            if (success)
            {
                Console.WriteLine($"{consumerName} successfully deregistered.");
            }
            else
            {
                Console.WriteLine($"{consumerName} was not successfully
 deregistered.");
            }
        }

        /// <summary>
        /// Deregisters a consumer from a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of a Kinesis stream.</param>
        /// <param name="consumerARN">The ARN of the consumer.</param>
        /// <param name="consumerName">The name of the consumer.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeregisterConsumerAsync(
            IAmazonKinesis client,
            string streamARN,
            string consumerARN,
            string consumerName)
        {
            var request = new DeregisterStreamConsumerRequest
            {
                StreamARN = streamARN,
                ConsumerARN = consumerARN,
                ConsumerName = consumerName,
            };
```

```
            var response = await client.DeregisterStreamConsumerAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Einzelheiten zur API finden Sie DeregisterStreamConsumerin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So heben Sie die Registrierung eines Datenstromkonsumenten auf

Im folgenden Beispiel für `deregister-stream-consumer` wird die Registrierung des angegebenen Konsumenten beim angegebenen Datenstrom aufgehoben.

```
aws kinesis deregister-stream-consumer \
    --stream-arn arn:aws:kinesis:us-west-2:123456789012:stream/samplestream \
    --consumer-name KinesisConsumerApplication
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Developing Consumer with Enhanced Fan-Out Using the Kinesis Data Streams API im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie DeregisterStreamConsumerin der AWS CLI Befehlsreferenz.

Verwenden Sie **`DescribeStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeStream` verwendet wird.

CLI

AWS CLI

So beschreiben Sie einen Datenstrom

Im folgenden Beispiel für `describe-stream` werden Details zum angegebenen Datenstrom abgerufen.

```
aws kinesis describe-stream \
    --stream-name samplestream
```

Ausgabe:

```
{
    "StreamDescription": {
        "Shards": [
            {
                "ShardId": "shardId-000000000000",
                "HashKeyRange": {
                    "StartingHashKey": "0",
                    "EndingHashKey": "113427455640312821154458202477256070484"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
 "49600871682957036442365024926191073437251060580128653314"
                }
            },
            {
                "ShardId": "shardId-000000000001",
                "HashKeyRange": {
                    "StartingHashKey": "113427455640312821154458202477256070485",
                    "EndingHashKey": "226854911280625642308916404954512140969"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
 "49600871682979337187563555549332609155523708941634633746"
                }
            },
            {
                "ShardId": "shardId-000000000002",
                "HashKeyRange": {
                    "StartingHashKey": "226854911280625642308916404954512140970",
                    "EndingHashKey": "340282366920938463463374607431768211455"
                },
                "SequenceNumberRange": {
                    "StartingSequenceNumber":
 "49600871683001637932762086172474144873796357303140614178"
                }
```

```
            }
        ],
        "StreamARN": "arn:aws:kinesis:us-west-2:123456789012:stream/
samplestream",
        "StreamName": "samplestream",
        "StreamStatus": "ACTIVE",
        "RetentionPeriodHours": 24,
        "EnhancedMonitoring": [
            {
                "ShardLevelMetrics": []
            }
        ],
        "EncryptionType": "NONE",
        "KeyId": null,
        "StreamCreationTimestamp": 1572297168.0
    }
}
```

Weitere Informationen finden Sie unter Erstellen und Verwalten von Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie DescribeStreamin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details des angegebenen Streams zurück.

```
Get-KINStream -StreamName "mystream"
```

Ausgabe:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Einzelheiten zur API finden Sie unter DescribeStream AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details des angegebenen Streams zurück.

```
Get-KINStream -StreamName "mystream"
```

Ausgabe:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- Einzelheiten zur API finden Sie unter DescribeStream AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")
```

```
    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- Einzelheiten zur API finden Sie DescribeStreamin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();

    println!("Stream description:");
    println!("  Name:              {}:", desc.stream_name());
    println!("  Status:            {:?}", desc.stream_status());
```

```
    println!("  Open shards:       {:?}", desc.shards.len());
    println!("  Retention (hours): {}", desc.retention_period_hours());
    println!("  Encryption:        {:?}", desc.encryption_type.unwrap());

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeStreamin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->describestream(
            iv_streamname = iv_stream_name ).
        DATA(lt_stream_description) = oo_result->get_streamdescription( ).
        MESSAGE 'Streams retrieved.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
 exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeStreamin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetRecords`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetRecords` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie Datensätze aus einem Shard ab

Im folgenden Beispiel für `get-records` werden Datensätze aus dem Shard eines Kinesis-Datenstroms über den angegebenen Shard-Iterator abgerufen.

```
aws kinesis get-records \
    --shard-iterator AAAAAAAAAAF7/0mWD7IuHj1yGv/
TKuNgx2ukD5xipCY4cy4gU96orWwZwcSXh3K9tAmGYeOZyLZrvzzeOFVf9iN99hUPw/w/
b0YWYeehfNvnf1DYt5XpDJghLKr3DzgznkTmMymDP3R+3wRKeuEw6/kdxY2yKJH0veaiekaVc4N2VwK/
GvaGP2Hh9Fg7N++q0Adg6fIDQPt4p8RpavDbk+A4sL9SWGE1
```

Ausgabe:

```
{
    "Records": [],
    "MillisBehindLatest": 80742000
}
```

Weitere Informationen finden Sie unter Developing Consumer Using the Kinesis Data Streams API with the AWS SDK for Java im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie GetRecordsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to read from (for
 example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
```

```
                .region(region)
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
                .streamName(streamName)
                .shardIteratorType("TRIM_HORIZON")
                .shardId(lastShardId)
                .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
                .shardIterator(shardIterator)
```

```
                .limit(1000)
                .build();

        GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

        // Put result into record list. Result may be empty.
        records = result.records();

        // Print records
        for (Record record : records) {
            SdkBytes byteBuffer = record.data();
            System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
 String(byteBuffer.asByteArray()));
        }
    }
}
```

- Einzelheiten zur API finden Sie GetRecordsin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird gezeigt, wie Daten aus einer Reihe von einem oder mehreren Datensätzen zurückgegeben und extrahiert werden können. Der Iterator, an den geliefert Get-KINRecord wird, bestimmt die Startposition der zurückzugebenden Datensätze, die in diesem Beispiel in einer Variablen, $records, erfasst werden. Auf jeden einzelnen Datensatz kann dann durch Indizierung der $records-Sammlung zugegriffen werden. Unter der Annahme, dass es sich bei den Daten im Datensatz um UTF-8 codierten Text handelt, zeigt der letzte Befehl, wie Sie die Daten aus dem Objekt extrahieren und als Text MemoryStream an die Konsole zurückgeben können.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Ausgabe:

```
MillisBehindLatest NextShardIterator           Records
------------------ -----------------           -------
```

```
0                   AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Ausgabe:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Ausgabe:

```
test data from string
```

- Einzelheiten zur API finden Sie unter GetRecords AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird gezeigt, wie Daten aus einer Reihe von einem oder mehreren Datensätzen zurückgegeben und extrahiert werden können. Der Iterator, an den geliefert Get-KINRecord wird, bestimmt die Startposition der zurückzugebenden Datensätze, die in diesem Beispiel in einer Variablen, $records, erfasst werden. Auf jeden einzelnen Datensatz kann dann durch Indizierung der $records-Sammlung zugegriffen werden. Unter der Annahme, dass es sich bei den Daten im Datensatz um UTF-8 codierten Text handelt, zeigt der letzte Befehl, wie Sie die Daten aus dem Objekt extrahieren und als Text MemoryStream an die Konsole zurückgeben können.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Ausgabe:

```
MillisBehindLatest NextShardIterator            Records
------------------ -----------------            -------
0                   AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Ausgabe:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Ausgabe:

```
test data from string
```

- Einzelheiten zur API finden Sie unter GetRecords AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")
```

```
    def get_records(self, max_records):
        """
        Gets records from the stream. This function is a generator that first
gets
        a shard iterator for the stream, then uses the shard iterator to get
records
        in batches from the stream. The shard iterator can be accessed through
the
        'details' property, which is populated using the 'describe' function of
this class.
        Each batch of records is yielded back to the caller until the specified
        maximum number of records has been retrieved.

        :param max_records: The maximum number of records to retrieve.
        :return: Yields the current batch of retrieved records.
        """
        try:
            response = self.kinesis_client.get_shard_iterator(
                StreamName=self.name,
                ShardId=self.details["Shards"][0]["ShardId"],
                ShardIteratorType="LATEST",
            )
            shard_iter = response["ShardIterator"]
            record_count = 0
            while record_count < max_records:
                response = self.kinesis_client.get_records(
                    ShardIterator=shard_iter, Limit=10
                )
                shard_iter = response["NextShardIterator"]
                records = response["Records"]
                logger.info("Got %s records.", len(records))
                record_count += len(records)
                yield records
        except ClientError:
            logger.exception("Couldn't get records from stream %s.", self.name)
            raise



    def describe(self, name):
        """
        Gets metadata about a stream.
```

```
        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- Einzelheiten zur API finden Sie GetRecordsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->getrecords(             " oo_result is returned for
testing purposes. "
            iv_sharditerator = iv_shard_iterator ).
        DATA(lt_records) = oo_result->get_records( ).
        MESSAGE 'Record retrieved.' TYPE 'I'.
      CATCH /aws1/cx_knsexpirediteratorex.
        MESSAGE 'Iterator expired.' TYPE 'E'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
```

```
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetRecordsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GetShardIterator`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetShardIterator` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So rufen Sie einen Shard-Iterator ab

Im folgenden Beispiel für `get-shard-iterator` wird der Shard-Iteratortyp `AT_SEQUENCE_NUMBER` verwendet und ein Shard-Iterator generiert, um mit dem Lesen von Datensätzen genau an der Position zu beginnen, die durch die angegebene Sequenznummer gekennzeichnet ist.

```
aws kinesis get-shard-iterator \
    --stream-name samplestream \
    --shard-id shardId-000000000001 \
    --shard-iterator-type LATEST
```

Ausgabe:

```
{
    "ShardIterator": "AAAAAAAAAAFEvJjIYI+3jw/4aqgH9FifJ+n48XWTh/
IFIsbILP6o5eDueD39NXNBfpZ10WL5K6ADXk8w+5H+Qhd9cFA9k268CPXCz/kebq1TGYI7Vy
+lUkA9BuN3xvATxMBGxRY3zYK05gqgvaIRn94O8SqeEqwhigwZxNWxID3Ej7YYYcxQi8Q/fIrCjGAy/
n2r5Z9G864YpWDfN9upNNQAR/iiOWKs"
}
```

Weitere Informationen finden Sie unter Developing Consumer Using the Kinesis Data Streams API with the AWS SDK for Java im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie GetShardIteratorin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt einen Shard-Iterator für den angegebenen Shard und die Startposition zurück. Einzelheiten zu den Shard-IDs und Sequenznummern können der Ausgabe des Get-KINStream Cmdlets entnommen werden, indem auf die Shards-Auflistung des zurückgegebenen Stream-Objekts verwiesen wird. Der zurückgegebene Iterator kann zusammen mit dem Get-KINRecord Cmdlet verwendet werden, um Datensätze im Shard abzurufen.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Ausgabe:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (GetShardIteratorV4).

Tools für V5 PowerShell

Beispiel 1: Gibt einen Shard-Iterator für den angegebenen Shard und die Startposition zurück. Einzelheiten zu den Shard-IDs und Sequenznummern können der Ausgabe des Get-KINStream Cmdlets entnommen werden, indem auf die Shards-Auflistung des zurückgegebenen Stream-Objekts verwiesen wird. Der zurückgegebene Iterator kann zusammen mit dem Get-KINRecord Cmdlet verwendet werden, um Datensätze im Shard abzurufen.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Ausgabe:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (GetShardIteratorV5).

Verwenden Sie mit einem **`ListStreamConsumers`** AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListStreamConsumers`.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
```

```
    /// List the consumers of an Amazon Kinesis stream.
    /// </summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;

            var consumers = await ListConsumersAsync(client, streamARN,
 maxResults);

            if (consumers.Count > 0)
            {
                consumers
                    .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
 {c.ConsumerARN}"));
            }
            else
            {
                Console.WriteLine("No consumers found.");
            }
        }

        /// <summary>
        /// Retrieve a list of the consumers for a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of the stream for which we want to
        /// retrieve a list of clients.</param>
        /// <param name="maxResults">The maximum number of results to return.</
param>
        /// <returns>A list of Consumer objects.</returns>
        public static async Task<List<Consumer>>
 ListConsumersAsync(IAmazonKinesis client, string streamARN, int maxResults)
        {
            var request = new ListStreamConsumersRequest
            {
                StreamARN = streamARN,
                MaxResults = maxResults,
            };
```

```
            var response = await client.ListStreamConsumersAsync(request);

            return response.Consumers;
        }
    }
```

- Einzelheiten zur API finden Sie ListStreamConsumersin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`ListStreams`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListStreams` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Retrieves and displays a list of existing Amazon Kinesis streams.
    /// </summary>
    public class ListStreams
    {
        public static async Task Main(string[] args)
        {
            IAmazonKinesis client = new AmazonKinesisClient();
```

```
            var response = await client.ListStreamsAsync(new
 ListStreamsRequest());

            List<string> streamNames = response.StreamNames;

            if (streamNames.Count > 0)
            {
                streamNames
                    .ForEach(s => Console.WriteLine($"Stream name: {s}"));
            }
            else
            {
                Console.WriteLine("No streams were found.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListStreamsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Datenströme auf

Im folgenden Beispiel für `list-streams` werden alle Parameter im aktuellen Konto und in der aktuellen Region aufgelistet.

```
aws kinesis list-streams
```

Ausgabe:

```
{
    "StreamNames": [
        "samplestream",
        "samplestream1"
    ]
}
```

Weitere Informationen finden Sie unter Auflisten eines Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie ListStreamsin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names;
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListStreamsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_kns->liststreams(        " oo_result is returned for
testing purposes. "
           "Set Limit to specify that a maximum of streams should be returned."
           iv_limit = iv_limit ).
       DATA(lt_streams) = oo_result->get_streamnames( ).
       MESSAGE 'Streams listed.' TYPE 'I'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ListStreamsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTagsForStream`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForStream` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Shows how to list the tags that have been attached to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class ListTags
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string streamName = "AmazonKinesisStream";

            await ListTagsAsync(client, streamName);
        }

        /// <summary>
        /// List the tags attached to a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the Kinesis stream for which you
        /// wish to display tags.</param>
        public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
        {
            var request = new ListTagsForStreamRequest
            {
                StreamName = streamName,
                Limit = 10,
            };

            var response = await client.ListTagsForStreamAsync(request);
            DisplayTags(response.Tags);

            while (response.HasMoreTags)
            {
                request.ExclusiveStartTagKey = response.Tags[response.Tags.Count
- 1].Key;
                response = await client.ListTagsForStreamAsync(request);
```

```
            }
        }

        /// <summary>
        /// Displays the items in a list of Kinesis tags.
        /// </summary>
        /// <param name="tags">A list of the Tag objects to be displayed.</param>
        public static void DisplayTags(List<Tag> tags)
        {
            tags
                .ForEach(t => Console.WriteLine($"Key: {t.Key} Value:
 {t.Value}"));
        }
    }
```

- Einzelheiten zur API finden Sie ListTagsForStreamin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Tags für einen Datenstrom auf

Im folgenden Beispiel für `list-tags-for-stream` werden alle Tags aufgelistet, die dem angegebenen Datenstrom zugeordnet sind.

```
aws kinesis list-tags-for-stream \
    --stream-name samplestream
```

Ausgabe:

```
{
    "Tags": [
        {
            "Key": "samplekey",
            "Value": "example"
        }
    ],
    "HasMoreTags": false
}
```

Weitere Informationen finden Sie unter Markieren Ihrer Streams im Entwicklerhandbuch für Amazon Kinesis Data Streams.

- Einzelheiten zur API finden Sie ListTagsForStreamin der AWS CLI Befehlsreferenz.

Verwenden Sie **PutRecord**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecord` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So schreiben Sie einen Datensatz in einen Datenstrom

Im folgenden Beispiel für `put-record` wird ein einzelner Datensatz mithilfe des angegebenen Partitionsschlüssels in den angegebenen Datenstrom geschrieben.

```
aws kinesis put-record \
    --stream-name samplestream \
    --data sampledatarecord \
    --partition-key samplepartitionkey
```

Ausgabe:

```
{
    "ShardId": "shardId-000000000009",
    "SequenceNumber": "49600902273357540915989931256901506243878407835297513618",
    "EncryptionType": "KMS"
}
```

Weitere Informationen finden Sie unter Entwickeln von Produzenten, die die Amazon Kinesis Data Streams-API mit dem AWS SDK for Java verwenden im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie PutRecordin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to which records
 are written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String
streamName) {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in
between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON
serialization by
        // the Jackson JSON library.
```

```
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
                .partitionKey(trade.getTickerSymbol()) // We use the ticker
 symbol as the partition key, explained in
                                                        // the Supplemental
 Information section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
 streamName) {
        try {
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
                System.err.println("Stream " + streamName + " is not active.
 Please wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
```

```
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutRecordin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Schreibt einen Datensatz, der die an den Parameter -Text übergebene Zeichenfolge enthält.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -
PartitionKey "Key1"
```

Beispiel 2: Schreibt einen Datensatz, der die Daten aus der angegebenen Datei enthält. Die Datei wird als Folge von Bytes behandelt. Wenn sie Text enthält, sollte sie daher vor der Verwendung mit diesem Cmdlet mit der erforderlichen Kodierung geschrieben werden.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Einzelheiten zur API finden Sie unter PutRecord AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Schreibt einen Datensatz, der die an den Parameter -Text übergebene Zeichenfolge enthält.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -
PartitionKey "Key1"
```

Beispiel 2: Schreibt einen Datensatz, der die Daten aus der angegebenen Datei enthält. Die Datei wird als Folge von Bytes behandelt. Wenn sie Text enthält, sollte sie daher vor der Verwendung mit diesem Cmdlet mit der erforderlichen Kodierung geschrieben werden.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- Einzelheiten zur API finden Sie unter PutRecord AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def put_record(self, data, partition_key):
        """
        Puts data into the stream. The data is formatted as JSON before it is
 passed
        to the stream.

        :param data: The data to put in the stream.
        :param partition_key: The partition key to use for the data.
        :return: Metadata about the record, including its shard ID and sequence
 number.
        """
        try:
```

```
            response = self.kinesis_client.put_record(
                StreamName=self.name, Data=json.dumps(data),
 PartitionKey=partition_key
            )
            logger.info("Put record in stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't put record in stream %s.", self.name)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie PutRecordin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) ->
 Result<(), Error> {
    let blob = Blob::new(data);

    client
        .put_record()
        .data(blob)
        .partition_key(key)
        .stream_name(stream)
        .send()
        .await?;

    println!("Put data into stream.");

    Ok(())
}
```

- Einzelheiten zur API finden Sie PutRecordin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_kns->putrecord(            " oo_result is returned for
testing purposes. "
           iv_streamname = iv_stream_name
           iv_data       = iv_data
           iv_partitionkey = iv_partition_key ).
       MESSAGE 'Record created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request
quotas.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested
data is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutRecordin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **PutRecords**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutRecords` verwendet wird.

CLI

AWS CLI

So schreiben Sie mehrere Datensätze in einen Datenstrom

Im folgenden Beispiel für `put-records` werden ein Datensatz mit dem angegebenen Partitionsschlüssel und ein anderer Datensatz mit einem anderen Partitionsschlüssel in einen einzelnen Aufruf geschrieben.

```
aws kinesis put-records \
    --stream-name samplestream \
    --
records Data=blob1,PartitionKey=partitionkey1 Data=blob2,PartitionKey=partitionkey2
```

Ausgabe:

```
{
    "FailedRecordCount": 0,
    "Records": [
        {
            "SequenceNumber":
 "49600883331171471519674795588238531498465399900093808706",
            "ShardId": "shardId-000000000004"
        },
        {
            "SequenceNumber":
 "49600902273357540915989931256902715169698037101720764562",
            "ShardId": "shardId-000000000009"
        }
    ],
```

```
    "EncryptionType": "KMS"
}
```

Weitere Informationen finden Sie unter Entwickeln von Produzenten, die die Amazon Kinesis Data Streams-API mit dem AWS SDK for Java verwenden im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie PutRecordsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutRecordsCommand, KinesisClient } from "@aws-sdk/client-kinesis";

/**
 * Put multiple records into a Kinesis stream.
 * @param {{ streamArn: string }} config
 */
export const main = async ({ streamArn }) => {
  const client = new KinesisClient({});
  try {
    await client.send(
      new PutRecordsCommand({
        StreamARN: streamArn,
        Records: [
          {
            Data: new Uint8Array(),
            /**
             * Determines which shard in the stream the data record is assigned
 to.
             * Partition keys are Unicode strings with a maximum length limit of
 256
             * characters for each key. Amazon Kinesis Data Streams uses the
 partition
```

```
             * key as input to a hash function that maps the partition key and
             * associated data to a specific shard.
             */
            PartitionKey: "TEST_KEY",
          },
          {
            Data: new Uint8Array(),
            PartitionKey: "TEST_KEY",
          },
        ],
      }),
    );
  } catch (caught) {
    if (caught instanceof Error) {
      //
    } else {
      throw caught;
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    streamArn: {
      type: "string",
      description: "The ARN of the stream.",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Einzelheiten zur API finden Sie PutRecordsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`RegisterStreamConsumer`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterStreamConsumer` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// This example shows how to register a consumer to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class RegisterConsumer
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string consumerName = "NEW_CONSUMER_NAME";
            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";

            var consumer = await RegisterConsumerAsync(client, consumerName,
 streamARN);

            if (consumer is not null)
            {
                Console.WriteLine($"{consumer.ConsumerName}");
            }
        }

        /// <summary>
        /// Registers the consumer to a Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client object.</param>
```

```
        /// <param name="consumerName">A string representing the consumer.</
param>
        /// <param name="streamARN">The ARN of the stream.</param>
        /// <returns>A Consumer object that contains information about the
 consumer.</returns>
        public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis
 client, string consumerName, string streamARN)
        {
            var request = new RegisterStreamConsumerRequest
            {
                ConsumerName = consumerName,
                StreamARN = streamARN,
            };

            var response = await client.RegisterStreamConsumerAsync(request);
            return response.Consumer;
        }
    }
```

- Einzelheiten zur API finden Sie RegisterStreamConsumerin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So registrieren Sie einen Datenstromkonsumenten

Im folgenden Beispiel für `register-stream-consumer` wird ein Konsument namens `KinesisConsumerApplication` beim angegebenen Datenstrom registriert.

```
aws kinesis register-stream-consumer \
    --stream-arn arn:aws:kinesis:us-west-2:012345678912:stream/samplestream \
    --consumer-name KinesisConsumerApplication
```

Ausgabe:

```
{
    "Consumer": {
```

```
        "ConsumerName": "KinesisConsumerApplication",
        "ConsumerARN": "arn:aws:kinesis:us-west-2: 123456789012:stream/
samplestream/consumer/KinesisConsumerApplication:1572383852",
        "ConsumerStatus": "CREATING",
        "ConsumerCreationTimestamp": 1572383852.0
    }
}
```

Weitere Informationen finden Sie unter Developing Consumer with Enhanced Fan-Out Using the Kinesis Data Streams API im Amazon Kinesis Data Streams Developer Guide.

- Einzelheiten zur API finden Sie RegisterStreamConsumerin der AWS CLI Befehlsreferenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_kns->registerstreamconsumer(       " oo_result is returned
for testing purposes. "
           iv_streamarn = iv_stream_arn
           iv_consumername = iv_consumer_name ).
       MESSAGE 'Stream consumer registered.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie RegisterStreamConsumerin der API-Referenz zum AWS SDK für SAP ABAP.

Serverless-Beispiele für Kinesis

Die folgenden Codebeispiele zeigen, wie Kinesis mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
```

```
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
```

```
        return data;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
```

```
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
```

```
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+"
 records");
        return null;
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Ein Kinesis-Ereignis mit Lambda verwenden. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
```

```
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Ein Kinesis-Ereignis mit Lambda verwenden. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
```

```
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
```

```
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
```

```
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error>
 {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);
```

```
        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
```

```
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure
 { ItemIdentifier = record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
```

```
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }
```

```
  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
```

```
public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord =
 kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Melden von Fehlern Kinesis Kinesis-Batch-Elementen mit Lambda unter Verwendung von. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
```

```
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
```

```
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
```

```
  sleep(1)
  data
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );
```

```
        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
```

```
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Codebeispiele für AWS KMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Key Management Service mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS KMS Entwicklerhandbuch — Weitere Informationen zu AWS KMS.
- AWS KMS API-Referenz — Details zu allen verfügbaren AWS KMS Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS KMS verwenden AWS SDKs
  - Hallo AWS Key Management Service
  - Lernen Sie die Grundlagen von AWS KMS mit einem AWS SDK
  - Aktionen für AWS KMS verwenden AWS SDKs

- Verwenden Sie mit CreateAliaseinem AWS SDK oder CLI
- Verwenden Sie CreateGrantmit einem AWS SDK oder CLI
- Verwenden Sie CreateKeymit einem AWS SDK oder CLI
- Verwenden Sie Decrypt mit einem AWS SDK oder CLI
- Verwenden Sie mit DeleteAliaseinem AWS SDK oder CLI
- Verwenden Sie DescribeKeymit einem AWS SDK oder CLI
- Verwenden Sie DisableKeymit einem AWS SDK oder CLI
- Verwenden Sie EnableKeymit einem AWS SDK oder CLI
- Verwenden Sie EnableKeyRotationmit einem AWS SDK oder CLI
- Verwenden Sie Encrypt mit einem AWS SDK oder CLI
- Verwenden Sie mit GenerateDataKeyeinem AWS SDK oder CLI
- Verwenden Sie GenerateDataKeyWithoutPlaintextmit einem AWS SDK oder CLI
- Verwenden Sie GenerateRandommit einem AWS SDK oder CLI
- Verwenden Sie GetKeyPolicymit einem AWS SDK oder CLI
- Verwenden Sie ListAliasesmit einem AWS SDK oder CLI
- Verwenden Sie ListGrantsmit einem AWS SDK oder CLI
- Verwenden Sie ListKeyPoliciesmit einem AWS SDK oder CLI
- Verwenden Sie ListKeysmit einem AWS SDK oder CLI
- Verwenden Sie PutKeyPolicymit einem AWS SDK oder CLI
- Verwenden Sie ReEncryptmit einem AWS SDK oder CLI
- Verwenden Sie RetireGrantmit einem AWS SDK oder CLI
- Verwenden Sie RevokeGrantmit einem AWS SDK oder CLI
- Verwenden Sie ScheduleKeyDeletionmit einem AWS SDK oder CLI
- Verwenden Sie „Signieren“ mit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UpdateAliasmit einem AWS SDK oder CLI
- Verwenden Sie Verify mit einem AWS SDK oder CLI

- Szenarien für AWS KMS verwenden AWS SDKs
  - Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

Grundlegende Beispiele für AWS KMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Key Management Service mit AWS SDKs verwendet werden.

Beispiele

- Hallo AWS Key Management Service
- Lernen Sie die Grundlagen von AWS KMS mit einem AWS SDK
- Aktionen für AWS KMS verwenden AWS SDKs
  - Verwenden Sie mit CreateAliaseinem AWS SDK oder CLI
  - Verwenden Sie CreateGrantmit einem AWS SDK oder CLI
  - Verwenden Sie CreateKeymit einem AWS SDK oder CLI
  - Verwenden Sie Decrypt mit einem AWS SDK oder CLI
  - Verwenden Sie mit DeleteAliaseinem AWS SDK oder CLI
  - Verwenden Sie DescribeKeymit einem AWS SDK oder CLI
  - Verwenden Sie DisableKeymit einem AWS SDK oder CLI
  - Verwenden Sie EnableKeymit einem AWS SDK oder CLI
  - Verwenden Sie EnableKeyRotationmit einem AWS SDK oder CLI
  - Verwenden Sie Encrypt mit einem AWS SDK oder CLI
  - Verwenden Sie mit GenerateDataKeyeinem AWS SDK oder CLI
  - Verwenden Sie GenerateDataKeyWithoutPlaintextmit einem AWS SDK oder CLI
  - Verwenden Sie GenerateRandommit einem AWS SDK oder CLI
  - Verwenden Sie GetKeyPolicymit einem AWS SDK oder CLI
  - Verwenden Sie ListAliasesmit einem AWS SDK oder CLI
  - Verwenden Sie ListGrantsmit einem AWS SDK oder CLI
  - Verwenden Sie ListKeyPoliciesmit einem AWS SDK oder CLI
  - Verwenden Sie ListKeysmit einem AWS SDK oder CLI
  - Verwenden Sie PutKeyPolicymit einem AWS SDK oder CLI
  - Verwenden Sie ReEncryptmit einem AWS SDK oder CLI
  - Verwenden Sie RetireGrantmit einem AWS SDK oder CLI
  - Verwenden Sie RevokeGrantmit einem AWS SDK oder CLI
  - Verwenden Sie ScheduleKeyDeletionmit einem AWS SDK oder CLI

- Verwenden Sie „Signieren“ mit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UpdateAliasmit einem AWS SDK oder CLI
- Verwenden Sie Verify mit einem AWS SDK oder CLI

Hallo AWS Key Management Service

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
```

```
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
 the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
 to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für PHP API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_kms->listkeys( ).
    MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für SAP ABAP.

Lernen Sie die Grundlagen von AWS KMS mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen eines KMS-Schlüssels.
- Listen Sie die KMS-Schlüssel für Ihr Konto auf und rufen Sie Details über sie ab.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.
- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Taggen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String accountId = "";
```

```
   private static final Logger logger =
LoggerFactory.getLogger(KMSScenario.class);

   static KMSActions kmsActions = new KMSActions();

   static Scanner scanner = new Scanner(System.in);

   static String aliasName = "alias/dev-encryption-key";

   public static void main(String[] args) {
       final String usage = """
           Usage: <granteePrincipal>

           Where:
              granteePrincipal - The principal (user, service account, or group)
to whom the grant or permission is being given.
           """;

       if (args.length != 1) {
           logger.info(usage);
           return;
       }
       String granteePrincipal = args[0];
       String policyName = "default";

       accountId = kmsActions.getAccountId();
       String keyDesc = "Created by the AWS KMS API";

       logger.info(DASHES);
       logger.info("""
           Welcome to the AWS Key Management SDK Basics scenario.

           This program demonstrates how to interact with AWS Key Management
using the AWS SDK for Java (v2).
           The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
           and manage AWS KMS keys and control their use across a wide range of
AWS services and applications.
           KMS provides a centralized and unified approach to managing
encryption keys, making it easier to meet your
           data protection and regulatory compliance requirements.

           This Basics scenario creates two key types:
```

```
            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " +
targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next
bit of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
```

```
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt
data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
```

```
            """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately
365 days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along
with key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
```

```
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
```

```
            The revocation of a grant immediately removes the permissions and
access that the grant had provided.
            This means that any principal (user, role, or service) that was
granted access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
```

```
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy.
The statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is
created that
            gives the AWS account that owns the KMS key full access to the KMS
key.

            Let's try to replace the automatically created policy with the
following policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
```

```
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make
sure it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
             for your data signing needs. By using an AWS KMS key, you can
leverage the
             security controls and compliance features provided by AWS,
             which can help you meet various regulatory requirements and enhance
the overall security posture
             of your organization.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String
targetKeyId) {
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
```

```
            but you can specify a waiting period of 7-30 days. When this
 operation is successful,
            the key state of the KMS key changes to PendingDeletion and the key
 can't be used in any
            cryptographic operations. It remains in this state for the duration
 of the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
 rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
 kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
 code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
```

```
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " +
rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
        }

        try {
            CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
```

```
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definieren Sie eine Klasse, die KMS-Aktionen umschließt.

```
public class KMSActions {
    private static final Logger logger =
 LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async
 client, with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
 creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
```

```
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
```

```
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and
key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }

    /**
     * Enables key rotation asynchronously for the specified key ID.
     *
     * @param keyId the ID of the key for which to enable key rotation
     * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
     * @throws RuntimeException if there was an error enabling key rotation,
either due to a KMS exception or an unexpected error
     */
    public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
```

```
        EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: "
+ kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
```

```
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
     * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
     *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *         if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
```

```
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key
ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
```

```
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean
indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
```

```
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and
policy name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
```

```
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
     *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes =
signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
```

```
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
```

```
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();
```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
 key ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion

- Sign
- TagResource

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "\n";
        echo "-------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using
 the AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "-------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
```

```
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
        echo "First, we will create a symmetric KMS key that is used to encrypt
 and decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code
 checks to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt
 data.\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
```

```
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key:
 {$key['KeyId']}\n";
        $this->pressEnter();
```

```
        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon
 Web Services accounts, IAM users, IAM roles, federated users, and assumed
 role users. For help with the ARN syntax for a principal, see IAM ARNs in the
 Identity and Access Management User Guide. \nTo skip this step, press enter
 without any other values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
```

```
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted
 earlier in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are
 the primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in
 the key policy determine who has permission to use the KMS key and how they can
 use it.\n";
        echo " You can also use IAM policies and grants to control access to the
 KMS key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
```

```
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
 screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
 make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
 compliance features provided by AWS, which can help you meet various regulatory
 requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text,
 $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
 $sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
```

```
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
 and governance of your KMS keys, making it easier to organize, track, and
 control access to your encrypted data within your AWS environment.\n";
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n
\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS
 key is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources
 created during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "--------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
```

```
        echo
 "------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
```

```
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
not enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
```

```
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
```

```
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
```

```
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
```

```
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature,
 string $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
```

```
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS
 SDK for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service
 that allows you to create
```

```
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys,
 making it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can
 used to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
```

```
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly
 name that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
 from the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
```

```
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use
 KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
 grants.
When authorizing access to a KMS key, grants are considered along with key
 policies and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee
 account_id, use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM
 roles, federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
```

```
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access
 to perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way
 to control
access to KMS keys. Every KMS key must have exactly one key policy. The
 statements in the key policy
determine who has permission to use the KMS key and how they can use it.
You can also use IAM policies and grants to control access to the KMS key, but
 every KMS key
must have a key policy.
```

```
By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version":"2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                        "Action": "kms:*",
                        "Resource": "*",
                    }
                ],
            }

            self.key_policy.set_new_policy(self.key_id, policy)
            print("Key policy replacement succeeded.")
            q.ask("Press Enter to continue...")
        else:
            print("Skipping replacing the key policy.")

        print(DASHES)
        print(f"12. Get the key policy\n")
        print(
            f"The next bit of code that runs gets the key policy to make sure it
 exists."
        )
        q.ask("Press Enter to continue...")
        policy = self.key_policy.get_policy(self.key_id)
```

```
        print(f"The key policy is: {policy}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"13. Create an asymmetric KMS key and sign your data\n")
        print(
            """
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
        """
        )
        q.ask("Press Enter to continue...")
        print(f"Sign and verify data operation succeeded.")
        self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
        message = "Here is the message that will be digitally signed"
        signature = self.key_encryption.sign(self.asymmetric_key_id, message)
        if self.key_encryption.verify(self.asymmetric_key_id, message,
signature):
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to your
encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()
```

```
    def is_key_enabled(self, key_id: str) -> bool:
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a
 KMS key is deleted,
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
                q.is_yesno,
```

```
            ):
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a
 window of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverwaltung.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
```

```
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
```

```
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselaliase.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
```

```
                raise


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverschlüsselung.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
```

```
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
CiphertextBlob=cipher_text)[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
```

```
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselberechtigungen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselrichtlinien.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
 principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
 that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id,
 Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
```

```
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey

- EnableKey
- Encrypt
- GetKeyPolicy
- ListAliases
- ListGrants
- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Aktionen für AWS KMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS KMS Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS KMS API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS KMS verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Key Management Service -API-Referenz.

Beispiele

- Verwenden Sie mit CreateAliaseinem AWS SDK oder CLI
- Verwenden Sie CreateGrantmit einem AWS SDK oder CLI
- Verwenden Sie CreateKeymit einem AWS SDK oder CLI
- Verwenden Sie Decrypt mit einem AWS SDK oder CLI
- Verwenden Sie mit DeleteAliaseinem AWS SDK oder CLI
- Verwenden Sie DescribeKeymit einem AWS SDK oder CLI
- Verwenden Sie DisableKeymit einem AWS SDK oder CLI
- Verwenden Sie EnableKeymit einem AWS SDK oder CLI

- Verwenden Sie EnableKeyRotationmit einem AWS SDK oder CLI
- Verwenden Sie Encrypt mit einem AWS SDK oder CLI
- Verwenden Sie mit GenerateDataKeyeinem AWS SDK oder CLI
- Verwenden Sie GenerateDataKeyWithoutPlaintextmit einem AWS SDK oder CLI
- Verwenden Sie GenerateRandommit einem AWS SDK oder CLI
- Verwenden Sie GetKeyPolicymit einem AWS SDK oder CLI
- Verwenden Sie ListAliasesmit einem AWS SDK oder CLI
- Verwenden Sie ListGrantsmit einem AWS SDK oder CLI
- Verwenden Sie ListKeyPoliciesmit einem AWS SDK oder CLI
- Verwenden Sie ListKeysmit einem AWS SDK oder CLI
- Verwenden Sie PutKeyPolicymit einem AWS SDK oder CLI
- Verwenden Sie ReEncryptmit einem AWS SDK oder CLI
- Verwenden Sie RetireGrantmit einem AWS SDK oder CLI
- Verwenden Sie RevokeGrantmit einem AWS SDK oder CLI
- Verwenden Sie ScheduleKeyDeletionmit einem AWS SDK oder CLI
- Verwenden Sie „Signieren“ mit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie UpdateAliasmit einem AWS SDK oder CLI
- Verwenden Sie Verify mit einem AWS SDK oder CLI

Verwenden Sie mit **`CreateAlias`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAlias` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Alias, {aliasName}, successfully created.");
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Alias für einen KMS-Schlüssel

Der folgende `create-alias`-Befehl erstellt einen Alias namens `example-alias` für den KMS-Schlüssel, der durch die Schlüssel-ID `1234abcd-12ab-34cd-56ef-1234567890ab` identifiziert wird.

Aliasnamen müssen mit `alias/` beginnen. Verwenden Sie keine Aliasnamen, die mit `alias/aws` beginnen. Diese sind für die Verwendung durch reserviert AWS.

```
aws kms create-alias \
    --alias-name alias/example-alias \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Dieser Befehl gibt keine Ausgabe zurück. Verwenden Sie den Befehl `list-aliases`, um den neuen Alias anzuzeigen.

Weitere Informationen finden Sie unter Verwenden von Aliasnamen im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie CreateAliasin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
               } else if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
               }
           }
```

```
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- Einzelheiten zur API finden Sie CreateAliasin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name
is not valid.";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie CreateAliasin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
```

```
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie CreateAliasin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->createalias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_key_id
        ).
        MESSAGE 'Alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsalreadyexistsex.
        MESSAGE 'Alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidaliasnameex.
        MESSAGE 'Invalid alias name.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateAliasin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateGrant`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGrant` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identity that is given permission to perform the operations
            // specified in the grant.
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS
key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
                KeyId = keyId,

                // A list of operations that the grant allows.
```

```
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
 grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Erteilung

Im folgenden Beispiel für `create-grant` wird eine Erteilung erstellt, die dem Benutzer `exampleUser` die Verwendung des Befehls `decrypt` im Beispiel-KMS-Schlüssel `1234abcd-12ab-34cd-56ef-1234567890ab` ermöglicht. Der zurückziehende Prinzipal ist die `adminRole`-Rolle. Die Erteilung verwendet die Erteilungseinschränkung `EncryptionContextSubset`, um diese Berechtigung nur dann zuzulassen, wenn der Verschlüsselungskontext in der Anforderung `decrypt` das Verschlüsselungskontextpaar `"Department": "IT"` enthält.

```
aws kms create-grant \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --grantee-principal arn:aws:iam::123456789012:user/exampleUser \
    --operations Decrypt \
    --constraints EncryptionContextSubset={Department=IT} \
    --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

Ausgabe:

```
{
    "GrantId":
 "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",
    "GrantToken": "<grant token here>"
}
```

Verwenden Sie zum Anzeigen detaillierter Informationen zur Erteilung den Befehl `list-grants`.

Weitere Informationen finden Sie unter Grants in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie CreateGrantunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID
of the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
```

```
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGrant(
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- Einzelheiten zur API finden Sie CreateGrantin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /***
    * @param string $keyId
    * @param string $granteePrincipal
    * @param array $operations
    * @param array $grantTokens
    * @return Result
    */
   public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
   {
       $args = [
           'KeyId' => $keyId,
           'GranteePrincipal' => $granteePrincipal,
           'Operations' => $operations,
       ];
       if($grantTokens){
           $args['GrantTokens'] = $grantTokens;
       }
       try{
           return $this->client->createGrant($args);
       }catch(KmsException $caught){
           if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
               echo "The request was rejected because the specified grant token
is not valid.\n";
           }
           throw $caught;
       }
   }
```

- Einzelheiten zur API finden Sie CreateGrantin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
```

```
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateGrantin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grantee_principal = 'arn:aws:iam::123456789012:role/my-role'
        " it_operations contains 'Encrypt', 'Decrypt', 'GenerateDataKey'
        oo_result = lo_kms->creategrant(
          iv_keyid = iv_key_id
          iv_granteeprincipal = iv_grantee_principal
          it_operations = it_operations
        ).
        MESSAGE 'Grant created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
```

```
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateGrantin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateKey`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateKey` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Arbeiten mit der Tabellenverschlüsselung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Shows how to create a new AWS Key Management Service (AWS KMS)
/// key.
/// </summary>
public class CreateKey
{
    public static async Task Main()
    {
```

```
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
 key.
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
 created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen vom Kunden verwalteten KMS-Schlüssel in AWS KMS

Im folgenden Beispiel für `create-key` wird ein KMS-Schlüssel für die symmetrische Verschlüsselung erstellt.

Zum Erstellen des grundlegenden KMS-Schlüssels, einem symmetrischen Verschlüsselungsschlüssel, müssen Sie keine Parameter angeben. Die Standardwerte für diese Parameter erstellen einen symmetrischen Verschlüsselungsschlüssel.

Da dieser Befehl keine Schlüsselrichtlinie angibt, erhält der KMS-Schlüssel die Standardschlüsselrichtlinie für KMS-Schlüssel, die programmgesteuert erstellt werden. Verwenden Sie den Befehl `get-key-policy`, um die Schlüsselrichtlinie anzuzeigen. Verwenden Sie den Befehl `put-key-policy`, um die Schlüsselrichtlinie zu ändern.

```
aws kms create-key
```

Der Befehl `create-key` gibt die wichtigsten Metadaten zurück, einschließlich der Schlüssel-ID und des ARN des neuen KMS-Schlüssels. Sie können diese Werte verwenden, um den KMS-Schlüssel in anderen AWS KMS-Vorgängen zu identifizieren. Die Ausgabe enthält keine Tags. Verwenden Sie den `list-resource-tags command`, um die Tags für einen KMS-Schlüssel anzuzeigen.

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2017-07-05T14:04:55-07:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_KMS"
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ]
    }
```

```
}
```

Hinweis: Mit dem Befehl `create-key` können Sie keinen Alias angeben. Verwenden Sie den Befehl `create-alias`, um einen Alias für den neuen KMS-Schlüssel zu erstellen.

Weitere Informationen finden Sie unter Erstellen von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

Beispiel 2: So erstellen Sie einen asymmetrischen RSA-KMS-Schlüssel für die Verschlüsselung und Entschlüsselung

Im folgenden Beispiel für `create-key` wird ein KMS-Schlüssel erstellt, der ein asymmetrisches RSA-Schlüsselpaar für die Verschlüsselung und Entschlüsselung enthält. Die Schlüsselspezifikation und -verwendung können nach dem Erstellen des Schlüssels nicht geändert werden.

```
aws kms create-key \
   --key-spec RSA_4096 \
   --key-usage ENCRYPT_DECRYPT
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "RSA_4096",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "RSAES_OAEP_SHA_1",
            "RSAES_OAEP_SHA_256"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_4096",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
```

```
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 3: So erstellen Sie einen KMS-Schlüssel mit asymmetrischer Elliptic-Curve für die Signatur und Verifizierung

Um einen asymmetrischen KMS-Schlüssel zu erstellen, der ein asymmetrisches Schlüsselpaar mit elliptischer Kurve (ECC) für Signatur und Verifizierung enthält. Der Parameter `--key-usage` ist erforderlich, obwohl `SIGN_VERIFY` der einzig gültige Wert für ECC-KMS-Schlüssel ist. Die Schlüsselspezifikation und -verwendung können nach dem Erstellen des Schlüssels nicht geändert werden.

```
aws kms create-key \
    --key-spec ECC_NIST_P521 \
    --key-usage SIGN_VERIFY
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "ECC_NIST_P521",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ECC_NIST_P521",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ECDSA_SHA_512"
        ]
    }
```

```
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 4: So erstellen Sie einen asymmetrischen ML-DSA KMS-Schlüssel zum Signieren und Überprüfen

In diesem Beispiel wird ein Modul-Gitter-Schlüssel für digitale Signaturen (ML-DSA) zum Signieren und Überprüfen erstellt. Der Schlüsselverwendungsparameter ist erforderlich, obwohl er der einzig gültige Wert für Schlüssel `SIGN_VERIFY` ist. ML-DSA

```
aws kms create-key \
    --key-spec ML_DSA_65 \
    --key-usage SIGN_VERIFY
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "ML_DSA_65",
        "KeyState": "Enabled",
        "KeyUsage": "SIGN_VERIFY",
        "MultiRegion": false,
        "Origin": "AWS_KMS",
        "SigningAlgorithms": [
            "ML_DSA_SHAKE_256"
        ]
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 5: So erstellen Sie einen HMAC-KMS-Schlüssel

Im folgenden Beispiel für `create-key` wird ein HMAC-KMS-Schlüssel von 384 Bit erstellt. Der Wert `GENERATE_VERIFY_MAC` für den Parameter `--key-usage` ist erforderlich, obwohl er der einzig gültige Wert für HMAC-KMS-Schlüssel ist.

```
aws kms create-key \
    --key-spec HMAC_384 \
    --key-usage GENERATE_VERIFY_MAC
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-04-05T14:04:55-07:00",
        "CustomerMasterKeySpec": "HMAC_384",
        "Description": "",
        "Enabled": true,
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "HMAC_384",
        "KeyState": "Enabled",
        "KeyUsage": "GENERATE_VERIFY_MAC",
        "MacAlgorithms": [
            "HMAC_SHA_384"
        ],
        "MultiRegion": false,
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter HMAC-Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 6: So erstellen Sie einen multiregionalen KMS-Schlüssel

Im folgenden Beispiel für `create-key` wird ein multiregionaler Primärschlüssel erstellt. Da die Standardwerte für alle Parameter einen symmetrischen Verschlüsselungsschlüssel

erzeugen, ist nur der Parameter `--multi-region` für diesen KMS-Schlüssel erforderlich. Um in der AWS CLI anzugeben, dass ein boolescher Parameter wahr ist, geben Sie einfach den Parameternamen an.

```
aws kms create-key \
    --multi-region
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2021-09-02T016:15:21-09:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
          "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": true,
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
                "Region": "us-west-2"
            },
            "ReplicaKeys": []
        },
        "Origin": "AWS_KMS"
    }
}
```

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 7: So erstellen Sie einen KMS-Schlüssel für importiertes Schlüsselmaterial

Im folgenden Beispiel für `create-key` wird ein KMS-Schlüssel ohne Schlüsselmaterial erstellt. Wenn der Vorgang abgeschlossen ist, können Sie Ihr eigenes Schlüsselmaterial in den KMS-Schlüssel importieren. Um diesen KMS-Schlüssel zu erstellen, legen Sie den Parameter `--origin` auf `EXTERNAL` fest.

```
aws kms create-key \
    --origin EXTERNAL
```

Ausgabe:

```
{
     "KeyMetadata": {
         "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
         "AWSAccountId": "111122223333",
         "CreationDate": "2019-12-02T07:48:55-07:00",
         "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
         "Description": "",
         "Enabled": false,
         "EncryptionAlgorithms": [
             "SYMMETRIC_DEFAULT"
         ],
         "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
         "KeyManager": "CUSTOMER",
         "KeySpec": "SYMMETRIC_DEFAULT",
         "KeyState": "PendingImport",
         "KeyUsage": "ENCRYPT_DECRYPT",
         "MultiRegion": false,
         "Origin": "EXTERNAL"
     }
 }
```

Weitere Informationen finden Sie unter Importieren von Schlüsselmaterial in AWS KMS-Schlüsseln im AWS Key Management Service Developer Guide.

Beispiel 6: So erstellen Sie einen KMS-Schlüssel in einem AWS CloudHSM-Schlüsselspeicher

Das folgende `create-key` Beispiel erstellt einen KMS-Schlüssel im angegebenen AWS CloudHSM-Schlüsselspeicher. Der Vorgang erstellt den KMS-Schlüssel und seine Metadaten in AWS KMS und erstellt das Schlüsselmaterial im AWS CloudHSM-Cluster, der dem benutzerdefinierten Schlüsselspeicher zugeordnet ist. Die Parameter `--custom-key-store-id` und `--origin` müssen angegeben werden.

```
aws kms create-key \
    --origin AWS_CLOUDHSM \
    --custom-key-store-id cks-1234567890abcdef0
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CloudHsmClusterId": "cluster-1a23b4cdefg",
        "CreationDate": "2019-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-1234567890abcdef0",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "AWS_CLOUDHSM"
    }
}
```

Weitere Informationen finden Sie unter AWS -CloudHSM-Schlüsselspeicher im Entwicklerhandbuch für AWS Key Management Service.

Beispiel 8: So erstellen Sie einen KMS-Schlüssel in einem externen Schlüsselspeicher

Im folgenden Beispiel für `create-key` wird ein KMS-Schlüssel im angegebenen externen Schlüsselspeicher erstellt. Die Parameter `--custom-key-store-id`, `--origin` und `--xks-key-id` sind in diesem Befehl erforderlich.

Der Parameter `--xks-key-id` gibt die ID eines vorhandenen symmetrischen Verschlüsselungsschlüssels in Ihrem externen Schlüsselmanager an. Dieser Schlüssel dient als externes Schlüsselmaterial für den KMS-Schlüssel. Der Wert des Parameters `--origin` muss `EXTERNAL_KEY_STORE` lauten. Der Parameter `custom-key-store-id` muss einen externen Schlüsselspeicher identifizieren, der mit dem Proxy seines externen Schlüsselspeichers verbunden ist.

```
aws kms create-key \
    --origin EXTERNAL_KEY_STORE \
    --custom-key-store-id cks-9876543210fedcba9 \
    --xks-key-id bb8562717f809024
```

Ausgabe:

```
{
    "KeyMetadata": {
        "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "AWSAccountId": "111122223333",
        "CreationDate": "2022-12-02T07:48:55-07:00",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "CustomKeyStoreId": "cks-9876543210fedcba9",
        "Description": "",
        "Enabled": true,
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "EXTERNAL_KEY_STORE",
        "XksKeyConfiguration": {
            "Id": "bb8562717f809024"
        }
    }
```

```
}
```

Weitere Informationen finden Sie unter Externe Schlüsselspeicher im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie CreateKeyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new symmetric encryption key asynchronously.
    *
    * @param keyDesc the description of the key to be created
    * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
    * @throws RuntimeException if an error occurs while creating the key
    */
   public CompletableFuture<String> createKeyAsync(String keyDesc) {
       CreateKeyRequest keyRequest = CreateKeyRequest.builder()
           .description(keyDesc)
           .keySpec(KeySpec.SYMMETRIC_DEFAULT)
           .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
           .build();

       return getAsyncClient().createKey(keyRequest)
           .thenApply(resp -> resp.keyMetadata().keyId())
           .exceptionally(ex -> {
               throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
           });
   }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
```

```
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
```

```
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateKeyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_description = 'Created by the AWS SDK for SAP ABAP'
        oo_result = lo_kms->createkey( iv_description = iv_description ).
        MESSAGE 'KMS key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
      CATCH /aws1/cx_kmslimitexceededex.
```

```
        MESSAGE 'Limit exceeded for KMS resources.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`Decrypt mit einem`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Decrypt` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So verschlüsseln Sie eine verschlüsselte Nachricht mit einem symmetrischen KMS-Schlüssel (Linux und macOS)

Das folgende `decrypt` Befehlsbeispiel zeigt die empfohlene Methode zum Entschlüsseln von Daten mit der AWS CLI. Diese Version zeigt, wie Daten unter einem symmetrischen KMS-Schlüssel entschlüsselt werden.

Geben Sie den Geheimtext in einer Datei an. Verwenden Sie im Wert des `--ciphertext-blob`-Parameters das `fileb://`-Präfix, das die CLI anweist, die Daten aus einer Binärdatei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten aus einer Datei finden Sie unter AWS CLI-Parameter aus einer Datei laden < https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html > im AWS Command Line Interface User Guide und Best Practices for Local File Parameters< https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/ > im AWS Command Line Tool Blog .Geben Sie den KMS-Schlüssel an, um den Chiffretext zu entschlüsseln. Der `--key-id` Parameter ist bei der Entschlüsselung mit einem symmetrischen KMS-Schlüssel nicht erforderlich. AWS KMS kann die Schlüssel-ID des KMS-Schlüssels, der zur Verschlüsselung der Daten verwendet wurde, aus den Metadaten im

Chiffretext abrufen. Es ist jedoch immer eine bewährte Methode, den von Ihnen verwendeten KMS-Schlüssel anzugeben. Diese Methode stellt sicher, dass Sie den beabsichtigten KMS-Schlüssel verwenden, und verhindert, dass Sie versehentlich einen Geheimtext mit einem KMS-Schlüssel entschlüsseln, dem Sie nicht vertrauen. Fordern Sie die Klartextausgabe als Textwert an. Der Parameter `--query` weist die CLI an, nur den Wert des Felds `Plaintext` aus der Ausgabe abzurufen. Der `--output` Parameter gibt die Ausgabe als Text zurück. Base64-decode den Klartext und speichern Sie ihn in einer Datei. Im folgenden Beispiel wird der Wert des `Plaintext` Parameters über die Pipeline (|) an das Base64-Hilfsprogramm übergeben, das ihn dekodiert. Anschließend erfolgt eine Umleitung (>) der dekodierten Ausgabe in die Datei `ExamplePlaintext`.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem Konto. AWS

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des Befehls `decrypt` wird Base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie unter Decrypt in der API-Referenz zu AWS Key Management Service.

Beispiel 2: Um eine verschlüsselte Nachricht mit einem symmetrischen KMS-Schlüssel zu entschlüsseln (Windows-Befehlszeile)

Das folgende Beispiel ist dasselbe wie das vorherige, außer dass es das `certutil` Hilfsprogramm für Base64-decode die Klartextdaten verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie in den folgenden Beispielen gezeigt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms decrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
```

```
    --output text ^
    --query Plaintext > ExamplePlaintextFile.base64
```

Führen Sie den Befehl `certutil` aus.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie unter Decrypt in der API-Referenz zu AWS Key Management Service.

Beispiel 3: So entschlüsseln Sie eine verschlüsselte Nachricht mit einem asymmetrischen KMS-Schlüssel (Linux und macOS)

Das folgende Beispiel für den Befehl `decrypt` zeigt, wie Daten entschlüsselt werden, die unter einem asymmetrischen RSA-KMS-Schlüssel verschlüsselt wurden.

Bei Verwendung eines asymmetrischen KMS-Schlüssels ist der Parameter `encryption-algorithm` erforderlich, der den Algorithmus zum Verschlüsseln des Klartextes angibt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms decrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --output text \
    --query Plaintext | base64 \
    --decode > ExamplePlaintextFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des Befehls `decrypt` wird Base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

- Weitere API-Informationen finden Sie unter Decrypt in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously decrypts the given encrypted data using the specified key
ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
               } else {
```

```
                    throw new RuntimeException("An unexpected error occurred
 while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
 decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
```

```
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
```

```
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
 = "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.
```

```
        :return: An instance of KeyEncrypt initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id,
CiphertextBlob=cipher_text)[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base
 64 characters.")
        })
```

```
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to
 UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS -SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->decrypt(
```

```
          iv_keyid = iv_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Text decrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter Decrypt in AWS SDK for SAP ABAP API-Referenz.

Verwenden Sie mit **`DeleteAlias`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAlias` verwendet wird.

CLI

AWS CLI

Um einen AWS KMS-Alias zu löschen

Im folgenden Beispiel für `delete-alias` wird die Tabelle `alias/example-alias` gelöscht. Der Aliasname muss mit „alias/“ beginnen.

```
aws kms delete-alias \
    --alias-name alias/example-alias
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `list-aliases`, um den Alias zu suchen.

Weitere Informationen finden Sie unter Löschen eines Alias im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie DeleteAliasin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }
```

- Einzelheiten zur API finden Sie DeleteAliasin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteAliasin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteAliasin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_alias_name = 'alias/my-key-alias'
    lo_kms->deletealias( iv_aliasname = iv_alias_name ).
    MESSAGE 'Alias deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Alias not found.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteAliasin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeKey`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;

        Console.WriteLine($"{metadata.KeyId} created on:
{metadata.CreationDate}");
        Console.WriteLine($"State: {metadata.KeyState}");
        Console.WriteLine($"{metadata.Description}");
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So finden Sie detaillierte Informationen zu einem KMS-Schlüssel

Im folgenden `describe-key` Beispiel werden detaillierte Informationen zum AWS verwalteten Schlüssel für Amazon S3 im Beispielkonto und in der Region abgerufen. Sie können diesen Befehl verwenden, um Details zu AWS verwalteten Schlüsseln und kundenverwalteten Schlüsseln zu finden.

Verwenden Sie den Parameter `key-id`, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird der Wert für einen Aliasnamen verwendet, Sie können in diesem Befehl jedoch eine Schlüssel-ID, einen Schlüssel-ARN, einen Aliasnamen oder einen Alias-ARN verwenden.

```
aws kms describe-key \
    --key-id alias/aws/s3
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "846764612917",
        "KeyId": "b8a9477d-836c-491f-857e-07937918959b",
        "Arn": "arn:aws:kms:us-west-2:846764612917:key/
b8a9477d-836c-491f-857e-07937918959b",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "CreationDate": 2017-06-30T21:44:32.140000+00:00,
        "Enabled": true,
        "Description": "Default KMS key that protects my S3 objects when no other
 key is defined",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "AWS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
```

```
            "SYMMETRIC_DEFAULT"
        ]
    }
}
```

Weitere Informationen finden Sie unter Anzeigen von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

Beispiel 2: So rufen Sie Details zu einem asymmetrischen RSA-KMS-Schlüssel ab

Im folgenden Beispiel für `describe-key` werden detaillierte Informationen zu einem asymmetrischen RSA-KMS-Schlüssel abgerufen, der für Signatur und Verifizierung verwendet wird.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "111122223333",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2019-12-02T19:47:14.861000+00:00",
        "CustomerMasterKeySpec": "RSA_2048",
        "Enabled": false,
        "Description": "",
        "KeyState": "Disabled",
        "Origin": "AWS_KMS",
        "MultiRegion": false,
        "KeyManager": "CUSTOMER",
        "KeySpec": "RSA_2048",
        "KeyUsage": "SIGN_VERIFY",
        "SigningAlgorithms": [
            "RSASSA_PKCS1_V1_5_SHA_256",
            "RSASSA_PKCS1_V1_5_SHA_384",
            "RSASSA_PKCS1_V1_5_SHA_512",
            "RSASSA_PSS_SHA_256",
            "RSASSA_PSS_SHA_384",
            "RSASSA_PSS_SHA_512"
        ]
```

```
    }
}
```

Beispiel 3: So rufen Sie Details zu einem Replikatschlüssel für mehrere Regionen ab

Im folgenden Beispiel für `describe-key` werden Metadaten für einen multiregionalen Replikatschlüssel abgerufen. Der multiregionale Schlüssel ist ein symmetrischer Verschlüsselungsschlüssel. Die Ausgabe eines `describe-key`-Befehls für einen beliebigen Schlüssel mit mehreren Regionen gibt Informationen zum Primärschlüssel und allen zugehörigen Replikaten zurück.

```
aws kms describe-key \
    --key-id arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "MultiRegion": true,
        "AWSAccountId": "111122223333",
        "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "CreationDate": "2021-06-28T21:09:16.114000+00:00",
        "CurrentKeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
        "Description": "",
        "Enabled": true,
        "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
        "KeyManager": "CUSTOMER",
        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "Origin": "AWS_KMS",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegionConfiguration": {
            "MultiRegionKeyType": "PRIMARY",
            "PrimaryKey": {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
```

```
                "Region": "us-west-2"
            },
            "ReplicaKeys": [
                {
                    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "eu-west-1"
                },
                {
                    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "ap-northeast-1"
                },
                {
                    "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                    "Region": "sa-east-1"
                }
            ]
        }
    }
}
```

Beispiel 4: So rufen Sie Details zu einem asymmetrischen HMAC-KMS-Schlüssel ab

Im folgenden Beispiel für `describe-key` werden detaillierte Informationen zu einem HMAC-KMS-Schlüssel abgerufen.

```
aws kms describe-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2022-04-03T22:23:10.194000+00:00",
        "Enabled": true,
        "Description": "Test key",
        "KeyUsage": "GENERATE_VERIFY_MAC",
```

```
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "HMAC_256",
        "MacAlgorithms": [
            "HMAC_SHA_256"
        ],
        "MultiRegion": false
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
```

```
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
 keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeKeyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->describekey( iv_keyid = iv_key_id ).
        DATA(lo_key) = oo_result->get_keymetadata( ).
        MESSAGE 'Retrieved key information successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DisableKey`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DisableKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been disabled.
```

```
                var describeResponse = await client.DescribeKeyAsync(new
 DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
 {describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So deaktivieren Sie einen KMS-Schlüssel vorübergehend

Der folgende Befehl `disable-key` deaktiviert einen kundenseitig verwalteten KMS-Schlüssel. Verwenden Sie den Befehl `enable-key`, um den KMS-Schlüssel erneut zu aktivieren.

```
aws kms disable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Aktivieren und Deaktivieren von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie DisableKeyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key
has been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- Einzelheiten zur API finden Sie DisableKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
     */
```

```
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie DisableKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DisableKeyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->disablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DisableKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableKey`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
        {
            KeyId = keyId,
        };
```

```
            var response = await client.EnableKeyAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been enabled.
                var describeResponse = await client.DescribeKeyAsync(new
 DescribeKeyRequest
                {
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
 {describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So aktivieren Sie einen KMS-Schlüssel

Im folgenden Beispiel für `enable-key` wird ein kundenseitig verwalteter Schlüssel aktiviert. Sie können einen Befehl wie diesen verwenden, um einen KMS-Schlüssel zu aktivieren, den Sie mit dem Befehl `disable-key` vorübergehend deaktiviert haben. Sie können ihn auch verwenden, um einen deaktivierten KMS-Schlüssel zu aktivieren, er zum Löschen vorgesehen war und der Löschvorgang abgebrochen wurde.

Verwenden Sie den Parameter `key-id`, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, Sie können jedoch eine Schlüssel-ID- oder einen Schlüssel-ARN-Wert in diesem Befehl verwenden.

Ersetzen Sie vor Ausführung dieses Befehls die Beispielschlüssel-ID durch eine gültige.

```
aws kms enable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `describe-key`, um zu überprüfen, ob der KMS-Schlüssel aktiviert ist. Sehen Sie sich die Werte der Felder `KeyState` und `Enabled` und in der Ausgabe von `describe-key` an.

Weitere Informationen finden Sie unter Aktivieren und Deaktivieren von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie EnableKeyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
```

```
                    throw new RuntimeException("An unexpected error occurred
 while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- Einzelheiten zur API finden Sie EnableKeyin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie EnableKeyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie EnableKeyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie EnableKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`EnableKeyRotation`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `EnableKeyRotation` verwendet wird.

CLI

AWS CLI

So aktivieren Sie die automatische Rotation eines KMS-Schlüssels

Das folgende Beispiel für `enable-key-rotation` ermöglicht die automatische Rotation eines kundenseitig verwalteten KMS-Schlüssels mit einem Rotationszeitraum von 180 Tagen. Der KMS-Schlüssel wird ab dem Datum, an dem dieser Befehl abgeschlossen wurde, ein Jahr (ungefähr 365 Tage) rotiert, und danach jedes Jahr.

Der Parameter `--key-id` identifiziert den KMS-Schlüssel. In diesem Beispiel wird ein Schlüssel-ARN-Wert verwendet, Sie können jedoch entweder die Schlüssel-ID oder den ARN des KMS-Schlüssels verwenden. Der Parameter `--rotation-period-in-days` gibt die Anzahl der Tage zwischen den einzelnen Rotationsdatumsangaben an. Geben Sie einen Wert zwischen 90 und 2 560 an. Wenn kein Wert angegeben wird, lautet der Standardwert 365 Tage.

```
aws kms enable-key-rotation \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --rotation-period-in-days 180
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `get-key-rotation-status`, um zu überprüfen, ob der KMS-Schlüssel aktiviert ist.

Weitere Informationen finden Sie unter Rotieren von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie EnableKeyRotationin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```
    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie EnableKeyRotationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekeyrotation( iv_keyid = iv_key_id ).
        MESSAGE 'Key rotation enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsunsupportedopex.
        MESSAGE 'Operation not supported for this key.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie EnableKeyRotationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`Encrypt mit einem`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Encrypt` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So verschlüsseln Sie den Inhalt einer Datei unter Linux oder MacOS

Der folgende `encrypt` Befehl demonstriert die empfohlene Methode zum Verschlüsseln von Daten mit der AWS CLI.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
```

```
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

Der Befehl führt mehrere Aktionen aus:

Er verwendet den Parameter `--plaintext`, um die zu verschlüsselnden Daten anzugeben. Dieser Parameterwert muss Base64-kodiert sein. Der Wert des `plaintext` Parameters muss Base64-kodiert sein, oder Sie müssen das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Beispiel: `fileb:///var/tmp/ExamplePlaintextFile` oder `fileb://C:\Temp\ExamplePlaintextFile`. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten aus einer Datei finden Sie unter Laden von Parametern aus einer Datei im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle und Bewährte Methoden für lokale Dateiparameter im AWS Befehlszeilentool `--output` sowie unter `--query` Parameter zur Steuerung Blog.Uses der Befehlsausgabe. Diese Parameter extrahieren die verschlüsselten Daten, den so genannten Chiffretext, aus der Befehlsausgabe. Weitere Informationen zur Steuerung der Ausgabe finden Sie unter Steuerung des Befehls. Ausgabe im AWS Command Line Interface User Guide. Verwendet das `base64` Hilfsprogramm, um die extrahierte Ausgabe in Binärdaten zu dekodieren. Der Chiffretext, der von einem erfolgreichen `encrypt` Befehl zurückgegeben wird, ist Base64-codierter Text. Sie müssen diesen Text dekodieren, bevor Sie ihn mit der AWS CLI entschlüsseln können. Speichert den binären Chiffretext in einer Datei. Der letzte Teil des Befehls (`> ExampleEncryptedFile`) speichert den binären Chiffretext in einer Datei, um die Entschlüsselung zu erleichtern. Einen Beispielbefehl, der die AWS CLI zum Entschlüsseln von Daten verwendet, finden Sie in den Entschlüsselungsbeispielen.

Beispiel 2: Verwenden der AWS CLI zum Verschlüsseln von Daten unter Windows

Dieses Beispiel ist mit dem vorherigen identisch, mit dem Unterschied, dass es das Tool `certutil` anstelle von `base64` verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie in den folgenden Beispielen gezeigt.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64

certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

Beispiel 3: Verschlüsseln mit einem asymmetrischen KMS-Schlüssel

Der folgende Befehl `encrypt` zeigt, wie Klartext mit einem asymmetrischen KMS-Schlüssel verschlüsselt wird. Der Parameter `--encryption-algorithm` muss angegeben werden. Wie bei allen `encrypt` CLI-Befehlen muss der `plaintext` Parameter base64-codiert sein, oder Sie müssen das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

```
aws kms encrypt \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --encryption-algorithm RSAES_OAEP_SHA_256 \
    --plaintext fileb://ExamplePlaintextFile \
    --output text \
    --query CiphertextBlob | base64 \
    --decode > ExampleEncryptedFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Weitere API-Informationen finden Sie unter Encrypt in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Encrypts the given text asynchronously using the specified KMS client and
 key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
 SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
 text) {
```

```
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
 getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
 algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
```

```
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is
not enabled.\n";
            }
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
```

```
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");
```

```
    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS -SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_plaintext contains the data to encrypt
        oo_result = lo_kms->encrypt(
          iv_keyid = iv_key_id
          iv_plaintext = iv_plaintext
        ).
        MESSAGE 'Text encrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie in der API-Referenz zum Verschlüsseln im AWS SDK für SAP ABAP.

Verwenden Sie mit **`GenerateDataKey`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateDataKey` verwendet wird.

CLI

AWS CLI

Beispiel 1: So generieren Sie einen symmetrischen 256-Bit-Datenschlüssel

Im folgenden `generate-data-key` Beispiel wird ein symmetrischer 256-Bit-Datenschlüssel zur Verwendung außerhalb von angefordert. AWS Der Befehl gibt einen Klartextdatenschlüssel zur sofortigen Verwendung und Löschung sowie eine Kopie dieses Datenschlüssels zurück, die unter dem angegebenen KMS-Schlüssel verschlüsselt ist. Sie können den verschlüsselten Datenschlüssel sicher neben den verschlüsselten Daten speichern.

Um einen 256-Bit-Datenschlüssel anzufordern, verwenden Sie den Parameter `key-spec` mit dem Wert `AES_256`. Um einen 128-Bit-Datenschlüssel anzufordern, verwenden Sie den Parameter `key-spec` mit dem Wert `AES_128`. Verwenden Sie für alle anderen Datenschlüssellängen den Parameter `number-of-bytes`.

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit KeySpec-Wert von SYMMETRIC_DEFAULT.

```
aws kms generate-data-key \
    --key-id alias/ExampleAlias \
    --key-spec AES_256
```

Ausgabe:

```
{
    "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
```

```
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
+YdhV8MrkBQPeac0ReRVNDt9qleAt+SHgIRF8P0H+7U="
}
```

Der `Plaintext` (Klartextdatenschlüssel) und der `CiphertextBlob` (verschlüsselte Datenschlüssel) werden im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im Entwicklerhandbuch für AWS Key Management Service. Beispiel 2: So generieren Sie einen symmetrischen 512-Bit-Datenschlüssel

Im folgenden Beispiel für `generate-data-key` wird ein symmetrischer 512-Bit-Datenschlüssel für Ver- und Entschlüsselung angefordert. Der Befehl gibt einen Klartextdatenschlüssel zur sofortigen Verwendung und Löschung sowie eine Kopie dieses Datenschlüssels zurück, die unter dem angegebenen KMS-Schlüssel verschlüsselt ist. Sie können den verschlüsselten Datenschlüssel sicher neben den verschlüsselten Daten speichern.

Verwenden Sie den Parameter `number-of-bytes`, um eine andere Schlüssellänge als 128 oder 256 Bit anzufordern. Um einen 512-Bit-Datenschlüssel anzufordern, verwendet das folgende Beispiel den Parameter `number-of-bytes` mit einem Wert von 64 (Byte).

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit KeySpec-Wert von SYMMETRIC_DEFAULT.

HINWEIS: Die Werte in der Ausgabe dieses Beispiels sind zur Anzeige gekürzt.

```
aws kms generate-data-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --number-of-bytes 64
```

Ausgabe:

```
{
    "CiphertextBlob": "AQIBAHi6LtupRpdKl2aJTzkK6FbhOtQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSIb3DQEHBqCCAlUwggJRAgEAMIICSgYJKoZ...",
```

```
    "Plaintext": "ty8Lr0Bk6OF07M2BWt6qbFdNB
+G00ZLtf5MSEb4al3R2UKWGOp06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm4OovEXiVfBveyX3DQxDzRSwbKDPk/...",
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

Der `Plaintext` (Klartextdatenschlüssel) und der `CiphertextBlob` (verschlüsselte Datenschlüssel) werden im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie unter Befehlsreferenz GenerateDataKey.AWS CLI

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side
 encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/
n)? "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)
```

- Einzelheiten zur API finden Sie GenerateDataKeyin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
```

```
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateDataKeyin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_keyspec = 'AES_256'
        oo_result = lo_kms->generatedatakey(
          iv_keyid = iv_key_id
          iv_keyspec = 'AES_256'
```

```
        ).
        MESSAGE 'Data key generated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GenerateDataKeyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GenerateDataKeyWithoutPlaintext`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateDataKeyWithoutPlaintext` verwendet wird.

CLI

AWS CLI

So generieren Sie einen symmetrischen 256-Bit-Datenschlüssel ohne Klartextschlüssel

Im folgenden Beispiel für `generate-data-key-without-plaintext` wird ein symmetrisches RSA-Datenschlüsselpaar mit 256 Bit zur Verwendung außerhalb von AWS angefordert. Sie können AWS KMS aufrufen, um den Datenschlüssel zu entschlüsseln, wenn Sie bereit sind, ihn zu verwenden.

Um einen 256-Bit-Datenschlüssel anzufordern, verwenden Sie den Parameter `key-spec` mit dem Wert `AES_256`. Um einen 128-Bit-Datenschlüssel anzufordern, verwenden Sie den Parameter `key-spec` mit dem Wert `AES_128`. Verwenden Sie für alle anderen Datenschlüssellängen den Parameter `number-of-bytes`.

Bei dem von Ihnen angegebenen KMS-Schlüssel muss es sich um einen KMS-Schlüssel mit symmetrischer Verschlüsselung handeln, d. h. um einen KMS-Schlüssel mit KeySpec-Wert von SYMMETRIC_DEFAULT.

```
aws kms generate-data-key-without-plaintext \
    --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
```

```
    --key-spec AES_256
```

Ausgabe:

```
{
    "CiphertextBlob":
 "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
    "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyMaterialId":
 "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

Der `CiphertextBlob` (verschlüsselte Datenschlüssel) wird im Base64-codierten Format zurückgegeben.

Weitere Informationen finden Sie unter Datenschlüssel im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie GenerateDataKeyWithoutPlaintextin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;
```

```
    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateDataKeyWithoutPlaintextin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`GenerateRandom`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GenerateRandom` verwendet wird.

CLI

AWS CLI

Beispiel 1: So generieren Sie eine 256-Bit-Zufallsbytezeichenfolge (Linux oder macOs)

Im folgenden Beispiel für `generate-random` wird eine Base64-kodierte 256-Bit-Zufallsbytezeichenfolge (32 Byte) generiert. Im Beispiel wird die Bytezeichenfolge dekodiert und in der Zufallsdatei gespeichert.

Wenn Sie diesen Befehl ausführen, müssen Sie den Parameter `number-of-bytes` verwenden, um die Länge des Zufallswerts in Byte anzugeben.

Sie geben keinen KMS-Schlüssel an, wenn Sie diesen Befehl ausführen. Die Zufallsbytezeichenfolge steht in keinem Zusammenhang mit einem KMS-Schlüssel.

Standardmäßig generiert AWS KMS die Zufallszahl. Wenn Sie jedoch einen benutzerdefinierten Schlüsselspeicher angeben, wird die zufällige Bytezeichenfolge in dem AWS CloudHSM-Cluster generiert, der dem benutzerdefinierten Schlüsselspeicher zugeordnet ist.

Im folgenden Beispiel werden die folgenden Parameter und Werte verwendet:

Es verwendet den erforderlichen `--number-of-bytes` Parameter mit dem Wert von, `32` um eine 32-Byte-Zeichenfolge (256-Bit) anzufordern.Es verwendet den `--output` Parameter mit dem Wert von, `text` um die AWS CLI anzuweisen, die Ausgabe als Text zurückzugeben, anstatt den Wert der `Plaintext` Eigenschaft aus der Antwort `--query parameter` zu extrahieren.Es leitet (|) die Ausgabe des Befehls an das `base64` Hilfsprogramm weiter, das die extrahierte Ausgabe dekodiert. Es JSON.It verwendet den Umleitungsoperator (**>**), um die dekodierte Bytezeichenfolge in der Datei zu speichern.IT verwendet den Umleitungsoperator (**>** `ExampleRandom`), um den binären Chiffretext in einer Datei zu speichern.

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext | base64 --decode > ExampleRandom
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie GenerateRandomin der API-Referenz für den AWS Key Management Service.

Beispiel 2: So generieren Sie eine 256-Bit-Zufallszahl zu generieren (Windows-Befehlszeile)

Im folgenden Beispiel wird der Befehl `generate-random` verwendet, um eine Base64-kodierte 256-Bit-Zufallsbytezeichenfolge (32 Byte) zu generieren. Im Beispiel wird die Bytezeichenfolge dekodiert und in der Zufallsdatei gespeichert. Dieses Beispiel ist mit dem vorherigen Beispiel identisch, mit dem Unterschied, dass es das Dienstprogramm `certutil` in Windows verwendet, um die zufällige Bytefolge vor dem Speichern in einer Datei mit Base64 zu dekodieren.

Generieren Sie zunächst eine Base64-kodierte Zufallsbytezeichenfolge und speichern Sie sie in einer temporären Datei, `ExampleRandom.base64`.

```
aws kms generate-random \
    --number-of-bytes 32 \
    --output text \
    --query Plaintext > ExampleRandom.base64
```

Da die Ausgabe des Befehls `generate-random` in einer Datei gespeichert wird, erzeugt dieses Beispiel keine Ausgabe.

Verwenden Sie nun den Befehl `certutil -decode`, um die Base64-kodierte Bytezeichenfolge in der Datei `ExampleRandom.base64` zu dekodieren. Anschließend wird die dekodierte Bytezeichenfolge in der Datei `ExampleRandom` gespeichert.

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie GenerateRandomin der API-Referenz für den AWS Key Management Service.

- Einzelheiten zur API finden Sie GenerateRandomunter AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
```

```
    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- Einzelheiten zur API finden Sie GenerateRandomin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`GetKeyPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetKeyPolicy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So kopieren Sie eine Schlüsselrichtlinie von einem KMS-Schlüssel in einen anderen

Im folgenden Beispiel für `get-key-policy` wird die Schlüsselrichtlinie von einem KMS-Schlüssel abgerufen und in einer Textdatei gespeichert. Anschließend wird die Richtlinie eines anderen KMS-Schlüssels ersetzt, wobei die Textdatei als Richtlinieneingabe verwendet wird.

Da für den Parameter `--policy` von `put-key-policy` eine Zeichenfolge erforderlich ist, müssen Sie die Option `--output text` verwenden, um die Ausgabe als Textzeichenfolge statt als JSON zurückzugeben.

```
aws kms get-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --query Policy \
    --output text > policy.txt

aws kms put-key-policy \
```

```
    --policy-name default \
    --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --policy file://policy.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie PutKeyPolicyin der AWS KMS-API-Referenz.

- Einzelheiten zur API finden Sie GetKeyPolicyunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
```

```
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Einzelheiten zur API finden Sie GetKeyPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->getkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
```

```
        ).
        MESSAGE 'Retrieved key policy successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetKeyPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAliases`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAliases` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) aliases that have been
defined for
    /// the keys in the same AWS Region as the default user. If you want to list
```

```
    /// the aliases in a different Region, pass the Region to the client
    /// constructor.
    /// </summary>
    public class ListAliases
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListAliasesRequest();
            var response = new ListAliasesResponse();

            do
            {
                response = await client.ListAliasesAsync(request);

                response.Aliases.ForEach(alias =>
                {
                    Console.WriteLine($"Created: {alias.CreationDate} Last
 Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Um alle Aliase in einem AWS Konto und einer Region aufzulisten

Im folgenden Beispiel wird der `list-aliases` Befehl verwendet, um alle Aliase in der Standardregion des AWS Kontos aufzulisten. Die Ausgabe umfasst Aliase, die AWS verwalteten KMS-Schlüsseln und kundenverwalteten KMS-Schlüsseln zugeordnet sind.

```
aws kms list-aliases
```

Ausgabe:

```
{
    "Aliases": [
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
            "AliasName": "alias/testKey",
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
            "AliasName": "alias/FinanceDept",
            "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
            "AliasName": "alias/aws/dynamodb",
            "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        },
        {
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
            "AliasName": "alias/aws/ebs",
            "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
        },
        ...
    ]
}
```

Beispiel 2: So listen Sie alle Aliasnamen für einen bestimmten KMS-Schlüssel auf

Im folgenden Beispiel werden der Befehl `list-aliases` und sein Parameter `key-id` verwendet, um alle Aliasname aufzulisten, die einem bestimmten KMS-Schlüssel zugeordnet sind.

Jeder Alias ist nur mit einem KMS-Schlüssel verbunden, ein KMS-Schlüssel kann jedoch über mehrere Aliasse verfügen. Dieser Befehl ist sehr nützlich, da die AWS KMS-Konsole nur einen Alias für jeden KMS-Schlüssel auflistet. Um alle Aliasnamen für einen KMS-Schlüssel zu suchen, müssen Sie den Befehl `list-aliases` verwenden.

In diesem Beispiel wird die Schlüssel-ID des KMS-Schlüssels für den Parameter `--key-id` verwendet. Sie können in diesem Befehl jedoch eine Schlüssel-ID, einen Schlüssel-ARN, einen Aliasnamen oder einen Alias-ARN verwenden.

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "Aliases": [
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-
key",
            "AliasName": "alias/oregon-test-key"
        },
        {
            "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
            "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-
test",
            "AliasName": "alias/project121-test"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Aliasnamen im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie ListAliasesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of
 aliases has been processed
```

```
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
 getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
 e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
 ex.getMessage());
                }
            });
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
```

```
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListAliasesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
```

```
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
 where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListAliasesin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
```

```
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListAliasesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kms->listaliases( ).
        MESSAGE 'Retrieved KMS aliases list.' TYPE 'I'.
```

```
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListAliasesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListGrants`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListGrants` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) grants that are associated
with
    /// a specific key.
    /// </summary>
    public class ListGrants
    {
        public static async Task Main()
```

```
        {
            // The identifier of the AWS KMS key to disable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListGrantsRequest
            {
                KeyId = keyId,
            };

            var response = new ListGrantsResponse();

            do
            {
                response = await client.ListGrantsAsync(request);

                response.Grants.ForEach(grant =>
                {
                    Console.WriteLine($"{grant.GrantId}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So zeigen Sie die Grants für einen AWS KMS-Schlüssel an

Im folgenden `list-grants` Beispiel werden alle Grants für den angegebenen AWS verwalteten KMS-Schlüssel für Amazon DynamoDB in Ihrem Konto angezeigt. Diese Erteilung ermöglicht es DynamoDB, den KMS-Schlüssel in Ihrem Namen zu verwenden, um eine DynamoDB-Tabelle zu verschlüsseln, bevor sie auf die Festplatte geschrieben wird. Sie können einen Befehl wie diesen verwenden, um die Zuschüsse für die AWS verwalteten KMS-

Schlüssel und die vom Kunden verwalteten KMS-Schlüssel für das AWS Konto und die Region anzuzeigen.

Dieser Befehl verwendet den Parameter `key-id` mit einer Schlüssel-ID, um den KMS-Schlüssel zu identifizieren. Sie können den KMS-Schlüssel mit einer Schlüssel-ID oder einem Schlüssel-ARN identifizieren. Verwenden Sie den `list-aliases` Befehl or, um die Schlüssel-ID oder den Schlüssel-ARN eines AWS verwalteten KMS-Schlüssels abzurufen. `list-keys`

```
aws kms list-grants \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Die Ausgabe zeigt, dass Amazon DynamoDB durch die Erteilung dazu berechtigt ist, den KMS-Schlüssel für kryptografische Vorgänge zu verwenden und Details zum KMS-Schlüssel (`DescribeKey`) anzuzeigen und Erteilungen (`RetireGrant`) zurückzuziehen. Die Einschränkung `EncryptionContextSubset` beschränkt diese Berechtigung auf Anforderungen, die die angegebenen Verschlüsselungskontextpaare enthalten. Daher sind die Berechtigungen in der Erteilung nur für das angegebene Konto und die angegebene DynamoDB-Tabelle wirksam.

```
{
    "Grants": [
        {
            "Constraints": {
                "EncryptionContextSubset": {
                    "aws:dynamodb:subscriberId": "123456789012",
                    "aws:dynamodb:tableName": "Services"
                }
            },
            "IssuingAccount": "arn:aws:iam::123456789012:root",
            "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
            "Operations": [
                "Decrypt",
                "Encrypt",
                "GenerateDataKey",
                "ReEncryptFrom",
                "ReEncryptTo",
                "RetireGrant",
                "DescribeKey"
            ],
```

```
            "GrantId":
 "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
            "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
            "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
            "CreationDate": "2021-05-13T18:32:45.144000+00:00"
        }
    ]
}
```

Weitere Informationen finden Sie unter Grants in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ListGrantsunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null
 if the operation succeeded, or will throw a {@link RuntimeException} if the
 operation failed
     * @throws RuntimeException if there was an error listing the grants, either
 due to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
 getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
 cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
 cause.getMessage(), cause);
                }
            });
    }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
```

```
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListGrantsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity
 or resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListGrantsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])
```

```
            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListGrantsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listgrants( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved grants list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListGrantsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListKeyPolicies`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListKeyPolicies` verwendet wird.

CLI

AWS CLI

So rufen Sie die Namen von Schlüsselrichtlinien für einen KMS-Schlüssel ab

Im folgenden Beispiel für `list-key-policies` werden die Namen der Schlüsselrichtlinien für einen kundenseitig verwalteten Schlüssel im Beispielkonto und in der Beispielregion abgerufen. Mit diesem Befehl können Sie die Namen der wichtigsten Richtlinien für AWS verwaltete Schlüssel und vom Kunden verwaltete Schlüssel ermitteln.

Da der einzige gültige Name der Schlüsselrichtlinie `default` lautet, ist dieser Befehl nicht hilfreich.

Verwenden Sie den Parameter `key-id`, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, Sie können jedoch eine Schlüssel-ID oder einen Schlüssel-ARN in diesem Befehl verwenden.

```
aws kms list-key-policies \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Ausgabe:

```
{
    "PolicyNames": [
    "default"
    ]
}
```

Weitere Informationen zu AWS KMS-Schlüsselrichtlinien finden Sie unter Verwenden von Schlüsselrichtlinien in AWS KMS im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie ListKeyPoliciesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously retrieves the key policy for the specified key ID and
policy name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the
policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
   public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
       GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .build();

       return getAsyncClient().getKeyPolicy(policyRequest)
           .thenApply(response -> {
               String policy = response.policy();
               logger.info("The response is: " + policy);
               return policy;
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Failed to get key policy", ex);
           });
   }
```

- Einzelheiten zur API finden Sie ListKeyPoliciesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- Einzelheiten zur API finden Sie ListKeyPoliciesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listkeypolicies( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved key policies list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListKeyPoliciesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListKeys`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListKeys` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
/// of the default user. To list keys in another AWS Region, supply the
Region
/// as a parameter to the client constructor.
/// </summary>
public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();

        do
        {
            response = await client.ListKeysAsync(request);

            response.Keys.ForEach(key =>
            {
                Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
            });

            // Set the Marker property when response.Truncated is true
            // in order to get the next keys.
```

```
                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die KMS-Schlüssel für ein Konto und eine Region ab

Im folgenden Beispiel für `list-keys` werden die KMS-Schlüssel für ein Konto und eine Region abgerufen. Dieser Befehl gibt sowohl AWS verwaltete Schlüssel als auch vom Kunden verwaltete Schlüssel zurück.

```
aws kms list-keys
```

Ausgabe:

```
{
    "Keys": [
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "KeyArn": "arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
            "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
        },
        {
            "KeyArn": "arn:aws:kms:us-
east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
            "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
        }
```

```
    ]
}
```

Weitere Informationen finden Sie unter Anzeigen von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie ListKeysin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
```

```
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
 Id is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
 exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
```

```
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ListKeysin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- Einzelheiten zur API finden Sie ListKeysin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kms->listkeys( ).
        MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListKeysin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutKeyPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutKeyPolicy` verwendet wird.

CLI

AWS CLI

So ändern Sie die Schlüsselrichtlinie für einen KMS-Schlüssel

Im folgenden Beispiel für `put-key-policy` wird die Schlüsselrichtlinie für einen kundenseitig verwalteten Schlüssel geändert.

Erstellen Sie zunächst eine Schlüsselrichtlinie und speichern Sie sie in einer lokalen JSON-Datei. In diesem Beispiel handelt es sich um die Datei `key_policy.json`. Sie können die Schlüsselrichtlinie auch als Zeichenfolgenwert des Parameters `policy` angeben.

Die erste Anweisung in dieser Schlüsselrichtlinie erteilt dem AWS Konto die Erlaubnis, IAM-Richtlinien zur Steuerung des Zugriffs auf den KMS-Schlüssel zu verwenden. Die zweite Anweisung erteilt dem Benutzer `test-user` die Berechtigung, die Befehle `describe-key` und `list-keys` auf dem KMS-Schlüssel auszuführen.

Inhalt von `key_policy.json`:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
```

```
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
        },
        {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [
                "kms:DescribeKey",
                "kms:ListKeys"
            ],
            "Resource" : "*"
        }
    ]
}
```

In diesem Beispiel wird der KMS-Schlüssel mit der Schlüssel-ID identifiziert, Sie können jedoch auch einen Schlüssel-ARN verwenden. Der Befehl verwendet den Parameter `policy`, um die Schlüsselrichtlinie festzulegen. Um anzugeben, dass sich die Richtlinie in einer Datei befindet, wird das erforderliche Präfix `file://` verwendet. Dieses Präfix ist erforderlich, um Dateien auf allen unterstützten Betriebssystemen zu identifizieren. Schließlich verwendet der Befehl den Parameter `policy-name` mit dem Wert `default`. Wenn kein Richtlinienwert angegeben wird, lautet der Standardwert `default`. Der einzige gültige Wert ist `default`.

```
aws kms put-key-policy \
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --policy file://key_policy.json
```

Dieser Befehl erzeugt keine Ausgabe. Verwenden Sie den Befehl `get-key-policy`, um zu überprüfen, ob der Befehl wirksam war. Mit dem folgenden Beispielbefehl wird die Schlüsselrichtlinie für denselben KMS-Schlüssel abgerufen. Der Parameter `output` mit dem Wert `text` gibt ein leicht lesbares Textformat zurück.

```
aws kms get-key-policy \
```

```
    --policy-name default \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --output text
```

Ausgabe:

```
{
    "Version":"2012-10-17",
    "Id" : "key-default-1",
    "Statement" : [
        {
            "Sid" : "Enable IAM User Permissions",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:root"
            },
            "Action" : "kms:*",
            "Resource" : "*"
            },
            {
            "Sid" : "Allow Use of Key",
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "arn:aws:iam::111122223333:user/test-user"
            },
            "Action" : [ "kms:Describe", "kms:List" ],
            "Resource" : "*"
        }
    ]
}
```

Weitere Informationen finden Sie unter Ändern einer Schlüsselrichtlinie im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie PutKeyPolicyin der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie PutKeyPolicyin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
 existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
            "Enter the ARN of an IAM role to set as the principal on the policy:
 "
        )
```

```
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new
statement to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
            )
            try:
                self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- Einzelheiten zur API finden Sie PutKeyPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_policy = '{"Version":"2012-10-17",        "Statement": [...]}'
        lo_kms->putkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
          iv_policy = iv_policy
        ).
        MESSAGE 'Key policy updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsmalformedplydocex.
        MESSAGE 'Malformed policy document.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutKeyPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ReEncrypt`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ReEncrypt` verwendet wird.

CLI

AWS CLI

Beispiel 1: So verschlüsseln Sie eine verschlüsselte Nachricht unter einem symmetrischen KMS-Schlüssel (Linux und macOS) erneut).

Das folgende `re-encrypt` Befehlsbeispiel zeigt die empfohlene Methode zum erneuten Verschlüsseln von Daten mit der AWS CLI.

Geben Sie den Geheimtext in einer Datei an. Verwenden Sie im Wert des `--ciphertext-blob`-Parameters das `fileb://`-Präfix, das die CLI anweist, die Daten aus einer Binärdatei zu lesen. Wenn sich die Datei nicht im aktuellen Verzeichnis befindet, geben Sie den vollständigen Dateipfad ein. Weitere Informationen zum Lesen von AWS CLI-Parameterwerten aus einer Datei finden Sie unter Laden von AWS CLI-Parametern aus einer Datei im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle und Best Practices für lokale Dateiparameter im AWS Command Line Tool-Blog. Geben Sie den KMS-Quellschlüssel an, der den Chiffretext entschlüsselt. Der `--source-key-id` Parameter ist bei der Entschlüsselung mit symmetrischer Verschlüsselung von KMS-Schlüsseln nicht erforderlich. AWS KMS kann den KMS-Schlüssel, der zur Verschlüsselung der Daten verwendet wurde, aus den Metadaten im Chiffretext-Blob abrufen. Es ist jedoch immer eine bewährte Methode, den von Ihnen verwendeten KMS-Schlüssel anzugeben. Diese Methode stellt sicher, dass Sie den beabsichtigten KMS-Schlüssel verwenden, und verhindert, dass Sie versehentlich einen Geheimtext mit einem KMS-Schlüssel entschlüsseln, dem Sie nicht vertrauen. Geben Sie den KMS-Schlüssel an, der die Daten erneut verschlüsselt. Der Parameter `--destination-key-id` ist immer erforderlich. In diesem Beispiel wird ein Schlüssel-ARN verwendet, Sie können jedoch jede gültige Schlüsselkennung verwenden. Fordern Sie die Klartextausgabe als Textwert an. Der Parameter `--query` weist die CLI an, nur den Wert des Feldes `Plaintext` aus der Ausgabe abzurufen. Der `--output` Parameter gibt die Ausgabe als Text zurück. Base64-decode den Klartext und speichern Sie ihn in einer Datei. Im folgenden Beispiel wird der Wert des `Plaintext` Parameters über die Pipeline (|) an das Base64-Hilfsprogramm übergeben, das ihn dekodiert. Anschließend erfolgt eine Umleitung (>) der dekodierten Ausgabe in die Datei `ExamplePlaintext`.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-IDs durch gültige Schlüsselkennungen aus Ihrem Konto. AWS

```
aws kms re-encrypt \
    --ciphertext-blob fileb://ExampleEncryptedFile \
```

```
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
    --query CiphertextBlob \
    --output text | base64 --decode > ExampleReEncryptedFile
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Die Ausgabe des Befehls `re-encrypt` wird Base64-dekodiert und in einer Datei gespeichert.

Weitere Informationen finden Sie ReEncryptin der API-Referenz für den AWS Key Management Service.

Beispiel 2: So verschlüsseln Sie eine verschlüsselte Nachricht unter einem symmetrischen KMS-Schlüssel (Windows-Befehlszeile) erneut).

Das folgende `re-encrypt` Befehlsbeispiel ist dasselbe wie das vorherige, außer dass es das `certutil` Hilfsprogramm für Base64-decode Klartextdaten verwendet. Für dieses Verfahren sind zwei Befehle erforderlich, wie in den folgenden Beispielen gezeigt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms re-encrypt ^
    --ciphertext-blob fileb://ExampleEncryptedFile ^
    --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
    --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^
    --query CiphertextBlob ^
    --output text > ExampleReEncryptedFile.base64
```

Verwenden Sie dann das Dienstprogramm `certutil`.

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

Ausgabe:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

Weitere Informationen finden Sie ReEncryptin der API-Referenz für den AWS Key Management Service.

- Einzelheiten zur API finden Sie ReEncryptunter AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by
 using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt
 the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
```

```
            f"Enter another key ID or ARN to reencrypt it: "
        )
        if destination_key_id != "":
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- Einzelheiten zur API finden Sie ReEncryptin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593
sourceCiphertextBlob = [blob].pack('H*')
```

```
# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Einzelheiten zur API finden Sie ReEncryptin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
```

```
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ReEncryptin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_source_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_destination_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->reencrypt(
          iv_sourcekeyid = iv_source_key_id
          iv_destinationkeyid = iv_destination_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Ciphertext reencrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect source key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ReEncryptin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RetireGrant`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RetireGrant` verwendet wird.

CLI

AWS CLI

So ziehen Sie eine Erteilung für einen Kundenhauptschlüssel zurück

Im folgenden Beispiel für `retire-grant` wird ein Zuschuss aus einem KMS-Schlüssel gelöscht.

Der folgende Beispielbefehl legt die Parameter `grant-id` und `key-id` fest. Der Wert des Parameters `key-id` muss der Schlüssel-ARN des KMS-Schlüssels sein.

```
aws kms retire-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `list-grants`, um zu bestätigen, dass die Erteilung zurückgezogen wurde.

Weitere Informationen finden Sie unter Zurückziehen und Widerrufen von Erteilungen im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie RetireGrantin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
```

```
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- Einzelheiten zur API finden Sie RetireGrantin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_grant_token = 'AQpAM2RhZ...'
        lo_kms->retiregrant( iv_granttoken = iv_grant_token ).
        MESSAGE 'Grant retired successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvgranttokenex.
        MESSAGE 'Invalid grant token.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RetireGrantin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RevokeGrant`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RevokeGrant` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So ziehen Sie eine Erteilung für einen Kundenhauptschlüssel zurück

Im folgenden Beispiel für `revoke-grant` wird ein Zuschuss aus einem KMS-Schlüssel gelöscht. Der folgende Beispielbefehl legt die Parameter `grant-id` und `key-id` fest. Der Wert des Parameters `key-id` kann die Schlüssel-ID oder der Schlüssel-ARN des KMS-Schlüssels sein.

```
aws kms revoke-grant \
    --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `list-grants`, um zu bestätigen, dass die Erteilung zurückgezogen wurde.

Weitere Informationen finden Sie unter Zurückziehen und Widerrufen von Erteilungen im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie RevokeGrantin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
                   } else {
```

```
                        throw new RuntimeException("KMS error occurred: " +
 kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
 exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- Einzelheiten zur API finden Sie RevokeGrantin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
```

```
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie RevokeGrantin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
```

```
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie RevokeGrantin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grant_id = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
        lo_kms->revokegrant(
          iv_keyid = iv_key_id
          iv_grantid = iv_grant_id
        ).
        MESSAGE 'Grant revoked successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidgrantidex.
        MESSAGE 'Invalid grant ID.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie RevokeGrantin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ScheduleKeyDeletion`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ScheduleKeyDeletion` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So planen Sie das Löschen eines kundenseitig verwalteten KMS-Schlüssels.

Im folgenden Beispiel für `schedule-key-deletion` wird geplant, dass der angegebene kundenseitig verwaltete KMS-Schlüssel innerhalb von 15 Tagen gelöscht wird.

Der Parameter `--key-id` identifiziert den KMS-Schlüssel. In diesem Beispiel wird ein Schlüssel-ARN-Wert verwendet, Sie können jedoch entweder die Schlüssel-ID oder den ARN des KMS-Schlüssels verwenden. Der Parameter `--pending-window-in-days` gibt die Dauer der Wartezeit von 7 bis 30 Tagen an. Die Wartezeit beträgt standardmäßig 30 Tage. In diesem Beispiel wird der Wert 15 angegeben, der angibt, dass der KMS-Schlüssel 15 Tage nach Abschluss des Befehls dauerhaft gelöscht werden AWS soll.

```
aws kms schedule-key-deletion \
    --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --pending-window-in-days 15
```

Die Antwort enthält den Schlüssel-ARN, den Schlüsselstatus, die Wartezeit (`PendingWindowInDays`) und das Löschdatum in Unix-Zeit. Verwenden Sie die AWS KMS-Konsole, um das Löschdatum in Ortszeit anzuzeigen. KMS-Schlüssel im Schlüsselstatus `PendingDeletion` können nicht in kryptografischen Vorgängen verwendet werden.

```
{
```

```
    "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
    "KeyState": "PendingDeletion",
    "PendingWindowInDays": 15
}
```

Weitere Informationen finden Sie unter Löschen von Schlüsseln im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
```

```
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
key ID: " + keyId, throwable);
            });
    }
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
```

```
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_key(self, key_id: str, window: int) -> None:
        """
```

```
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_pending_window_days = 7
        oo_result = lo_kms->schedulekeydeletion(
          iv_keyid = iv_key_id
          iv_pendingwindowindays = iv_pending_window_days
        ).
        MESSAGE 'Key scheduled for deletion.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ScheduleKeyDeletionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie „**Signieren**“ mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Sign` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So generieren Sie eine digitale Signatur für eine Nachricht

Im folgenden Beispiel für `sign` wird eine kryptografische Signatur für eine Kurznachricht generiert. Die Ausgabe des Befehls enthält das mit base-64 codiertes Feld `Signature`, das Sie mit dem Befehl `verify` überprüfen können.

Sie müssen eine zu signierende Nachricht und einen Signierungsalgorithmus angeben, den Ihr asymmetrischer KMS-Schlüssel unterstützt. Verwenden Sie den Befehl `describe-key`, um den Signierungsalgorithmus für Ihren KMS-Schlüssel abzurufen.

In AWS CLI v2 muss der Wert des `message` Parameters sein Base64-encoded. Oder Sie können die Nachricht in einer Datei speichern und das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto. Die Schlüssel-ID muss einen asymmetrischen KMS-Schlüssel mit der Schlüsselverwendung SIGN_VERIFY darstellen.

```
msg=(echo 'Hello World' | base64)

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://UnsignedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

Ausgabe:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "Signature": "ABCDEFhpyVYyTxbafE74ccSvEJLJr3zuoV1Hfymz4qv+/
fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

Beispiel 2: So speichern Sie eine digitale Signatur in einer Datei (Linux und macOs)

Im folgenden Beispiel für `sign` wird eine kryptografische Signatur für eine Kurznachricht generiert, die in einer lokalen Datei gespeichert ist. Der Befehl ruft auch die `Signature` Eigenschaft aus der Antwort ab und speichert Base64-decodes sie in der ExampleSignature Datei. Sie können die Signaturdatei in einem `verify`-Befehl verwenden, der die Signatur überprüft.

Für den `sign` Befehl sind eine Base64-encoded Nachricht und ein Signaturalgorithmus erforderlich, die Ihr asymmetrischer KMS-Schlüssel unterstützt. Verwenden Sie den Befehl `describe-key`, um den Signaturalgorithmus abzurufen, den Ihr KMS-Schlüssel unterstützt.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto. Die Schlüssel-ID muss einen asymmetrischen KMS-Schlüssel mit der Schlüsselverwendung SIGN_VERIFY darstellen.

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --output text \
    --query Signature | base64 --decode > ExampleSignature
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. In diesem Beispiel wird die Eigenschaft `Signature` der Ausgabe extrahiert und in einer Datei gespeichert.

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Asymmetrische Schlüssel in AWS KMS im AWS Key Management Service Developer Guide.

- Weitere API-Informationen finden Sie unter Sign in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
```

```
    *     <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
    *     <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
    * </ol>
    *
    * @return a {@link CompletableFuture} that completes with the result of the
signature verification,
    *         {@code true} if the signature is valid, {@code false} otherwise.
    * @throws KmsException if any error occurs during the KMS operations.
    * @throws RuntimeException if an unexpected error occurs.
    */
   public CompletableFuture<Boolean> signVerifyDataAsync() {
       String signMessage = "Here is the message that will be digitally signed";

       // Create an AWS KMS key used to digitally sign data.
       CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
           .keySpec(KeySpec.RSA_2048)
           .keyUsage(KeyUsageType.SIGN_VERIFY)
           .origin(OriginType.AWS_KMS)
           .build();

       return getAsyncClient().createKey(createKeyRequest)
           .thenCompose(createKeyResponse -> {
               String keyId = createKeyResponse.keyMetadata().keyId();

               SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
               SignRequest signRequest = SignRequest.builder()
                   .keyId(keyId)
                   .message(messageBytes)
                   .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                   .build();

               return getAsyncClient().sign(signRequest)
                   .thenCompose(signResponse -> {
                       byte[] signedBytes =
signResponse.signature().asByteArray();

                       VerifyRequest verifyRequest = VerifyRequest.builder()
                           .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
```

```
.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }
```

- Weitere API-Informationen finden Sie unter Sign in der AWS SDK for Java 2.x -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/***
 * @param string $keyId
 * @param string $message
 * @param string $algorithm
 * @return Result
 */
public function sign(string $keyId, string $message, string $algorithm)
{
```

```
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Sign in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter Sign in der API-Referenz zum AWS -SDK für Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the message to sign
```

```
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->sign(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signingalgorithm = iv_signing_algorithm
        ).
        MESSAGE 'Message signed successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidkeyusageex.
        MESSAGE 'Key cannot be used for signing.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter Referenz zum AWS AnmeldesDK für SAP ABAP API.

Verwenden Sie **`TagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So fügen Sie einem KMS-Schlüssel ein Tag hinzu

Im folgenden Beispiel für `tag-resource` werden einem kundenseitig verwalteten KMS-Schlüssel die Tags `"Purpose":"Test"` und `"Dept":"IT"` hinzugefügt. Sie können Tags wie diese verwenden, um KMS-Schlüssel zu kennzeichnen und Kategorien von KMS-Schlüsseln für Berechtigungen und Prüfungen zu erstellen.

Verwenden Sie den Parameter `key-id`, um den KMS-Schlüssel anzugeben. In diesem Beispiel wird ein Schlüssel-ID-Wert verwendet, Sie können jedoch eine Schlüssel-ID oder einen Schlüssel-ARN in diesem Befehl verwenden.

```
aws kms tag-resource \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
    --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den `list-resource-tags` Befehl, um die Tags auf einem AWS KMS-KMS-Schlüssel anzuzeigen.

Weitere Informationen zur Verwendung von Tags in AWS KMS finden Sie unter Tagging Keys im AWS Key Management Service Developer Guide.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging
 operation is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
```

```
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key",
 throwable);
            });
    }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
```

```
            throw $caught;
        }
    }
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
```

```
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie TagResourcein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_tags TYPE /aws1/cl_kmstag=>tt_taglist.

    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_tag_key = 'Environment'
        " iv_tag_value = 'Production'
        APPEND NEW /aws1/cl_kmstag(
          iv_tagkey = iv_tag_key
          iv_tagvalue = iv_tag_value
        ) TO lt_tags.

        lo_kms->tagresource(
```

```
          iv_keyid = iv_key_id
          it_tags = lt_tags
        ).
        MESSAGE 'Tag added to KMS key successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmstagexception.
        MESSAGE 'Invalid tag format.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateAlias`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAlias` verwendet wird.

CLI

AWS CLI

So verknüpfen Sie einen Alias mit einem anderen KMS-Schlüssel

Im folgenden Beispiel für `update-alias` wird der Alias `alias/test-key` einem anderen KMS-Schlüssel zugeordnet.

Der Parameter `--alias-name` gibt den Alias an. Der Wert des Aliasnamens muss mit `alias/` beginnen. Der Parameter `--target-key-id` gibt den KMS-Schlüssel an, der dem Alias zugeordnet werden soll. Sie müssen den aktuellen KMS-Schlüssel für den Alias nicht angeben.

```
aws kms update-alias \
    --alias-name alias/test-key \
    --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben. Verwenden Sie den Befehl `list-aliases`, um den Alias zu suchen.

Weitere Informationen finden Sie unter Aktualisieren von Aliasnamen im Entwicklerhandbuch für AWS Key Management Service.

- Einzelheiten zur API finden Sie UpdateAliasin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
 the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
```

```
            f"Enter another key ID or ARN that you want to associate with
 {alias}: "
        )
        if new_key_id != "":
            try:
                self.kms_client.update_alias(AliasName=alias,
 TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- Einzelheiten zur API finden Sie UpdateAliasin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_target_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        lo_kms->updatealias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_target_key_id
        ).
        MESSAGE 'Alias updated successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Alias or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateAliasin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`Verify`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Verify` verwendet wird.

CLI

AWS CLI

So überprüfen Sie eine digitale Signatur

Der folgende `verify` Befehl überprüft eine kryptografische Signatur für eine kurze Nachricht. Base64-encoded Die Schlüssel-ID, die Nachricht, der Nachrichtentyp und der Signaturalgorithmus müssen mit denen identisch sein, die zum Signieren der Nachricht verwendet wurden.

In AWS CLI v2 muss der Wert des `message` Parameters sein Base64-encoded. Oder Sie können die Nachricht in einer Datei speichern und das `fileb://` Präfix verwenden, das die AWS CLI anweist, Binärdaten aus der Datei zu lesen.

Die von Ihnen angegebene Signatur kann nicht Base64-codiert sein. Hilfe zum Entschlüsseln der Signatur, die der Befehl `sign` zurückgibt, finden Sie in den Beispielen für den Befehl `sign`.

Die Ausgabe des Befehls enthält ein boolesches `SignatureValid`-Feld, das angibt, dass die Signatur verifiziert wurde. Wenn die Signaturüberprüfung fehlschlägt, schlägt auch der Befehl `verify` fehl.

Bevor Sie diesen Befehl ausführen, ersetzen Sie die Beispielschlüssel-ID durch eine gültige Schlüssel-ID aus Ihrem AWS Konto.

```
aws kms verify \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
```

```
    --message fileb://EncodedMessage \
    --message-type RAW \
    --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
    --signature fileb://ExampleSignature
```

Ausgabe:

```
{
    "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "SignatureValid": true,
    "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

Weitere Informationen zur Verwendung asymmetrischer KMS-Schlüssel in AWS KMS finden Sie unter Using asymmetric keys im AWS Key Management Service Developer Guide.

- Weitere API-Informationen finden Sie unter Verify in der AWS CLI -Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Weitere Informationen finden Sie unter. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
 client.
        """
```

```
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie unter Verify in der API-Referenz zum AWS -SDK für Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the original message
        " iv_signature contains the signature to verify
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->verify(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signature = iv_signature
          iv_signingalgorithm = iv_signing_algorithm
        ).
        DATA(lv_valid) = oo_result->get_signaturevalid( ).
        IF lv_valid = abap_true.
          MESSAGE 'Signature is valid.' TYPE 'I'.
        ELSE.
          MESSAGE 'Signature is invalid.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinvalidsigex.
        MESSAGE 'Invalid signature.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter Verify in AWS SDK for SAP ABAP API-Referenz.

Szenarien für AWS KMS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS KMS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS KMS oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

Arbeiten Sie mit der DynamoDB-Tabellenverschlüsselung unter AWS Command Line Interface v2

Im folgenden Codebeispiel wird gezeigt, wie Sie Verschlüsselungsoptionen für DynamoDB-Tabellen verwalten.

- Erstellen Sie eine Tabelle mit Standardverschlüsselung.
- Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.
- Aktualisieren Sie die Einstellungen für die Tabellenverschlüsselung.
- Beschreiben Sie die Tabellenverschlüsselung.

Bash

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Standardverschlüsselung.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
```

```
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Aktualisieren Sie die Tabellenverschlüsselung.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Beschreiben Sie die Tabellenverschlüsselung.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Codebeispiele für Lambda mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Lambda mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden AWS. Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Weitere -Quellen

- Lambda-Entwicklerhandbuch – Weitere Informationen zu Lambda.
- Lambda-API-Referenz – Details zu allen verfügbaren Lambda-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Lambda AWS SDKs
  - Hallo Lambda

- Lernen Sie die Grundlagen von Lambda mit einem AWS SDK
- Aktionen für Lambda mit AWS SDKs
  - Verwendung von CreateAlias mit einer CLI
  - Verwenden Sie mit einem CreateFunction AWS SDK oder CLI
  - Verwendung von DeleteAlias mit einer CLI
  - Verwenden Sie mit einem DeleteFunction AWS SDK oder CLI
  - Verwendung von DeleteFunctionConcurrency mit einer CLI
  - Verwendung von DeleteProvisionedConcurrencyConfig mit einer CLI
  - Verwendung von GetAccountSettings mit einer CLI
  - Verwendung von GetAlias mit einer CLI
  - Verwenden Sie mit einem GetFunction AWS SDK oder CLI
  - Verwendung von GetFunctionConcurrency mit einer CLI
  - Verwendung von GetFunctionConfiguration mit einer CLI
  - Verwendung von GetPolicy mit einer CLI
  - Verwendung von GetProvisionedConcurrencyConfig mit einer CLI
  - Verwenden Sie Invoke mit einem AWS SDK oder CLI
  - Verwenden Sie ListFunctionsmit einem AWS SDK oder CLI
  - Verwendung von ListProvisionedConcurrencyConfigs mit einer CLI
  - Verwendung von ListTags mit einer CLI
  - Verwendung von ListVersionsByFunction mit einer CLI
  - Verwendung von PublishVersion mit einer CLI
  - Verwendung von PutFunctionConcurrency mit einer CLI
  - Verwendung von PutProvisionedConcurrencyConfig mit einer CLI
  - Verwendung von RemovePermission mit einer CLI
  - Verwendung von TagResource mit einer CLI
  - Verwendung von UntagResource mit einer CLI
  - Verwendung von UpdateAlias mit einer CLI
  - Verwenden Sie mit einem UpdateFunctionCode AWS SDK oder CLI
  - Verwenden Sie UpdateFunctionConfigurationmit einem AWS SDK oder CLI

- Szenarien für die Verwendung von Lambda AWS SDKs

- Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Leihbibliothek-REST-API erstellen
- Erstellen einer Messenger-Anwendung mit Step Functions
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen einer Websocket-Chat-Anwendung mit API Gateway
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Ein Monitoring-Dashboard mit dem Funktionsnamen als Variable erstellen
- Erstellen Sie Ihre erste serverlose Funktion
- Aufrufen einer Lambda-Funktion von einem Browser aus
- Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion
- Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

- Serverless-Beispiele für Lambda
  - Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion
  - Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser
  - Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser
  - Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser
  - Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

- Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser
- Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen
- Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser
- AWS Community-Beiträge für Lambda
  - Erstellen und Testen einer Serverless-Anwendung

Grundlegende Beispiele für die Verwendung von Lambda AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von AWS Lambda mit AWS SDKs verwenden können.

Beispiele

- Hallo Lambda
- Lernen Sie die Grundlagen von Lambda mit einem AWS SDK
- Aktionen für Lambda mit AWS SDKs
  - Verwendung von CreateAlias mit einer CLI
  - Verwenden Sie mit einem CreateFunction AWS SDK oder CLI
  - Verwendung von DeleteAlias mit einer CLI
  - Verwenden Sie mit einem DeleteFunction AWS SDK oder CLI
  - Verwendung von DeleteFunctionConcurrency mit einer CLI
  - Verwendung von DeleteProvisionedConcurrencyConfig mit einer CLI
  - Verwendung von GetAccountSettings mit einer CLI
  - Verwendung von GetAlias mit einer CLI
  - Verwenden Sie mit einem GetFunction AWS SDK oder CLI
  - Verwendung von GetFunctionConcurrency mit einer CLI
  - Verwendung von GetFunctionConfiguration mit einer CLI
  - Verwendung von GetPolicy mit einer CLI
  - Verwendung von GetProvisionedConcurrencyConfig mit einer CLI
  - Verwenden Sie Invoke mit einem AWS SDK oder CLI

- Verwenden Sie ListFunctionsmit einem AWS SDK oder CLI
- Verwendung von ListProvisionedConcurrencyConfigs mit einer CLI
- Verwendung von ListTags mit einer CLI
- Verwendung von ListVersionsByFunction mit einer CLI
- Verwendung von PublishVersion mit einer CLI
- Verwendung von PutFunctionConcurrency mit einer CLI
- Verwendung von PutProvisionedConcurrencyConfig mit einer CLI
- Verwendung von RemovePermission mit einer CLI
- Verwendung von TagResource mit einer CLI
- Verwendung von UntagResource mit einer CLI
- Verwendung von UpdateAlias mit einer CLI
- Verwenden Sie mit einem UpdateFunctionCode AWS SDK oder CLI
- Verwenden Sie UpdateFunctionConfigurationmit einem AWS SDK oder CLI

Hallo Lambda

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Lambda beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace LambdaActions;

using Amazon.Lambda;

public class HelloLambda
{
    static async Task Main(string[] args)
    {
```

```
        var lambdaClient = new AmazonLambdaClient();

        Console.WriteLine("Hello AWS Lambda");
        Console.WriteLine("Let's get started with AWS Lambda by listing your
 existing Lambda functions:");

        var response = await lambdaClient.ListFunctionsAsync();
        response.Functions.ForEach(function =>
        {

 Console.WriteLine($"{function.FunctionName}\t{function.Description}");
        });
    }
}
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

# Set this project's name.
project("hello_lambda")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_lambda.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
 client and lists the Lambda functions.
 *
 *  main function
 *
 *  Usage: 'hello_lambda'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);
        std::vector<Aws::String> functions;
        Aws::String marker; // Used for pagination.

        do {
            Aws::Lambda::Model::ListFunctionsRequest request;
            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::Lambda::Model::ListFunctionsOutcome outcome =
 lambdaClient.ListFunctions(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Lambda::Model::ListFunctionsResult
 &listFunctionsResult = outcome.GetResult();
                std::cout << listFunctionsResult.GetFunctions().size()
                          << " lambda functions were retrieved." << std::endl;

                for (const Aws::Lambda::Model::FunctionConfiguration
 &functionConfiguration: listFunctionsResult.GetFunctions()) {
                    functions.push_back(functionConfiguration.GetFunctionName());
                    std::cout << functions.size() << "  "
```

```
                                << functionConfiguration.GetDescription() <<
 std::endl;
                    std::cout << "   "
                                <<
 Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                        functionConfiguration.GetRuntime()) << ": "
                                << functionConfiguration.GetHandler()
                                << std::endl;
                }
                marker = listFunctionsResult.GetNextMarker();
            } else {
                std::cerr << "Error with Lambda::ListFunctions. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!marker.empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListFunctionsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
```

```
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
)

// main uses the AWS SDK for Go (v2) to create an AWS Lambda client and list up
 to 10
// functions in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 lambdaClient := lambda.NewFromConfig(sdkConfig)

 maxItems := 10
 fmt.Printf("Let's list up to %v functions for your account.\n", maxItems)
 result, err := lambdaClient.ListFunctions(ctx, &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 if err != nil {
  fmt.Printf("Couldn't list functions for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Functions) == 0 {
  fmt.Println("You don't have any functions!")
 } else {
  for _, function := range result.Functions {
   fmt.Printf("\t%v\n", *function.FunctionName)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists the AWS Lambda functions associated with the current AWS account.
    *
    * @param awsLambda an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    *
    * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
    */
   public static void listFunctions(LambdaClient awsLambda) {
       try {
           ListFunctionsResponse functionResult = awsLambda.listFunctions();
           List<FunctionConfiguration> list = functionResult.functions();
           for (FunctionConfiguration config : list) {
               System.out.println("The function name is " +
config.functionName());
           }

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { LambdaClient, paginateListFunctions } from "@aws-sdk/client-lambda";

const client = new LambdaClient({});

export const helloLambda = async () => {
  const paginator = paginateListFunctions({ client }, {});
  const functions = [];

  for await (const page of paginator) {
    const funcNames = page.Functions.map((f) => f.FunctionName);
    functions.push(...funcNames);
  }

  console.log("Functions:");
  console.log(functions.join("\n"));
  return functions;
};
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def main():
    """
    List the Lambda functions in your AWS account.
    """
    # Create the Lambda client
    lambda_client = boto3.client("lambda")

    # Use the paginator to list the functions
    paginator = lambda_client.get_paginator("list_functions")
    response_iterator = paginator.paginate()

    print("Here are the Lambda functions in your account:")
    for page in response_iterator:
        for function in page["Functions"]:
            print(f"  {function['FunctionName']}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ListFunctionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-lambda'
```

```
# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
  Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
 necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end

  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Ruby API-Referenz.

Lernen Sie die Grundlagen von Lambda mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.

- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Methoden, die Lambda-Aktionen ausführen.

```
namespace LambdaActions;

using Amazon.Lambda;
using Amazon.Lambda.Model;

/// <summary>
/// A class that implements AWS Lambda methods.
/// </summary>
public class LambdaWrapper
{
    private readonly IAmazonLambda _lambdaService;

    /// <summary>
    /// Constructor for the LambdaWrapper class.
    /// </summary>
    /// <param name="lambdaService">An initialized Lambda service client.</param>
    public LambdaWrapper(IAmazonLambda lambdaService)
    {
        _lambdaService = lambdaService;
    }

    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function.</param>
```

```
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };

        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }


    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
```

```
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</
returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }


    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string
 functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }


    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
```

```
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue =
System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }


    /// <summary>
    /// Get a list of Lambda functions.
    /// </summary>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
    {
        var functionList = new List<FunctionConfiguration>();

        var functionPaginator =
            _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
        await foreach (var function in functionPaginator.Functions)
        {
            functionList.Add(function);
        }

        return functionList;
    }


    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
```

```
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }


    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment
 variables.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
```

```
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
        };

        var response = await
 _lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

Erstellen Sie eine Funktion, die das Szenario ausführt.

```
global using System.Threading.Tasks;
global using Amazon.IdentityManagement;
global using Amazon.Lambda;
global using LambdaActions;
global using LambdaScenarioCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


using Amazon.Lambda.Model;
using Microsoft.Extensions.Configuration;

namespace LambdaBasics;

public class LambdaBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonLambda>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<LambdaWrapper>()
            .AddTransient<LambdaRoleWrapper>()
            .AddTransient<UIWrapper>()
        )
        .Build();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();


        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<LambdaBasics>();

        var lambdaWrapper = host.Services.GetRequiredService<LambdaWrapper>();
        var lambdaRoleWrapper =
host.Services.GetRequiredService<LambdaRoleWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        string functionName = configuration["FunctionName"]!;
        string roleName = configuration["RoleName"]!;
        string policyDocument = "{" +
            " \"Version\": \"2012-10-17\"," +
            " \"Statement\": [ " +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Principal\": {" +
            "            \"Service\": \"lambda.amazonaws.com\" " +
            "    }," +
            "        \"Action\": \"sts:AssumeRole\" " +
            "    }" +
```

```
            "]" +
        "}";

        var incrementHandler = configuration["IncrementHandler"];
        var calculatorHandler = configuration["CalculatorHandler"];
        var bucketName = configuration["BucketName"];
        var incrementKey = configuration["IncrementKey"];
        var calculatorKey = configuration["CalculatorKey"];
        var policyArn = configuration["PolicyArn"];

        uiWrapper.DisplayLambdaBasicsOverview();

        // Create the policy to use with the AWS Lambda functions and then attach
the
        // policy to a new role.
        var roleArn = await lambdaRoleWrapper.CreateLambdaRoleAsync(roleName,
policyDocument);

        Console.WriteLine("Waiting for role to become active.");
        uiWrapper.WaitABit(15, "Wait until the role is active before trying to
use it.");

        // Attach the appropriate AWS Identity and Access Management (IAM) role
policy to the new role.
        var success = await
lambdaRoleWrapper.AttachLambdaRolePolicyAsync(policyArn, roleName);
        uiWrapper.WaitABit(10, "Allow time for the IAM policy to be attached to
the role.");

        // Create the Lambda function using a zip file stored in an Amazon Simple
Storage Service
        // (Amazon S3) bucket.
        uiWrapper.DisplayTitle("Create Lambda Function");
        Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
        var lambdaArn = await lambdaWrapper.CreateLambdaFunctionAsync(
            functionName,
            bucketName,
            incrementKey,
            roleArn,
            incrementHandler);

        Console.WriteLine("Waiting for the new function to be available.");
        Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");
```

```
        // Get the Lambda function.
        Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
        FunctionConfiguration config;
        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.State != State.Active);

        Console.WriteLine($"\nThe function, {functionName} has been created.");
        Console.WriteLine($"The runtime of this Lambda function is
{config.Runtime}.");

        uiWrapper.PressEnter();

        // List the Lambda functions.
        uiWrapper.DisplayTitle("Listing all Lambda functions.");
        var functions = await lambdaWrapper.ListFunctionsAsync();
        DisplayFunctionList(functions);

        uiWrapper.DisplayTitle("Invoke increment function");
        Console.WriteLine("Now that it has been created, invoke the Lambda
increment function.");
        string? value;
        do
        {
            Console.Write("Enter a value to increment: ");
            value = Console.ReadLine();
        }
        while (string.IsNullOrEmpty(value));

        string functionParameters = "{" +
            "\"action\": \"increment\", " +
            "\"x\": \"" + value + "\"" +
        "}";
        var answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
        Console.WriteLine($"{value} + 1 = {answer}.");

        uiWrapper.DisplayTitle("Update function");
        Console.WriteLine("Now update the Lambda function code.");
        await lambdaWrapper.UpdateFunctionCodeAsync(functionName, bucketName,
calculatorKey);
```

```
        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        await lambdaWrapper.UpdateFunctionConfigurationAsync(
            functionName,
            calculatorHandler,
            new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        uiWrapper.DisplayTitle("Call updated function");
        Console.WriteLine("Now call the updated function...");

        bool done = false;

        do
        {
            string? opSelected;

            Console.WriteLine("Select the operation to perform:");
            Console.WriteLine("\t1. add");
            Console.WriteLine("\t2. subtract");
            Console.WriteLine("\t3. multiply");
            Console.WriteLine("\t4. divide");
            Console.WriteLine("\tOr enter \"q\" to quit.");
            Console.WriteLine("Enter the number (1, 2, 3, 4, or q) of the
operation you want to perform: ");
            do
            {
                Console.Write("Your choice? ");
                opSelected = Console.ReadLine();
            }
            while (opSelected == string.Empty);
```

```
            var operation = (opSelected) switch
            {
                "1" => "add",
                "2" => "subtract",
                "3" => "multiply",
                "4" => "divide",
                "q" => "quit",
                _ => "add",
            };

            if (operation == "quit")
            {
                done = true;
            }
            else
            {
                // Get two numbers and an action from the user.
                value = string.Empty;
                do
                {
                    Console.Write("Enter the first value: ");
                    value = Console.ReadLine();
                }
                while (value == string.Empty);

                string? value2;
                do
                {
                    Console.Write("Enter a second value: ");
                    value2 = Console.ReadLine();
                }
                while (value2 == string.Empty);

                functionParameters = "{" +
                    "\"action\": \"" + operation + "\", " +
                    "\"x\": \"" + value + "\"," +
                    "\"y\": \"" + value2 + "\"" +
                "}";

                answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
                Console.WriteLine($"The answer when we {operation} the two
numbers is: {answer}.");
            }
```

```
            uiWrapper.PressEnter();
        } while (!done);

        // Delete the function created earlier.

        uiWrapper.DisplayTitle("Clean up resources");
        // Detach the IAM policy from the IAM role.
        Console.WriteLine("First detach the IAM policy from the role.");
        success = await lambdaRoleWrapper.DetachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(15, "Let's wait for the policy to be fully detached
from the role.");

        Console.WriteLine("Delete the AWS Lambda function.");
        success = await lambdaWrapper.DeleteFunctionAsync(functionName);
        if (success)
        {
            Console.WriteLine($"The {functionName} function was deleted.");
        }
        else
        {
            Console.WriteLine($"Could not remove the function {functionName}");
        }

        // Now delete the IAM role created for use with the functions
        // created by the application.
        Console.WriteLine("Now we can delete the role that we created.");
        success = await lambdaRoleWrapper.DeleteLambdaRoleAsync(roleName);
        if (success)
        {
            Console.WriteLine("The role has been successfully removed.");
        }
        else
        {
            Console.WriteLine("Couldn't delete the role.");
        }

        Console.WriteLine("The Lambda Scenario is now complete.");
        uiWrapper.PressEnter();

        // Displays a formatted list of existing functions returned by the
        // LambdaMethods.ListFunctions.
        void DisplayFunctionList(List<FunctionConfiguration> functions)
```

```
        {
            functions.ForEach(functionConfig =>
            {

 Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
            });
        }
    }
}


namespace LambdaActions;

using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class LambdaRoleWrapper
{
    private readonly IAmazonIdentityManagementService _lambdaRoleService;

    public LambdaRoleWrapper(IAmazonIdentityManagementService lambdaRoleService)
    {
        _lambdaRoleService = lambdaRoleService;
    }

    /// <summary>
    /// Attach an AWS Identity and Access Management (IAM) role policy to the
    /// IAM role to be assumed by the AWS Lambda functions created for the
 scenario.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
 policy.</param>
    /// <param name="roleName">The name of the IAM role to attach the IAM policy
 to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to create.</param>
    /// <param name="policyDocument">The policy document for the new IAM role.</
param>
    /// <returns>A string representing the ARN for newly created role.</returns>
    public async Task<string> CreateLambdaRoleAsync(string roleName, string
 policyDocument)
    {
        var request = new CreateRoleRequest
        {
            AssumeRolePolicyDocument = policyDocument,
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.CreateRoleAsync(request);
        return response.Role.Arn;
    }

    /// <summary>
    /// Deletes an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</
returns>
    public async Task<bool> DeleteLambdaRoleAsync(string roleName)
    {
        var request = new DeleteRoleRequest
        {
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.DeleteRoleAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    public async Task<bool> DetachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```
namespace LambdaScenarioCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the AWS Lambda Basics scenario.
    /// </summary>
    public void DisplayLambdaBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to AWS Lambda Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an AWS Identity and Access Management
 (IAM) role that will be assumed by the functions we create.");
        Console.WriteLine("\t2. Attaches an IAM role policy that has Lambda
 permissions.");
        Console.WriteLine("\t3. Creates a Lambda function that increments the
 value passed to it.");
        Console.WriteLine("\t4. Calls the increment function and passes a
 value.");
        Console.WriteLine("\t5. Updates the code so that the function is a simple
 calculator.");
        Console.WriteLine("\t6. Calls the calculator function with the values
 entered.");
        Console.WriteLine("\t7. Deletes the Lambda function.");
        Console.WriteLine("\t7. Detaches the IAM role policy.");
        Console.WriteLine("\t8. Deletes the IAM role.");
        PressEnter();
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }
```

```
    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaIncrement;

public class Function
{

    /// <summary>
    /// A simple function increments the integer parameter.
    /// </summary>
    /// <param name="input">A JSON string containing an action, which must be
    /// "increment" and a string representing the value to increment.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</
param>
    /// <returns>A string representing the incremented value of the parameter.</
returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        if (input["action"] == "increment")
        {
            int inputValue = Convert.ToInt32(input["x"]);
            return inputValue + 1;
        }
        else
        {
            return 0;
        }
    }
}
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaCalculator;

public class Function
{

    /// <summary>
    /// A simple function that takes two number in string format and performs
    /// the requested arithmetic function.
    /// </summary>
    /// <param name="input">JSON data containing an action, and x and y values.
    /// Valid actions include: add, subtract, multiply, and divide.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</
param>
    /// <returns>A string representing the results of the calculation.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        var action = input["action"];
        int x = Convert.ToInt32(input["x"]);
        int y = Convert.ToInt32(input["y"]);
        int result;
        switch (action)
        {
            case "add":
                result = x + y;
                break;
            case "subtract":
                result = x - y;
                break;
            case "multiply":
                result = x * y;
                break;
            case "divide":
                if (y == 0)
```

```
                {
                    Console.Error.WriteLine("Divide by zero error.");
                    result = 0;
                }
                else
                    result = x / y;
                break;
            default:
                Console.Error.WriteLine($"{action} is not a valid operation.");
                result = 0;
                break;
        }
        return result;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get started with functions scenario.
/*!
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
 function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same
 architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
```

```
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                               buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome =
 client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " <<
 seconds
                      << " seconds elapsed." << std::endl;
            break;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&
                 outcome.GetError().GetMessage().find("role") >= 0) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout
                        << "Waiting for the IAM role to become available as a
 CreateFunction parameter. "
                        << seconds
```

```
                        << " seconds elapsed." << std::endl;

                std::cout << outcome.GetError().GetMessage() << std::endl;
            }
        }
        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (60 > seconds);

    std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

    // 3.  Invoke the Lambda function.
    {
        int increment = askQuestionForInt("Enter an increment integer: ");

        Aws::Lambda::Model::InvokeResult invokeResult;
        Aws::Utils::Json::JsonValue jsonPayload;
        jsonPayload.WithString("action", "increment");
        jsonPayload.WithInteger("number", increment);
        if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
                                 invokeResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
                {
                    std::cout << INCREMENT_RESULT_PREFIX
                              << iter->second.AsInteger() << std::endl;
                }
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
```

```
                        invokeResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() <<
 std::endl;
            }
        }
    }

    std::cout
            << "The Lambda function will now be updated with new code. Press
 return to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

    // 4.  Update the Lambda function code.
    {
        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." <<
std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "This function uses an environment variable to control the logging
level."
            << std::endl;
    std::cout
            << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
            << std::endl;
    seconds = 0;

    // 5.  Update the Lambda function configuration.
    do {
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

            // RESOURCE_IN_USE: function code update not completed.
        else if (outcome.GetError().GetErrorType() !=
```

```
                 Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
            if ((seconds % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Lambda function update in progress . After " <<
 seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (0 < seconds);

    if (0 > seconds) {
        std::cerr << "Function failed to become active." << std::endl;
    }
    else {
        std::cout << "Updated function active after " << seconds << " seconds."
                  << std::endl;
    }

    std::cout
            << "\nThe new code applies an arithmetic operator to two variables, x
 an y."
            << std::endl;
    std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-
by"};
    for (size_t i = 0; i < operators.size(); ++i) {
        std::cout << "   " << i + 1 << " " << operators[i] << std::endl;
    }

    // 6.  Invoke the updated Lambda function.
    do {
        int operatorIndex = askQuestionForIntRange("Select an operator index 1 -
 4 ", 1,
                                                   4);
        int x = askQuestionForInt("Enter an integer for the x value ");
        int y = askQuestionForInt("Enter an integer for the y value ");

        Aws::Utils::Json::JsonValue calculateJsonPayload;
        calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
        calculateJsonPayload.WithInteger("x", x);
```

```
        calculateJsonPayload.WithInteger("y", y);
        Aws::Lambda::Model::InvokeResult calculatedResult;
        if (invokeLambdaFunction(calculateJsonPayload,
                                 Aws::Lambda::Model::LogType::Tail,
                                 calculatedResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
                std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsInteger() <<
std::endl;
            }
            else if (iter != values.end() && iter->second.IsFloatingPointType())
{
                std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsDouble() << std::endl;
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                        calculatedResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() <<
std::endl;
            }
        }

        answer = askQuestion("Would you like to try another operation? (y/n) ");
    } while (answer == "y");

    std::cout
            << "A list of the lambda functions will be retrieved. Press return to
continue, ";
    std::getline(std::cin, answer);

    // 7.  List the Lambda functions.

    std::vector<Aws::String> functions;
    Aws::String marker;
```

```
    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          <<
Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    // 8.  Get a Lambda function.
    if (!functions.empty()) {
        std::stringstream question;
        question << "Choose a function to retrieve between 1 and " <<
functions.size()
                 << " ";
        int functionIndex = askQuestionForIntRange(question.str(), 1,
```

```
 static_cast<int>(functions.size()));

        Aws::String functionName = functions[functionIndex - 1];

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
 client.GetFunction(request);

        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<

 outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout << "The resources will be deleted. Press return to continue, ";
    std::getline(std::cin, answer);

    // 9.  Delete the Lambda function.
    bool result = deleteLambdaFunction(client);

    // 10. Delete the IAM role.
    return result && deleteIamRole(clientConfig);
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
```

```
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue
 &jsonPayload,
                                     Aws::Lambda::Model::LogType logType,
                                     Aws::Lambda::Model::InvokeResult
 &invokeResult,
                                     const Aws::Lambda::LambdaClient &client) {
    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
 resources are
     * available.  The Invoke function is called repeatedly until the resources
 are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload =
 Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

            // ACCESS_DENIED: because the role is not available yet.
            // RESOURCE_CONFLICT: because the Lambda function is being created or
 updated.
        else if ((outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
                 (outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout << "Waiting for the invoke api to be available, status
 " <<
                          ((outcome.GetError().GetErrorType() ==
```

```
                                Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                                "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
 seconds
                            << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (seconds < 60);

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein interaktives Szenario, das Ihnen zeigt, wie Sie mit Lambda-Funktionen loslegen können.

```
import (
 "archive/zip"
 "bytes"
 "context"
 "encoding/base64"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "os"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 iamtypes "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/lambda/actions"
)

// GetStartedFunctionsScenario shows you how to use AWS Lambda to perform the
 following
// actions:
//
//  1. Create an AWS Identity and Access Management (IAM) role and Lambda
 function, then upload handler code.
//  2. Invoke the function with a single parameter and get results.
//  3. Update the function code and configure with an environment variable.
//  4. Invoke the function with new parameters and get results. Display the
 returned execution log.
//  5. List the functions for your account, then clean up resources.
type GetStartedFunctionsScenario struct {
 sdkConfig       aws.Config
 functionWrapper actions.FunctionWrapper
 questioner      demotools.IQuestioner
 helper          IScenarioHelper
 isTestRun       bool
}
```

```
// NewGetStartedFunctionsScenario constructs a GetStartedFunctionsScenario
 instance from a configuration.
// It uses the specified config to get a Lambda client and create wrappers for
 the actions
// used in the scenario.
func NewGetStartedFunctionsScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedFunctionsScenario {
 lambdaClient := lambda.NewFromConfig(sdkConfig)
 return GetStartedFunctionsScenario{
  sdkConfig:       sdkConfig,
  functionWrapper: actions.FunctionWrapper{LambdaClient: lambdaClient},
  questioner:      questioner,
  helper:          helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedFunctionsScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Lambda get started with functions demo.")
 log.Println(strings.Repeat("-", 88))

 role := scenario.GetOrCreateRole(ctx)
 funcName := scenario.CreateFunction(ctx, role)
 scenario.InvokeIncrement(ctx, funcName)
 scenario.UpdateFunction(ctx, funcName)
 scenario.InvokeCalculator(ctx, funcName)
 scenario.ListFunctions(ctx)
 scenario.Cleanup(ctx, role, funcName)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

```
// GetOrCreateRole checks whether the specified role exists and returns it if it
 does.
// Otherwise, a role is created that specifies Lambda as a trusted principal.
// The AWSLambdaBasicExecutionRole managed policy is attached to the role and the
 role
// is returned.
func (scenario GetStartedFunctionsScenario) GetOrCreateRole(ctx context.Context)
 *iamtypes.Role {
 var role *iamtypes.Role
 iamClient := iam.NewFromConfig(scenario.sdkConfig)
 log.Println("First, we need an IAM role that Lambda can assume.")
 roleName := scenario.questioner.Ask("Enter a name for the role:",
 demotools.NotEmpty{})
 getOutput, err := iamClient.GetRole(ctx, &iam.GetRoleInput{
  RoleName: aws.String(roleName)})
 if err != nil {
  var noSuch *iamtypes.NoSuchEntityException
  if errors.As(err, &noSuch) {
   log.Printf("Role %v doesn't exist. Creating it....\n", roleName)
  } else {
   log.Panicf("Couldn't check whether role %v exists. Here's why: %v\n",
    roleName, err)
  }
 } else {
  role = getOutput.Role
  log.Printf("Found role %v.\n", *role.RoleName)
 }
 if role == nil {
  trustPolicy := PolicyDocument{
   Version: "2012-10-17",
   Statement: []PolicyStatement{{
    Effect:    "Allow",
    Principal: map[string]string{"Service": "lambda.amazonaws.com"},
    Action:    []string{"sts:AssumeRole"},
   }},
  }
  policyArn := "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
  createOutput, err := iamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(trustPolicy.String()),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  }
```

```
  role = createOutput.Role
  _, err = iamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
   PolicyArn: aws.String(policyArn),
   RoleName:  aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't attach a policy to role %v. Here's why: %v\n", roleName,
 err)
  }
  log.Printf("Created role %v.\n", *role.RoleName)
  log.Println("Let's give AWS a few seconds to propagate resources...")
  scenario.helper.Pause(10)
 }
 log.Println(strings.Repeat("-", 88))
 return role
}

// CreateFunction creates a Lambda function and uploads a handler written in
 Python.
// The code for the Python handler is packaged as a []byte in .zip format.
func (scenario GetStartedFunctionsScenario) CreateFunction(ctx context.Context,
 role *iamtypes.Role) string {
 log.Println("Let's create a function that increments a number.\n" +
  "The function uses the 'lambda_handler_basic.py' script found in the \n" +
  "'handlers' directory of this project.")
 funcName := scenario.questioner.Ask("Enter a name for the Lambda function:",
 demotools.NotEmpty{})
 zipPackage := scenario.helper.CreateDeploymentPackage("lambda_handler_basic.py",
 fmt.Sprintf("%v.py", funcName))
 log.Printf("Creating function %v and waiting for it to be ready.", funcName)
 funcState := scenario.functionWrapper.CreateFunction(ctx, funcName,
 fmt.Sprintf("%v.lambda_handler", funcName),
  role.Arn, zipPackage)
 log.Printf("Your function is %v.", funcState)
 log.Println(strings.Repeat("-", 88))
 return funcName
}

// InvokeIncrement invokes a Lambda function that increments a number. The
 function
// parameters are contained in a Go struct that is used to serialize the
 parameters to
// a JSON payload that is passed to the function.
```

```
// The result payload is deserialized into a Go struct that contains an int
 value.
func (scenario GetStartedFunctionsScenario) InvokeIncrement(ctx context.Context,
 funcName string) {
 parameters := actions.IncrementParameters{Action: "increment"}
 log.Println("Let's invoke our function. This function increments a number.")
 parameters.Number = scenario.questioner.AskInt("Enter a number to increment:",
 demotools.NotEmpty{})
 log.Printf("Invoking %v with %v...\n", funcName, parameters.Number)
 invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, parameters,
 false)
 var payload actions.LambdaResultInt
 err := json.Unmarshal(invokeOutput.Payload, &payload)
 if err != nil {
  log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
   funcName, err)
 }
 log.Printf("Invoking %v with %v returned %v.\n", funcName, parameters.Number,
 payload)
 log.Println(strings.Repeat("-", 88))
}

// UpdateFunction updates the code for a Lambda function by uploading a simple
 arithmetic
// calculator written in Python. The code for the Python handler is packaged as a
// []byte in .zip format.
// After the code is updated, the configuration is also updated with a new log
// level that instructs the handler to log additional information.
func (scenario GetStartedFunctionsScenario) UpdateFunction(ctx context.Context,
 funcName string) {
 log.Println("Let's update the function to an arithmetic calculator.\n" +
  "The function uses the 'lambda_handler_calculator.py' script found in the \n" +
  "'handlers' directory of this project.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 log.Println("Creating deployment package...")
 zipPackage :=
 scenario.helper.CreateDeploymentPackage("lambda_handler_calculator.py",
  fmt.Sprintf("%v.py", funcName))
 log.Println("...and updating the Lambda function and waiting for it to be
 ready.")
 funcState := scenario.functionWrapper.UpdateFunctionCode(ctx, funcName,
 zipPackage)
 log.Printf("Updated function %v. Its current state is %v.", funcName, funcState)
```

```
 log.Println("This function uses an environment variable to control logging
 level.")
 log.Println("Let's set it to DEBUG to get the most logging.")
 scenario.functionWrapper.UpdateFunctionConfiguration(ctx, funcName,
  map[string]string{"LOG_LEVEL": "DEBUG"})
 log.Println(strings.Repeat("-", 88))
}

// InvokeCalculator invokes the Lambda calculator function. The parameters are
 stored in a
// Go struct that is used to serialize the parameters to a JSON payload. That
 payload is then passed
// to the function.
// The result payload is deserialized to a Go struct that stores the result as
 either an
// int or float32, depending on the kind of operation that was specified.
func (scenario GetStartedFunctionsScenario) InvokeCalculator(ctx context.Context,
 funcName string) {
 wantInvoke := true
 choices := []string{"plus", "minus", "times", "divided-by"}
 for wantInvoke {
  choice := scenario.questioner.AskChoice("Select an arithmetic operation:\n",
 choices)
  x := scenario.questioner.AskInt("Enter a value for x:", demotools.NotEmpty{})
  y := scenario.questioner.AskInt("Enter a value for y:", demotools.NotEmpty{})
  log.Printf("Invoking %v %v %v...", x, choices[choice], y)
  calcParameters := actions.CalculatorParameters{
   Action: choices[choice],
   X:      x,
   Y:      y,
  }
  invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, calcParameters,
 true)
  var payload any
  if choice == 3 { // divide-by results in a float.
   payload = actions.LambdaResultFloat{}
  } else {
   payload = actions.LambdaResultInt{}
  }
  err := json.Unmarshal(invokeOutput.Payload, &payload)
  if err != nil {
   log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
    funcName, err)
  }
```

```
  log.Printf("Invoking %v with %v %v %v returned %v.\n", funcName,
   calcParameters.X, calcParameters.Action, calcParameters.Y, payload)
  scenario.questioner.Ask("Press Enter to see the logs from the call.")
  logRes, err := base64.StdEncoding.DecodeString(*invokeOutput.LogResult)
  if err != nil {
   log.Panicf("Couldn't decode log result. Here's why: %v\n", err)
  }
  log.Println(string(logRes))
  wantInvoke = scenario.questioner.AskBool("Do you want to calculate again? (y/
n)", "y")
 }
 log.Println(strings.Repeat("-", 88))
}

// ListFunctions lists up to the specified number of functions for your account.
func (scenario GetStartedFunctionsScenario) ListFunctions(ctx context.Context) {
 count := scenario.questioner.AskInt(
  "Let's list functions for your account. How many do you want to see?",
 demotools.NotEmpty{})
 functions := scenario.functionWrapper.ListFunctions(ctx, count)
 log.Printf("Found %v functions:", len(functions))
 for _, function := range functions {
  log.Printf("\t%v", *function.FunctionName)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup removes the IAM and Lambda resources created by the example.
func (scenario GetStartedFunctionsScenario) Cleanup(ctx context.Context, role
 *iamtypes.Role, funcName string) {
 if scenario.questioner.AskBool("Do you want to clean up resources created for
 this example? (y/n)",
  "y") {
  iamClient := iam.NewFromConfig(scenario.sdkConfig)
  policiesOutput, err := iamClient.ListAttachedRolePolicies(ctx,
   &iam.ListAttachedRolePoliciesInput{RoleName: role.RoleName})
  if err != nil {
   log.Panicf("Couldn't get policies attached to role %v. Here's why: %v\n",
    *role.RoleName, err)
  }
  for _, policy := range policiesOutput.AttachedPolicies {
   _, err = iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
    PolicyArn: policy.PolicyArn, RoleName: role.RoleName,
   })
```

```
    if err != nil {
     log.Panicf("Couldn't detach policy %v from role %v. Here's why: %v\n",
      *policy.PolicyArn, *role.RoleName, err)
    }
   }
   _, err = iamClient.DeleteRole(ctx, &iam.DeleteRoleInput{RoleName:
  role.RoleName})
   if err != nil {
    log.Panicf("Couldn't delete role %v. Here's why: %v\n", *role.RoleName, err)
   }
   log.Printf("Deleted role %v.\n", *role.RoleName)

   scenario.functionWrapper.DeleteFunction(ctx, funcName)
   log.Printf("Deleted function %v.\n", funcName)
  } else {
   log.Println("Okay. Don't forget to delete the resources when you're done with
  them.")
  }
}

// IScenarioHelper abstracts I/O and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 CreateDeploymentPackage(sourceFile string, destinationFile string) *bytes.Buffer
}

// ScenarioHelper lets the caller specify the path to Lambda handler functions.
type ScenarioHelper struct {
 HandlerPath string
}

// Pause waits for the specified number of seconds.
func (helper *ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// CreateDeploymentPackage creates an AWS Lambda deployment package from a source
 file. The
// deployment package is stored in .zip format in a bytes.Buffer. The buffer can
 be
// used to pass a []byte to Lambda when creating the function.
// The specified destinationFile is the name to give the file when it's deployed
 to Lambda.
```

```
func (helper *ScenarioHelper) CreateDeploymentPackage(sourceFile string,
 destinationFile string) *bytes.Buffer {
 var err error
 buffer := &bytes.Buffer{}
 writer := zip.NewWriter(buffer)
 zFile, err := writer.Create(destinationFile)
 if err != nil {
  log.Panicf("Couldn't create destination archive %v. Here's why: %v\n",
 destinationFile, err)
 }
 sourceBody, err := os.ReadFile(fmt.Sprintf("%v/%v", helper.HandlerPath,
 sourceFile))
 if err != nil {
  log.Panicf("Couldn't read handler source file %v. Here's why: %v\n",
   sourceFile, err)
 } else {
  _, err = zFile.Write(sourceBody)
  if err != nil {
   log.Panicf("Couldn't write handler %v to zip archive. Here's why: %v\n",
    sourceFile, err)
  }
 }
 err = writer.Close()
 if err != nil {
  log.Panicf("Couldn't close zip writer. Here's why: %v\n", err)
 }
 return buffer
}
```

Erstellen einer Struktur, die die einzelnen Lambda-Aktionen umschließt.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}


// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName
 string) types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx,
 &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
```

```
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in
 the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context,
 functionName string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
```

```
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n",
 functionName, err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionConfiguration updates a map of environment variables configured
 for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v",
 functionName, err)
 }
}



// ListFunctions lists up to maxItems functions for the account. This function
 uses a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
```

```
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}



// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
```

```
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}



// IncrementParameters is used to serialize parameters to the increment Lambda
 handler.
type IncrementParameters struct {
 Action string `json:"action"`
 Number int    `json:"number"`
}

// CalculatorParameters is used to serialize parameters to the calculator Lambda
 handler.
type CalculatorParameters struct {
 Action string `json:"action"`
 X      int    `json:"x"`
 Y      int    `json:"y"`
}

// LambdaResultInt is used to deserialize an int result from a Lambda handler.
type LambdaResultInt struct {
 Result int `json:"result"`
}

// LambdaResultFloat is used to deserialize a float32 result from a Lambda
 handler.
type LambdaResultFloat struct {
 Result float32 `json:"result"`
}
```

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
```

```
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the
 numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
```

```
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <functionName> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                role - The AWS Identity and Access Management (IAM) service role
 that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
 example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 name that contains the .zip or .jar used to update the Lambda function's code.\s
                key - The Amazon S3 key name that represents the .zip or .jar
 (for example, LambdaHello-1.0-SNAPSHOT.jar).
                """;

        if (args.length != 5) {
              System.out.println(usage);
              return;
        }

        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
 bucketName, role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda
 function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
 again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
```

```
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();
```

```
            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which
is used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response =
awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " +
config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with
the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String
functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
```

```
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter
                .waitUntilFunctionUpdated(getFunctionConfigRequest);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which
is used to interact with the AWS Lambda service
```

```
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda
 function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine AWS Identity and Access Management (IAM-) Rolle, die Lambda die Berechtigung erteilt, in Protokolle zu schreiben.

```
    logger.log(`Creating role (${NAME_ROLE_LAMBDA})...`);
    const response = await createRole(NAME_ROLE_LAMBDA);

import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

Erstellen Sie eine Lambda-Funktion und laden Sie Handlercode hoch.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
```

```
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

Aktualisieren Sie den Funktionscode und konfigurieren Sie seine Lambda-Umgebung mit einer Umgebungsvariablen.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
```

```
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

Listen Sie die Funktionen für Ihr Konto auf.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Löschen Sie die IAM-Rolle für Ihre Lambda-Funktion.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

```
/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName>
 <key>

        Where:
            functionName - The name of the AWS Lambda function.
```

```
            role - The AWS Identity and Access Management (IAM) service role that
has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
name that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file
(for example, LambdaHello-1.0-SNAPSHOT.jar).
            """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)
```

```
    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}
```

```
suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is
 ${res.payload?.toString(Charsets.UTF_8)}")
    }
}
```

```
suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }
```

```
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
```

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.
\n";

        echo "\nNow let's create an S3 bucket and upload our Lambda code there.
\n";
        $bucketName = "amzn-s3-demo-bucket-$uniqid";
        $s3client->createBucket([
```

```
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName,
 $role, $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);

        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
```

```
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for
 more information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName,
 $calculatorParams, "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does
 equal $result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";
```

```
        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
 "Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
 $listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
```

```
        echo "Deleted the bucket.\n";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.
```

```
    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the
 numbers,
```

```
    and returns the result.

    :param event: The event dict that contains the parameters sent when the
 function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

Erstellen Sie Funktionen, die Lambda-Aktionen umschließen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
```

```
        Creates a Lambda deployment package in .zip format in an in-memory
buffer. This
        buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
```

```
        Creates an IAM role that grants the Lambda function basic permissions. If
 a
        role with the specified name already exists, it is used for the demo.

        :param iam_role_name: The name of the role to create.
        :return: The role and a value that indicates whether the role is newly
 created.
        """
        role = self.get_iam_role(iam_role_name)
        if role is not None:
            return role, False

        lambda_assume_role_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "lambda.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

        try:
            role = self.iam_resource.create_role(
                RoleName=iam_role_name,
                AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy),
            )
            logger.info("Created role %s.", role.name)
            role.attach_policy(PolicyArn=policy_arn)
            logger.info("Attached basic execution policy to role %s.", role.name)
        except ClientError as error:
            if error.response["Error"]["Code"] == "EntityAlreadyExists":
                role = self.iam_resource.Role(iam_role_name)
                logger.warning("The role %s already exists. Using it.",
 iam_role_name)
            else:
                logger.exception(
                    "Couldn't create role %s or attach policy %s.",
                    iam_role_name,
                    policy_arn,
                )
```

```
                raise

        return role, True

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response =
self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function.
This
                             must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the
function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
```

```
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
```

```
        :param function_params: The parameters of the function as a dict. This
dict
                                is serialized to JSON before it is sent to
Lambda.
        :param get_log: When true, the last 4 KB of the execution log are
included in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes
in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return response


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Funktion, die das Szenario ausführt.

```
class UpdateFunctionWaiter(CustomWaiter):
    """A custom waiter that waits until a function is successfully updated."""

    def __init__(self, client):
        super().__init__(
            "UpdateSuccess",
            "GetFunction",
            "Configuration.LastUpdateStatus",
            {"Successful": WaitState.SUCCESS, "Failed": WaitState.FAILURE},
            client,
        )

    def wait(self, function_name):
        self._wait(FunctionName=function_name)


def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
 lambda_name):
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda
 handler.
    :param calculator_file: The name of the file that contains the calculator
 Lambda handler.
    :param lambda_name: The name to give resources created for the scenario, such
 as the
                        IAM role and the Lambda function.
```

```
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print("-" * 88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(
            basic_file, f"{lambda_name}.py"
        )
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f"{lambda_name}.lambda_handler", iam_role,
deployment_package
        )
    else:
        print(f"Function {lambda_name} already exists.")
    print("-" * 88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
        "action": "increment",
        "number": q.ask("Give me a number to increment: ", q.is_int),
    }
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(
        f"Incrementing {action_params['number']} resulted in "
        f"{json.load(response['Payload'])}"
    )
    print("-" * 88)
```

```
    print(f"Let's update the function to an arithmetic calculator.")
    q.ask("Press Enter when you're ready.")
    print("Creating a new deployment package...")
    deployment_package = wrapper.create_deployment_package(
        calculator_file, f"{lambda_name}.py"
    )
    print(f"...and updating the {lambda_name} Lambda function.")
    update_waiter = UpdateFunctionWaiter(lambda_client)
    wrapper.update_function_code(lambda_name, deployment_package)
    update_waiter.wait(lambda_name)
    print(f"This function uses an environment variable to control logging
level.")
    print(f"Let's set it to DEBUG to get the most logging.")
    wrapper.update_function_configuration(
        lambda_name, {"LOG_LEVEL": logging.getLevelName(logging.DEBUG)}
    )

    actions = ["plus", "minus", "times", "divided-by"]
    want_invoke = True
    while want_invoke:
        print(f"Let's invoke {lambda_name}. You can invoke these actions:")
        for index, action in enumerate(actions):
            print(f"{index + 1}: {action}")
        action_params = {}
        action_index = q.ask(
            "Enter the number of the action you want to take: ",
            q.is_int,
            q.in_range(1, len(actions)),
        )
        action_params["action"] = actions[action_index - 1]
        print(f"You've chosen to invoke 'x {action_params['action']} y'.")
        action_params["x"] = q.ask("Enter a value for x: ", q.is_int)
        action_params["y"] = q.ask("Enter a value for y: ", q.is_int)
        print(f"Invoking {lambda_name}...")
        response = wrapper.invoke_function(lambda_name, action_params, True)
        print(
            f"Calculating {action_params['x']} {action_params['action']}
{action_params['y']} "
            f"resulted in {json.load(response['Payload'])}"
        )
        q.ask("Press Enter to see the logs from the call.")
        print(base64.b64decode(response["LogResult"]).decode())
        want_invoke = q.ask("That was fun. Shall we do it again? (y/n) ",
q.is_yesno)
```

```
    print("-" * 88)

    if q.ask(
        "Do you want to list all of the functions in your account? (y/n) ",
 q.is_yesno
    ):
        wrapper.list_functions()
    print("-" * 88)

    if q.ask("Ready to delete the function and role? (y/n) ", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            boto3.client("lambda"),
            boto3.resource("iam"),
            "lambda_handler_basic.py",
            "lambda_handler_calculator.py",
            "doc_example_lambda_calculator",
        )
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions

- UpdateFunctionCode
- UpdateFunctionConfiguration

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die erforderlichen IAM-Berechtigungen für eine Lambda-Funktion ein, die Protokolle schreiben kann.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
  role_policy = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Principal': { 'Service': 'lambda.amazonaws.com' },
```

```
          'Action': 'sts:AssumeRole'
        }
      ]
    }
    role = @iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    @iam_client.attach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    wait_for_role_to_exist(iam_role_name)
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    sleep(10)
    [role, role_policy.to_json]
  end

  def destroy_iam_role(iam_role_name)
    @iam_client.detach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    @iam_client.delete_role(role_name: iam_role_name)
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  end

  def wait_for_role_to_exist(iam_role_name)
    @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  end
```

Definieren Sie einen Lambda-Handler, der eine als Aufrufparameter bereitgestellte Zahl inkrementiert.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
                   Logger::DEBUG
                 when 'info'
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Komprimieren (ZIP) Sie Ihre Lambda-Funktion in ein Bereitstellungspaket.

```
  # Creates a Lambda deployment package in .zip format.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda
 file and zip.
  # @return: The deployment package.
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?('lambda_function.zip')
      File.delete('lambda_function.zip')
      @logger.debug('Deleting old zip: lambda_function.zip')
    end
```

```
    Zip::File.open('lambda_function.zip', create: true) do |zipfile|
      zipfile.add('lambda_function.rb', "#{source_file}.rb")
    end
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read('lambda_function.zip').to_s
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n
#{e.message}")
  end
```

Erstellen Sie eine neue Lambda-Funktion.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
```

```
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

Rufen Sie Ihre Lambda-Funktion mit optionalen Laufzeitparametern auf.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n
#{e.message}")
  end
```

Aktualisieren Sie die Konfiguration Ihrer Lambda-Funktion, um eine neue Umgebungsvariable einzufügen.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
```

```
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for
#{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

Aktualisieren Sie den Code Ihrer Lambda-Funktion mit einem anderen Bereitstellungspaket, das anderen Code enthält.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for:
#{function_name}:\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n
#{e.message}")
  end
```

Listen Sie alle vorhandenen Lambda-Funktionen mithilfe des eingebauten Paginators auf.

```
# Lists the Lambda functions for the current account.
def list_functions
  functions = []
  @lambda_client.list_functions.each do |response|
    response['functions'].each do |function|
      functions.append(function['function_name'])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

Löschen Sie eine bestimmte Lambda-Funktion.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Cargo.toml mit den Abhängigkeiten, die in diesem Szenario verwendet werden.

```
[package]
name = "lambda-code-examples"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-ec2 = { version = "1.3.0" }
aws-sdk-iam = { version = "1.3.0" }
aws-sdk-lambda = { version = "1.3.0" }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
tracing = "0.1.37"
serde_json = "1.0.94"
anyhow = "1.0.71"
uuid = { version = "1.3.3", features = ["v4"] }
lambda_runtime = "0.8.0"
serde = "1.0.164"
```

Eine Sammlung von Hilfsprogrammen, die das Aufrufen von Lambda für dieses Szenario optimieren. Diese Datei befindet sich in der Datei src/ations .rs in der Kiste.

```
use anyhow::anyhow;
use aws_sdk_iam::operation::{create_role::CreateRoleError,
 delete_role::DeleteRoleOutput};
use aws_sdk_lambda::{
    operation::{
        delete_function::DeleteFunctionOutput, get_function::GetFunctionOutput,
        invoke::InvokeOutput, list_functions::ListFunctionsOutput,
        update_function_code::UpdateFunctionCodeOutput,
        update_function_configuration::UpdateFunctionConfigurationOutput,
    },
    primitives::ByteStream,
    types::{Environment, FunctionCode, LastUpdateStatus, State},
};
use aws_sdk_s3::{
    error::ErrorMetadata,
    operation::{delete_bucket::DeleteBucketOutput,
 delete_object::DeleteObjectOutput},
    types::CreateBucketConfiguration,
};
use aws_smithy_types::Blob;
use serde::{ser::SerializeMap, Serialize};
use std::{fmt::Display, path::PathBuf, str::FromStr, time::Duration};
use tracing::{debug, info, warn};

/* Operation describes  */
#[derive(Clone, Copy, Debug, Serialize)]
pub enum Operation {
    #[serde(rename = "plus")]
    Plus,
    #[serde(rename = "minus")]
    Minus,
    #[serde(rename = "times")]
    Times,
    #[serde(rename = "divided-by")]
    DividedBy,
}

impl FromStr for Operation {
    type Err = anyhow::Error;

    fn from_str(s: &str) -> Result<Self, Self::Err> {
        match s {
```

```
            "plus" => Ok(Operation::Plus),
            "minus" => Ok(Operation::Minus),
            "times" => Ok(Operation::Times),
            "divided-by" => Ok(Operation::DividedBy),
            _ => Err(anyhow!("Unknown operation {s}")),
        }
    }
}

impl Display for Operation {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match self {
            Operation::Plus => write!(f, "plus"),
            Operation::Minus => write!(f, "minus"),
            Operation::Times => write!(f, "times"),
            Operation::DividedBy => write!(f, "divided-by"),
        }
    }
}

/**
 * InvokeArgs will be serialized as JSON and sent to the AWS Lambda handler.
 */
#[derive(Debug)]
pub enum InvokeArgs {
    Increment(i32),
    Arithmetic(Operation, i32, i32),
}

impl Serialize for InvokeArgs {
    fn serialize<S>(&self, serializer: S) -> Result<S::Ok, S::Error>
    where
        S: serde::Serializer,
    {
        match self {
            InvokeArgs::Increment(i) => serializer.serialize_i32(*i),
            InvokeArgs::Arithmetic(o, i, j) => {
                let mut map: S::SerializeMap =
 serializer.serialize_map(Some(3))?;
                map.serialize_key(&"op".to_string())?;
                map.serialize_value(&o.to_string())?;
                map.serialize_key(&"i".to_string())?;
                map.serialize_value(&i)?;
                map.serialize_key(&"j".to_string())?;
```

```
                map.serialize_value(&j)?;
                map.end()
            }
        }
    }
}

/** A policy document allowing Lambda to execute this function on the account's
 behalf. */
const ROLE_POLICY_DOCUMENT: &str = r#"{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "lambda.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
    ]
}"#;

/**
 * A LambdaManager gathers all the resources necessary to run the Lambda example
 scenario.
 * This includes instantiated aws_sdk clients and details of resource names.
 */
pub struct LambdaManager {
    iam_client: aws_sdk_iam::Client,
    lambda_client: aws_sdk_lambda::Client,
    s3_client: aws_sdk_s3::Client,
    lambda_name: String,
    role_name: String,
    bucket: String,
    own_bucket: bool,
}

// These unit type structs provide nominal typing on top of String parameters for
 LambdaManager::new
pub struct LambdaName(pub String);
pub struct RoleName(pub String);
pub struct Bucket(pub String);
pub struct OwnBucket(pub bool);

impl LambdaManager {
    pub fn new(
```

```
        iam_client: aws_sdk_iam::Client,
        lambda_client: aws_sdk_lambda::Client,
        s3_client: aws_sdk_s3::Client,
        lambda_name: LambdaName,
        role_name: RoleName,
        bucket: Bucket,
        own_bucket: OwnBucket,
    ) -> Self {
        Self {
            iam_client,
            lambda_client,
            s3_client,
            lambda_name: lambda_name.0,
            role_name: role_name.0,
            bucket: bucket.0,
            own_bucket: own_bucket.0,
        }
    }

    /**
     * Load the AWS configuration from the environment.
     * Look up lambda_name and bucket if none are given, or generate a random
name if not present in the environment.
     * If the bucket name is provided, the caller needs to have created the
bucket.
     * If the bucket name is generated, it will be created.
     */
    pub async fn load_from_env(lambda_name: Option<String>, bucket:
Option<String>) -> Self {
        let sdk_config = aws_config::load_from_env().await;
        let lambda_name = LambdaName(lambda_name.unwrap_or_else(|| {
            std::env::var("LAMBDA_NAME").unwrap_or_else(|_|
"rust_lambda_example".to_string())
        }));
        let role_name = RoleName(format!("{}_role", lambda_name.0));
        let (bucket, own_bucket) =
            match bucket {
                Some(bucket) => (Bucket(bucket), false),
                None => (
                    Bucket(std::env::var("LAMBDA_BUCKET").unwrap_or_else(|_| {
                        format!("rust-lambda-example-{}", uuid::Uuid::new_v4())
                    })),
                    true,
                ),
```

```
            };

        let s3_client = aws_sdk_s3::Client::new(&sdk_config);

        if own_bucket {
            info!("Creating bucket for demo: {}", bucket.0);
            s3_client
                .create_bucket()
                .bucket(bucket.0.clone())
                .create_bucket_configuration(
                    CreateBucketConfiguration::builder()

 .location_constraint(aws_sdk_s3::types::BucketLocationConstraint::from(
                            sdk_config.region().unwrap().as_ref(),
                        ))
                        .build(),
                )
                .send()
                .await
                .unwrap();
        }

        Self::new(
            aws_sdk_iam::Client::new(&sdk_config),
            aws_sdk_lambda::Client::new(&sdk_config),
            s3_client,
            lambda_name,
            role_name,
            bucket,
            OwnBucket(own_bucket),
        )
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
```

```
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }

    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
```

```
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Create an IAM execution role for the managed Lambda function.
     * If the role already exists, use that instead.
     */
    async fn create_role(&self) -> Result<aws_sdk_iam::types::Role,
CreateRoleError> {
        info!("Creating execution role for function");
        let get_role = self
            .iam_client
            .get_role()
            .role_name(self.role_name.clone())
            .send()
            .await;
        if let Ok(get_role) = get_role {
            if let Some(role) = get_role.role {
                return Ok(role);
            }
        }

        let create_role = self
            .iam_client
            .create_role()
            .role_name(self.role_name.clone())
            .assume_role_policy_document(ROLE_POLICY_DOCUMENT)
            .send()
            .await;

        match create_role {
            Ok(create_role) => match create_role.role {
                Some(role) => Ok(role),
```

```
                None => Err(CreateRoleError::generic(
                    ErrorMetadata::builder()
                        .message("CreateRole returned empty success")
                        .build(),
                )),
            },
            Err(err) => Err(err.into_service_error()),
        }
    }

    /**
     * Poll `is_function_ready` with a 1-second delay. It returns when the
function is ready or when there's an error checking the function's state.
     */
    pub async fn wait_for_function_ready(&self) -> Result<(), anyhow::Error> {
        info!("Waiting for function");
        while !self.is_function_ready(None).await? {
            info!("Function is not ready, sleeping 1s");
            tokio::time::sleep(Duration::from_secs(1)).await;
        }
        Ok(())
    }

    /**
     * Check if a Lambda function is ready to be invoked.
     * A Lambda function is ready for this scenario when its state is active and
its LastUpdateStatus is Successful.
     * Additionally, if a sha256 is provided, the function must have that as its
current code hash.
     * Any missing properties or failed requests will be reported as an Err.
     */
    async fn is_function_ready(
        &self,
        expected_code_sha256: Option<&str>,
    ) -> Result<bool, anyhow::Error> {
        match self.get_function().await {
            Ok(func) => {
                if let Some(config) = func.configuration() {
                    if let Some(state) = config.state() {
                        info!(?state, "Checking if function is active");
                        if !matches!(state, State::Active) {
                            return Ok(false);
                        }
                    }
```

```
                    match config.last_update_status() {
                        Some(last_update_status) => {
                            info!(?last_update_status, "Checking if function is
ready");
                            match last_update_status {
                                LastUpdateStatus::Successful => {
                                    // continue
                                }
                                LastUpdateStatus::Failed |
LastUpdateStatus::InProgress => {
                                    return Ok(false);
                                }
                                unknown => {
                                    warn!(
                                        status_variant = unknown.as_str(),
                                        "LastUpdateStatus unknown"
                                    );
                                    return Err(anyhow!(
                                        "Unknown LastUpdateStatus, fn config is
{config:?}"
                                    ));
                                }
                            }
                        }
                        None => {
                            warn!("Missing last update status");
                            return Ok(false);
                        }
                    };
                    if expected_code_sha256.is_none() {
                        return Ok(true);
                    }
                    if let Some(code_sha256) = config.code_sha256() {
                        return Ok(code_sha256 ==
expected_code_sha256.unwrap_or_default());
                    }
                }
            }
            Err(e) => {
                warn!(?e, "Could not get function while waiting");
            }
        }
        Ok(false)
    }
```

```
   /** Get the Lambda function with this Manager's name. */
   pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error>
{
       info!("Getting lambda function");
       self.lambda_client
           .get_function()
           .function_name(self.lambda_name.clone())
           .send()
           .await
           .map_err(anyhow::Error::from)
   }

   /** List all Lambda functions in the current Region. */
   pub async fn list_functions(&self) -> Result<ListFunctionsOutput,
anyhow::Error> {
       info!("Listing lambda functions");
       self.lambda_client
           .list_functions()
           .send()
           .await
           .map_err(anyhow::Error::from)
   }

   /** Invoke the lambda function using calculator InvokeArgs. */
   pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
       info!(?args, "Invoking {}", self.lambda_name);
       let payload = serde_json::to_string(&args)?;
       debug!(?payload, "Sending payload");
       self.lambda_client
           .invoke()
           .function_name(self.lambda_name.clone())
           .payload(Blob::new(payload))
           .send()
           .await
           .map_err(anyhow::Error::from)
   }

   /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
   pub async fn update_function_code(
       &self,
       zip_file: PathBuf,
```

```
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(updated)
    }
```

```
    /** Delete a function and its role, and if possible or necessary, its
associated code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };
```

```
        (delete_function, delete_role, delete_object)
    }

    pub async fn cleanup(
        &self,
        location: Option<String>,
    ) -> (
        (
            Result<DeleteFunctionOutput, anyhow::Error>,
            Result<DeleteRoleOutput, anyhow::Error>,
            Option<Result<DeleteObjectOutput, anyhow::Error>>,
        ),
        Option<Result<DeleteBucketOutput, anyhow::Error>>,
    ) {
        let delete_function = self.delete_function(location).await;

        let delete_bucket = if self.own_bucket {
            info!("Deleting bucket {}", self.bucket);
            if delete_function.2.is_none() ||
 delete_function.2.as_ref().unwrap().is_ok() {
                Some(
                    self.s3_client
                        .delete_bucket()
                        .bucket(self.bucket.clone())
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                None
            }
        } else {
            info!("No bucket to clean up");
            None
        };

        (delete_function, delete_bucket)
    }
}

/**
 * Testing occurs primarily as an integration test running the `scenario` bin
 successfully.
```

```
 * Each action relies deeply on the internal workings and state of Amazon Simple
 Storage Service (Amazon S3), Lambda, and IAM working together.
 * It is therefore infeasible to mock the clients to test the individual actions.
 */
#[cfg(test)]
mod test {
    use super::{InvokeArgs, Operation};
    use serde_json::json;

    /** Make sure that the JSON output of serializing InvokeArgs is what's
 expected by the calculator. */
    #[test]
    fn test_serialize() {
        assert_eq!(json!(InvokeArgs::Increment(5)), 5);
        assert_eq!(
            json!(InvokeArgs::Arithmetic(Operation::Plus, 5, 7)).to_string(),
            r#"{"op":"plus","i":5,"j":7}"#.to_string(),
        );
    }
}
```

Eine Binärdatei, um das Szenario vom Anfang bis Ende auszuführen, wobei Befehlszeilen-Flags verwendet werden, um einige Verhaltensweisen zu steuern. Diese Datei ist src/bin / scenario.rs in der Kiste.

```
/*
## Service actions

Service actions wrap the SDK call, taking a client and any specific parameters
 necessary for the call.

* CreateFunction
* GetFunction
* ListFunctions
* Invoke
* UpdateFunctionCode
* UpdateFunctionConfiguration
* DeleteFunction

## Scenario
```

```
A scenario runs at a command prompt and prints output to the user on the result
 of each service action. A scenario can run in one of two ways: straight through,
 printing out progress as it goes, or as an interactive question/answer script.

## Getting started with functions

Use an SDK to manage AWS Lambda functions: create a function, invoke it, update
 its code, invoke it again, view its output and logs, and delete it.

This scenario uses two Lambda handlers:
_Note: Handlers don't use AWS SDK API calls._

The increment handler is straightforward:

1. It accepts a number, increments it, and returns the new value.
2. It performs simple logging of the result.

The arithmetic handler is more complex:
1. It accepts a set of actions ['plus', 'minus', 'times', 'divided-by'] and two
 numbers, and returns the result of the calculation.
2. It uses an environment variable to control log level (such as DEBUG, INFO,
 WARNING, ERROR).
It logs a few things at different levels, such as:
    * DEBUG: Full event data.
    * INFO: The calculation result.
    * WARN~ING~: When a divide by zero error occurs.
    * This will be the typical `RUST_LOG` variable.


The steps of the scenario are:

1. Create an AWS Identity and Access Management (IAM) role that meets the
 following requirements:
    * Has an assume_role policy that grants 'lambda.amazonaws.com' the
 'sts:AssumeRole' action.
    * Attaches the 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' managed role.
    * _You must wait for ~10 seconds after the role is created before you can use
 it!_
2. Create a function (CreateFunction) for the increment handler by packaging it
 as a zip and doing one of the following:
    * Adding it with CreateFunction Code.ZipFile.
    * --or--
```

```
    * Uploading it to Amazon Simple Storage Service (Amazon S3) and adding it
 with CreateFunction Code.S3Bucket/S3Key.
    * _Note: Zipping the file does not have to be done in code._
    * If you have a waiter, use it to wait until the function is active.
 Otherwise, call GetFunction until State is Active.
3. Invoke the function with a number and print the result.
4. Update the function (UpdateFunctionCode) to the arithmetic handler by
 packaging it as a zip and doing one of the following:
    * Adding it with UpdateFunctionCode ZipFile.
    * --or--
    * Uploading it to Amazon S3 and adding it with UpdateFunctionCode S3Bucket/
S3Key.
5. Call GetFunction until Configuration.LastUpdateStatus is 'Successful' (or
 'Failed').
6. Update the environment variable by calling UpdateFunctionConfiguration and
 pass it a log level, such as:
    * Environment={'Variables': {'RUST_LOG': 'TRACE'}}
7. Invoke the function with an action from the list and a couple of values.
 Include LogType='Tail' to get logs in the result. Print the result of the
 calculation and the log.
8. [Optional] Invoke the function to provoke a divide-by-zero error and show the
 log result.
9. List all functions for the account, using pagination (ListFunctions).
10. Delete the function (DeleteFunction).
11. Delete the role.

Each step should use the function created in Service Actions to abstract calling
 the SDK.
 */

use aws_sdk_lambda::{operation::invoke::InvokeOutput, types::Environment};
use clap::Parser;
use std::{collections::HashMap, path::PathBuf};
use tracing::{debug, info, warn};
use tracing_subscriber::EnvFilter;

use lambda_code_examples::actions::{
    InvokeArgs::{Arithmetic, Increment},
    LambdaManager, Operation,
};

#[derive(Debug, Parser)]
pub struct Opt {
    /// The AWS Region.
```

```
    #[structopt(short, long)]
    pub region: Option<String>,

    // The bucket to use for the FunctionCode.
    #[structopt(short, long)]
    pub bucket: Option<String>,

    // The name of the Lambda function.
    #[structopt(short, long)]
    pub lambda_name: Option<String>,

    // The number to increment.
    #[structopt(short, long, default_value = "12")]
    pub inc: i32,

    // The left operand.
    #[structopt(long, default_value = "19")]
    pub num_a: i32,

    // The right operand.
    #[structopt(long, default_value = "23")]
    pub num_b: i32,

    // The arithmetic operation.
    #[structopt(short, long, default_value = "plus")]
    pub operation: Operation,

    #[structopt(long)]
    pub cleanup: Option<bool>,

    #[structopt(long)]
    pub no_cleanup: Option<bool>,
}

fn code_path(lambda: &str) -> PathBuf {
    PathBuf::from(format!("../target/lambda/{lambda}/bootstrap.zip"))
}

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
```

```
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}

async fn main_block(
    opt: &Opt,
    manager: &LambdaManager,
    code_location: String,
) -> Result<(), anyhow::Error> {
    let invoke = manager.invoke(Increment(opt.inc)).await?;
    log_invoke_output(&invoke, "Invoked function configured as increment");

    let update_code = manager
        .update_function_code(code_path("arithmetic"), code_location.clone())
        .await?;

    let code_sha256 = update_code.code_sha256().unwrap_or("Unknown SHA");
    info!(?code_sha256, "Updated function code with arithmetic.zip");

    let arithmetic_args = Arithmetic(opt.operation, opt.num_a, opt.num_b);
    let invoke = manager.invoke(arithmetic_args).await?;
    log_invoke_output(&invoke, "Invoked function configured as arithmetic");

    let update = manager
        .update_function_configuration(
            Environment::builder()
                .set_variables(Some(HashMap::from([(
                    "RUST_LOG".to_string(),
                    "trace".to_string(),
                )])))
                .build(),
        )
        .await?;
    let updated_environment = update.environment();
    info!(?updated_environment, "Updated function configuration");

    let invoke = manager
        .invoke(Arithmetic(opt.operation, opt.num_a, opt.num_b))
        .await?;
    log_invoke_output(
```

```
        &invoke,
        "Invoked function configured as arithmetic with increased logging",
    );

    let invoke = manager
        .invoke(Arithmetic(Operation::DividedBy, opt.num_a, 0))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with divide by zero",
    );

    Ok::<(), anyhow::Error>(())
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt()
        .without_time()
        .with_file(true)
        .with_line_number(true)
        .with_env_filter(EnvFilter::from_default_env())
        .init();

    let opt = Opt::parse();
    let manager = LambdaManager::load_from_env(opt.lambda_name.clone(),
 opt.bucket.clone()).await;

    let key = match manager.create_function(code_path("increment")).await {
        Ok(init) => {
            info!(?init, "Created function, initially with increment.zip");
            let run_block = main_block(&opt, &manager, init.clone()).await;
            info!(?run_block, "Finished running example, cleaning up");
            Some(init)
        }
        Err(err) => {
            warn!(?err, "Error happened when initializing function");
            None
        }
    };

    if Some(false) == opt.cleanup || Some(true) == opt.no_cleanup {
        info!("Skipping cleanup")
    } else {
```

```
        let delete = manager.cleanup(key).await;
        info!(?delete, "Deleted function & cleaned up resources");
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       "Create an AWS Identity and Access Management (IAM) role that grants AWS
Lambda permission to write to logs."
       DATA(lv_policy_document) = `{` &&
           `"Version":"2012-10-17",        ` &&
                 `"Statement": [` &&
                   `{` &&
                     `"Effect": "Allow",` &&
                     `"Action": [` &&
                       `"sts:AssumeRole"` &&
                     `],` &&
```

```
                      `"Principal": {` &&
                        `"Service": [` &&
                          `"lambda.amazonaws.com"` &&
                        `]` &&
                      `}` &&
                    `}` &&
                  `]` &&
                `}`.
        TRY.
            DATA(lo_create_role_output) = lo_iam->createrole(
                    iv_rolename = iv_role_name
                    iv_assumerolepolicydocument = lv_policy_document
                    iv_description = 'Grant lambda permission to write to
 logs' ).
            DATA(lv_role_arn) = lo_create_role_output->get_role( )->get_arn( ).
            MESSAGE 'IAM role created.' TYPE 'I'.
            WAIT UP TO 10 SECONDS.            " Make sure that the IAM role is
 ready for use. "
          CATCH /aws1/cx_iamentityalrdyexex.
            DATA(lo_role) = lo_iam->getrole( iv_rolename = iv_role_name ).
            lv_role_arn = lo_role->get_role( )->get_arn( ).
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iammalformedplydocex.
            MESSAGE 'Policy document in the request is malformed.' TYPE 'E'.
        ENDTRY.

        TRY.
            lo_iam->attachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Attached policy to the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.
```

```
        " Create a Lambda function and upload handler code. "
        " Lambda function performs 'increment' action on a number. "
        TRY.
            lo_lmd->createfunction(
                 iv_functionname = iv_function_name
                 iv_runtime = `python3.9`
                 iv_role = lv_role_arn
                 iv_handler = iv_handler
                 io_code = io_initial_zip_file
                 iv_description = 'AWS Lambda code example' ).
            MESSAGE 'Lambda function created.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        " Verify the function is in Active state "
        WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
          IF sy-index = 10.
            EXIT.               " Maximum 10 seconds. "
          ENDIF.
          WAIT UP TO 1 SECONDS.
        ENDWHILE.

        "Invoke the function with a single parameter and get results."
        TRY.
            DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "increment",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_initial_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
            ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
           " ov_initial_invoke_payload is returned for testing purposes. "
            DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
```

```
            CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT
 XML lo_writer_json.
            DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " Update the function code and configure its Lambda environment with an
 environment variable. "
        " Lambda function is updated to perform 'decrement' action also. "
        TRY.
            lo_lmd->updatefunctioncode(
                  iv_functionname = iv_function_name
                  iv_zipfile = io_updated_zip_file ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function code updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        TRY.
            DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
            DATA ls_variable LIKE LINE OF lt_variables.
            ls_variable-key = 'LOG_LEVEL'.
            ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
 'info' ).
            INSERT ls_variable INTO TABLE lt_variables.

            lo_lmd->updatefunctionconfiguration(
```

```
                  iv_functionname = iv_function_name
                  io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
 lt_variables ) ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourceconflictex.
            MESSAGE 'Resource already exists or another operation is in
 progress.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        "Invoke the function with new parameters and get results. Display the
 execution log that's returned from the invocation."
        TRY.
            lv_json = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "decrement",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_updated_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
            ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
           " ov_updated_invoke_payload is returned for testing purposes. "
            lo_writer_json = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT
 XML lo_writer_json.
            lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
```

```
        ENDTRY.

        " List the functions for your account. "
        TRY.
            DATA(lo_list_output) = lo_lmd->listfunctions( ).
            DATA(lt_functions) = lo_list_output->get_functions( ).
            MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        ENDTRY.

        " Delete the Lambda function. "
        TRY.
            lo_lmd->deletefunction( iv_functionname = iv_function_name ).
            MESSAGE 'Lambda function deleted.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'W'.
        ENDTRY.

        " Detach role policy. "
        TRY.
            lo_iam->detachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Detached policy from the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

        " Delete the IAM role. "
        TRY.
            lo_iam->deleterole( iv_rolename = iv_role_name ).
            MESSAGE 'IAM role deleted.' TYPE 'I'.
```

```
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

      CATCH /aws1/cx_rt_service_generic INTO lo_exception.
        DATA(lv_error) = lo_exception->get_longtext( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie die erste Lambda-Funktion, die einfach den angegebenen Wert erhöht.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
```

```
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "increment",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "increment",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
```

```
}

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function body.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let incrementLambdaRuntime = LambdaRuntime {
        (event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?

    if action != "increment" {
        context.logger.error("Unrecognized operation: \"\(action)\". The only
 supported action is \"increment\".")
    } else {
        answer = event.number + 1
        context.logger.info("The calculated answer is \(answer!).")
    }

    let response = Response(answer: answer)
    return response
}

// Run the Lambda runtime code.

try await incrementLambdaRuntime.run()
```

Definieren Sie die zweite Lambda-Funktion, die eine arithmetische Operation mit zwei Zahlen ausführt.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "calculator",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "calculator",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
import AWSLambdaRuntime
```

```
/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

/// A dictionary mapping operation names to closures that perform that
/// operation and return the result.
let actions = [
    "plus": { (x: Int, y: Int) -> Int in
        return x + y
    },
    "minus": { (x: Int, y: Int) -> Int in
        return x - y
    },
    "times": { (x: Int, y: Int) -> Int in
        return x * y
    },
    "divided-by": { (x: Int, y: Int) -> Int in
        return x / y
    }
]

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function's entry point. Called by the Lambda runtime.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
```

```
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let calculatorLambdaRuntime = LambdaRuntime {
        (_ event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?
    var actionFunc: ((Int, Int) -> Int)?

    // Get the closure to run to perform the calculation.

    actionFunc = await actions[action]

    guard let actionFunc else {
        context.logger.error("Unrecognized operation '\(action)\'")
        return Response(answer: nil)
    }

    // Perform the calculation and return the answer.

    answer = actionFunc(event.x, event.y)

    guard let answer else {
        context.logger.error("Error computing \(event.x) \(action) \(event.y)")
    }
    context.logger.info("\(event.x) \(action) \(event.y) = \(answer)")

    return Response(answer: answer)
}

try await calculatorLambdaRuntime.run()
```

Definieren Sie das Hauptprogramm, das die beiden Lambda-Funktionen aufruft.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription
```

```
let package = Package(
    name: "lambda-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "lambda-basics",
            dependencies: [
                .product(name: "AWSLambda", package: "aws-sdk-swift"),
                .product(name: "AWSIAM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources"
        )
    ]
)

//
/// An example demonstrating a variety of important AWS Lambda functions.

import ArgumentParser
import AWSIAM
import SmithyWaitersAPI
import AWSClientRuntime
import AWSLambda
import Foundation
```

```
/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct IncrementRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

struct Response: Encodable, Decodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}

struct CalculatorRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

let exampleName = "SwiftLambdaRoleExample"
let basicsFunctionName = "lambda-basics-function"

/// The ARN of the standard IAM policy for execution of Lambda functions.
let policyARN = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Option(help: "Name of the IAM Role to use for the Lambda functions")
    var role = exampleName
    @Option(help: "Zip archive containing the 'increment' lambda function")
    var incpath: String
    @Option(help: "Zip archive containing the 'calculator' lambda function")
    var calcpath: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
```

```
        commandName: "lambda-basics",
        abstract: """
        This example demonstrates several common operations using AWS Lambda.
        """,
        discussion: """
        """
    )

    /// Returns the specified IAM role object.
    ///
    /// - Parameters:
    ///   - iamClient: `IAMClient` to use when looking for the role.
    ///   - roleName: The name of the role to check.
    ///
    /// - Returns: The `IAMClientTypes.Role` representing the specified role.
    func getRole(iamClient: IAMClient, roleName: String) async throws
                 -> IAMClientTypes.Role {
        do {
            let roleOutput = try await iamClient.getRole(
                input: GetRoleInput(
                    roleName: roleName
                )
            )

            guard let role = roleOutput.role else {
                throw ExampleError.roleNotFound
            }
            return role
        } catch {
            throw ExampleError.roleNotFound
        }
    }

    /// Create the AWS IAM role that will be used to access AWS Lambda.
    ///
    /// - Parameters:
    ///   - iamClient: The AWS `IAMClient` to use.
    ///   - roleName: The name of the AWS IAM role to use for Lambda.
    ///
    /// - Throws: `ExampleError.roleCreateError`
    ///
    /// - Returns: The `IAMClientTypes.Role` struct that describes the new role.
    func createRoleForLambda(iamClient: IAMClient, roleName: String) async throws
-> IAMClientTypes.Role {
```

```
        let output = try await iamClient.createRole(
            input: CreateRoleInput(
                assumeRolePolicyDocument:
                """
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"Service": "lambda.amazonaws.com"},
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
                """,
                roleName: roleName
            )
        )

        guard let role = output.role else {
            throw ExampleError.roleCreateError
        }

        // Wait for the role to be ready for use.

        _ = try await iamClient.waitUntilRoleExists(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetRoleInput(roleName: roleName)
        )

        return role
    }

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
```

```
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async
-> Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }

    /// Create the specified AWS Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to create.
    ///   - roleArn: The ARN of the role to apply to the function.
    ///   - path: The path of the Zip archive containing the function.
    ///
    /// - Returns: `true` if the AWS Lambda was successfully created; `false`
    ///   if it wasn't.
    func createFunction(lambdaClient: LambdaClient, functionName: String,
                                roleArn: String?, path: String) async throws ->
Bool {
        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
```

```
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }

        // Wait for a while to be sure the function is done being created.

        let output = try await lambdaClient.waitUntilFunctionActiveV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(functionName: functionName)
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Update the AWS Lambda function with new code to run when the function
    /// is invoked.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to update.
    ///   - path: The pathname of the Zip file containing the packaged Lambda
    ///     function.
    /// - Throws: `ExampleError.zipFileReadError`
    /// - Returns: `true` if the function's code is updated successfully.
    ///   Otherwise, returns `false`.
    func updateFunctionCode(lambdaClient: LambdaClient, functionName: String,
                            path: String) async throws -> Bool {
        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data

        // Read the function's Zip file.
```

```
        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }

        let output = try await lambdaClient.waitUntilFunctionUpdatedV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(
                functionName: functionName
            )
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
```

```
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String)
 async throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
```

```
        }
    }

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
 Int {
        do {
```

```
            let incRequest = IncrementRequest(action: "increment", number:
number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }

    /// Invoke the calculator Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - action: Which arithmetic operation to perform: "plus", "minus",
    ///     "times", or "divided-by".
    ///   - x: The first number to use in the computation.
    ///   - y: The second number to use in the computation.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: The computed answer as an `Int`.
    func invokeCalculator(lambdaClient: LambdaClient, action: String, x: Int, y:
Int) async throws -> Int {
        do {
            let calcRequest = CalculatorRequest(action: action, x: x, y: y)
            let calcData = try! JSONEncoder().encode(calcRequest)
```

```
            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: calcData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }

    }

    /// Perform the example's tasks.
    func basics() async throws {
        let iamClient = try await IAMClient(
            config: IAMClient.IAMClientConfiguration(region: region)
        )

        let lambdaClient = try await LambdaClient(
            config: LambdaClient.LambdaClientConfiguration(region: region)
        )

        /// The IAM role to use for the example.
        var iamRole: IAMClientTypes.Role

        // Look for the specified role. If it already exists, use it. If not,
        // create it and attach the desired policy to it.

        do {
            iamRole = try await getRole(iamClient: iamClient, roleName: role)
        } catch ExampleError.roleNotFound {
            // The role wasn't found, so create it and attach the needed
            // policy.
```

```
            iamRole = try await createRoleForLambda(iamClient: iamClient,
roleName: role)

            do {
                _ = try await iamClient.attachRolePolicy(
                    input: AttachRolePolicyInput(policyArn: policyARN, roleName:
role)
                )
            } catch {
                throw ExampleError.policyError
            }
        }

        // Give the policy time to attach to the role.

        sleep(5)

        // Look to see if the function already exists. If it does, throw an
        // error.

        if await doesLambdaFunctionExist(lambdaClient: lambdaClient, name:
basicsFunctionName) {
            throw ExampleError.functionAlreadyExists
        }

        // Create, then invoke, the "increment" version of the calculator
        // function.

        print("Creating the increment Lambda function...")
        if try await createFunction(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    roleArn: iamRole.arn, path: incpath) {
            print("Running increment function calls...")
            for number in 0...4 {
                do {
                    let answer = try await invokeIncrement(lambdaClient:
lambdaClient, number: number)
                    print("Increment \(number) = \(answer)")
                } catch {
                    print("Error incrementing \(number): ",
error.localizedDescription)
                }
            }
```

```
        } else {
            print("*** Failed to create the increment function.")
        }

        // Enable debug logging.

        print("\nEnabling debug logging...")
        try await enableDebugLogging(lambdaClient: lambdaClient, functionName:
basicsFunctionName)

        // Change it to a basic arithmetic calculator. Then invoke it a few
        // times.

        print("\nReplacing the Lambda function with a calculator...")

        if try await updateFunctionCode(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    path: calcpath) {
            print("Running calculator function calls...")
            for x in [6, 10] {
                for y in [2, 4] {
                    for action in ["plus", "minus", "times", "divided-by"] {
                        do {
                            let answer = try await invokeCalculator(lambdaClient:
lambdaClient, action: action, x: x, y: y)
                            print("\(x) \(action) \(y) = \(answer)")
                        } catch {
                            print("Error calculating \(x) \(action) \(y): ",
error.localizedDescription)
                        }
                    }
                }
            }
        }

        // List all lambda functions.

        let functionNames = try await getFunctionNames(lambdaClient:
lambdaClient)

        if functionNames.count > 0 {
            print("\nAWS Lambda functions available on your account:")
            for name in functionNames {
                print("  \(name)")
```

```
            }
        }

        // Delete the lambda function.

        print("Deleting lambda function...")

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }

        // Detach the role from the policy, then delete the role.

        print("Deleting the AWS IAM role...")

        do {
            _ = try await iamClient.detachRolePolicy(
                input: DetachRolePolicyInput(
                    policyArn: policyARN,
                    roleName: role
                )
            )
            _ = try await iamClient.deleteRole(
                input: DeleteRoleInput(
                    roleName: role
                )
            )
        } catch {
            throw ExampleError.deleteRoleError
        }
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
```

```
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.basics()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}


/// Errors thrown by the example's functions.
enum ExampleError: Error {
    /// An AWS Lambda function with the specified name already exists.
    case functionAlreadyExists
    /// The specified role doesn't exist.
    case roleNotFound
    /// Unable to create the role.
    case roleCreateError
    /// Unable to delete the role.
    case deleteRoleError
    /// Unable to attach a policy to the role.
    case policyError
    /// Unable to get the executable directory.
    case executableNotFound
    /// An error occurred creating a lambda function.
    case createLambdaError
    /// An error occurred invoking the lambda function.
    case invokeError
    /// No answer received from the invocation.
    case noAnswerReceived
    /// Unable to list the AWS Lambda functions.
    case listFunctionsError
    /// Unable to update the AWS Lambda function.
    case updateFunctionError
    /// Unable to update the function configuration.
    case updateFunctionConfigurationError
    /// The environment response is missing after an
    /// UpdateEnvironmentConfiguration attempt.
    case environmentResponseMissingError
    /// The environment variables are missing from the EnvironmentResponse and
    /// no errors occurred.
```

```
    case environmentVariablesMissingError
    /// The log level is incorrect after attempting to set it.
    case logLevelIncorrectError
    /// Unable to load the AWS Lambda function's Zip file.
    case zipFileReadError

    var errorDescription: String? {
        switch self {
        case .functionAlreadyExists:
            return "An AWS Lambda function with that name already exists."
        case .roleNotFound:
            return "The specified role doesn't exist."
        case .deleteRoleError:
            return "Unable to delete the AWS IAM role."
        case .roleCreateError:
            return "Unable to create the specified role."
        case .policyError:
            return "An error occurred attaching the policy to the role."
        case .executableNotFound:
            return "Unable to find the executable program directory."
        case .createLambdaError:
            return "An error occurred creating a lambda function."
        case .invokeError:
            return "An error occurred invoking a lambda function."
        case .noAnswerReceived:
            return "No answer received from the lambda function."
        case .listFunctionsError:
            return "Unable to list the AWS Lambda functions."
        case .updateFunctionError:
            return "Unable to update the AWS lambda function."
        case .updateFunctionConfigurationError:
            return "Unable to update the AWS lambda function configuration."
        case .environmentResponseMissingError:
            return "The environment is missing from the response after updating
the function configuration."
        case .environmentVariablesMissingError:
            return "While no error occurred, no environment variables were
returned following function configuration."
        case .logLevelIncorrectError:
            return "The log level is incorrect after attempting to set it to
DEBUG."
        case .zipFileReadError:
            return "Unable to read the AWS Lambda function."
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen für Lambda mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Lambda-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Lambda-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Lambda AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Lambda -API-Referenz.

Beispiele

- Verwendung von CreateAlias mit einer CLI
- Verwenden Sie mit einem CreateFunction AWS SDK oder CLI
- Verwendung von DeleteAlias mit einer CLI
- Verwenden Sie mit einem DeleteFunction AWS SDK oder CLI
- Verwendung von DeleteFunctionConcurrency mit einer CLI
- Verwendung von DeleteProvisionedConcurrencyConfig mit einer CLI
- Verwendung von GetAccountSettings mit einer CLI

- Verwendung von GetAlias mit einer CLI
- Verwenden Sie mit einem GetFunction AWS SDK oder CLI
- Verwendung von GetFunctionConcurrency mit einer CLI
- Verwendung von GetFunctionConfiguration mit einer CLI
- Verwendung von GetPolicy mit einer CLI
- Verwendung von GetProvisionedConcurrencyConfig mit einer CLI
- Verwenden Sie Invoke mit einem AWS SDK oder CLI
- Verwenden Sie ListFunctionsmit einem AWS SDK oder CLI
- Verwendung von ListProvisionedConcurrencyConfigs mit einer CLI
- Verwendung von ListTags mit einer CLI
- Verwendung von ListVersionsByFunction mit einer CLI
- Verwendung von PublishVersion mit einer CLI
- Verwendung von PutFunctionConcurrency mit einer CLI
- Verwendung von PutProvisionedConcurrencyConfig mit einer CLI
- Verwendung von RemovePermission mit einer CLI
- Verwendung von TagResource mit einer CLI
- Verwendung von UntagResource mit einer CLI
- Verwendung von UpdateAlias mit einer CLI
- Verwenden Sie mit einem UpdateFunctionCode AWS SDK oder CLI
- Verwenden Sie UpdateFunctionConfigurationmit einem AWS SDK oder CLI

Verwendung von **`CreateAlias`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAlias` verwendet wird.

CLI

AWS CLI

So erstellen Sie einen Alias für eine Lambda-Funktion

Im folgenden Beispiel für `create-alias` wird ein Alias namens `LIVE` erstellt, der auf Version 1 der `my-function`-Lambda-Funktion verweist.

```
aws lambda create-alias \
    --function-name my-function \
    --description "alias for live version of function" \
    --function-version 1 \
    --name LIVE
```

Ausgabe:

```
{
    "FunctionVersion": "1",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",
    "Description": "alias for live version of function"
}
```

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz CreateAlias.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neuer Lambda-Alias für die angegebene Version und Routing-Konfiguration erstellt, um den Prozentsatz der Aufrufe für dieses System anzugeben.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias
 for version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Einzelheiten zur API finden Sie unter CreateAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer Lambda-Alias für die angegebene Version und Routing-Konfiguration erstellt, um den Prozentsatz der Aufrufe für dieses System anzugeben.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias
 for version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- Einzelheiten zur API finden Sie unter CreateAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateFunction`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateFunction` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function.</param>
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
```

```
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };

        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same
 architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
```

```
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                                buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome =
 client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " <<
 seconds
                      << " seconds elapsed." << std::endl;
            break;
        }

        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Lambda-Funktion

Im folgenden Beispiel für `create-function` wird eine Lambda-Funktion mit dem Namen `my-function` erstellt.

```
aws lambda create-function \
    --function-name my-function \
    --runtime nodejs22.x \
    --zip-file fileb://my-function.zip \
    --handler my-function.handler \
    --role arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
tges6bf4
```

Inhalt von `my-function.zip`:

```
This file is a deployment package that contains your function code and any
 dependencies.
```

Ausgabe:

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "PFn4S+er27qk+UuZSTKEQfNKG/XNn7QJs90mJgq6oH8=",
    "FunctionName": "my-function",
    "CodeSize": 308,
    "RevisionId": "873282ed-4cd3-4dc8-a069-d0c647e470c6",
    "MemorySize": 128,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
zgur6bf4",
    "Timeout": 3,
    "LastModified": "2025-10-14T22:26:11.234+0000",
    "Handler": "my-function.handler",
```

```
    "Runtime": "nodejs22.x",
    "Description": ""
}
```

Weitere Informationen finden Sie unter Konfigurieren des Speichers der Lambda-Funktion im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie CreateFunctionin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
```

```
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
 awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
 waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);
```

```
  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }
```

```
    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn
    }
}
```

- Einzelheiten zur API finden Sie CreateFunctionin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
 $bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
```

```
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue C#-Funktion (dotnetcore1.0 Runtime) mit dem Namen AWS Lambda erstellt, die die kompilierten Binärdateien für die Funktion aus einer ZIP-Datei MyFunction im lokalen Dateisystem bereitstellt (relative oder absolute Pfade können verwendet werden). C#-Lambda-Funktionen spezifizieren den Handler für die Funktion mit der Bezeichnung AssemblyName:::Namespace.ClassName:. MethodName Sie sollten die Teile Assembly-Name (ohne .dll-Suffix), Namespace, Klassenname und Methodenname der Handler-Spezifikation entsprechend ersetzen. Für die neue Funktion werden die Umgebungsvariablen 'envvar1' und 'envvar2' aus den bereitgestellten Werten eingerichtet.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Ausgabe:

```
CodeSha256       : /NgBMd...gq71I=
CodeSize         : 214784
DeadLetterConfig :
```

```
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, mit dem Unterschied, dass die Funktions-Binärdateien zunächst in einen Amazon S3-Bucket hochgeladen werden (der sich in derselben Region befinden muss wie die vorgesehene Lambda-Funktion) und das resultierende S3-Objekt dann bei der Erstellung der Funktion referenziert wird.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -
File .\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Einzelheiten zur API finden Sie unter CreateFunction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue C#-Funktion (dotnetcore1.0 Runtime) mit dem Namen AWS Lambda erstellt, die die kompilierten Binärdateien für die Funktion aus einer ZIP-Datei MyFunction im lokalen Dateisystem bereitstellt (relative oder absolute Pfade können verwendet werden). C#-Lambda-Funktionen spezifizieren den Handler für die Funktion mit der Bezeichnung AssemblyName:::Namespace.ClassName:. MethodName Sie sollten die Teile Assembly-Name (ohne .dll-Suffix), Namespace, Klassenname und Methodenname der Handler-Spezifikation entsprechend ersetzen. Für die neue Funktion

werden die Umgebungsvariablen 'envvar1' und 'envvar2' aus den bereitgestellten Werten eingerichtet.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Ausgabe:

```
CodeSha256       : /NgBMd...gq71I=
CodeSize         : 214784
DeadLetterConfig :
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, mit dem Unterschied, dass die Funktions-Binärdateien zunächst in einen Amazon S3-Bucket hochgeladen werden (der sich in derselben Region befinden muss wie die vorgesehene Lambda-Funktion) und das resultierende S3-Objekt dann bei der Erstellung der Funktion referenziert wird.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -
File .\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
```

```
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- Einzelheiten zur API finden Sie unter CreateFunction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function.
 This
                              must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the
 function
                                     code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
```

```
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- Einzelheiten zur API finden Sie CreateFunctionin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
```

```
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name:
function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
```

```
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Einzelheiten zur API finden Sie CreateFunctionin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_lmd->createfunction(
           iv_functionname = iv_function_name
           iv_runtime = `python3.9`
           iv_role = iv_role_arn
           iv_handler = iv_handler
           io_code = io_zip_file
           iv_description = 'AWS Lambda code example' ).
       MESSAGE 'Lambda function created.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateFunctionin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }
```

- Einzelheiten zur API finden Sie CreateFunctionin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteAlias`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAlias` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Alias einer Lambda-Funktion

Im folgenden Beispiel für `delete-alias` wird der Alias mit dem Namen `LIVE` aus der `my-function`-Lambda-Funktion gelöscht.

```
aws lambda delete-alias \
    --function-name my-function \
    --name LIVE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz DeleteAlias.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die im Befehl erwähnte Lambda-Funktion Alias gelöscht.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Einzelheiten zur API finden Sie unter DeleteAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die im Befehl erwähnte Lambda-Funktion Alias gelöscht.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- Einzelheiten zur API finden Sie unter DeleteAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteFunction`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFunction` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</
returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
```

```
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::DeleteFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);

    Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
            request);

    if (outcome.IsSuccess()) {
```

```
        std::cout << "The lambda function was successfully deleted." <<
 std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So löschen Sie eine Lambda-Funktion anhand des Funktionsnamens

Im folgenden Beispiel für `delete-function` wird die Lambda-Funktion `my-function` durch Angabe des Funktionsnamens gelöscht.

```
aws lambda delete-function \
    --function-name my-function
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: Eine Lambda-Funktion anhand des Funktions-ARN löschen

Im folgenden Beispiel für `delete-function` wird die Lambda-Funktion `my-function` durch Angabe des ARN der Funktion gelöscht.

```
aws lambda delete-function \
    --function-name arn:aws:lambda:us-west-2:123456789012:function:my-function
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: Eine Lambda-Funktion anhand eines teilweisen Funktions-ARN löschen

Im folgenden Beispiel für `delete-function` wird die Lambda-Funktion `my-function` durch Angabe des teilweisen ARN der Funktion gelöscht.

```
aws lambda delete-function \
    --function-name 123456789012:function:my-function
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Konfigurieren von AWS -Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an AWS Lambda function.
    *
    * @param awsLambda     an instance of the {@link LambdaClient} class, which
 is used to interact with the AWS Lambda service
    * @param functionName  the name of the Lambda function to be deleted
    *
    * @throws LambdaException if an error occurs while deleting the Lambda
 function
    */
   public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
       try {
           DeleteFunctionRequest request = DeleteFunctionRequest.builder()
               .functionName(functionName)
               .build();
```

```
            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Einzelheiten zur API finden Sie DeleteFunctionin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
```

```
        ]);
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine bestimmte Version einer Lambda-Funktion gelöscht

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Einzelheiten zur API finden Sie unter DeleteFunction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine bestimmte Version einer Lambda-Funktion gelöscht

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- Einzelheiten zur API finden Sie unter DeleteFunction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
```

```
        self.iam_resource = iam_resource


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

- Einzelheiten zur API finden Sie DeleteFunctionin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end
```

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /** Delete a function and its role, and if possible or necessary, its
associated code object and bucket. */
   pub async fn delete_function(
       &self,
       location: Option<String>,
   ) -> (
       Result<DeleteFunctionOutput, anyhow::Error>,
       Result<DeleteRoleOutput, anyhow::Error>,
       Option<Result<DeleteObjectOutput, anyhow::Error>>,
   ) {
       info!("Deleting lambda function {}", self.lambda_name);
       let delete_function = self
           .lambda_client
           .delete_function()
           .function_name(self.lambda_name.clone())
           .send()
           .await
```

```
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_lmd->deletefunction( iv_functionname = iv_function_name ).
        MESSAGE 'Lambda function deleted.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteFunctionin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
```

```
        } catch {
            print("Error: Unable to delete the function.")
        }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteFunctionConcurrency`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFunctionConcurrency` verwendet wird.

CLI

AWS CLI

So entfernen Sie das Limit für die reservierte gleichzeitige Ausführung einer Funktion

Im folgenden `delete-function-concurrency`-Beispiel wird das reservierte Limit für gleichzeitige Ausführung aus der `my-function`-Funktion gelöscht.

```
aws lambda delete-function-concurrency \
    --function-name  my-function
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Gleichzeitigkeit für eine Lambda-Funktion reservieren im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteFunctionConcurrencyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Funktionsgleichzeitigkeit der Lambda-Funktion aufgehoben.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter DeleteFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Funktionsgleichzeitigkeit der Lambda-Funktion aufgehoben.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter DeleteFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteProvisionedConcurrencyConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteProvisionedConcurrencyConfig` verwendet wird.

CLI

AWS CLI

So löschen Sie eine bereitgestellte Gleichzeitigkeitskonfiguration

Das folgende `delete-provisioned-concurrency-config`-Beispiel löscht die bereitgestellte Gleichzeitigkeitskonfiguration für den `GREEN`-Alias der angegebenen Funktion.

```
aws lambda delete-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier GREEN
```

- Einzelheiten zur API finden Sie DeleteProvisionedConcurrencyConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für einen bestimmten Alias entfernt.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

- Einzelheiten zur API finden Sie unter DeleteProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für einen bestimmten Alias entfernt.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

- Einzelheiten zur API finden Sie unter DeleteProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetAccountSettings`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetAccountSettings` verwendet wird.

CLI

AWS CLI

Um Details zu Ihrem Konto in einer AWS Region abzurufen

Im folgenden `get-account-settings`-Beispiel werden die Lambda-Grenzwerte und Nutzungsinformationen für Ihr Konto angezeigt.

```
aws lambda get-account-settings
```

Ausgabe:

```
{
    "AccountLimit": {
       "CodeSizeUnzipped": 262144000,
       "UnreservedConcurrentExecutions": 1000,
       "ConcurrentExecutions": 1000,
       "CodeSizeZipped": 52428800,
       "TotalCodeSize": 80530636800
    },
    "AccountUsage": {
```

```
        "FunctionCount": 4,
        "TotalCodeSize": 9426
    }
}
```

Weitere Informationen finden Sie unter AWS Lambda Limits im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetAccountSettingsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel wird angezeigt, um das Kontolimit und die Kontonutzung zu vergleichen

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Ausgabe:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Einzelheiten zur API finden Sie unter GetAccountSettings AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel wird angezeigt, um das Kontolimit und die Kontonutzung zu vergleichen

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Ausgabe:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- Einzelheiten zur API finden Sie unter GetAccountSettings AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetAlias`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetAlias` verwendet wird.

CLI

AWS CLI

So rufen Sie Details über einen Funktionsalias ab

Das folgende `get-alias`-Beispiel zeigt Details für den Alias namens `LIVE` für die `my-function`-Lambda-Funktion an.

```
aws lambda get-alias \
    --function-name my-function \
    --name LIVE
```

Ausgabe:

```
{
    "FunctionVersion": "3",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",
    "Description": "alias for live version of function"
}
```

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz GetAlias.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Gewichtungen der Routing-Konfiguration für einen bestimmten Lambda-Funktionsalias abgerufen.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Ausgabe:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Einzelheiten zur API finden Sie unter GetAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Gewichtungen der Routing-Konfiguration für einen bestimmten Lambda-Funktionsalias abgerufen.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Ausgabe:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- Einzelheiten zur API finden Sie unter GetAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetFunction`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetFunction` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen Sie Ihre erste serverlose Funktion

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string
functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Lambda::LambdaClient client(clientConfig);

       Aws::Lambda::Model::GetFunctionRequest request;
       request.SetFunctionName(functionName);

       Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

       if (outcome.IsSuccess()) {
           std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                     << std::endl;
       }
       else {
           std::cerr << "Error with Lambda::GetFunction. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
       }
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen über eine Funktion ab

Das folgende Beispiel für `get-function` zeigt Informationen zur Funktion `my-function` an.

```
aws lambda get-function \
    --function-name  my-function
```

Ausgabe:

```
{
    "Concurrency": {
        "ReservedConcurrentExecutions": 100
    },
    "Code": {
        "RepositoryType": "S3",
        "Location": "https://awslambda-us-west-2-tasks.s3.us-
west-2.amazonaws.com/snapshots/123456789012/my-function..."
    },
    "Configuration": {
        "TracingConfig": {
            "Mode": "PassThrough"
        },
        "Version": "$LATEST",
        "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
        "FunctionName": "my-function",
        "VpcConfig": {
            "SubnetIds": [],
            "VpcId": "",
            "SecurityGroupIds": []
        },
        "MemorySize": 128,
        "RevisionId": "28f0fb31-5c5c-43d3-8955-03e76c5c1075",
        "CodeSize": 304,
        "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
        "Handler": "index.handler",
        "Role": "arn:aws:iam::123456789012:role/service-role/helloWorldPython-
role-uy3l9qyq",
        "Timeout": 3,
```

```
        "LastModified": "2025-09-24T18:20:35.054+0000",
        "Runtime": "nodejs22.x",
        "Description": ""
    }
}
```

Weitere Informationen finden Sie unter Konfigurieren des Speichers der Lambda-Funktion im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetFunctionin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName
 string) types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx,
 &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which
 is used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
 information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
```

```
                .functionName(functionName)
                .build();

            GetFunctionResponse response =
 awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
 response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new GetFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def get_function(self, function_name):
        """
        Gets data about a Lambda function.
```

```
        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response =
 self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- Einzelheiten zur API finden Sie GetFunctionin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
```

```
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error>
{
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from)
    }
```

- Einzelheiten zur API finden Sie GetFunctionin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).
   " oo_result is returned for testing purposes. "
       MESSAGE 'Lambda function information retrieved.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetFunctionin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async
 -> Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }
```

- Einzelheiten zur API finden Sie GetFunctionin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`GetFunctionConcurrency`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetFunctionConcurrency` verwendet wird.

CLI

AWS CLI

So zeigen Sie die reservierte Gleichzeitigkeitseinstellung für eine Funktion an

Das folgende `get-function-concurrency`-Beispiel ruft die reservierte Gleichzeitigkeitseinstellung für die angegebene Funktion ab.

```
aws lambda get-function-concurrency \
```

```
    --function-name my-function
```

Ausgabe:

```
{
    "ReservedConcurrentExecutions": 250
}
```

- Einzelheiten zur API finden Sie GetFunctionConcurrencyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel ruft die reservierte Gleichzeitigkeit für die Lambda-Funktion ab

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Ausgabe:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Einzelheiten zur API finden Sie unter GetFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel ruft die reservierte Gleichzeitigkeit für die Lambda-Funktion ab

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Ausgabe:

```
ReservedConcurrentExecutions
----------------------------
100
```

- Einzelheiten zur API finden Sie unter GetFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetFunctionConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetFunctionConfiguration` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen Sie Ihre erste serverlose Funktion

CLI

AWS CLI

So rufen Sie die versionsspezifischen Einstellungen einer Lambda-Funktion ab

Das folgende `get-function-configuration`-Beispiel zeigt die Einstellungen für Version 2 der `my-function`-Funktion.

```
aws lambda get-function-configuration \
    --function-name  my-function:2
```

Ausgabe:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
    "Version": "2",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
```

```
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:2",
    "Handler": "index.handler"
}
```

Weitere Informationen finden Sie unter Konfigurieren von AWS -Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetFunctionConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt die versionsspezifische Konfiguration einer Lambda-Funktion zurück.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Ausgabe:

```
CodeSha256                 : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                   : 1426
DeadLetterConfig           : Amazon.Lambda.Model.DeadLetterConfig
Description                : Verson 3 to test Aliases
Environment                : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                             :PowershellAlias
FunctionName               : MylambdaFunction123
Handler                    : lambda_function.launch_instance
KMSKeyArn                  :
LastModified               : 2019-12-25T09:52:59.872+0000
LastUpdateStatus           : Successful
LastUpdateStatusReason     :
```

```
LastUpdateStatusReasonCode :
Layers                     : {}
MasterArn                  :
MemorySize                 : 128
RevisionId                 : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                       : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                    : python3.8
State                      : Active
StateReason                :
StateReasonCode            :
Timeout                    : 600
TracingConfig              : Amazon.Lambda.Model.TracingConfigResponse
Version                    : 4
VpcConfig                  : Amazon.Lambda.Model.VpcConfigDetail
```

- Einzelheiten zur API finden Sie unter GetFunctionConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt die versionsspezifische Konfiguration einer Lambda-Funktion zurück.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Ausgabe:

```
CodeSha256                 : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                   : 1426
DeadLetterConfig           : Amazon.Lambda.Model.DeadLetterConfig
Description                : Verson 3 to test Aliases
Environment                : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
                             :PowershellAlias
FunctionName               : MylambdaFunction123
Handler                    : lambda_function.launch_instance
KMSKeyArn                  :
LastModified               : 2019-12-25T09:52:59.872+0000
LastUpdateStatus           : Successful
LastUpdateStatusReason     :
LastUpdateStatusReasonCode :
```

```
Layers                      : {}
MasterArn                   :
MemorySize                  : 128
RevisionId                  : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                        : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                     : python3.8
State                       : Active
StateReason                 :
StateReasonCode             :
Timeout                     : 600
TracingConfig               : Amazon.Lambda.Model.TracingConfigResponse
Version                     : 4
VpcConfig                   : Amazon.Lambda.Model.VpcConfigDetail
```

- Einzelheiten zur API finden Sie unter GetFunctionConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPolicy`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPolicy` verwendet wird.

CLI

AWS CLI

So rufen Sie die ressourcenbasierte IAM-Richtlinie für eine Funktion, eine Version oder einen Alias ab

Das folgende `get-policy`-Beispiel zeigt Richtlinieninformationen über die `my-function`-Lambda-Funktion an.

```
aws lambda get-policy \
    --function-name my-function
```

Ausgabe:

```
{
    "Policy": {
        "Version":"2012-10-17",
        "Id":"default",
        "Statement":
        [
```

```
            {
                "Sid":"iot-events",
                "Effect":"Allow",
                "Principal": {"Service":"iotevents.amazonaws.com"},
                "Action":"lambda:InvokeFunction",
                "Resource":"arn:aws:lambda:us-west-2:123456789012:function:my-
function"
            }
        ]
    },
    "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668"
}
```

Weitere Informationen finden Sie unter Using Resource-based Policies for AWS Lambda im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter GetPolicy AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie der Lambda-Funktion angezeigt

```
Get-LMPolicy -FunctionName test -Select Policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lamb
east-1:123456789102:function:test"}]}
```

- Einzelheiten zur API finden Sie unter GetPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie der Lambda-Funktion angezeigt

```
Get-LMPolicy -FunctionName test -Select Policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lamb
east-1:123456789102:function:test"}]}
```

- Einzelheiten zur API finden Sie unter GetPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetProvisionedConcurrencyConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetProvisionedConcurrencyConfig` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine bereitgestellte Gleichzeitigkeitskonfiguration an

Im folgenden `get-provisioned-concurrency-config`-Beispiel werden Details für die bereitgestellte Gleichzeitigkeitskonfiguration für den `BLUE`-Alias der angegebenen Funktion angezeigt.

```
aws lambda get-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier BLUE
```

Ausgabe:

```
{
    "RequestedProvisionedConcurrentExecutions": 100,
    "AvailableProvisionedConcurrentExecutions": 100,
    "AllocatedProvisionedConcurrentExecutions": 100,
    "Status": "READY",
    "LastModified": "2019-12-31T20:28:49+0000"
}
```

- Einzelheiten zur API finden Sie GetProvisionedConcurrencyConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für den angegebenen Alias der Lambda-Funktion abgerufen.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Ausgabe:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Einzelheiten zur API finden Sie unter GetProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für den angegebenen Alias der Lambda-Funktion abgerufen.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Ausgabe:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- Einzelheiten zur API finden Sie unter GetProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie Invoke mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Invoke` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Erstellen Sie Ihre erste serverlose Funktion
- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
```

```
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue =
System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload =
Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
```

```
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: Eine Lambda-Funktion synchron aufrufen

Im folgenden Beispiel für `invoke` wird die Funktion `my-function` synchron aufgerufen. Die `cli-binary-format` Option ist erforderlich, wenn Sie AWS CLI Version 2 verwenden. Weitere Informationen finden Sie unter Von der AWS CLI unterstützte globale Befehlszeilenoptionen im AWS -CLI-Benutzerhandbuch.

```
aws lambda invoke \
    --function-name my-function \
    --cli-binary-format raw-in-base64-out \
    --payload '{ "name": "Bob" }' \
    response.json
```

Ausgabe:

```
{
    "ExecutedVersion": "$LATEST",
    "StatusCode": 200
}
```

Weitere Informationen finden Sie unter Synchrones Aufrufen einer Lambda-Funktion im Entwicklerhandbuch für AWS Lambda.

Beispiel 2: Eine Lambda-Funktion asynchron aufrufen

Im folgenden Beispiel für `invoke` wird die Funktion `my-function` asynchron aufgerufen. Die `cli-binary-format` Option ist erforderlich, wenn Sie AWS CLI Version 2 verwenden. Weitere Informationen finden Sie unter Von der AWS CLI unterstützte globale Befehlszeilenoptionen im AWS -CLI-Benutzerhandbuch.

```
aws lambda invoke \
    --function-name my-function \
    --invocation-type Event \
    --cli-binary-format raw-in-base64-out \
    --payload '{ "name": "Bob" }' \
    response.json
```

Ausgabe:

```
{
    "StatusCode": 202
}
```

Weitere Informationen finden Sie unter Asynchrones Aufrufen einer Lambda-Funktion im Entwicklerhandbuch für AWS Lambda.

- Weitere API-Informationen finden Sie unter Invoke in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
```

```
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Invokes a specific AWS Lambda function.
    *
    * @param awsLambda    an instance of {@link LambdaClient} to interact with
the AWS Lambda service
    * @param functionName the name of the AWS Lambda function to be invoked
    */
   public static void invokeFunction(LambdaClient awsLambda, String
functionName) {
       InvokeResponse res;
       try {
           // Need a SdkBytes instance for the payload.
           JSONObject jsonObj = new JSONObject();
           jsonObj.put("inputValue", "2000");
           String json = jsonObj.toString();
           SdkBytes payload = SdkBytes.fromUtf8String(json);

           InvokeRequest request = InvokeRequest.builder()
               .functionName(functionName)
               .payload(payload)
               .build();

           res = awsLambda.invoke(request);
           String value = res.payload().asUtf8String();
           System.out.println(value);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function invoke($functionName, $params, $logType = 'None')
    {
        return $this->lambdaClient->invoke([
            'FunctionName' => $functionName,
            'Payload' => json_encode($params),
            'LogType' => $logType,
        ]);
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This
 dict
                                is serialized to JSON before it is sent to
 Lambda.
        :param get_log: When true, the last 4 KB of the execution log are
 included in
                        the response.
        :return: The response from the function invocation.
```

```
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
```

```
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n
 #{e.message}")
  end
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
 anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
```

```
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}
```

- Weitere API-Informationen finden Sie unter Aufrufen in der API-Referenz zum AWS -SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
          `{`  &&
            `"action": "increment",`  &&
            `"number": 10` &&
          `}` ).
        oo_result = lo_lmd->invoke(                  " oo_result is returned for
testing purposes. "
                iv_functionname = iv_function_name
                iv_payload = lv_json ).
        MESSAGE 'Lambda function invoked.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvrequestcontex.
        MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidzipfileex.
```

```
        MESSAGE 'The deployment package could not be unzipped.' TYPE 'E'.
      CATCH /aws1/cx_lmdrequesttoolargeex.
        MESSAGE 'Invoke request body JSON input limit was exceeded by the request
payload.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
      CATCH /aws1/cx_lmdunsuppedmediatyp00.
        MESSAGE 'Invoke request body does not have JSON as its content type.'
TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Invoke (Aufrufen) in der API-Referenz für das AWS -SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
```

```
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
Int {
        do {
            let incRequest = IncrementRequest(action: "increment", number:
number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }
```

- Detaillierte API-Informationen finden Sie unter Aufrufen in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListFunctions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFunctions` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get a list of Lambda functions.
/// </summary>
/// <returns>A list of FunctionConfiguration objects.</returns>
public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
{
    var functionList = new List<FunctionConfiguration>();

    var functionPaginator =
        _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
    await foreach (var function in functionPaginator.Functions)
    {
        functionList.Add(function);
    }

    return functionList;
}
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
```

```
                            <<
 Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                    functionConfiguration.GetRuntime()) << ": "
                            << functionConfiguration.GetHandler()
                            << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Eine Liste der Lambda-Funktionen abrufen

Im folgenden Beispiel für `list-functions` wird eine Liste aller Funktionen für den aktuellen Benutzer angezeigt.

```
aws lambda list-functions
```

Ausgabe:

```
{
    "Functions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbL/RMr0fT/I=",
            "FunctionName": "helloworld",
            "MemorySize": 128,
            "RevisionId": "1718e831-badf-4253-9518-d0644210af7b",
```

```
            "CodeSize": 294,
            "FunctionArn": "arn:aws:lambda:us-
west-2:123456789012:function:helloworld",
            "Handler": "helloworld.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-
role-zgur6bf4",
            "Timeout": 3,
            "LastModified": "2025-09-23T18:32:33.857+0000",
            "Runtime": "nodejs22.x",
            "Description": ""
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2025-10-01T16:47:28.490+0000",
            "Runtime": "nodejs22.x",
            "Description": ""
        },
        {
            "Layers": [
                {
                    "CodeSize": 41784542,
                    "Arn": "arn:aws:lambda:us-
west-2:420165488524:layer:AWSLambda-Python37-SciPy1x:2"
                },
                {
```

```
                    "CodeSize": 4121,
                    "Arn": "arn:aws:lambda:us-
west-2:123456789012:layer:pythonLayer:1"
                }
            ],
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "ZQukCqxtkqFgyF2cU41Avj99TKQ/hNihPtDtRcc08mI=",
            "FunctionName": "my-python-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 128,
            "RevisionId": "80b4eabc-acf7-4ea8-919a-e874c213707d",
            "CodeSize": 299,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
python-function",
            "Handler": "lambda_function.lambda_handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/my-python-
function-role-z5g7dr6n",
            "Timeout": 3,
            "LastModified": "2025-10-01T19:40:41.643+0000",
            "Runtime": "python3.11",
            "Description": ""
        }
    ]
}
```

Weitere Informationen finden Sie unter Konfigurieren des Speichers der Lambda-Funktion im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie ListFunctionsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// ListFunctions lists up to maxItems functions for the account. This function
 uses a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
```

```
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function listFunctions($maxItems = 50, $marker = null)
    {
        if (is_null($marker)) {
            return $this->lambdaClient->listFunctions([
                'MaxItems' => $maxItems,
            ]);
        }

        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Lambda-Funktionen mit sortierter Codegröße angezeigt

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Ausgabe:

```
FunctionName                                                Runtime   Timeout
 CodeSize
```

```
------------                                                    -------   -------
 --------
test                                                            python2.7       3
   243
MylambdaFunction123                                             python3.8     600
   659
myfuncpython1                                                   python3.8     303
   675
```

- Einzelheiten zur API finden Sie unter ListFunctions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Lambda-Funktionen mit sortierter Codegröße angezeigt

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Ausgabe:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
test                                                            python2.7       3
   243
MylambdaFunction123                                             python3.8     600
   659
myfuncpython1                                                   python3.8     303
   675
```

- Einzelheiten zur API finden Sie unter ListFunctions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListFunctionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput,
anyhow::Error> {
        info!("Listing lambda functions");
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }
```

- Einzelheiten zur API finden Sie ListFunctionsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_lmd->listfunctions( ).       " oo_result is returned for
testing purposes. "
        DATA(lt_functions) = oo_result->get_functions( ).
        MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
```

```
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListFunctionsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []
```

```
        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }
```

- Einzelheiten zur API finden Sie ListFunctionsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`ListProvisionedConcurrencyConfigs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListProvisionedConcurrencyConfigs` verwendet wird.

CLI

AWS CLI

So erhalten Sie eine Liste der bereitgestellten Gleichzeitigkeitskonfigurationen

Das folgende `list-provisioned-concurrency-configs`-Beispiel listet die bereitgestellten Gleichzeitigkeitskonfigurationen für die angegebene Funktion auf.

```
aws lambda list-provisioned-concurrency-configs \
    --function-name my-function
```

Ausgabe:

```
{
    "ProvisionedConcurrencyConfigs": [
        {
            "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-
function:GREEN",
            "RequestedProvisionedConcurrentExecutions": 100,
```

```
            "AvailableProvisionedConcurrentExecutions": 100,
            "AllocatedProvisionedConcurrentExecutions": 100,
            "Status": "READY",
            "LastModified": "2019-12-31T20:29:00+0000"
        },
        {
            "FunctionArn": "arn:aws:lambda:us-east-2:123456789012:function:my-
function:BLUE",
            "RequestedProvisionedConcurrentExecutions": 100,
            "AvailableProvisionedConcurrentExecutions": 100,
            "AllocatedProvisionedConcurrentExecutions": 100,
            "Status": "READY",
            "LastModified": "2019-12-31T20:28:49+0000"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListProvisionedConcurrencyConfigsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der bereitgestellten Gleichzeitigkeitskonfigurationen für eine Lambda-Funktion abgerufen.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter ListProvisionedConcurrencyConfigs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der bereitgestellten Gleichzeitigkeitskonfigurationen für eine Lambda-Funktion abgerufen.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- Einzelheiten zur API finden Sie unter ListProvisionedConcurrencyConfigs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListTags` verwendet wird.

CLI

AWS CLI

So rufen Sie die Liste der Tags für eine Lambda-Funktion ab

Das folgende `list-tags`-Beispiel zeigt die an die `my-function`-Lambda-Funktion angehängten Tags an.

```
aws lambda list-tags \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function
```

Ausgabe:

```
{
    "Tags": {
        "Category": "Web Tools",
        "Department": "Sales"
    }
}
```

Weitere Informationen finden Sie unter Tagging von Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie ListTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft die Tags und ihre Werte ab, die derzeit für die angegebene Funktion festgelegt sind.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Ausgabe:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Einzelheiten zur API finden Sie unter ListTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft die Tags und ihre Werte ab, die derzeit für die angegebene Funktion festgelegt sind.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Ausgabe:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- Einzelheiten zur API finden Sie unter ListTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListVersionsByFunction`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListVersionsByFunction` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste der Versionen einer Funktion ab

Im folgenden `list-versions-by-function`-Beispiel wird die Liste der Versionen für die `my-function`-Lambda-Funktion angezeigt.

```
aws lambda list-versions-by-function \
    --function-name my-function
```

Ausgabe:

```
{
    "Versions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "93017fc9-59cb-41dc-901b-4845ce4bf668",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:$LATEST",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-10-01T16:47:28.490+0000",
            "Runtime": "nodejs10.x",
            "Description": ""
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "1",
            "CodeSha256": "5tT2qgzYUHoqwR616pZ2dpkn/0J1FrzJmlKidWaaCgk=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
```

```
            },
            "MemorySize": 256,
            "RevisionId": "949c8914-012e-4795-998c-e467121951b1",
            "CodeSize": 304,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:1",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-09-26T20:28:40.438+0000",
            "Runtime": "nodejs10.x",
            "Description": "new version"
        },
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "2",
            "CodeSha256": "sU0cJ2/hOZevwV/lTxCuQqK3gDZP3i8gUoqUUVRmY6E=",
            "FunctionName": "my-function",
            "VpcConfig": {
                "SubnetIds": [],
                "VpcId": "",
                "SecurityGroupIds": []
            },
            "MemorySize": 256,
            "RevisionId": "cd669f21-0f3d-4e1c-9566-948837f2e2ea",
            "CodeSize": 266,
            "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:2",
            "Handler": "index.handler",
            "Role": "arn:aws:iam::123456789012:role/service-role/
helloWorldPython-role-uy3l9qyq",
            "Timeout": 3,
            "LastModified": "2019-10-01T16:47:28.490+0000",
            "Runtime": "nodejs10.x",
            "Description": "newer version"
        }
    ]
}
```

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz ListVersionsByFunction.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt die Liste der versionsspezifischen Konfigurationen für jede Version der Lambda-Funktion zurück.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Ausgabe:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
      RoleName
------------        -------   ---------- ------- -------- ------------
      --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Einzelheiten zur API finden Sie unter ListVersionsByFunction AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt die Liste der versionsspezifischen Konfigurationen für jede Version der Lambda-Funktion zurück.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Ausgabe:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
      RoleName
------------        -------   ---------- ------- -------- ------------
      --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- Einzelheiten zur API finden Sie unter ListVersionsByFunction AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PublishVersion`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PublishVersion` verwendet wird.

CLI

AWS CLI

So veröffentlichen Sie eine neue Version einer Funktion

Im folgenden Beispiel für `publish-version` wird eine neue Version der `my-function`-Lambda-Funktion veröffentlicht.

```
aws lambda publish-version \
    --function-name my-function
```

Ausgabe:

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "dBG9m8SGdmlEjw/JYXlhhvCrAv5TxvXsbL/RMr0fT/I=",
    "FunctionName": "my-function",
    "CodeSize": 294,
```

```
    "RevisionId": "f31d3d39-cc63-4520-97d4-43cd44c94c20",
    "MemorySize": 128,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:3",
    "Version": "2",
    "Role": "arn:aws:iam::123456789012:role/service-role/MyTestFunction-role-
zgur6bf4",
    "Timeout": 3,
    "LastModified": "2019-09-23T18:32:33.857+0000",
    "Handler": "my-function.handler",
    "Runtime": "nodejs10.x",
    "Description": ""
}
```

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz PublishVersion.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Version für den vorhandenen Snapshot von Lambda-Funktionscode erstellt

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Einzelheiten zur API finden Sie unter PublishVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Version für den vorhandenen Snapshot von Lambda-Funktionscode erstellt

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- Einzelheiten zur API finden Sie unter PublishVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutFunctionConcurrency`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutFunctionConcurrency` verwendet wird.

CLI

AWS CLI

So konfigurieren Sie ein Limit für reservierte Gleichzeitigkeit für eine Funktion

Im folgenden `put-function-concurrency`-Beispiel werden 100 reservierte gleichzeitige Ausführungen für die `my-function`-Funktion konfiguriert.

```
aws lambda put-function-concurrency \
    --function-name  my-function  \
    --reserved-concurrent-executions 100
```

Ausgabe:

```
{
    "ReservedConcurrentExecutions": 100
}
```

Weitere Informationen finden Sie unter Gleichzeitigkeit für eine Lambda-Funktion reservieren im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie PutFunctionConcurrencyin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Gleichzeitigkeitseinstellungen für die Funktion als Ganzes angewendet.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Einzelheiten zur API finden Sie unter PutFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Gleichzeitigkeitseinstellungen für die Funktion als Ganzes angewendet.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- Einzelheiten zur API finden Sie unter PutFunctionConcurrency AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutProvisionedConcurrencyConfig`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutProvisionedConcurrencyConfig` verwendet wird.

CLI

AWS CLI

So weisen Sie bereitgestellte Gleichzeitigkeit zu

Das folgende `put-provisioned-concurrency-config`-Beispiel weist dem `BLUE`-Alias der angegebenen Funktion 100 bereitgestellte Gleichzeitigkeit zu.

```
aws lambda put-provisioned-concurrency-config \
    --function-name my-function \
    --qualifier BLUE \
    --provisioned-concurrent-executions 100
```

Ausgabe:

```
{
    "Requested ProvisionedConcurrentExecutions": 100,
    "Allocated ProvisionedConcurrentExecutions": 0,
    "Status": "IN_PROGRESS",
    "LastModified": "2019-11-21T19:32:12+0000"
}
```

- Einzelheiten zur API finden Sie PutProvisionedConcurrencyConfigin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird dem Alias einer Funktion eine bereitgestellte Gleichzeitigkeitskonfiguration hinzugefügt

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Einzelheiten zur API finden Sie unter PutProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird dem Alias einer Funktion eine bereitgestellte Gleichzeitigkeitskonfiguration hinzugefügt

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- Einzelheiten zur API finden Sie unter PutProvisionedConcurrencyConfig AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemovePermission`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemovePermission` verwendet wird.

CLI

AWS CLI

So entfernen Sie Berechtigungen aus einer vorhandenen Lambda-Funktion

Das folgende `remove-permission`-Beispiel entfernt die Berechtigung zum Aufrufen einer Funktion namens `my-function`.

```
aws lambda remove-permission \
    --function-name my-function \
    --statement-id sns
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Using Resource-based Policies for AWS Lambda im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter RemovePermission AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie für die angegebene StatementId Lambda-Funktion entfernt.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId
 $policy[0].Sid
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie für die angegebene StatementId Lambda-Funktion entfernt.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId
 $policy[0].Sid
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`TagResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie einer vorhandenen Lambda-Funktion Tags hinzu

Das folgende `tag-resource`-Beispiel fügt der angegebenen Lambda-Funktion ein Tag mit dem Schlüsselnamen `DEPARTMENT` und einem Wert von `Department A` hinzu.

```
aws lambda tag-resource \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \
    --tags "DEPARTMENT=Department A"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Tagging von Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Fügt die drei Tags (Washington, Oregon und Kalifornien) und ihre zugehörigen Werte zu der angegebenen Funktion hinzu, die durch ihre ARN identifiziert wird.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia";
 "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Einzelheiten zur API finden Sie unter TagResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Fügt die drei Tags (Washington, Oregon und Kalifornien) und ihre zugehörigen Werte zu der angegebenen Funktion hinzu, die durch ihre ARN identifiziert wird.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia";
 "Oregon" = "Salem"; "California" = "Sacramento" }
```

- Einzelheiten zur API finden Sie unter TagResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UntagResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResource` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einer vorhandenen Lambda-Funktion

Das folgende `untag-resource`-Beispiel entfernt das Tag mit dem Schlüsselnamen `DEPARTMENT`-Tag aus der Lambda-Funktion `my-function`.

```
aws lambda untag-resource \
    --resource arn:aws:lambda:us-west-2:123456789012:function:my-function \
    --tag-keys DEPARTMENT
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Tagging von Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie UntagResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey
 "Washington","Oregon","California"
```

Beispiel 2: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Sobald der Dienst pro bereitgestelltem Tag aufgerufen wurde.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Einzelheiten zur API finden Sie unter UntagResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey
 "Washington","Oregon","California"
```

Beispiel 2: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Sobald der Dienst pro bereitgestelltem Tag aufgerufen wurde.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- Einzelheiten zur API finden Sie unter UntagResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateAlias`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAlias` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie einen Funktionsalias

Das folgende `update-alias`-Beispiel aktualisiert den Alias mit dem Namen `LIVE` so, dass er auf Version 3 der `my-function`-Lambda-Funktion verweist.

```
aws lambda update-alias \
```

```
    --function-name my-function \
    --function-version 3 \
    --name LIVE
```

Ausgabe:

```
{
    "FunctionVersion": "3",
    "Name": "LIVE",
    "AliasArn": "arn:aws:lambda:us-west-2:123456789012:function:my-
function:LIVE",
    "RevisionId": "594f41fb-b85f-4c20-95c7-6ca5f2a92c93",
    "Description": "alias for live version of function"
}
```

Weitere Informationen finden Sie unter Konfiguration von AWS Lambda-Funktionsaliasen im AWS Lambda Developer Guide.

- Einzelheiten zur API finden Sie unter Befehlsreferenz UpdateAlias.AWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel aktualisiert die Konfiguration einer vorhandenen Lambda-Funktion-Alias. Der RoutingConfiguration Wert wird aktualisiert, sodass 60% (0,6) des Datenverkehrs auf Version 1 umgestellt werden

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description
 " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6}
```

- Einzelheiten zur API finden Sie unter UpdateAlias AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel aktualisiert die Konfiguration einer vorhandenen Lambda-Funktion-Alias. Der RoutingConfiguration Wert wird aktualisiert, sodass 60% (0,6) des Datenverkehrs auf Version 1 umgestellt werden

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description
 " Alias for version 2" -FunctionVersion 2 -Name "newlabel1" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6}
```

- Einzelheiten zur API finden Sie unter UpdateAlias AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateFunctionCode`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateFunctionCode` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
```

```
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
_lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
{response.LastModified}.");
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
```

```
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." <<
 std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Den Code einer Lambda-Funktion aktualisieren

Im folgenden Beispiel für `update-function-code` wird der Code der unveröffentlichten Version ($LATEST) der Funktion `my-function` durch den Inhalt der angegebenen ZIP-Datei ersetzt.

```
aws lambda update-function-code \
    --function-name  my-function \
    --zip-file fileb://my-function.zip
```

Ausgabe:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Handler": "index.handler"
}
```

Weitere Informationen finden Sie unter Konfigurieren von AWS -Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in
 the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context,
 functionName string, zipPackage *bytes.Buffer) types.State {
 var state types.State
```

```
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n",
 functionName, err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
 located
```

```
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter
                .waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
```

```
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert die Funktion mit dem Namen 'MyFunction' mit neuem Inhalt, der in der angegebenen ZIP-Datei enthalten ist. Für eine C#.NET-Core-Lambda-Funktion sollte die ZIP-Datei die kompilierte Assembly enthalten.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, verwendet aber ein Amazon S3-Objekt, das den aktualisierten Code enthält, um die Funktion zu aktualisieren.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -
Key UpdatedCode.zip
```

- Einzelheiten zur API finden Sie unter UpdateFunctionCode AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert die Funktion mit dem Namen 'MyFunction' mit neuem Inhalt, der in der angegebenen ZIP-Datei enthalten ist. Für eine C#.NET-Core-Lambda-Funktion sollte die ZIP-Datei die kompilierte Assembly enthalten.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, verwendet aber ein Amazon S3-Objekt, das den aktualisierten Code enthält, um die Funktion zu aktualisieren.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -
Key UpdatedCode.zip
```

- Einzelheiten zur API finden Sie unter UpdateFunctionCode AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
 contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes
 in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
```

```
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
```

```
 # @return: Data about the update, including the status.
 def update_function_code(function_name, deployment_package)
   @lambda_client.update_function_code(
     function_name: function_name,
     zip_file: deployment_package
   )
   @lambda_client.wait_until(:function_updated_v2, { function_name:
function_name }) do |w|
     w.max_attempts = 5
     w.delay = 5
   end
 rescue Aws::Lambda::Errors::ServiceException => e
   @logger.error("There was an error updating function code for:
#{function_name}:\n #{e.message}")
   nil
 rescue Aws::Waiters::Errors::WaiterFailed => e
   @logger.error("Failed waiting for #{function_name} to update:\n
#{e.message}")
 end
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
   pub async fn update_function_code(
       &self,
       zip_file: PathBuf,
       key: String,
   ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
```

```
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --
output-format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
```

```
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->updatefunctioncode(     " oo_result is returned for
testing purposes. "
             iv_functionname = iv_function_name
             iv_zipfile = io_zip_file ).

       MESSAGE 'Lambda function code updated.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
```

```
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data

        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
```

```
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`UpdateFunctionConfiguration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateFunctionConfiguration` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
actions.</param>
    /// <param name="environmentVariables">A dictionary of environment
variables.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
environmentVariables },
        };

        var response = await
_lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
 client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Die Konfiguration einer Funktion ändern

Im folgenden Beispiel für `update-function-configuration` wird die Speichergröße für die unveröffentlichte Version ($LATEST) der Funktion `my-function` auf 256 MB geändert.

```
aws lambda update-function-configuration \
    --function-name  my-function \
    --memory-size 256
```

Ausgabe:

```
{
    "FunctionName": "my-function",
    "LastModified": "2019-09-26T20:28:40.438+0000",
    "RevisionId": "e52502d4-9320-4688-9cd6-152a6ab7490d",
    "MemorySize": 256,
    "Version": "$LATEST",
    "Role": "arn:aws:iam::123456789012:role/service-role/my-function-role-
uy3l9qyq",
    "Timeout": 3,
    "Runtime": "nodejs10.x",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "CodeSha256": "5tT2qgzYUHaqwR716pZ2dpkn/0J1FrzJmlKidWoaCgk=",
    "Description": "",
    "VpcConfig": {
        "SubnetIds": [],
        "VpcId": "",
        "SecurityGroupIds": []
    },
    "CodeSize": 304,
    "FunctionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
    "Handler": "index.handler"
}
```

Weitere Informationen finden Sie unter Konfigurieren von AWS -Lambda-Funktionen im AWS -Lambda-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// UpdateFunctionConfiguration updates a map of environment variables configured
 for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v",
 functionName, err)
 }
}
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the configuration of an AWS Lambda function.
    *
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function updateFunctionConfiguration($functionName, $handler,
 $environment = '')
    {
```

```
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel aktualisiert die vorhandene Lambda-Funktionskonfiguration

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Einzelheiten zur API finden Sie unter UpdateFunctionConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel aktualisiert die vorhandene Lambda-Funktionskonfiguration

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- Einzelheiten zur API finden Sie unter UpdateFunctionConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                   function_name: function_name,
                                                   environment: {
                                                     variables: {
                                                       'LOG_LEVEL' => log_level
                                                     }
                                                   }
                                                 })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
```

```
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for
#{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;
```

```
        self.wait_for_function_ready().await?;

        Ok(updated)
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_lmd->updatefunctionconfiguration(     " oo_result is
returned for testing purposes. "
             iv_functionname = iv_function_name
             iv_runtime = iv_runtime
             iv_description  = 'Updated Lambda function'
             iv_memorysize  = iv_memory_size ).

        MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
      CATCH /aws1/cx_lmdcodesigningcfgno00.
        MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdcodeverification00.
        MESSAGE 'Code signature failed one or more validation checks for
signature mismatch or expiration.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidcodesigex.
        MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.'
TYPE 'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
```

```
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
 TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String)
 async throws {
```

```
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der API-Referenz zum AWS SDK für Swift.

Szenarien für die Verwendung von Lambda AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Lambda mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen, indem Sie mehrere Funktionen innerhalb von Lambda oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK
- Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung
- Leihbibliothek-REST-API erstellen
- Erstellen einer Messenger-Anwendung mit Step Functions
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen einer Websocket-Chat-Anwendung mit API Gateway
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Ein Monitoring-Dashboard mit dem Funktionsnamen als Variable erstellen
- Erstellen Sie Ihre erste serverlose Funktion
- Aufrufen einer Lambda-Funktion von einem Browser aus
- Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion
- Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt

- Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen
- Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

Bestätigen Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
```

```
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
```

```
 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
```

```
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
```

```
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
```

```
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
```

```
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
```

```
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
```

```
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
```

```
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

```
// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
```

```
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}
```

```
// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
```

```
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
```

```
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
```

```
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
```

```
    log.Printf("Something went wrong during cleanup.\n%v\n", r)
    log.Println("Use the AWS Management Console to remove any remaining resources
  \n" +
     "that were created for this scenario.")
   }
  }()

  wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
  resources that were created "+
   "during this demo (y/n)?", "y")
  if wantDelete {
   for _, accessToken := range resources.userAccessTokens {
    err := resources.cognitoActor.DeleteUser(ctx, accessToken)
    if err != nil {
     log.Println("Couldn't delete user during cleanup.")
     panic(err)
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - DeleteUser

- InitiateAuth
- SignUp
- UpdateUserPool

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie einen interaktiven „Szenario“-Lauf. Die JavaScript (v3) -Beispiele verwenden gemeinsam einen Szenario-Runner, um komplexe Beispiele zu vereinfachen. Der komplette Quellcode ist aktiviert GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
```

```
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Dieses Szenario demonstriert die automatische Bestätigung eines bekannten Benutzers. Es orchestriert die Beispielschritte.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
```

```
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
```

```
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
```

```
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
```

```
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);
```

```
    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
```

```
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);
```

```
const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
```

```
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Dies sind Schritte, die mit anderen Szenarien geteilt werden.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";
```

```
export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Ein Handler für den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
```

```
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
 '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
```

```
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName
 '${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Modul für CloudWatch Logs-Aktionen.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
```

```
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
```

```
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für Amazon-Cognito-Aktionen.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
```

```
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
```

```
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
```

```
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für DynamoDB-Aktionen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
```

```
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migrieren Sie automatisch bekannte Amazon Cognito Cognito-Benutzer mit einer Lambda-Funktion mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch migriert werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `MigrateUser`-Trigger aufzurufen.
- Melden Sie sich bei Amazon Cognito mit einem Benutzernamen und einer E-Mail-Adresse an, die sich nicht im Benutzerpool befinden.
- Die Lambda-Funktion durchsucht eine DynamoDB-Tabelle und migriert bekannte Benutzer automatisch in den Benutzerpool.
- Führen Sie den Ablauf „Passwort vergessen“ aus, um das Passwort für den migrierten Benutzer zurückzusetzen.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
```

```
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}
```

```
// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
```

```
    runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  *authResult.AccessToken)
    signedIn = true
   }
  }

  log.Println(strings.Repeat("-", 88))
  return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
  user actions.User) {
  wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
  to Cognito, you must be able to receive a confirmation\n"+
   "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
  if !wantCode {
   log.Println("To complete this example and successfully migrate a user to
  Cognito, you must enter an email\n" +
    "you own that can receive a confirmation code.")
   return
  }
  codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
  user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("\nA confirmation code has been sent to %v.",
  *codeDelivery.Destination)
  code := runner.questioner.Ask("Check your email and enter it here:")

  confirmed := false
  password := runner.questioner.AskPassword("\nEnter a password that has at least
  eight characters, uppercase, lowercase, numbers and symbols.\n"+
   "(the password will not display as you type):", 8)
  for !confirmed {
   log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
   err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
  user.UserName, password)
   if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
     password = runner.questioner.AskPassword("\nEnter another password:", 8)
    } else {
```

```
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
```

```
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `MigrateUser`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
```

```
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
```

```
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
```

```
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
```

```
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
```

```
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
```

```
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
```

```
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

```
// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
```

```
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}
```

```
// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
```

```
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
```

```
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
```

```
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
```

```
    log.Printf("Something went wrong during cleanup.\n%v\n", r)
    log.Println("Use the AWS Management Console to remove any remaining resources
  \n" +
     "that were created for this scenario.")
   }
  }()

  wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
  resources that were created "+
   "during this demo (y/n)?", "y")
  if wantDelete {
   for _, accessToken := range resources.userAccessTokens {
    err := resources.cognitoActor.DeleteUser(ctx, accessToken)
    if err != nil {
     log.Println("Couldn't delete user during cleanup.")
     panic(err)
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - ConfirmForgotPassword

- DeleteUser
- ForgotPassword
- InitiateAuth
- SignUp
- UpdateUserPool

Erstellen Sie eine API-Gateway-REST-API zur COVID-19 Datenverfolgung

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die anhand fiktiver Daten ein System zur Nachverfolgung täglicher Fälle COVID-19 in den Vereinigte Staaten simuliert.

Python

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon AWS Lambda DynamoDB verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle von Infektionen COVID-19 in Vereinigte Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation

- DynamoDB
- Lambda

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

Python

SDK für Python (Boto3)

Zeigt, wie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) API und AWS Chalice verwendet wird, um eine REST-API zu erstellen, die von einer Amazon Aurora Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldedaten verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen einer Messenger-Anwendung mit Step Functions

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

Python

SDK für Python (Boto3)

Zeigt, wie AWS SDK für Python (Boto3) mit AWS Step Functions dem with eine Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. Die Zustandsmaschine ist mit einer AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht werden kann.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Eine Lambda-Funktion erstellen und bereitstellen

- Erstellen Sie eine REST-API
- Lambda-Proxyintegration konfigurieren
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."
```

```
# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
```

```
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
```

```
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
```

```
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}
```

```
echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2
```

```
# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Websocket-Chat-Anwendung mit API Gateway

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon API Gateway V2 eine Websocket-API erstellen, die in Amazon DynamoDB integriert AWS Lambda werden kann.

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.
- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

Die folgenden Codebeispiele zeigen, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus ihrer Originalsprache übersetzt, ihre Stimmung ermittelt und aus dem übersetzten Text eine Audiodatei generiert.

.NET

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.

- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie der innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
```

```
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
```

```
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
```

```
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });
```

```
  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ein Monitoring-Dashboard mit dem Funktionsnamen als Variable erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
```

```
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
```

```
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Eine IAM-Rolle für Lambda erstellen
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES
```

```
##############################################################################
# Helper functions
##############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
```

```
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
```

```
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
```

```
##############################################################################
# Runtime selection
##############################################################################

echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
```

```
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10
```

```
##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi
```

```
echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
```

```
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

###############################################################################
# Step 4: View CloudWatch logs
###############################################################################

echo ""
echo "============================================"
echo "Step 4: View CloudWatch Logs"
echo "============================================"
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
```

```
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy

- CreateFunction
- CreateRole
- DeleteFunction
- DeleteLogGroup
- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

JavaScript

SDK für JavaScript (v2)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion verwendet, um eine Amazon DynamoDB-Tabelle mit Benutzerauswahlen zu aktualisieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

SDK für JavaScript (v3)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion verwendet, um eine Amazon DynamoDB-Tabelle mit Benutzerauswahlen zu aktualisieren. Diese App verwendet v3. AWS SDK für JavaScript

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda

Das folgende Codebeispiel zeigt, wie Sie Daten für Ihre Anwendung mit S3 Object Lambda transformieren können.

.NET

SDK für .NET

Zeigt, wie Sie benutzerdefinierten Code zu standardmäßigen S3-GET-Anforderungen hinzufügen, um das angeforderte Objekt, das von S3 abgerufen wurde, so zu ändern, dass das Objekt den Anforderungen des anfordernden Clients oder der anfordernden Anwendung entspricht.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Amazon S3

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten

Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie Sie eine AWS Step Functions Zustandsmaschine erstellen, die nacheinander AWS Lambda Funktionen aufruft.

Java

SDK für Java 2.x

Zeigt, wie Sie einen AWS serverlosen Workflow mithilfe von AWS Step Functions und erstellen. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt

Das folgende Codebeispiel zeigt, wie Sie mit der Neptune-API Grafikdaten abfragen.

Java

SDK für Java 2.x

Zeigt, wie die Java-API von Amazon Neptune verwendet wird, um eine Lambda-Funktion zu erstellen, die Grafikdaten innerhalb der VPC abfragt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Neptune

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Lambda-Funktionen für die Überwachung erstellen
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
```

```
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi
```

```
    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
```

```
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
```

```
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
```

```
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
```

```
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
```

```
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Schreiben Sie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito Cognito-Benutzerauthentifizierung mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon-Cognito-Benutzerauthentifizierung geschrieben werden.

- Verwenden Sie Administratorfunktionen, um einen Benutzer zu einem Benutzerpool hinzuzufügen.
- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PostAuthentication`-Trigger aufzurufen.
- Melden Sie den neuen Benutzer bei Amazon Cognito an.
- Die Lambda-Funktion schreibt benutzerdefinierte Informationen in CloudWatch Logs und in eine DynamoDB-Tabelle.
- Ermitteln Sie benutzerdefinierte Daten aus der DynamoDB-Tabelle, zeigen Sie diese an und bereinigen Sie dann die Ressourcen.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
```

```
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
```

```
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
```

```
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PostAuthentication`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "fmt"
```

```
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}
```

```
// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
```

```
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
```

```
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
```

```
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
```

```
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
```

```
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
```

```
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
```

```
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
```

```
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

```
// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
```

```
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
```

```
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
```

```
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)
```

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
```

```
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
```

```
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Serverless-Beispiele für Lambda

Die folgenden Codebeispiele zeigen, wie Lambda mit AWS SDKs verwendet wird.

Beispiele

- Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion
- Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser
- Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser
- Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser
- Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser
- Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser
- Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen
- Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion mithilfe von .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
```

```
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                 $"Pwd={authToken};";


        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";
```

```
            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The
 sum is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server
 error" })
        };
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
```

```
var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
var region string = os.Getenv("AWS_REGION")

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
 panic("configuration error: " + err.Error())
}

authenticationToken, err := auth.BuildAuthToken(
 context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
if err != nil {
 panic("failed to create authentication token: " + err.Error())
}

dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
 dbUser, authenticationToken, dbEndpoint, dbName,
)

db, err := sql.Open("mysql", dsn)
if err != nil {
 panic(err)
}

defer db.Close()

var sum int
err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
if err != nil {
 panic(err)
}
s := fmt.Sprint(sum)
message := fmt.Sprintf("The selected sum is: %s", s)

messageBytes, err := json.Marshal(message)
if err != nil {
 return nil, err
}

messageString := string(messageBytes)
return map[string]interface{}{
 "statusCode": 200,
 "headers":    map[string]string{"Content-Type": "application/json"},
 "body":       messageString,
}, nil
```

```
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    @Override
```

```
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
```

```
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
```

```
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}
```

```
export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von TypeScript.

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}
```

```
async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
```

```
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new
AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

"mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
```

```
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Python.

```
import json
import os
import boto3
import pymysql
```

```
# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )
```

```
  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
```

```
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
```

```
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
```

```
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
```

```
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }
```

```
 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
```

```
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+"
 records");
        return null;
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Ein Kinesis-Ereignis mit Lambda verwenden. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
```

```
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Ein Kinesis-Ereignis mit Lambda verwenden. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
```

```
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;
```

```
require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error>
 {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
```

```
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;
```

```
public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext
 context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
```

```
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string,
 error) {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
```

```
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new
 GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " +
 GSON.toJson(record.getDynamodb()));
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines DynamoDB-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
```

```
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Konsumieren eines DynamoDB-Ereignisses mit Lambda unter Verwendung. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity
```

```
use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be
 marked as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())
```

```
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
```

```
//Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });

        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
    }
```

```
public class Event
{
    [JsonPropertyName("eventSourceArn")]
    public string EventSourceArn { get; set; }

    [JsonPropertyName("events")]
    public List<DocumentDBEventRecord> Events { get; set; }

    [JsonPropertyName("eventSource")]
    public string EventSource { get; set; }
}

public class DocumentDBEventRecord
{
    [JsonPropertyName("event")]
    public EventData Event { get; set; }
}

public class EventData
{
    [JsonPropertyName("_id")]
    public IdData Id { get; set; }

    [JsonPropertyName("clusterTime")]
    public ClusterTime ClusterTime { get; set; }

    [JsonPropertyName("documentKey")]
    public DocumentKey DocumentKey { get; set; }

    [JsonPropertyName("fullDocument")]
    public Dictionary<string, object> FullDocument { get; set; }

    [JsonPropertyName("ns")]
    public Namespace Ns { get; set; }

    [JsonPropertyName("operationType")]
    public string OperationType { get; set; }
}

public class IdData
{
    [JsonPropertyName("_data")]
```

```
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
```

```
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
```

```
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verwenden eines Amazon DocumentDB DocumentDB-Ereignisses mit Lambda. JavaScript

```
console.log('Loading function');
```

```
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Ein Amazon DocumentDB DocumentDB-Ereignis mit Lambda verwenden TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
```

```
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(),
 Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
```

```
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Die folgenden Codebeispiele zeigen, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to
 process.</param>
    /// <param name="context">The ILambdaContext that provides methods for
 logging and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);
```

```
                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
```

```
    fmt.Println("Message:", message)
   }
  }
 }

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);
```

```
                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Ein Amazon MSK-Ereignis mit Lambda verwenden. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Ein Amazon MSK-Ereignis mit Lambda verwenden. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
```

```
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();
```

```
        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
```

```
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error>
 {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
```

```
            let message = std::str::from_utf8(&record_bytes)?;

            info!("Message: {}", message);
           }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von .NET

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");
```

```
                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
```

```
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
```

```
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung JavaScript.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
```

```
    }
};
```

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
```

```
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
```

```
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
```

```
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();
```

```
    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
```

```
}
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
```

```
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
```

```
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
```

```
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
```

```
  raise
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Die folgenden Codebeispiele zeigen, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;
```

```
// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
```

```
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SQS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SQS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";
```

```
export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
```

```
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
```

```
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
```

```
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace KinesisIntegration;

public class Function
{
```

```
   // Powertools Logger requires an environment variables against your function
   // POWERTOOLS_SERVICE_NAME
   [Logging(LogEvent = true)]
   public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
ILambdaContext context)
   {
       if (evnt.Records.Count == 0)
       {
           Logger.LogInformation("Empty Kinesis Event received");
           return new StreamsEventResponse();
       }

       foreach (var record in evnt.Records)
       {
           try
           {
               Logger.LogInformation($"Processed Event with EventId:
{record.EventId}");
               string data = await GetRecordDataAsync(record.Kinesis, context);
               Logger.LogInformation($"Data: {data}");
               // TODO: Do interesting work based on the new data
           }
           catch (Exception ex)
           {
               Logger.LogError($"An error occurred {ex.Message}");
               /* Since we are working with streams, we can return the failed
item immediately.
                  Lambda will immediately begin to retry processing from this
failed item onwards. */
               return new StreamsEventResponse
               {
                   BatchItemFailures = new
List<StreamsEventResponse.BatchItemFailure>
                   {
                       new StreamsEventResponse.BatchItemFailure
{ ItemIdentifier = record.Kinesis.SequenceNumber }
                   }
               };
           }
       }
       Logger.LogInformation($"Successfully processed {evnt.Records.Count}
records.");
       return new StreamsEventResponse();
   }
```

```
    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
```

```
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }

  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord =
 kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
```

```
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};
```

```
async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Melden von Fehlern Kinesis Kinesis-Batch-Elementen mit Lambda unter Verwendung von. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
```

```
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
```

```
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
```

```
  sleep(1)
  data
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );
```

```
        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
```

```
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace AWSLambda_DDB;
```

```
public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent)
 (*BatchResult, error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
```

```
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
```

```
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context
 context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed
 item immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier:
 curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }
```

```
  return { batchItemFailures: batchItemFailures };
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
```

```
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;
```

```
    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed
 item onwards. */
            return Ok(response);
        }
    }
```

```
    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
```

```
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
```

```
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();
```

```
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batch-Elementfehlern mit Lambda unter Verwendung von. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
```

```
    console.log(`Processed message: ${record.body}`);
}
```

Melden von SQS-Batch-Elementfehlern mit Lambda unter Verwendung von. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord }
 from 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier":
 record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) ->
 Result<SqsBatchResponse, Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
    run(service_fn(function_handler)).await
}
```

AWS Community-Beiträge für Lambda

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden AWS. Um Feedback zu geben, verwenden Sie den Mechanismus, der in den verlinkten Repositorys bereitgestellt wird.

Beispiele

- Erstellen und Testen einer Serverless-Anwendung

Erstellen und Testen einer Serverless-Anwendung

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

.NET

SDK für .NET

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des .NET SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Go

SDK für Go V2

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Java

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Rust

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Codebeispiele für Amazon Lex mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Lex mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Lex-Entwicklerhandbuch – Weitere Informationen zu Amazon Lex.
- API-Referenz für Amazon Lex – Details zu allen verfügbaren Amazon-Lex-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Amazon Lex AWS SDKs
  - Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher

Szenarien für die Verwendung von Amazon Lex AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Lex mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Lex oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher

Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher

Die folgenden Codebeispiele zeigen, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

Java

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel „Einen Amazon Lex-Chatbot erstellen“ im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Codebeispiele für Lightsail mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Lightsail mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Lightsail Developer Guide — Weitere Informationen zu Lightsail.
- Lightsail-API-Referenz — Details zu allen verfügbaren Lightsail-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Lightsail AWS SDKs
  - Erste Schritte mit einfachen virtuellen privaten Servern

Szenarien für die Verwendung von Lightsail AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Lightsail mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Lightsail oder in Kombination mit anderen aufrufen. AWS-Services Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit einfachen virtuellen privaten Servern

Erste Schritte mit einfachen virtuellen privaten Servern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer -Instance
- Fügen Sie Ihrer Instanz Speicher hinzu
- Snapshot erstellen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Lightsail Getting Started CLI Script
# This script demonstrates how to create and manage Lightsail resources using the
 AWS CLI

set -euo pipefail

# Set AWS region
export AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION"

# Set up logging
LOG_FILE="lightsail-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Lightsail Getting Started script at $(date)"

# Error handling function
handle_error() {
  echo "ERROR: $1" >&2
  echo "Attempting to clean up resources..."
  cleanup_resources || true
  exit 1
```

```
}

# Function to check if a command succeeded
check_status() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
INSTANCE_NAME="LightsailInstance-${RANDOM_ID}"
DISK_NAME="LightsailDisk-${RANDOM_ID}"
SNAPSHOT_NAME="LightsailSnapshot-${RANDOM_ID}"

# Array to track created resources
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
  CREATED_RESOURCES+=("$1:$2")
  echo "Created $1: $2"
}

# Function to clean up resources
cleanup_resources() {
  echo "Resources created by this script:"
  for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "  $resource"
  done

  # Reverse the array to delete resources in reverse order
  for (( idx=${#CREATED_RESOURCES[@]}-1 ; idx>=0 ; idx-- )); do
    IFS=':' read -r type name <<< "${CREATED_RESOURCES[idx]}"

    case "$type" in
      "instance_snapshot")
        echo "Deleting instance snapshot: $name"
        aws lightsail delete-instance-snapshot --instance-snapshot-name "$name"
 --region "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk_snapshot")
        echo "Deleting disk snapshot: $name"
```

```
        aws lightsail delete-disk-snapshot --disk-snapshot-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk")
        echo "Detaching disk: $name"
        aws lightsail detach-disk --disk-name "$name" --region "$AWS_REGION" 2>/
dev/null || true
        sleep 10
        echo "Deleting disk: $name"
        aws lightsail delete-disk --disk-name "$name" --region "$AWS_REGION" 2>/
dev/null || true
        ;;
      "instance")
        echo "Deleting instance: $name"
        INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name "$name"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
        if [ "$INSTANCE_STATE" == "pending" ]; then
          echo "Instance is in pending state. Waiting for it to be ready before
 deleting..."
          MAX_WAIT=30
          WAITED=0
          while [ "$INSTANCE_STATE" == "pending" ] && [ $WAITED -lt $MAX_WAIT ];
 do
            sleep 10
            WAITED=$((WAITED+1))
            INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name
 "$name" --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null ||
 echo "unknown")
            echo "Instance state: $INSTANCE_STATE"
          done
        fi
        aws lightsail delete-instance --instance-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
    esac
  done

  echo "Cleanup completed"
}

# Step 1: Verify AWS CLI configuration
echo "Step 1: Verifying AWS CLI configuration"
aws configure list
```

```
check_status "Failed to verify AWS CLI configuration"

# Step 2: Get available blueprints and bundles
echo "Step 2: Getting available blueprints and bundles"
echo "Available blueprints (showing first 5):"
aws lightsail get-blueprints --region "$AWS_REGION" --query 'blueprints[0:5].
[blueprintId,name]' --output table
check_status "Failed to get blueprints"

echo "Available bundles (showing first 5):"
aws lightsail get-bundles --region "$AWS_REGION" --query 'bundles[0:5].
[bundleId,name,price]' --output table
check_status "Failed to get bundles"

# Get available regions and availability zones
echo "Getting available regions and availability zones"
# Use a specific availability zone in us-west-2 region
AVAILABILITY_ZONE="us-west-2a"
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 3: Create a Lightsail instance
echo "Step 3: Creating Lightsail instance: $INSTANCE_NAME"
aws lightsail create-instances \
  --instance-names "$INSTANCE_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --blueprint-id amazon_linux_2023 \
  --bundle-id nano_3_0 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create Lightsail instance"
track_resource "instance" "$INSTANCE_NAME"

# Wait for the instance to be in a running state
echo "Waiting for instance to be in running state..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  STATUS=$(aws lightsail get-instance-state --instance-name "$INSTANCE_NAME"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
  if [ "$STATUS" == "running" ]; then
    echo "Instance is now running"
    break
  fi
```

```
  echo "Instance status: $STATUS. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Instance failed to reach running state after 5 minutes"
fi

# Get instance details
echo "Getting instance details"
INSTANCE_IP=$(aws lightsail get-instance --instance-name "$INSTANCE_NAME" --
region "$AWS_REGION" --query 'instance.publicIpAddress' --output text)
check_status "Failed to get instance IP address"
echo "Instance IP address: $INSTANCE_IP"

# Validate IP address format
if ! [[ "$INSTANCE_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
  handle_error "Invalid IP address format: $INSTANCE_IP"
fi

# Step 4: Download the default key pair
echo "Step 4: Downloading default key pair"
KEY_FILE="lightsail_key_${RANDOM_ID}.pem"
umask 077
aws lightsail download-default-key-pair --region "$AWS_REGION" --output text >
 "$KEY_FILE"
check_status "Failed to download key pair"

if [ ! -f "$KEY_FILE" ] || [ ! -s "$KEY_FILE" ]; then
  handle_error "Key pair file was not created or is empty"
fi

chmod 400 "$KEY_FILE"
check_status "Failed to set permissions on key pair"
echo "Key pair downloaded to $KEY_FILE"

echo "To connect to your instance, use:"
echo "ssh -i $KEY_FILE ec2-user@$INSTANCE_IP"

# Step 5: Create a block storage disk
echo "Step 5: Creating block storage disk: $DISK_NAME"
aws lightsail create-disk \
```

```
  --disk-name "$DISK_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --size-in-gb 8 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create disk"
track_resource "disk" "$DISK_NAME"

# Wait for the disk to be available using polling
echo "Waiting for disk to be available..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  DISK_STATE=$(aws lightsail get-disk --disk-name "$DISK_NAME" --region
 "$AWS_REGION" --query 'disk.state' --output text 2>/dev/null || echo "unknown")
  if [ "$DISK_STATE" == "available" ]; then
    echo "Disk is now available"
    break
  fi
  echo "Disk status: $DISK_STATE. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Disk failed to become available after 5 minutes"
fi

# Attach the disk to the instance
echo "Attaching disk to instance"
aws lightsail attach-disk \
  --disk-name "$DISK_NAME" \
  --instance-name "$INSTANCE_NAME" \
  --disk-path /dev/xvdf \
  --region "$AWS_REGION"
check_status "Failed to attach disk to instance"

echo "Disk attached. To format and mount the disk, connect to your instance and
 run:"
echo "sudo mkfs -t ext4 /dev/xvdf"
echo "sudo mkdir -p /mnt/my-data"
echo "sudo mount /dev/xvdf /mnt/my-data"
echo "sudo chown ec2-user:ec2-user /mnt/my-data"
```

```
# Step 6: Create a snapshot of the instance
echo "Step 6: Creating snapshot of the instance: $SNAPSHOT_NAME"
aws lightsail create-instance-snapshot \
  --instance-name "$INSTANCE_NAME" \
  --instance-snapshot-name "$SNAPSHOT_NAME" \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create instance snapshot"
track_resource "instance_snapshot" "$SNAPSHOT_NAME"

# Wait for the snapshot to complete using polling
echo "Waiting for snapshot to complete... (this may take several minutes)"
MAX_ATTEMPTS=60
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  SNAPSHOT_STATE=$(aws lightsail get-instance-snapshot --instance-snapshot-name
 "$SNAPSHOT_NAME" --region "$AWS_REGION" --query 'instanceSnapshot.state' --
output text 2>/dev/null || echo "unknown")
  if [ "$SNAPSHOT_STATE" == "available" ]; then
    echo "Snapshot creation completed"
    break
  fi
  echo "Snapshot status: $SNAPSHOT_STATE. Waiting... ($ATTEMPTS/$MAX_ATTEMPTS)"
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  echo "Warning: Snapshot creation is taking longer than expected but will
 continue in the background."
  echo "You can check its status later with: aws lightsail get-instance-snapshot
 --instance-snapshot-name $SNAPSHOT_NAME --region $AWS_REGION"
fi

# Step 7: Clean up resources
echo "Step 7: Clean up resources"
echo "The script has created the following resources:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
  echo "  $resource"
done

echo "Cleaning up these resources automatically..."
cleanup_resources
```

```
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachDisk
  - CreateDisk
  - CreateInstanceSnapshot
  - CreateInstances
  - DeleteDisk
  - DeleteInstance
  - DeleteInstanceSnapshot
  - DetachDisk
  - DownloadDefaultKeyPair
  - GetBlueprints
  - GetBundles
  - GetDisk
  - GetInstance
  - GetInstanceSnapshot
  - GetInstanceState

Codebeispiele für Amazon Location mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Location Service mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Amazon-Location-Entwicklerhandbuch – Weitere Informationen zu Amazon Location.

- API-Referenz für Amazon Location – Details zu allen verfügbaren Aktionen von Amazon Location.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Location AWS SDKs
  - Hello Amazon Location Service
  - Lernen Sie die Grundlagen von Amazon Location mit einem AWS SDK
  - Aktionen für Amazon Location mit AWS SDKs
    - Verwenden Sie BatchUpdateDevicePositionmit einem AWS SDK
    - Verwenden Sie CalculateRoutemit einem AWS SDK
    - Verwenden Sie CreateGeofenceCollectionmit einem AWS SDK
    - Verwenden Sie CreateKeymit einem AWS SDK
    - Verwenden Sie CreateMapmit einem AWS SDK
    - Verwenden Sie CreateRouteCalculatormit einem AWS SDK
    - Verwenden Sie CreateTrackermit einem AWS SDK
    - Verwenden Sie DeleteGeofenceCollectionmit einem AWS SDK
    - Verwenden Sie DeleteKeymit einem AWS SDK
    - Verwenden Sie DeleteMapmit einem AWS SDK
    - Verwenden Sie DeleteRouteCalculatormit einem AWS SDK
    - Verwenden Sie DeleteTrackermit einem AWS SDK
    - Verwenden Sie GetDevicePositionmit einem AWS SDK
    - Verwenden Sie PutGeofencemit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Location AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Location Service mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Location Service
- Lernen Sie die Grundlagen von Amazon Location mit einem AWS SDK
- Aktionen für Amazon Location mit AWS SDKs
  - Verwenden Sie BatchUpdateDevicePositionmit einem AWS SDK
  - Verwenden Sie CalculateRoutemit einem AWS SDK
  - Verwenden Sie CreateGeofenceCollectionmit einem AWS SDK
  - Verwenden Sie CreateKeymit einem AWS SDK
  - Verwenden Sie CreateMapmit einem AWS SDK
  - Verwenden Sie CreateRouteCalculatormit einem AWS SDK
  - Verwenden Sie CreateTrackermit einem AWS SDK
  - Verwenden Sie DeleteGeofenceCollectionmit einem AWS SDK
  - Verwenden Sie DeleteKeymit einem AWS SDK
  - Verwenden Sie DeleteMapmit einem AWS SDK
  - Verwenden Sie DeleteRouteCalculatormit einem AWS SDK
  - Verwenden Sie DeleteTrackermit einem AWS SDK
  - Verwenden Sie GetDevicePositionmit einem AWS SDK
  - Verwenden Sie PutGeofencemit einem AWS SDK

Hello Amazon Location Service

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon Location Service beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
```

```
    }

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list
geofences from.
     * @return A {@link CompletableFuture} representing the result of the
asynchronous operation.
     *         The future completes when all geofences have been processed and
logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
```

```
                );
            }
        });
        return future;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListGeofenceCollections
  - ListGeofences

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  LocationClient,
  ListGeofenceCollectionsCommand,
} from "@aws-sdk/client-location";

/**
 * Lists geofences from a specified geofence collection asynchronously.
 */
export const main = async () => {
  const region = "eu-west-1";
  const locationClient = new LocationClient({ region: region });
  const listGeofenceCollParams = {
    MaxResults: 100,
  };
  try {
    const command = new ListGeofenceCollectionsCommand(listGeofenceCollParams);
```

```
    const response = await locationClient.send(command);
    const geofenceEntries = response.Entries;
    if (geofenceEntries.length === 0) {
      console.log("No Geofences were found in the collection.");
    } else {
      for (const geofenceEntry of geofenceEntries) {
        console.log(`Geofence ID: ${geofenceEntry.CollectionName}`);
      }
    }
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - ListGeofenceCollections
  - ListGeofences

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
```

```
                }
            }
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - ListGeofenceCollections
  - ListGeofences

Lernen Sie die Grundlagen von Amazon Location mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen einer Karte mit Amazon Location
- Erstellen Sie einen API-Schlüssel für Amazon Location.
- Zeigen Sie die Karten-URL an.
- Erstellen Sie eine Geofence-Sammlung.
- Speichern einer Geofence-Geometrie
- Erstellen einer Tracker-Ressource
- Aktualisieren der Position eines Geräts
- Rufen Sie die neueste Positionsaktualisierung für ein bestimmtes Gerät ab.
- Erstellen Sie einen Routenrechner.
- Ermitteln der Entfernung zwischen Seattle und Vancouver
- Verwenden Sie Amazon-Location-APIs höherer Ebene.
- Löschen Sie die Amazon-Location-Komponenten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features von Amazon Location Service zeigt.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """

            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
```

```
             keyName - The name of the API key to create (e.g., "AWSApiKey").
             collectionName - The name of the geofence collection (e.g.,
"AWSLocationCollection").
             geoId - The geographic identifier used for the geofence or map
(e.g., "geoId").
             trackerName - The name of the tracker (e.g., "geoTracker").
             calculatorName - The name of the route calculator (e.g.,
"AWSRouteCalc").
             deviceId - The ID of the device (e.g., "iPhone-112356").
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }

       String mapName = args[0];
       String keyName = args[1];
       String collectionName = args[2];
       String geoId = args[3];
       String trackerName = args[4];
       String calculatorName = args[5];
       String deviceId = args[6];

       logger.info("""
           AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
           provides location-based services for developers. This service
simplifies
           the integration of location-based features into applications, making
it
           easier to build and deploy location-aware applications.

           The AWS Location Service offers a range of location-based services,
           including:

           Maps: The service provides access to high-quality maps, satellite
imagery,\s
           and geospatial data from various providers, allowing developers to\s
           easily embed maps into their applications.

           Tracking: The Location Service enables real-time tracking of mobile
devices,\s
```

```
            assets, or other entities, allowing developers to build applications
\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),
\s
            and vice versa, enabling developers to integrate location-based
 search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
 calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName,
 calculatorName);
            logger.info(e.getMessage());
        }
    }

    public static void runScenario(String mapName, String keyName, String
 collectionName, String geoId, String trackerName, String calculatorName, String
 deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
 code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
map.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);

        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
```

```
            AWS SDK. For more information about getting the key value, see
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName
+"/tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
 geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence
 collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a
 geographic area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific
 region.

            To define a geofence, you need to specify the coordinates of a
```

```
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.

            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating
geofence: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn =
locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
```

```
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for
deviceId, but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
```

```
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver
using the route calculator.");
        waitForInputToContinue(scanner);
```

```
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
            logger.info("Successfully calculated route. The distance in
kilometers is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic
coordinates into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific
location.
            """);

        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
```

```
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName,
calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence
collection, tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
```

```
     */
    private static void cleanUp(String mapName, String keyName, String
 collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
 cause);
            } else {
                logger.info("An unexpected error occurred: {}",
 cause.getMessage(), cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon Location Service SDK.

```
public class LocationActions {
```

```
    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
    private static final Logger logger =
LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
```

```
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic
 coordinates (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes
 successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
 occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
 search", exception);
```

```
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the
detailed information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
```

```
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType ->
 {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
 occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
 search", exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
     * Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
     * This method uses the latitude and longitude of San Francisco as the input,
 and prints the resulting address.
     */
```

```
    public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: "
+ cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse
geocoding", exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }


    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the
route
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
```

```
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
```

```
                        throw new CompletionException("A conflict error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device
position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device
position: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
```

```
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();

       CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof ResourceNotFoundException) {
                   throw new CompletionException("The resource was not found: "
+ cause.getMessage(), cause);
               } else {
                   throw new CompletionException("Error updating device
position: " + exception.getMessage(), exception);
               }
           }
       });
   }


   /**
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
```

```
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
   }



   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the
geofence to
    * @param geoId          the unique identifier for the geofence
    */
   public CompletableFuture<PutGeofenceResponse> putGeofence(String
collectionName, String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
           .polygon(List.of(
               List.of(
                   List.of(-122.3381, 47.6101), // First point
                   List.of(-122.3281, 47.6101),
```

```
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String
collectionName) {
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection
was not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the
ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon
Resource Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
```

```
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map,
such as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
```

```
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }



    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
```

```
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error
occurred during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();

       return getClient().deleteKey(keyRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The key was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
               }
               logger.info("The key {} was deleted.", keyName);
           })
           .thenApply(response -> null);
   }


   /**
    * Deletes a map with the specified name.
    *
    * @param mapName the name of the map to be deleted
    * @return a {@link CompletableFuture} that completes when the map deletion
is successful, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<Void> deleteMap(String mapName) {
       DeleteMapRequest mapRequest = DeleteMapRequest.builder()
           .mapName(mapName)
           .build();

       return getClient().deleteMap(mapRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has
been deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not
found.", cause);
                    }
                    throw new CompletionException("Failed to delete the tracker:
" + exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

```
    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route
 calculator has been deleted
     * @throws CompletionException if an error occurs while deleting the route
 calculator
     *                             - If the route calculator was not found, a
 {@link ResourceNotFoundException} will be thrown
     *                             - If any other error occurs, a generic {@link
 CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
 DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was
 not found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchUpdateDevicePosition

- CalculateRoute
- CreateGeofenceCollection
- CreateKey
- CreateMap
- CreateRouteCalculator
- CreateTracker
- DeleteGeofenceCollection
- DeleteKey
- DeleteMap
- DeleteRouteCalculator
- DeleteTracker
- GetDevicePosition
- PutGeofence

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates how to use the AWS SDK for JavaScript (v3) to work with
 Amazon Location Service.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-
started.html
*/
```

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import {
  CreateMapCommand,
  CreateGeofenceCollectionCommand,
  PutGeofenceCommand,
  CreateTrackerCommand,
  BatchUpdateDevicePositionCommand,
  GetDevicePositionCommand,
  CreateRouteCalculatorCommand,
  CalculateRouteCommand,
  LocationClient,
  ConflictException,
  ResourceNotFoundException,
  DeleteGeofenceCollectionCommand,
  DeleteRouteCalculatorCommand,
  DeleteTrackerCommand,
  DeleteMapCommand,
} from "@aws-sdk/client-location";

import {
  GeoPlacesClient,
  ReverseGeocodeCommand,
  SearchNearbyCommand,
  SearchTextCommand,
  GetPlaceCommand,
  ValidationException,
} from "@aws-sdk/client-geo-places";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";

/*The inputs for this example can be edited in the ./input.json.*/
import data from "./inputs.json" with { type: "json" };

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
```

```
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
  verbose: "false",
});

const pressEnterConfirm = new ScenarioInput(
  "confirm",
  "Press Enter to continue",
  {
    type: "confirm",
    verbose: "false",
  },
);

const region = "eu-west-1";

const locationClient = new LocationClient({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Location Use demo! \n" +
    "AWS Location Service is a fully managed service offered by Amazon Web
 Services (AWS) that " +
    "provides location-based services for developers. This service simplifies " +
    "the integration of location-based features into applications, making it " +
    "Maps: The service provides access to high-quality maps, satellite imagery, "
 +
    "and geospatial data from various providers, allowing developers to " +
    "easily embed maps into their applications:\n" +
    "Tracking: The Location Service enables real-time tracking of mobile devices,
 " +
    "assets, or other entities, allowing developers to build applications " +
    "that can monitor the location of people, vehicles, or other objects.\n" +
    "Geocoding: The service provides the ability to convert addresses or " +
    "location names into geographic coordinates (latitude and longitude), " +
    "and vice versa, enabling developers to integrate location-based search " +
    "and routing functionality into their applications. " +
    "Please define values ./inputs.json for each user-defined variable used in
 this app. Otherwise the default is used:\n" +
    "- mapName: The name of the map to be create (default is 'AWSMap').\n" +
    "- keyName: The name of the API key to create (default is ' AWSApiKey')\n" +
```

```
    "- collectionName: The name of the geofence collection (default is
 'AWSLocationCollection')\n" +
    "- geoId: The geographic identifier used for the geofence or map (default is
 'geoId')\n" +
    "- trackerName: The name of the tracker (default is 'geoTracker')\n" +
    "- calculatorName: The name of the route calculator (default is
 'AWSRouteCalc')\n" +
    "- deviceId: The ID of the device (default is 'iPhone-112356')",

  { header: true },
);
const displayCreateAMap = new ScenarioOutput(
  "displayCreateAMap",
  "1. Create a map\n" +
    "An AWS Location map can enhance the user experience of your " +
    " application by providing accurate and personalized location-based " +
    " features. For example, you could use the geocoding capabilities to " +
    " allow users to search for and locate businesses, landmarks, or " +
    " other points of interest within a specific region.",
);

const sdkCreateAMap = new ScenarioAction(
  "sdkCreateAMap",
  async (/** @type {State} */ state) => {
    const createMapParams = {
      MapName: `${data.inputs.mapName}`,
      Configuration: { style: "VectorEsriNavigation" },
    };
    try {
      const command = new CreateMapCommand(createMapParams);
      const response = await locationClient.send(command);
      state.MapName = response.MapName;
      console.log("Map created. Map ARN is: ", state.MapName);
    } catch (error) {
      console.error("Error creating map: ", error);
      throw error;
    }
  },
);

const displayMapUrl = new ScenarioOutput(
  "displayMapUrl",
  "2. Display Map URL\n" +
    "When you embed a map in a web app or website, the API key is " +
```

```
    "included in the map tile URL to authenticate requests. You can " +
    "restrict API keys to specific AWS Location operations (e.g., only " +
    "maps, not geocoding). API keys can expire, ensuring temporary " +
    "access control.\n" +
    "In order to get the MAP URL you need to create and get the API Key value. "
 +
    "You can create and get the key value using the AWS Management Console under
 " +
    "Location Services. These operations cannot be completed using the " +
    "AWS SDK. For more information about getting the key value, see " +
    "the AWS Location Documentation.",
);

const sdkDisplayMapUrl = new ScenarioAction(
  "sdkDisplayMapUrl",
  async (/** @type {State} */ state) => {
    const mapURL = `https://maps.geo.aws.amazon.com/maps/v0/maps/
${state.MapName}/tiles/{z}/{x}/{y}?key=API_KEY_VALUE`;
    state.mapURL = mapURL;
    console.log(
      `Replace \'API_KEY_VALUE\' in the following URL with the value for the API
 key you create and get from the AWS Management Console under Location Services.
 This is then the Map URL you can embed this URL in your Web app:\n
${state.mapURL}`,
    );
  },
);
const displayCreateGeoFenceColl = new ScenarioOutput(
  "displayCreateGeoFenceColl",
  "3. Create a geofence collection, which manages and stores geofences.",
);

const sdkCreateGeoFenceColl = new ScenarioAction(
  "sdkCreateGeoFenceColl",
  async (/** @type {State} */ state) => {
    // Creates a new geofence collection.
    const geoFenceCollParams = {
      CollectionName: `${data.inputs.collectionName}`,
    };
    try {
      const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
      const response = await locationClient.send(command);
      state.CollectionName = response.CollectionName;
      console.log(
```

```
        `The geofence collection was successfully created:
 ${state.CollectionName}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayStoreGeometry = new ScenarioOutput(
  "displayStoreGeometry",
  "4. Store a geofence geometry in a given geofence collection. " +
    "An AWS Location geofence is a virtual boundary that defines a geographic
 area " +
    "on a map. It is a useful feature for tracking the location of " +
    "assets or monitoring the movement of objects within a specific region. " +
    "To define a geofence, you need to specify the coordinates of a " +
    "polygon that represents the area of interest. The polygon must be " +
    "defined in a counter-clockwise direction, meaning that the points of " +
    "the polygon must be listed in a counter-clockwise order. " +
    "This is a requirement for the AWS Location service to correctly " +
    "interpret the geofence and ensure that the location data is " +
    "accurately processed within the defined area.",
);

const sdkStoreGeometry = new ScenarioAction(
  "sdkStoreGeometry",
  async (/** @type {State} */ state) => {
    const geoFenceGeoParams = {
      CollectionName: `${data.inputs.collectionName}`,
      GeofenceId: `${data.inputs.geoId}`,
      Geometry: {
        Polygon: [
          [
            [-122.3381, 47.6101],
            [-122.3281, 47.6101],
            [-122.3281, 47.6201],
            [-122.3381, 47.6201],
            [-122.3381, 47.6101],
```

```
          ],
        ],
      },
    };
    try {
      const command = new PutGeofenceCommand(geoFenceGeoParams);
      const response = await locationClient.send(command);
      state.GeoFencId = response.GeofenceId;
      console.log("GeoFence created. GeoFence ID is: ", state.GeoFencId);
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message}
 \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateTracker = new ScenarioOutput(
  "displayCreateTracker",
  "5. Create a tracker resource which lets you retrieve current and historical
 location of devices.",
);

const sdkCreateTracker = new ScenarioAction(
  "sdkCreateTracker",
  async (/** @type {State} */ state) => {
    //Creates a new tracker resource in your AWS account, which you can use to
 track the location of devices.
    const createTrackerParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Description: "Created using the JavaScript V3 SDK",
      PositionFiltering: "TimeBased",
    };
    try {
      const command = new CreateTrackerCommand(createTrackerParams);
      const response = await locationClient.send(command);
      state.trackerName = response.TrackerName;
      console.log("Tracker created. Tracker name is : ", state.trackerName);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
```

```
          `A validation error occurred while creating geofence: ${caught.message}
 \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
      }
      return;
    }
  },
);
const displayUpdatePosition = new ScenarioOutput(
  "displayUpdatePosition",
  "6. Update the position of a device in the location tracking system." +
    "The AWS Location Service does not enforce a strict format for deviceId, but
 it must:\n " +
    "- Be a string (case-sensitive).\n" +
    "- Be 1-100 characters long.\n" +
    "- Contain only: Alphanumeric characters (A-Z, a-z, 0-9); Underscores (_);
 Hyphens (-); and be the same ID used when sending and retrieving positions.",
);

const sdkUpdatePosition = new ScenarioAction(
  "sdkUpdatePosition",
  async (/** @type {State} */ state) => {
    // Updates the position of a device in the location tracking system.

    const updateDevicePosParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Updates: [
        {
          DeviceId: `${data.inputs.deviceId}`,
          SampleTime: new Date(),
          Position: [-122.4194, 37.7749],
        },
      ],
    };
    try {
      const command = new BatchUpdateDevicePositionCommand(
        updateDevicePosParams,
      );
      const response = await locationClient.send(command);
      console.log(
```

```
        `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while updating the device:
 ${caught.message} \n Exiting program.`,
        );
      }
    }
  },
);
const displayRetrievePosition = new ScenarioOutput(
  "displayRetrievePosition",
  "7. Retrieve the most recent position update for a specified device.",
);

const sdkRetrievePosition = new ScenarioAction(
  "sdkRetrievePosition",
  async (/** @type {State} */ state) => {
    const devicePositionParams = {
      TrackerName: `${data.inputs.trackerName}`,
      DeviceId: `${data.inputs.deviceId}`,
    };
    try {
      const command = new GetDevicePositionCommand(devicePositionParams);
      const response = await locationClient.send(command);
      state.position = response.Position;
      console.log("Successfully fetched device position: : ", state.position);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `"The resource was not found: ${caught.message} \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
      }
      return;
    }
  },
);
const displayCreateRouteCalc = new ScenarioOutput(
```

```
  "displayCreateRouteCalc",
  "8. Create a route calculator.",
);

const sdkCreateRouteCalc = new ScenarioAction(
  "sdkCreateRouteCalc",
  async (/** @type {State} */ state) => {
    const routeCalcParams = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DataSource: "Esri",
    };
    try {
      // Creates a new route calculator with the specified name and data source.
      const command = new CreateRouteCalculatorCommand(routeCalcParams);
      const response = await locationClient.send(command);
      state.CalculatorName = response.CalculatorName;
      console.log(
        "Route calculator created successfully. Calculator name is: ",
        state.CalculatorName,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayDetermineDist = new ScenarioOutput(
  "displayDetermineDist",
  "9. Determine the distance between Seattle and Vancouver using the route
 calculator.",
);

const sdkDetermineDist = new ScenarioAction(
  "sdkDetermineDist",
  async (/** @type {State} */ state) => {
    // Calculates the distance between two locations asynchronously.
    const determineDist = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DeparturePosition: [-122.3321, 47.6062],
      DestinationPosition: [-123.1216, 49.2827],
```

```
      TravelMode: "Car",
      DistanceUnit: "Kilometers",
    };
    try {
      const command = new CalculateRouteCommand(determineDist);
      const response = await locationClient.send(command);

      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.Summary.Distance,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `Failed to calculate route: ${caught.message} \n Exiting program.`,
        );
      }
      return;
    }
  },
);
const displayUseGeoPlacesClient = new ScenarioOutput(
  "displayUseGeoPlacesClient",
  "10. Use the GeoPlacesAsyncClient to perform additional operations. " +
    "This scenario will show use of the GeoPlacesClient that enables" +
    "location search and geocoding capabilities for your applications. " +
    "We are going to use this client to perform these AWS Location tasks: \n" +
    " - Reverse Geocoding (reverseGeocode): Converts geographic coordinates into
 addresses.\n " +
    " - Place Search (searchText): Finds places based on search queries.\n " +
    " - Nearby Search (searchNearby): Finds places near a specific location.\n "
 +
    "First we will perform a Reverse Geocoding operation",
);

const sdkUseGeoPlacesClient = new ScenarioAction(
  "sdkUseGeoPlacesClient",
  async (/** @type {State} */ state) => {
    const geoPlacesClient = new GeoPlacesClient({ region: region });

    const reverseGeoCodeParams = {
      QueryPosition: [-122.4194, 37.7749],
    };
    const searchTextParams = {
```

```
      QueryText: "coffee shop",
      BiasPosition: [-122.4194, 37.7749], //San Fransisco
    };
    const searchNearbyParams = {
      QueryPosition: [-122.4194, 37.7749],
      QueryRadius: Number("1000"),
    };
    try {
      /*   Performs reverse geocoding using the AWS Geo Places API.
     Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
     This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.*/

      console.log("Use latitude 37.7749 and longitude -122.4194.");
      const command = new ReverseGeocodeCommand(reverseGeoCodeParams);
      const response = await geoPlacesClient.send(command);
      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.ResultItems[0].Distance,
      );
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log(
        "Now we are going to perform a text search using coffee shop",
      );

      /*Searches for a place using the provided search query and prints the
detailed information of the first result.
 @param searchTextParams the search query to be used for the place search (ex,
coffee shop)*/

      const command = new SearchTextCommand(searchTextParams);
      const response = await geoPlacesClient.send(command);
      const placeId = response.ResultItems[0].PlaceId.toString();
      const getPlaceCommand = new GetPlaceCommand({
        PlaceId: placeId,
```

```
      });
      const getPlaceResponse = await geoPlacesClient.send(getPlaceCommand);
      console.log(
        `Detailed Place Information: \n Name and address:
 ${getPlaceResponse.Address.Label}`,
      );

      const foodTypes = getPlaceResponse.FoodTypes;
      if (foodTypes.length) {
        console.log("Food Types: ");
        for (const foodType of foodTypes) {
          console.log("- ", foodType.LocalizedName);
        }
      } else {
        console.log("No food types available.");
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log("\nNow we are going to perform a nearby search.");
      const command = new SearchNearbyCommand(searchNearbyParams);
      const response = await geoPlacesClient.send(command);
      const resultItems = response.ResultItems;
      console.log("\nSuccessfully performed nearby search.");
      for (const resultItem of resultItems) {
        console.log("Name and address: ", resultItem.Address.Label);
        console.log("Distance: ", resultItem.Distance);
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
```

```
const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "11. Delete the AWS Location Services resources. " +
    "Would you like to delete the AWS Location Services resources? (y/n)",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    const deleteGeofenceCollParams = {
      CollectionName: `${state.CollectionName}`,
    };
    const deleteRouteCalculatorParams = {
      CalculatorName: `${state.CalculatorName}`,
    };
    const deleteTrackerParams = { TrackerName: `${state.trackerName}` };
    const deleteMapParams = { MapName: `${state.MapName}` };
    try {
      const command = new DeleteMapCommand(deleteMapParams);
      const response = await locationClient.send(command);
      console.log("Map deleted.");
    } catch (error) {
      console.log("Error deleting map: ", error);
    }
    try {
      const command = new DeleteGeofenceCollectionCommand(
        deleteGeofenceCollParams,
      );
      const response = await locationClient.send(command);
      console.log("Geofence collection deleted.");
    } catch (error) {
      console.log("Error deleting geofence collection: ", error);
    }
    try {
      const command = new DeleteRouteCalculatorCommand(
        deleteRouteCalculatorParams,
      );
      const response = await locationClient.send(command);
      console.log("Route calculator deleted.");
    } catch (error) {
      console.log("Error deleting route calculator: ", error);
    }
    try {
```

```
      const command = new DeleteTrackerCommand(deleteTrackerParams);
      const response = await locationClient.send(command);
      console.log("Tracker deleted.");
    } catch (error) {
      console.log("Error deleting tracker: ", error);
    }
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you "
 +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming
 languages, have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Location Services Scenario", [
  greet,
  pressEnter,
  displayCreateAMap,
  sdkCreateAMap,
  pressEnter,
  displayMapUrl,
  sdkDisplayMapUrl,
  pressEnter,
  displayCreateGeoFenceColl,
  sdkCreateGeoFenceColl,
  pressEnter,
  displayStoreGeometry,
  sdkStoreGeometry,
  pressEnter,
  displayCreateTracker,
  sdkCreateTracker,
  pressEnter,
  displayUpdatePosition,
  sdkUpdatePosition,
  pressEnter,
  displayRetrievePosition,
  sdkRetrievePosition,
  pressEnter,
```

```
  displayCreateRouteCalc,
  sdkCreateRouteCalc,
  pressEnter,
  displayDetermineDist,
  sdkDetermineDist,
  pressEnter,
  displayUseGeoPlacesClient,
  sdkUseGeoPlacesClient,
  pressEnter,
  displayDeleteResources,
  pressEnterConfirm,
  sdkDeleteResources,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap

- CreateRouteCalculator
- CreateTracker
- DeleteGeofenceCollection
- DeleteKey
- DeleteMap
- DeleteRouteCalculator
- DeleteTracker
- GetDevicePosition
- PutGeofence

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
```

```
            keyName - The name of the API key to create (e.g., "AWSApiKey").
            collectionName - The name of the geofence collection (e.g.,
"AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
"geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
"AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
        """
    AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile
devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.
```

```
    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)
```

```
    println(DASHES)
    println("3. Display Map URL")
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/
{z}/{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores
 geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
 area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
```

```
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking
system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId,
but it must:
            - Be a string (case-sensitive).
            - Be 1-100 characters long.
            - Contain only:
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking
system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
```

```
    println("8. Retrieve the most recent position update for a specified
device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully:
${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and
Vancouver using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific
location.

        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/
n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
```

```
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}


/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence
 collection has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}


/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)
```

```
    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)

        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex,
 coffee shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")
```

```
            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
```

```
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}


/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the
 route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters,
 Kilometers, Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
```

```
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
```

```
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude,
 latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the
 Amazon Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence
 to
 * @param geoId          the unique identifier for the geofence
```

```
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
```

```
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
```

```
    }
}

fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
            println("Invalid input. Please try again.")
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Aktionen für Amazon Location mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Location-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Location Service.

Beispiele

- Verwenden Sie BatchUpdateDevicePositionmit einem AWS SDK
- Verwenden Sie CalculateRoutemit einem AWS SDK
- Verwenden Sie CreateGeofenceCollectionmit einem AWS SDK
- Verwenden Sie CreateKeymit einem AWS SDK
- Verwenden Sie CreateMapmit einem AWS SDK
- Verwenden Sie CreateRouteCalculatormit einem AWS SDK
- Verwenden Sie CreateTrackermit einem AWS SDK
- Verwenden Sie DeleteGeofenceCollectionmit einem AWS SDK
- Verwenden Sie DeleteKeymit einem AWS SDK
- Verwenden Sie DeleteMapmit einem AWS SDK
- Verwenden Sie DeleteRouteCalculatormit einem AWS SDK
- Verwenden Sie DeleteTrackermit einem AWS SDK
- Verwenden Sie GetDevicePositionmit einem AWS SDK
- Verwenden Sie PutGeofencemit einem AWS SDK

Verwenden Sie **`BatchUpdateDevicePosition`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `BatchUpdateDevicePosition` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();

       CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
```

```
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: "
 + cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device
 position: " + exception.getMessage(), exception);
                }
            }
        });
    }
```

- Einzelheiten zur API finden Sie BatchUpdateDevicePositionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  BatchUpdateDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
const updateDevicePosParams = {
  TrackerName: `${data.inputs.trackerName}`,
  Updates: [
    {
      DeviceId: `${data.inputs.deviceId}`,
      SampleTime: new Date(),
      Position: [-122.4194, 37.7749],
```

```
    },
  ],
};
export const main = async () => {
  try {
    const command = new BatchUpdateDevicePositionCommand(updateDevicePosParams);
    const response = await locationClient.send(command);
    //console.log("response ", response.Errors[0].Error);

    console.log(
      `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      response,
    );
  } catch (error) {
    console.log("error ", error);
  }
};
```

- Einzelheiten zur API finden Sie BatchUpdateDevicePositionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
```

```
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude,
 latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- Einzelheiten zur API finden Sie BatchUpdateDevicePositionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CalculateRoute`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CalculateRoute` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the
route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
```

```
                    throw new CompletionException("Failed to calculate route: " +
 exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie CalculateRouteein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  CalculateRouteCommand,
  ResourceNotFoundException,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DeparturePosition: [-122.3321, 47.6062],
    DestinationPosition: [-123.1216, 49.2827],
    TravelMode: "Car",
    DistanceUnit: "Kilometers",
  };
  try {
    const command = new CalculateRouteCommand(routeCalcParams);
    const response = await locationClient.send(command);
```

```
    console.log(
      "Successfully calculated route. The distance in kilometers is : ",
      response.Summary.Distance,
    );
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie CalculateRoutein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the
 route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
```

```
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters,
 Kilometers, Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- Einzelheiten zur API finden Sie CalculateRoutein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateGeofenceCollection`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateGeofenceCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String
collectionName) {
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection
was not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the
ARN
    }
```

- Einzelheiten zur API finden Sie CreateGeofenceCollectionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateGeofenceCollectionCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const geoFenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
    const response = await locationClient.send(command);
    console.log(
      "Collection created. Collection name is: ",
      response.CollectionName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error("A conflict occurred. Exiting program.");
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateGeofenceCollectionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- Einzelheiten zur API finden Sie CreateGeofenceCollectionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateKey`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon
Resource Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
```

```
                    }
                    throw new CompletionException("Failed to create API key: " +
 exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
 null if the response reaches here
    }
```

- Einzelheiten zur API finden Sie CreateKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- Einzelheiten zur API finden Sie CreateKeyin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateMap`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateMap` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map,
such as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
```

```
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- Einzelheiten zur API finden Sie CreateMapin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import { CreateMapCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
```

```
const region = "eu-west-1";

export const main = async () => {
  const CreateMapCommandInput = {
    MapName: `${data.inputs.mapName}`,
    Configuration: { style: "VectorEsriNavigation" },
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateMapCommand(CreateMapCommandInput);
    const response = await locationClient.send(command);
    console.log("Map created. Map ARN is : ", response.MapArn);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Einzelheiten zur API finden Sie CreateMapin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }
```

```
    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- Einzelheiten zur API finden Sie CreateMapin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateRouteCalculator`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateRouteCalculator` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
```

```
    public CompletableFuture<CreateRouteCalculatorResponse>
 createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
 CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
 calculator: " + exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateRouteCalculatorin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateRouteCalculatorCommand,
```

```
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DataSource: "Esri",
  };
  try {
    const command = new CreateRouteCalculatorCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Route calculator created successfully. Calculator name is ",
      response.CalculatorName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateRouteCalculatorin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- Einzelheiten zur API finden Sie CreateRouteCalculatorin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateTracker`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateTracker` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
   }
```

- Einzelheiten zur API finden Sie CreateTrackerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import { CreateTrackerCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const createTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateTrackerCommand(createTrackerParams);
    const response = await locationClient.send(command);
    //state.trackerName - response.TrackerName;
    console.log("Tracker created. Tracker name is : ", response.TrackerName);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- Einzelheiten zur API finden Sie CreateTrackerin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the
 Amazon Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- Einzelheiten zur API finden Sie CreateTrackerin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteGeofenceCollection`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteGeofenceCollection` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a geofence collection asynchronously.
    *
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
                   throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
               }
               logger.info("The geofence collection {} was deleted.",
collectionName);
           })
           .thenApply(response -> null);
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteGeofenceCollectionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteGeofenceCollectionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteGeofenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new DeleteGeofenceCollectionCommand(
      deleteGeofenceCollParams,
    );
    const response = await locationClient.send(command);
    console.log("Collection deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
```

```
        `${data.inputs.collectionName} Geofence collection not found.`,
      );
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteGeofenceCollectionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence
 collection has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteGeofenceCollectionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteKey`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteKey` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error
occurred during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
 cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
 exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteKeyin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteKeyin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **DeleteMap**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteMap` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion
is successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
```

```
                    }
                    throw new CompletionException("Failed to delete map: " +
 exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteMapin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteMapCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteMapParams = {
    MapName: `${data.inputs.mapName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteMapCommand(deleteMapParams);
    const response = await locationClient.send(command);
    console.log("Map deleted.");
  } catch (caught) {
```

```
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.mapName} map not found.`);
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteMapin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteMapin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteRouteCalculator`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteRouteCalculator` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route
calculator has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                            - If the route calculator was not found, a
{@link ResourceNotFoundException} will be thrown
    *                            - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was
 not found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteRouteCalculatorin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteRouteCalculatorCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteRouteCalculatorParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
  };
```

```
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteRouteCalculatorCommand(
      deleteRouteCalculatorParams,
    );
    const response = await locationClient.send(command);
    console.log("Route calculator deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.calculatorName} route calculator not found.`,
      );
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteRouteCalculatorin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteRouteCalculatorin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteTracker`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteTracker` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has
been deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
```

```
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the tracker:
 " + exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- Einzelheiten zur API finden Sie DeleteTrackerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteTrackerCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
```

```
const region = "eu-west-1";

export const main = async () => {
  const deleteTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteTrackerCommand(deleteTrackerParams);
    const response = await locationClient.send(command);
    console.log("Tracker deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.trackerName} tracker not found.`);
      return;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteTrackerin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
```

```
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteTrackerin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetDevicePosition`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetDevicePosition` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device
position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
```

```
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
 found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device
 position: " + exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie GetDevicePositionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  GetDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
```

```
const region = "eu-west-1";

export const main = async () => {
  const locationClient = new LocationClient({ region: region });
  const deviceId = `${data.inputs.deviceId}`;
  const trackerName = `${data.inputs.trackerName}`;

  const devicePositionParams = {
    DeviceId: deviceId,
    TrackerName: trackerName,
  };
  try {
    const command = new GetDevicePositionCommand(devicePositionParams);
    const response = await locationClient.send(command);
    //state.position = response.position;
    console.log("Successfully fetched device position: ", response);
  } catch (error) {
    console.log("Error ", error);
    /*  if (caught instanceof ResourceNotFoundException) {
      console.error(
        `"The resource was not found: ${caught.message} \n Exiting program.`,
      );
    } else {
      `An unexpected error error occurred: ${caught.message} \n Exiting
 program.`;
    }
    return;*/
  }
};
```

- Einzelheiten zur API finden Sie GetDevicePositionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- Einzelheiten zur API finden Sie GetDevicePositionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`PutGeofence`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutGeofence` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the
geofence to
    * @param geoId          the unique identifier for the geofence
    */
   public CompletableFuture<PutGeofenceResponse> putGeofence(String
collectionName, String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
           .polygon(List.of(
               List.of(
                   List.of(-122.3381, 47.6101), // First point
                   List.of(-122.3281, 47.6101),
                   List.of(-122.3281, 47.6201),
                   List.of(-122.3381, 47.6201),
                   List.of(-122.3381, 47.6101) // Closing the polygon
               )
           ))
           .build();

       PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
           .collectionName(collectionName) // Specify the collection.
           .geofenceId(geoId) // Unique ID for the geofence.
           .geometry(geofenceGeometry)
           .build();

       return getClient().putGeofence(geofenceRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
 creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
 exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie PutGeofencein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  PutGeofenceCommand,
  LocationClient,
  ValidationException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
export const main = async () => {
  const geoFenceGeoParams = {
    CollectionName: `${data.inputs.collectionName}`,
    GeofenceId: `${data.inputs.geoId}`,
    Geometry: {
      Polygon: [
        [
          [-122.3381, 47.6101],
```

```
          [-122.3281, 47.6101],
          [-122.3281, 47.6201],
          [-122.3381, 47.6201],
          [-122.3381, 47.6101],
        ],
      ],
    },
  };
  try {
    const command = new PutGeofenceCommand(geoFenceGeoParams);
    const response = await locationClient.send(command);
    console.log("GeoFence created. GeoFence ID is: ", response.GeofenceId);
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Einzelheiten zur API finden Sie PutGeofencein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence
 to
 * @param geoId          the unique identifier for the geofence
 */
```

```
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- Einzelheiten zur API finden Sie PutGeofencein der API-Referenz zum AWS SDK für Kotlin.

Codebeispiele für Location Service Places mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Location Service Places mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Location Service Places – Weitere Informationen zu Location Service Places.

- API-Referenz für Location Service Places – Details zu allen verfügbaren Aktionen von Location Service Places.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Location Service Places mit AWS SDKs
  - Aktionen für Location Service Places mit AWS SDKs
    - Verwenden Sie ReverseGeocodemit einem AWS SDK
    - Verwenden Sie SearchNearbymit einem AWS SDK
    - Verwenden Sie SearchTextmit einem AWS SDK

Grundlegende Beispiele für Location Service Places mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Location Service Places mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Location Service Places mit AWS SDKs
  - Verwenden Sie ReverseGeocodemit einem AWS SDK
  - Verwenden Sie SearchNearbymit einem AWS SDK
  - Verwenden Sie SearchTextmit einem AWS SDK

Aktionen für Location Service Places mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Location Service Places-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Location Service.

Beispiele

- Verwenden Sie ReverseGeocodemit einem AWS SDK

- Verwenden Sie SearchNearbymit einem AWS SDK
- Verwenden Sie SearchTextmit einem AWS SDK

Verwenden Sie **`ReverseGeocode`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ReverseGeocode`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
```

```
                if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: "
 + cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse
 geocoding", exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- Einzelheiten zur API finden Sie ReverseGeocodein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`SearchNearby`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`SearchNearby`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Performs a nearby places search based on the provided geographic
 coordinates (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes
 successfully.
```

```
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
search", exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- Einzelheiten zur API finden Sie SearchNearbyin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`SearchText`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`SearchText`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Searches for a place using the provided search query and prints the
detailed information of the first result.
    *
    * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
    */
   public CompletableFuture<Void> searchText(String searchQuery) {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       List<Double> queryPosition = List.of(longitude, latitude);

       SearchTextRequest request = SearchTextRequest.builder()
               .queryText(searchQuery)
               .biasPosition(queryPosition)
               .build();

       return getGeoPlacesClient().searchText(request)
               .thenCompose(response -> {
                   if (response.resultItems().isEmpty()) {
                       logger.info("No places found.");
                       return CompletableFuture.completedFuture(null);
                   }

                   // Get the first place ID
                   String placeId = response.resultItems().get(0).placeId();
                   logger.info("Found Place with id: " + placeId);

                   // Fetch detailed info using getPlace
                   GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                           .placeId(placeId)
```

```
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType ->
 {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error
 occurred: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place
 search", exception);
                });
    }
```

- Einzelheiten zur API finden Sie SearchTextin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon Managed Grafana mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Managed Grafana mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon Managed Grafana Developer Guide — Weitere Informationen zu Amazon Managed Grafana.
- Amazon Managed Grafana-API-Referenz — Details zu allen verfügbaren Amazon Managed Grafana-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Amazon Managed Grafana AWS SDKs
  - Einen verwalteten Monitoring-Workspace erstellen

Szenarien für die Verwendung von Amazon Managed Grafana AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Managed Grafana mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon Managed Grafana aufrufen oder mit anderen AWS-Services kombinieren. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Einen verwalteten Monitoring-Workspace erstellen

Einen verwalteten Monitoring-Workspace erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Ihren Workspace
- Erstellen Sie einen Grafana-Arbeitsbereich
- Konfigurieren Sie die Authentifizierung
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
```

```
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"
```

```
# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."
```

```
# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
```

```
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
```

```
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
```

```
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"
```

```
    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Codebeispiele für Managed Service für Apache Flink mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Managed Service für Apache Flink mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Managed Service für Apache Flink – Weitere Informationen zu Managed Service für Apache Flink.
- API-Referenz für Managed Service für Apache Flink – Details zu allen verfügbaren Aktionen für Managed Service für Apache Flink.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Managed Service für Apache Flink mit AWS SDKs
  - Aktionen für Managed Service für Apache Flink mit AWS SDKs
    - Verwenden Sie AddApplicationInputmit einem AWS SDK
    - Verwenden Sie mit AddApplicationOutputeinem AWS SDK
    - Verwenden Sie mit CreateApplicationeinem AWS SDK
    - Verwenden Sie mit DeleteApplicationeinem AWS SDK
    - Verwenden Sie mit DescribeApplicationeinem AWS SDK
    - Verwenden Sie mit DescribeApplicationSnapshoteinem AWS SDK
    - Verwenden Sie mit DiscoverInputSchemaeinem AWS SDK
    - Verwenden Sie mit StartApplicationeinem AWS SDK
    - Verwenden Sie mit StopApplicationeinem AWS SDK
    - Verwenden Sie mit UpdateApplicationeinem AWS SDK
- Datengenerator für Managed Service für Apache Flink
  - Generieren Sie einen Kinesis-Stream mit einem Referrer mithilfe eines AWS SDK
  - Generieren Sie einen Kinesis-Stream mit Blutdruckanomalien mithilfe eines AWS SDK
  - Generieren Sie einen Kinesis-Stream mit Daten in Spalten mithilfe eines AWS SDK
  - Generieren Sie einen Kinesis-Stream mit Herzfrequenzanomalien mithilfe eines AWS SDK
  - Generieren Sie einen Kinesis-Stream mit Hotspots mithilfe eines AWS SDK

- Generieren Sie einen Kinesis-Stream mit Protokolleinträgen mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Staffeldaten mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Börsentickerdaten mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit zwei Datentypen mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Webprotokolldaten mithilfe eines AWS SDK

Grundlegende Beispiele für Managed Service für Apache Flink mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Managed Service für Apache Flink mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Managed Service für Apache Flink mit AWS SDKs
  - Verwenden Sie AddApplicationInputmit einem AWS SDK
  - Verwenden Sie mit AddApplicationOutputeinem AWS SDK
  - Verwenden Sie mit CreateApplicationeinem AWS SDK
  - Verwenden Sie mit DeleteApplicationeinem AWS SDK
  - Verwenden Sie mit DescribeApplicationeinem AWS SDK
  - Verwenden Sie mit DescribeApplicationSnapshoteinem AWS SDK
  - Verwenden Sie mit DiscoverInputSchemaeinem AWS SDK
  - Verwenden Sie mit StartApplicationeinem AWS SDK
  - Verwenden Sie mit StopApplicationeinem AWS SDK
  - Verwenden Sie mit UpdateApplicationeinem AWS SDK

Aktionen für Managed Service für Apache Flink mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Managed Service for Apache Flink-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Managed Service für Apache Flink.

Beispiele

- Verwenden Sie AddApplicationInputmit einem AWS SDK
- Verwenden Sie mit AddApplicationOutputeinem AWS SDK
- Verwenden Sie mit CreateApplicationeinem AWS SDK
- Verwenden Sie mit DeleteApplicationeinem AWS SDK
- Verwenden Sie mit DescribeApplicationeinem AWS SDK
- Verwenden Sie mit DescribeApplicationSnapshoteinem AWS SDK
- Verwenden Sie mit DiscoverInputSchemaeinem AWS SDK
- Verwenden Sie mit StartApplicationeinem AWS SDK
- Verwenden Sie mit StopApplicationeinem AWS SDK
- Verwenden Sie mit UpdateApplicationeinem AWS SDK

Verwenden Sie **`AddApplicationInput`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`AddApplicationInput`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
```

```
        self.create_timestamp = None


    def add_input(self, input_prefix, stream_arn, input_schema):
        """
        Adds an input stream to the application. The input stream data is mapped
        to an in-application stream that can be processed by your code running in
        Kinesis Data Analytics.

        :param input_prefix: The prefix prepended to in-application input stream
names.
        :param stream_arn: The ARN of the input stream.
        :param input_schema: A schema that maps the data in the input stream to
the
                             runtime environment. This can be automatically
generated
                             by using `discover_input_schema` or you can create
it
                             yourself.
        :return: Metadata about the newly added input.
        """
        try:
            response = self.analytics_client.add_application_input(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Input={
                    "NamePrefix": input_prefix,
                    "KinesisStreamsInput": {"ResourceARN": stream_arn},
                    "InputSchema": input_schema,
                },
            )
            self.version_id = response["ApplicationVersionId"]
            logger.info("Add input stream %s to application %s.", stream_arn,
self.name)
        except ClientError:
            logger.exception(
                "Couldn't add input stream %s to application %s.", stream_arn,
self.name
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie AddApplicationInputin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`AddApplicationOutput`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`AddApplicationOutput`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps
 data
        from the specified in-application stream to the output stream.

        :param in_app_stream_name: The name of the in-application stream to map
                                   to the output stream.
```

```
        :param output_arn: The ARN of the output stream.
        :return: A list of metadata about the output resources currently assigned
                 to the application.
        """
        try:
            response = self.analytics_client.add_application_output(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Output={
                    "Name": in_app_stream_name,
                    "KinesisStreamsOutput": {"ResourceARN": output_arn},
                    "DestinationSchema": {"RecordFormatType": "JSON"},
                },
            )
            outputs = response["OutputDescriptions"]
            self.version_id = response["ApplicationVersionId"]
            logging.info(
                "Added output %s to %s, which now has %s outputs.",
                output_arn,
                self.name,
                len(outputs),
            )
        except ClientError:
            logger.exception("Couldn't add output %s to %s.", output_arn,
self.name)
            raise
        else:
            return outputs
```

- Einzelheiten zur API finden Sie AddApplicationOutputin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def create(self, app_name, role_arn, env="SQL-1_0"):
        """
        Creates a Kinesis Data Analytics application.

        :param app_name: The name of the application.
        :param role_arn: The ARN of a role that can be assumed by Kinesis Data
                         Analytics and grants needed permissions.
        :param env: The runtime environment of the application, such as SQL. Code
                    uploaded to the application runs in this environment.
        :return: Metadata about the newly created application.
        """
        try:
            response = self.analytics_client.create_application(
                ApplicationName=app_name,
                RuntimeEnvironment=env,
                ServiceExecutionRole=role_arn,
            )
            details = response["ApplicationDetail"]
```

```
            self._update_details(details)
            logger.info("Application %s created.", app_name)
        except ClientError:
            logger.exception("Couldn't create application %s.", app_name)
            raise
        else:
            return details
```

- Einzelheiten zur API finden Sie CreateApplicationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None
```

```
    def delete(self):
        """
        Deletes an application.
        """
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp
            )
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- Einzelheiten zur API finden Sie DeleteApplicationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
```

```
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe(self, name):
        """
        Gets metadata about an application.

        :param name: The name of the application to look up.
        :return: Metadata about the application.
        """
        try:
            response =
self.analytics_client.describe_application(ApplicationName=name)
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Got metadata for application %s.", name)
        except ClientError:
            logger.exception("Couldn't get metadata for application %s.", name)
            raise
        else:
            return details
```

- Einzelheiten zur API finden Sie DescribeApplicationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeApplicationSnapshot`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeApplicationSnapshot`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe_snapshot(self, application_name, snapshot_name):
        """
        Gets metadata about a previously saved application snapshot.

        :param application_name: The name of the application.
        :param snapshot_name: The name of the snapshot.
        :return: Metadata about the snapshot.
        """
        try:
            response = self.analytics_client.describe_application_snapshot(
                ApplicationName=application_name, SnapshotName=snapshot_name
            )
            snapshot = response["SnapshotDetails"]
            logger.info(
                "Got metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
            raise
        else:
            return snapshot
```

- Einzelheiten zur API finden Sie DescribeApplicationSnapshotin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DiscoverInputSchema`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DiscoverInputSchema`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def discover_input_schema(self, stream_arn, role_arn):
        """
        Discovers a schema that maps data in a stream to a format that is usable
 by
        an application's runtime environment. The stream must be active and have
        enough data moving through it for the service to sample. The returned
 schema
        can be used when you add the stream as an input to the application or you
 can
        write your own schema.
```

```
        :param stream_arn: The ARN of the stream to map.
        :param role_arn: A role that lets Kinesis Data Analytics read from the
stream.
        :return: The discovered schema of the data in the input stream.
        """
        try:
            response = self.analytics_client.discover_input_schema(
                ResourceARN=stream_arn,
                ServiceExecutionRole=role_arn,
                InputStartingPositionConfiguration={"InputStartingPosition":
"NOW"},
            )
            schema = response["InputSchema"]
            logger.info("Discovered input schema for stream %s.", stream_arn)
        except ClientError:
            logger.exception(
                "Couldn't discover input schema for stream %s.", stream_arn
            )
            raise
        else:
            return schema
```

- Einzelheiten zur API finden Sie DiscoverInputSchemain AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`StartApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`StartApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def start(self, input_id):
        """
        Starts an application. After the application is running, it reads from
 the
        specified input stream and runs the application code on the incoming
 data.

        :param input_id: The ID of the input to read.
        """
        try:
            self.analytics_client.start_application(
                ApplicationName=self.name,
                RunConfiguration={
                    "SqlRunConfigurations": [
                        {
                            "InputId": input_id,
                            "InputStartingPositionConfiguration": {
                                "InputStartingPosition": "NOW"
                            },
                        }
                    ]
                },
            )
            logger.info("Started application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't start application %s.", self.name)
            raise
```

- Einzelheiten zur API finden Sie StartApplicationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`StopApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`StopApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def stop(self):
        """
        Stops an application. This stops the application from processing data but
        does not delete any resources.
        """
        try:
            self.analytics_client.stop_application(ApplicationName=self.name)
            logger.info("Stopping application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't stop application %s.", self.name)
            raise
```

- Einzelheiten zur API finden Sie StopApplicationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UpdateApplication`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateApplication`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Code aktualisiert, der in einer vorhandenen Anwendung ausgeführt wird.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def update_code(self, code):
        """
        Updates the code that runs in the application. The code must run in the
```

```
        runtime environment of the application, such as SQL. Application code
        typically reads data from in-application streams and transforms it in
some way.

        :param code: The code to upload. This completely replaces any existing
code
                     in the application.
        :return: Metadata about the application.
        """
        try:
            response = self.analytics_client.update_application(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                ApplicationConfigurationUpdate={
                    "ApplicationCodeConfigurationUpdate": {
                        "CodeContentTypeUpdate": "PLAINTEXT",
                        "CodeContentUpdate": {"TextContentUpdate": code},
                    }
                },
            )
            details = response["ApplicationDetail"]
            self.version_id = details["ApplicationVersionId"]
            logger.info("Update code for application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't update code for application %s.",
self.name)
            raise
        else:
            return details
```

- Einzelheiten zur API finden Sie UpdateApplicationin AWS SDK for Python (Boto3) API Reference.

Datengenerator für Managed Service für Apache Flink

Die folgenden Codebeispiele zeigen, wie Managed Service für Apache Flink mit SDKs verwendet wird. AWS

Beispiele

- Generieren Sie einen Kinesis-Stream mit einem Referrer mithilfe eines AWS SDK

- Generieren Sie einen Kinesis-Stream mit Blutdruckanomalien mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Daten in Spalten mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Herzfrequenzanomalien mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Hotspots mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Protokolleinträgen mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Staffeldaten mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Börsentickerdaten mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit zwei Datentypen mithilfe eines AWS SDK
- Generieren Sie einen Kinesis-Stream mit Webprotokolldaten mithilfe eines AWS SDK

Generieren Sie einen Kinesis-Stream mit einem Referrer mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit einem Referrer erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
```

```
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Blutdruckanomalien mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Blutdruckanomalien erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class PressureType(Enum):
    low = "LOW"
    normal = "NORMAL"
    high = "HIGH"
```

```
def get_blood_pressure(pressure_type):
    pressure = {"BloodPressureLevel": pressure_type.value}
    if pressure_type == PressureType.low:
        pressure["Systolic"] = random.randint(50, 80)
        pressure["Diastolic"] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure["Systolic"] = random.randint(90, 120)
        pressure["Diastolic"] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure["Systolic"] = random.randint(130, 200)
        pressure["Diastolic"] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure


def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low
            if rnd < 0.005
            else PressureType.high
            if rnd > 0.995
            else PressureType.normal
        )
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Daten in Spalten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Daten in Spalten erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"Col_A": "a", "Col_B": "b", "Col_C": "c", "Col_E_Unstructured":
 "x,y,z"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Herzfrequenzanomalien mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Herzfrequenzanomalien erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class RateType(Enum):
    normal = "NORMAL"
    high = "HIGH"


def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {"heartRate": rate, "rateType": rate_type.value}


def generate(stream_name, kinesis_client, output=True):
    while True:
```

```
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Hotspots mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit einem Hotspot erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"
```

```
def get_hotspot(field, spot_size):
    hotspot = {
        "left": field["left"] + random.random() * (field["width"] - spot_size),
        "width": spot_size,
        "top": field["top"] + random.random() * (field["height"] - spot_size),
        "height": spot_size,
    }
    return hotspot


def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        "x": rectangle["left"] + random.random() * rectangle["width"],
        "y": rectangle["top"] + random.random() * rectangle["height"],
        "is_hot": "Y" if rectangle is hotspot else "N",
    }
    return {"Data": json.dumps(point), "PartitionKey": "partition_key"}


def generate(
    stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client
):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)
        ]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)
```

```
if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={"left": 0, "width": 10, "top": 0, "height": 10},
        hotspot_size=1,
        hotspot_weight=0.2,
        batch_size=10,
        kinesis_client=boto3.client("kinesis"),
    )
```

Generieren Sie einen Kinesis-Stream mit Protokolleinträgen mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Protokolleinträgen erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "
        '"GET /index.php HTTP/1.1" 200 125 "-" '
        '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }
```

```
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Staffeldaten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit gestaffelten Daten erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
```

```
    return {
        "EVENT_TIME": event_time.isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and
 ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey",
            )
            time.sleep(10)


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Börsentickerdaten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Börsentickerdaten erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit zwei Datentypen mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit zwei Datentypen erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"


def get_order(order_id, ticker):
    return {
        "RecordType": "Order",
        "Oid": order_id,
        "Oticker": ticker,
        "Oprice": random.randint(500, 10000),
        "Otype": "Sell",
    }


def get_trade(order_id, trade_id, ticker):
    return {
        "RecordType": "Trade",
        "Tid": trade_id,
        "Toid": order_id,
        "Tticker": ticker,
        "Tprice": random.randint(0, 3000),
    }


def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
```

```
        ticker = random.choice(["AAAA", "BBBB", "CCCC"])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
 PartitionKey=PARTITION_KEY
        )
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY,
            )
        order_id += 1


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren Sie einen Kinesis-Stream mit Webprotokolldaten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Webprotokolldaten erstellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3
```

```
STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "log": "192.168.254.30 - John [24/May/2004:22:01:02 -0700] "
        '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Codebeispiele für AWS Marketplace Katalog-API verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie die AWS Marketplace Catalog API mit einem AWS Software Development Kit (SDK) verwenden.

Weitere -Quellen

- AWS Marketplace Verkäuferleitfaden zur Katalog-API — Weitere Informationen zur AWS Marketplace Katalog-API.
- AWS Marketplace API-Referenz für die Katalog-API — Einzelheiten zu allen verfügbaren AWS Marketplace Katalog-API-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- AMI-Produkte für AWS Marketplace Catalog-API
  - Fügen Sie einem vorhandenen AMI-Produkt eine Dimension hinzu und aktualisieren Sie die Preisbedingungen für das Angebot mithilfe eines AWS SDK
  - Fügen Sie eine Region hinzu, in der ein AMI-Produkt mithilfe eines bereitgestellt wird AWS SDK
  - Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung unter Verwendung eines AWS SDK
  - Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung unter Verwendung eines AWS SDK
  - Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung unter Verwendung eines AWS SDK
  - Erstellen Sie einen Entwurf eines AMI-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK
  - Beschränken Sie eine Region, in der ein AMI-Produkt bereitgestellt wird, mithilfe eines AWS SDK
  - Beschränken Sie die Sichtbarkeit von Produkten mit einem AWS SDK
  - Geben Sie mit einem an, ob AMI-Assets in neuen Regionen bereitgestellt werden AWS SDK
- Angebote von Channel-Partnern für AWS Marketplace Catalog-API
  - Erstellen Sie einen CPPO-Entwurf für einen beliebigen Produkttyp mithilfe eines AWS SDK
  - Erstellen Sie ein privates Ersatzangebot zur Wiederverkaufsgenehmigung auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK
  - Listet alle CPPOs auf, die von einem Channel-Partner mit einem erstellt wurden AWS SDK
  - Listet alle geteilten Wiederverkaufsberechtigungen auf, die einem Vertriebspartner unter Verwendung eines zur Verfügung stehen AWS SDK
  - Veröffentlichen Sie ein CPPO und fügen Sie eine Käufer-EULA mit einem an AWS SDK
  - Veröffentlichen Sie ein CPPO mit einer einmaligen Wiederverkaufsgenehmigung und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK
  - Veröffentlichen Sie einen CPPO-Entwurf und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK
  - Aktualisieren Sie das Ablaufdatum eines CPPO mithilfe eines AWS SDK
- Container-Produkte für AWS Marketplace Catalog-API
  - Erstellen Sie einen Entwurf eines Container-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK

- Erstellen Sie ein limitiertes Container-Produkt mit öffentlichem Angebot und Vertragspreisen unter Verwendung eines AWS SDK
- Entitäten für AWS Marketplace Catalog-API
  - Beschreiben Sie alle Entitäten in einem einzigen Aufruf mit einem AWS SDK
  - Listet alle Angebote im Zusammenhang mit einem Produkt auf und beschreibt sie mithilfe eines AWS SDK
- Angebote für AWS Marketplace Catalog-API
  - Erstellen Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt und erstellen Sie ein privates Angebot mit einem AWS SDK
  - Erstellen Sie einen Entwurf eines privaten Angebots für ein AMI- oder SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung und einem flexiblen Zahlungsplan für ein AMI-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit stündlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Angebot mit gestaffelten Vertragspreisen für ein SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK
  - Erstellen Sie ein privates Ersatzangebot auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK

- Beschreiben Sie ein öffentliches Angebot mit einem AWS SDK
- Verfallen Sie ein privates Angebot mit einem AWS SDK
- Listet alle privaten Angebote mit einem auf AWS SDK
- Listet veröffentlichte öffentliche und private Angebote für eine bestimmte Produkt-ID auf. Verwenden Sie dazu AWS SDK
- Aktualisieren Sie ein Angebot, um einen Vertrag mit Pay-As-You-Go Preisgestaltung anzuwenden, indem Sie einen AWS SDK
- Aktualisieren Sie ein Angebot so, dass es eine stündliche jährliche Preisgestaltung vorsieht. Verwenden Sie dazu AWS SDK
- Aktualisieren Sie ein Angebot, um Targeting auf bestimmte geografische Regionen anzuwenden, indem Sie AWS SDK
- Aktualisieren Sie den Namen und die Beschreibung eines öffentlichen Angebots mithilfe eines AWS SDK
- Aktualisieren Sie die EULA eines Angebots mithilfe eines AWS SDK
- Aktualisieren Sie das Ablaufdatum eines privaten Angebots auf ein future Datum mithilfe eines AWS SDK
- Aktualisieren Sie die kostenlose Testdauer eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt mithilfe eines AWS SDK
- Aktualisieren Sie die Rückerstattungsrichtlinie eines Angebots mithilfe eines AWS SDK

- Produkte für AWS Marketplace Catalog-API
  - Beschreiben Sie ein AMI-, SaaS- oder Container-Produkt mit einem AWS SDK
  - Alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten, indem Sie eine AWS SDK
- Genehmigung zum Weiterverkauf für AWS Marketplace Catalog-API
  - Erstellen Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp mithilfe eines AWS SDK
  - Beschreiben Sie eine Wiederverkaufsautorisierung mit einem AWS SDK
  - Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis mit einem privaten Angebot unter Verwendung eines AWS SDK
  - Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum mithilfe eines AWS SDK

- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufserlaubnis mit einem Ablaufdatum und einer EULA, die dem Käufer unter Verwendung eines AWS SDK
- Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie die mehrfach verwendbare Wiederverkaufsgenehmigung mit Ablauf und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum mithilfe eines AWS SDK
- Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie eine benutzerdefinierte EULA hinzu, indem Sie AWS SDK
- Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie mithilfe eines AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung und fügen Sie einen flexiblen Zahlungsplan hinzu, indem Sie einen AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis für einen beliebigen Produkttyp und fügen Sie eine EULA hinzu, indem Sie eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsautorisierung für und fügen Sie hinzu, ob es sich um eine Verlängerung handelt. Verwenden Sie dazu AWS SDK
- Beschränken Sie die Wiederverkaufsautorisierung mithilfe eines AWS SDK
- Aktualisieren Sie den Namen und die Beschreibung der einmaligen oder mehrfach verwendbaren Wiederverkaufsautorisierung mithilfe eines AWS SDK

- SaaS-Produkte für AWS Marketplace Catalog-API
  - Erstellen Sie einen Entwurf eines SaaS-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK

- Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertragspreisen unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertrag und Pay-As-You-Go Preisgestaltung unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder eingeschränktes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen unter Verwendung eines AWS SDK
- Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot mit einem AWS SDK
- Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot anhand eines vorhandenen Entwurfs mithilfe eines AWS SDK
- Aktualisieren Sie die Dimensionen eines AMI- oder SaaS-Produkts mithilfe eines AWS SDK

- Dienstprogramme für AWS Marketplace Catalog-API
  - Hilfsprogramme zum Starten eines Changesets mit einem AWS SDK

AMI-Produkte für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Fügen Sie einem vorhandenen AMI-Produkt eine Dimension hinzu und aktualisieren Sie die Preisbedingungen für das Angebot mithilfe eines AWS SDK
- Fügen Sie eine Region hinzu, in der ein AMI-Produkt mithilfe eines bereitgestellt wird AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung unter Verwendung eines AWS SDK
- Erstellen Sie einen Entwurf eines AMI-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK
- Beschränken Sie eine Region, in der ein AMI-Produkt bereitgestellt wird, mithilfe eines AWS SDK
- Beschränken Sie die Sichtbarkeit von Produkten mit einem AWS SDK

- Geben Sie mit einem an, ob AMI-Assets in neuen Regionen bereitgestellt werden AWS SDK

Fügen Sie einem vorhandenen AMI-Produkt eine Dimension hinzu und aktualisieren Sie die Preisbedingungen für das Angebot mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine Dimension zu einem vorhandenen AMI-Produkt hinzufügen und die Preisbedingungen für das Angebot aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
```

```
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a dimension to an existing
 AMI product and update the offer pricing terms.
```

```
CAPI-23
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Add dimension for AMI product")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Fügen Sie eine Region hinzu, in der ein AMI-Produkt mithilfe eines bereitgestellt wird AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine Region hinzufügen, in der ein AMI-Produkt bereitgestellt wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a region where my
AMI product is deployed
CAPI-25A
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)
```

```
    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
```

```
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
```

```
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
```

```
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "VideoUrls": [],
                "AdditionalResources": [],
                "SupportDescription": "For support, contact support@example.com"
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
```

```
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
```

```
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
```

```
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-annual pricing and standard or custom EULA
CAPI-06
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly-annual pricing
 and standard EULA",
    )
```

```
    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
```

```
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
```

```
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
```

```
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
```

```
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
```

```
                }
            ]
        }
    },
    {
        "ChangeType": "AddDimensions",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": [
            {
                "Key": "t2.micro",
                "Description": "t2.micro",
                "Name": "t2.micro",
                "Types": [
                    "Metered"
                ],
                "Unit": "Hrs"
            }
        ]
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PositiveTargeting": {
                "BuyerAccounts": [
                    "111111111111",
                    "222222222222"
                ]
            }
        }
    },
    {
        "ChangeType": "ReleaseProduct",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
```

```
                        }
                    ]
                },
                {
                    "Type": "RecurringPaymentTerm",
                    "CurrencyCode": "USD",
                    "BillingPeriod": "Monthly",
                    "Price": "15.0"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
```

```
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-monthly pricing and standard or custom EULA
CAPI-08
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "create limited AMI product and public offer with hourly-monthly pricing
 and standard EULA",
    )
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
```

```
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
```

```
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
```

```
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create a public or limited AMI
 product
and public offer with hourly pricing and standard or custom EULA
CAPI-07
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie einen Entwurf eines AMI-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie AMI-Produktentwürfe mit einem Entwurf eines öffentlichen Angebots erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an AMI draft product
with a draft public offer.
CAPI-02
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "AMI draft product with draft public offer",
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Beschränken Sie eine Region, in der ein AMI-Produkt bereitgestellt wird, mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine Region einschränken, in der ein AMI-Produkt bereitgestellt wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
```

```
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to restrict a region where my
AMI product is deployed
CAPI-25B
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Beschränken Sie die Sichtbarkeit von Produkten mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Sichtbarkeit von Produkten einschränken.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change a product visibility to
 restricted
CAPI-17
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict existing AMI")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Geben Sie mit einem an, ob AMI-Assets in neuen Regionen bereitgestellt werden AWS SDK

Die folgenden Codebeispiele zeigen, wie angegeben wird, ob AMI-Assets in neuen Regionen eingesetzt werden, die von AWS zur Unterstützung future Regionen entwickelt wurden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
```

```
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to modify a product to support
 all future regions
CAPI-26
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update future region support")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Angebote von Channel-Partnern für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Erstellen Sie einen CPPO-Entwurf für einen beliebigen Produkttyp mithilfe eines AWS SDK
- Erstellen Sie ein privates Ersatzangebot zur Wiederverkaufsgenehmigung auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK
- Listet alle CPPOs auf, die von einem Channel-Partner mit einem erstellt wurden AWS SDK
- Listet alle geteilten Wiederverkaufsberechtigungen auf, die einem Vertriebspartner unter Verwendung eines zur Verfügung stehen AWS SDK
- Veröffentlichen Sie ein CPPO und fügen Sie eine Käufer-EULA mit einem an AWS SDK
- Veröffentlichen Sie ein CPPO mit einer einmaligen Wiederverkaufsgenehmigung und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK
- Veröffentlichen Sie einen CPPO-Entwurf und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK
- Aktualisieren Sie das Ablaufdatum eines CPPO mithilfe eines AWS SDK

Erstellen Sie einen CPPO-Entwurf für einen beliebigen Produkttyp mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen CPPO-Entwurf für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn für Käufer veröffentlichen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer name"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” CPPO
for any product type (AMI/SaaS/Container) that can be reviewed internally
before publishing to buyers
CAPI-60
"""
import os

import utils.start_changeset as sc  # noqa: E402
```

```
import utils.stringify_details as sd  # noqa: E402

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Create a draft CPPO offer for a product")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Ersatzangebot zur Wiederverkaufsgenehmigung auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein privates Ersatzangebot zur Genehmigung des Wiederverkaufs aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with
 contract pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
```

```
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
```

```
                "Description": "Test private resale replacement offer with
contract pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
"$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier":
 "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a resale authorization
 replacement private offer
from an existing agreement with contract pricing
CAPI-96
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create resale authorization replacement private offer with contract
 pricing",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Listet alle CPPOs auf, die von einem Channel-Partner mit einem erstellt wurden AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle CPPOs auflisten, die von einem Channel-Partner erstellt wurden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
```

```
 List<String> entityIdList = new ArrayList<String>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER)
   .maxResults(10)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
  entityIdList.add(entitySummary.entityId());
 }

 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }
 }

 // filter for CPPO offers: ResaleAuthorizationId exists in Details

 List<String> cppoOfferIds = new ArrayList<String>();

 for (String entityId : entityIdList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entityId)
    .build();
```

```
    DescribeEntityResponse describeEntityResponse =
  marketplaceCatalogClient.describeEntity(describeEntityRequest);

    Document resaleAuthorizationDocument =
  describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
    String resaleAuthorizationId = resaleAuthorizationDocument != null ?
  resaleAuthorizationDocument.asString() : "";

    if (!resaleAuthorizationId.isEmpty()) {
        cppoOfferIds.add(resaleAuthorizationId);
    }
   }
   return cppoOfferIds;
  }

}
```

- Einzelheiten zur API finden Sie ListEntitiesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all Channel Partner
 Offers
in an account

Program executed with no arguments:
ie. python3 list_all_cppo_offers.py
```

```
CAPI-93
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_offer_entities():
    """
    Returns a list of all offers in the account
    """

    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(Catalog="AWSMarketplace",
 EntityType="Offer")
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results are paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="Offer",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise
```

```
        if "NextToken" in response:
            response_list.append(response)

    return response_list


def build_offer_list(response_list):
    """
    Cleans up list_entities response list with just list of offer IDs
    """
    offer_list = []

    for response in response_list:
        for entity in response["EntitySummaryList"]:
            offer_list.append(entity["EntityId"])

    return offer_list


def check_offer_resaleauth(offer_id):
    """
    Checks to see if an offer is based on a resale authorization
    """
    offer_response = describe_entity(offer_id)
    offer_details = json.loads(offer_response["Details"])
    if offer_details is None:
        offer_details = offer_response["DetailsDocument"]
    if "ResaleAuthorizationId" in offer_details and
 offer_details["ResaleAuthorizationId"] is not None:
        return offer_id
    else:
        return None


def describe_entity(entity_id):
    """
    General purpose describe entity call
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )
    except ClientError as e:
```

```
        logging.exception(f"Couldn't describe entity. {e}")
        raise

    return response


def get_resaleauth_offers():
    """
    Returns a list of all offers in the account that are
    based on a resale authorization
    """
    resale_offer_list = []

    response_list = get_offer_entities()
    offer_list = build_offer_list(response_list)
    for offer in offer_list:
        print ("offer id " + offer)
        offer_info = check_offer_resaleauth(offer)
        if offer_info is not None:
            resale_offer_list.append(offer_info)

    return resale_offer_list


if __name__ == "__main__":
    print(get_resaleauth_offers())
```

- Einzelheiten zur API finden Sie ListEntitiesin AWS SDK for Python (Boto3) API Reference.

Listet alle geteilten Wiederverkaufsberechtigungen auf, die einem Vertriebspartner unter Verwendung eines zur Verfügung stehen AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle gemeinsamen Genehmigungen des Wiederverkaufs auflisten, die einem Channel-Partner zur Verfügung stehen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {

 /*
  * list all resale authorizations shared to an account
  */
 public static void main(String[] args) {

  List<ListEntitiesResponse> responseList = getListEntityResponseList();
  ReferenceCodesUtils.formatOutput(responseList);
 }

 public static List<ListEntitiesResponse> getListEntityResponseList() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  List<ListEntitiesResponse> responseList = new
 ArrayList<ListEntitiesResponse>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  responseList.add(listEntitiesResponse);

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest = ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
     .maxResults(10)
     .ownershipType(OWNERSHIP_TYPE_SHARED)
     .nextToken(listEntitiesResponse.nextToken())
     .build();

   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

   responseList.add(listEntitiesResponse);
  }
  return responseList;
 }

}
```

- Einzelheiten zur API finden Sie ListEntitiesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all resale authorizations
shared to an account

Program executed with no arguments:
ie. python3 list_all_resale_authorizations.py

CAPI-94
"""

import logging

import boto3
import utils.helpers as hlp  # noqa: E402
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_shared_entities():
    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType="ResaleAuthorization",
            OwnershipType="SHARED",
```

```
        )
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results can be paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="ResaleAuthorization",
                OwnershipType="SHARED",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


if __name__ == "__main__":
    response_list = get_shared_entities()
    hlp.pretty_print_datetime(response_list)
```

- Einzelheiten zur API finden Sie ListEntitiesin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie ein CPPO und fügen Sie eine Käufer-EULA mit einem an AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein CPPO veröffentlichen und eine Käufer-EULA anhängen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
```

```
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
       ]
         }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
```

```
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
        ]
          }
         },
         {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
         },
         {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
         },
         {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish CPPO
for any product type (AMI/SaaS/Container) and append buyer EULA
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
```

```
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish CPPO for any product type and append buyer EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie ein CPPO mit einer einmaligen Wiederverkaufsgenehmigung und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie ein CPPO mithilfe einer einmaligen Wiederverkaufsgenehmigung für AMI-, SaaS- oder Container-Produkte veröffentlicht und der Preisaufschlag aktualisiert wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
                }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to Create CPPO using one-time
 resale
authorization on AMI, SaaS or Container products and update price markup
CAPI-63
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create CPPO using one-time resale authorization and update price markup"
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie einen CPPO-Entwurf und aktualisieren Sie den Preisaufschlag mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen CPPO-Entwurf veröffentlichen und den Preisaufschlag aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
```

```
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish “draft” CPPO
for any product type (AMI/SaaS/Container) and update price markup
CAPI-72
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish draft CPPO for any product type adn update price markup"
```

```
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie das Ablaufdatum eines CPPO mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie das Ablaufdatum eines CPPO aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update the expiry
date of a CPPO offer
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update the expiry date of a CPPO offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Container-Produkte für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Erstellen Sie einen Entwurf eines Container-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK
- Erstellen Sie ein limitiertes Container-Produkt mit öffentlichem Angebot und Vertragspreisen unter Verwendung eines AWS SDK

Erstellen Sie einen Entwurf eines Container-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Container-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
```

```
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
```

```
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an container draft
 product
with a draft public offer.
CAPI-03
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft container product with a draft public offer",
```

```
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein limitiertes Container-Produkt mit öffentlichem Angebot und Vertragspreisen unter Verwendung eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein limitiertes Container-Produkt mit einem öffentlichen Angebot, Vertragspreisen und Standard-EULA erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
```

```
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
```

```
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
```

```
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
```

```
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
```

```
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create limited container
product with public offer, contract pricing and standard EULA
CAPI-15
"""


import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited container product with public offer contract pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Entitäten für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Beschreiben Sie alle Entitäten in einem einzigen Aufruf mit einem AWS SDK
- Listet alle Angebote im Zusammenhang mit einem Produkt auf und beschreibt sie mithilfe eines AWS SDK

Beschreiben Sie alle Entitäten in einem einzigen Aufruf mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie alle Entitäten in einem einzigen Aufruf beschrieben werden können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {

    /*
     * BatchDescribe my entities in a single call and
```

```
    *  check if it contains all the information I need to know about the
entities.
    */
   public static void main(String[] args) {

       MarketplaceCatalogClient marketplaceCatalogClient =
               MarketplaceCatalogClient.builder()
                       .httpClient(ApacheHttpClient.builder().build())
                       .credentialsProvider(ProfileCredentialsProvider.create())
                       .build();

       BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
               BatchDescribeEntitiesRequest.builder()
                       .entityRequestList(Arrays.asList(
                               EntityRequest.builder()

.catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                       .build(),
                               EntityRequest.builder()

.catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
                                       .build()))
                       .build();

       BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

       // Reading the successful entities response
       Map<String, EntityDetail> entityDetailsMap =
batchDescribeEntitiesResponse.entityDetails();
       for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet())
{
           System.out.println("EntityId: " + entry.getKey());
           ReferenceCodesUtils.formatOutput(entry.getValue());
       }

       // Logging the failed entities error details
       Map<String, BatchDescribeErrorDetail> entityErrorsMap =
batchDescribeEntitiesResponse.errors();
       for (Map.Entry<String, BatchDescribeErrorDetail> entry :
entityErrorsMap.entrySet()) {
           System.out.println(String.format("EntityId: %s, ErrorCode: %s,
ErrorMessage: %s", entry.getKey(),
```

```
                    entry.getValue().errorCode(),
 entry.getValue().errorMessage()));
        }
    }
}
```

- Einzelheiten zur API finden Sie BatchDescribeEntitiesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to describe for multiple entities
 information in the AWS Marketplace Catalog
CAPI-98
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"
OFFER_ID = "offer-1111111111111"
MARKETPLACE_CATALOG = "AWSMarketplace"
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_entities_information(mp_client):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    entity_request_list_param = [
        {'EntityId': PRODUCT_ID, 'Catalog': MARKETPLACE_CATALOG},
        {'EntityId': OFFER_ID, 'Catalog': MARKETPLACE_CATALOG}
    ]
    try:
        response = mp_client.batch_describe_entities(
            EntityRequestList=entity_request_list_param
        )

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for entities in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = get_entities_information(mp_client)
    print("Successful entities response -")
    pretty_print(response["EntityDetails"])
    print("Failed entities response -")
    pretty_print(response["Errors"])
```

```
if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie BatchDescribeEntitiesin AWS SDK for Python (Boto3) API Reference.

Listet alle Angebote im Zusammenhang mit einem Produkt auf und beschreibt sie mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle mit einem Produkt verbundenen Angebote auflisten und beschreiben.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
 /*
  * retrieve all private offer information related to a single product
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryList();

  // for each offer id, output the offer detail using DescribeEntity API

  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   ReferenceCodesUtils.formatOutput(describeEntityResponse);
  }
 }
 public static List<EntitySummary> getEntitySummaryList() {
  // define list entities filters

  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
```

```
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(PRODUCT_ID)
        .build())
      .build())
    .build();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(50)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER).maxResults(50)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DescribeEntity
  - ListEntities

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to retrieve all offer information
related to a single product
CAPI-97
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError
from utils import helpers

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-catalog")


def get_entity_information(entity_id):
    """
    Returns information about a given entity
```

```
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def list_entity_details(entity_type, entity_id):
    """
    Returns details about a given entity and entity type
    """

    entity_summary_list = []

    # filter will return details for given entity_id with BuyerAccounts targeting
    filter_list_param = {
        'OfferFilters':{
            'ProductId':{
                'ValueList':[entity_id]
            },
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            EntityTypeFilters = filter_list_param,
            MaxResults=10
```

```
        )

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)

    # add results to entity_summary_list
    entity_summary_list.extend(response["EntitySummaryList"])

    # if there are more than 10 offers, paginate through the results
    while "NextToken" in response and response["NextToken"] is not None:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                EntityTypeFilters = filter_list_param,
                NextToken=response["NextToken"],
                MaxResults=10
            )

        except ClientError as e:
            if e.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Entity ID %s not found.", entity_id)
            else:
                logger.error("Unexpected error: %s", e)

        # add results to entity_summary_list
        entity_summary_list.extend(response["EntitySummaryList"])

        return entity_summary_list

if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "--entity-id",
        "-eid",
        help="Provide Entity ID corresponding to a product to filter offers on",
        required=True,
    )

    args = parser.parse_args()
```

```
    # Gets a offers associated with the entity_id
    response = list_entity_details(
        "Offer",
        entity_id=args.entity_id
    )

    if response: # if response is not empty

        # list_entity_details returns a list of offers
        for offer in response:

            print("-"*128)
            print(f"Terms for Offer ID: {offer['EntityId']}")
            print("-"*128)

            #retrieve offer information for each offer
            entity_information = get_entity_information(offer["EntityId"])

            helpers.pretty_print_datetime(entity_information)

    else:
        print(f"No information found for Entity ID: {args.entity_id}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeEntity
  - ListEntities

Angebote für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Erstellen Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt und erstellen Sie ein privates Angebot mit einem AWS SDK
- Erstellen Sie einen Entwurf eines privaten Angebots für ein AMI- oder SaaS-Produkt mithilfe eines AWS SDK

- Erstellen Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung und einem flexiblen Zahlungsplan für ein AMI-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit stündlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Angebot mit gestaffelten Vertragspreisen für ein SaaS-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK
- Erstellen Sie ein privates Ersatzangebot auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK
- Beschreiben Sie ein öffentliches Angebot mit einem AWS SDK
- Verfallen Sie ein privates Angebot mit einem AWS SDK
- Listet alle privaten Angebote mit einem auf AWS SDK
- Listet veröffentlichte öffentliche und private Angebote für eine bestimmte Produkt-ID auf. Verwenden Sie dazu AWS SDK
- Aktualisieren Sie ein Angebot, um einen Vertrag mit Pay-As-You-Go Preisgestaltung anzuwenden, indem Sie einen AWS SDK
- Aktualisieren Sie ein Angebot so, dass es eine stündliche jährliche Preisgestaltung vorsieht. Verwenden Sie dazu AWS SDK
- Aktualisieren Sie ein Angebot, um Targeting auf bestimmte geografische Regionen anzuwenden, indem Sie AWS SDK

- Aktualisieren Sie den Namen und die Beschreibung eines öffentlichen Angebots mithilfe eines AWS SDK
- Aktualisieren Sie die EULA eines Angebots mithilfe eines AWS SDK
- Aktualisieren Sie das Ablaufdatum eines privaten Angebots auf ein future Datum mithilfe eines AWS SDK
- Aktualisieren Sie die kostenlose Testdauer eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt mithilfe eines AWS SDK
- Aktualisieren Sie die Rückerstattungsrichtlinie eines Angebots mithilfe eines AWS SDK

Erstellen Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt und erstellen Sie ein privates Angebot mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt erstellen und ein privates Angebot erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
```

```
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
```

```
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a SaaS product custom
 dimension and private offer
CAPI-91
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create a SaaS product custom dimension and private offer"
    )
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie einen Entwurf eines privaten Angebots für ein AMI- oder SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen privaten Angebotsentwurf für ein beliebiges AMI- oder SaaS-Produkt erstellen, sodass Sie es intern überprüfen können, bevor Sie es für Käufer veröffentlichen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
```

```
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preis für ein SaaS-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
```

```
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract with PAYG pricing for my SaaS product
CAPI-34
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with contract with PAYG pricing for my SaaS
 product",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Some kind of refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer
with contract pricing and flexible payment schedule for my SaaS product
CAPI-39
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create private offer with contract pricing and flexible payment schedule
 for my SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
```

```
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
```

```
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer (target
 buyers)
for my Container product with contract pricing
CAPI-36
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish a private offer for my Container product with contract pricing",
    )
```

```
    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
```

```
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract pricing for my AMI product
CAPI-35
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)
```

```
change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with contract pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung und einem flexiblen Zahlungsplan für ein AMI-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen und einem flexiblen Zahlungsplan für ein AMI-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
```

```
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "FixedUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "Price": "0.0",
                    "Duration": "P365D",
                    "Grants": [
                        {
                            "DimensionKey": "t2.micro",
                            "MaxQuantity": 1
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementDuration": "P650D"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
```

```
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
```

```
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing and flexible payment schedule for my AMI product
CAPI-XX
"""

import os

import utils.start_changeset as sc
```

```
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with hourly annual pricing and flexible payment
 schedule for my AMI product",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit stündlicher jährlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen für ein AMI-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing for my AMI product
CAPI-31
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly annual pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit stündlicher Preisgestaltung für ein AMI-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit stündlichen Preisen für ein AMI-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly pricing for my AMI product
CAPI-32
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set, "create private offer with hourly pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 subscription pricing for my SaaS product
CAPI-33
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create private offer with subscription pricing for my SaaS
 product"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Angebot mit gestaffelten Vertragspreisen für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Angebot mit gestaffelten Preisen für ein SaaS-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 tiered contract pricing for my SaaS product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(
        change_set,
        "Create private offer with tiered contract pricing for my SaaS product",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public free trial
 offer with subscription pricing for SaaS product
CAPI-13
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create public free trial offer with subscription pricing for SaaS
 product",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein privates Ersatzangebot auf der Grundlage einer bestehenden Vereinbarung mit Vertragspreisen unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein privates Ersatzangebot aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract
pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
```

```
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract
 pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
```

```
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a replacement private
 offer
from an existing agreement with contract pricing
CAPI-95
"""


import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create replacement private offer with contract pricing..",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Beschreiben Sie ein öffentliches Angebot mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein öffentliches Angebot beschreiben.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
```

```
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Einzelheiten zur API finden Sie DescribeEntityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for offer information in
 the AWS Marketplace Catalog
CAPI-29
"""
```

```
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

OFFER_ID = "offer-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_offer_information(mp_client, entity_id):
    """
    Returns information about a given offer
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of offer information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Offer with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an offer in the AWS Marketplace Catalog.")
    print("-" * 88)
```

```
    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_offer_information(mp_client, OFFER_ID))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DescribeEntityin AWS SDK for Python (Boto3) API Reference.

Verfallen Sie ein privates Angebot mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie das Ablaufdatum eines privaten Angebots auf ein Datum in der Vergangenheit festlegen, sodass Käufer das Angebot nicht mehr sehen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
```

```
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Listet alle privaten Angebote mit einem auf AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle privaten Angebote auflisten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterD
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date,
 Offer Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 public static void main(String[] args) {

  String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
  String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

  List<EntitySummary> entitySummaryList =
 getEntitySummaryList(offerReleaseDateAfterValue,
 offerAvailableEndDateAfterValue);

  // for each offer id, output the offer detail using DescribeEntity API


  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
```

```
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
       .build())
     .releaseDate(OfferReleaseDateFilter.builder()
       .dateRange(OfferReleaseDateFilterDateRange.builder()
         .afterValue(offerReleaseDateAfterValue)
         .build())
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
```

```
  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }

  return entitySummaryList;
 }

}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) for listing offers in the AWS
 Marketplace Catalog
CAPI-40
```

```
"""
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# Constants
MAX_RESULTS = 10
CATALOG = "AWSMarketplace"
ENTITY_TYPE = "Offer"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def list_private_offers(mp_client, return_all_private_offers):
    """
    This method retrieves list of all Private Offers for this account.
    """
    entity_summary_list = []
    filter_list_param = {
        'OfferFilters': {
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }
    try:
        response = mp_client.list_entities(
            Catalog=CATALOG,
            EntityType=ENTITY_TYPE,
            EntityTypeFilters=filter_list_param,
            MaxResults=MAX_RESULTS
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request: %s", e)
        raise

    entity_summary_list.extend(response["EntitySummaryList"])
```

```
    logger.info("Number of results in first iteration: %d " %
 len(entity_summary_list))

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response and return_all_private_offers:
        try:
            logger.info("Getting Next Token results: %s " %
 response["NextToken"])
            response = mp_client.list_entities(
                Catalog=CATALOG,
                EntityType=ENTITY_TYPE,
                EntityTypeFilters=filter_list_param,
                MaxResults=MAX_RESULTS,
                NextToken=response["NextToken"]
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request: %s", e)
            raise

        entity_summary_list.extend(response["EntitySummaryList"])
        logger.info(
            "Number of results in the current iteration: %d "
            % len(response["EntitySummaryList"])
        )

    return entity_summary_list


def get_offer_details(mp_client, offer):
    """
    Describe the details of the Offer.
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=offer["EntityId"]
        )

        return response
    except ClientError:
        logger.exception("Error: Couldn't get details of the Offer.")
        raise


def usage_demo():
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Demo  - List Private offers.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    # Get list of all Offers.
    private_offers = list_private_offers(mp_client, False)
    count = len(private_offers)

    logger.info("Number of Offers: %d " % count)
    offer_counter = 0
    # Display details of each Offer.
    for offer in private_offers:
        print("-" * 88)
        offer_counter += 1
        print("Displaying Offer details for Offer# %d" % offer_counter)
        entity = get_offer_details(mp_client, offer)
        pretty_print(entity)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Listet veröffentlichte öffentliche und private Angebote für eine bestimmte Produkt-ID auf. Verwenden Sie dazu AWS SDK

Das folgende Codebeispiel zeigt, wie Sie freigegebene öffentliche und private Angebote für eine bestimmte Produkt-ID auflisten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
```

```
 * To change to released private offers, change OFFER_TARGETING_NONE (None) to
OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
 */
public static void main(String[] args) {

 List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
 ReferenceCodesUtils.formatOutput(entitySummaryList);
}

public static List<EntitySummary> getEntitySummaryLIst() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_NONE)
       .build())
     .state(OfferStateFilter.builder()
       .valueListWithStrings(OFFER_STATE_RELEASED)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER)
   .maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
```

```
  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Aktualisieren Sie ein Angebot, um einen Vertrag mit Pay-As-You-Go Preisgestaltung anzuwenden, indem Sie einen AWS SDK

Die folgenden Codebeispiele zeigen, wie ein Angebot aktualisiert wird, um einen Vertrag mit Pay-As-You-Go Preisgestaltung anzuwenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
```

```
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 contract with PAYG pricing
CAPI-21
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer to apply contract with PAYG pricing")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie ein Angebot so, dass es eine stündliche jährliche Preisgestaltung vorsieht. Verwenden Sie dazu AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein Angebot aktualisieren, um einen stündlichen jährlichen Preis anzuwenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
```

```
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 hourly annual pricing
CAPI-20
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer with hourly annual pricing")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie ein Angebot, um Targeting auf bestimmte geografische Regionen anzuwenden, indem Sie AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein Angebot aktualisieren, um das Targeting auf bestimmte geografische Regionen anzuwenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 targeting to certain geographic regions.
CAPI-19
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer targeting")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie den Namen und die Beschreibung eines öffentlichen Angebots mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Namen und die Beschreibung eines öffentlichen Angebots aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update name and description of
 my offer
CAPI-18
"""
```

```
import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update name and description of my offer")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie die EULA eines Angebots mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die EULA eines Angebots aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update EULA of my offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update EULA of my offer")


if __name__ == "__main__":
```

```
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie das Ablaufdatum eines privaten Angebots auf ein future Datum mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie das Ablaufdatum eines privaten Angebots auf ein zukünftiges Datum aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
```

```
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to set expiry date of a private
 offer to a date in the future so that my buyers get more time to evaluate and
 accept the offer.
CAPI-37
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer expiration date")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie die kostenlose Testdauer eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Dauer der kostenlosen Testversion eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
```

```
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change free trial duration for
 my SaaS product by modifying my free trial public offer
CAPI-14
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Change free trial duration for SaaS product")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie die Rückerstattungsrichtlinie eines Angebots mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Rückerstattungsrichtlinie eines Angebots aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update refund policy of my
 offer
```

```
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update refund policy of my public offer")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Produkte für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Beschreiben Sie ein AMI-, SaaS- oder Container-Produkt mit einem AWS SDK
- Alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten, indem Sie eine AWS SDK

Beschreiben Sie ein AMI-, SaaS- oder Container-Produkt mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein AMI-, SaaS- oder Container-Produkt beschreiben und prüfen, ob es alle Informationen enthält, die Sie über das Produkt haben möchten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
```

```
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Einzelheiten zur API finden Sie DescribeEntityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
CAPI-28
"""

import json
import logging
```

```
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")
```

```
    pretty_print(get_product_information(mp_client, PRODUCT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DescribeEntityin AWS SDK for Python (Boto3) API Reference.

Alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten, indem Sie eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import
 software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
  * List all my AMI or SaaS or Container products and associated public offers
  */
 public static void main(String[] args) {

  Map<String, List<EntitySummary>> allProductsWithOffers =
 getAllProductsWithOffers();

  ReferenceCodesUtils.formatOutput(allProductsWithOffers);
 }

 public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
 List<EntitySummary>> ();

  // get all product entities
  List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
```

```
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(null)
    .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }

 // loop through each product entity and get the public released offers
associated using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();
```

```
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(null)
      .build();

    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);

    // save all entitySummary of the results into entitySummaryList

    List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

    offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

    while ( listEntitiesResponse.nextToken() != null &&
  listEntitiesResponse.nextToken().length() > 0) {
     listEntitiesRequest =
       ListEntitiesRequest.builder()
       .catalog(AWS_MP_CATALOG)
       .entityType(ENTITY_TYPE_OFFER)
       .maxResults(10)
       .entityTypeFilters(entityTypeFilters)
       .nextToken(listEntitiesResponse.nextToken())
       .build();
     listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
     offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
    }

    // save final results into map; key = product id; value = offer entity summary
  list

    allProductsWithOffers.put(productEntitySummary.entityId(),
  offerEntitySummaryList);
   }
   return allProductsWithOffers;
  }

 }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DescribeEntity
  - ListEntities

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to display information about AMI
 products and their associated offers in the AWS Marketplace Catalog
CAPI-27
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

try:
    mp_client = boto3.client("marketplace-catalog")
except ClientError as e:
    logger.error("Could not create boto3 client.")
    raise
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def describe_entity(entity_id):
    """
    Returns entity details
    Args: entity_id str: The entity ID of the product or offer
    Returns: dict: The entity details
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=entity_id
        )
    except ClientError as e:
        logger.error("Could not complete describe_entity request.")
        raise

    # De-stringify the details
    response["Details"] = json.loads(response["Details"])

    return response


def get_entities(entity_type, visibility=None):
    """
    Returns list of entities for provided entity_type
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
 or Offer
    Returns: list: Abbreviated list of entity information
    """
    EntitySummaryList = []

    # Get the first page of results
    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            MaxResults=MAX_PAGE_RESULTS,
        )
    except ClientError as e:
```

```
        logger.error("Could not complete list_entities request.")
        raise

    EntitySummaryList.extend(response["EntitySummaryList"])

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                MaxResults=MAX_PAGE_RESULTS,
                NextToken=response["NextToken"],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        EntitySummaryList.extend(response["EntitySummaryList"])

    # if visibility is provided, filter the list to only include entities with
 that visibility
    if visibility is not None:
        EntitySummaryList = [
            entity for entity in EntitySummaryList if entity["Visibility"] ==
 visibility
        ]

    return EntitySummaryList


def get_enhanced_product_list(entity_type):
    """
    Returns an enhanced list of products with product details and offer details
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
    Returns: list: Enhanced list of dictionary objects containing product and
 offer details
    """

    product_list = get_entities(entity_type)

    # Loop through product list and append product details to each product
    for product in product_list:
        # appends product details to product dictionary
```

```
        product["ProductDetails"] = describe_entity(product["EntityId"])
["Details"]
        # creating an empty list for offer details
        product["OfferDetailsList"] = []

    return product_list


def attach_offer_details(product_list):
    """
    Loops through offer information and appends offer details to product list
    Args: product_list list: List of product dictionaries
    Returns: list: Enhanced list of dictionary objects containing product and
 offer details
    """
    offer_list = get_entities("Offer", "Public")

    # Loop through offer list and append offer details to each product
    for offer in offer_list:
        offer["OfferDetails"] = describe_entity(offer["EntityId"])["Details"]

        # Extracts product-id from offer
        product_id = offer["OfferDetails"]["ProductId"]

        # Determines if product-id referenced in offer matches product-id in
 product list
        product_dict = next(
            filter(lambda product: product["EntityId"] == product_id,
 product_list),
            None,
        )

        # If product-id matches, appends offer details to product dictionary
        if product_dict is not None:
            # logger.info(f"Offer product Id {offer['OfferDetails']['ProductId']}
 found in product dictionary. Updating product dictionary with offer details")
            product_dict["OfferDetailsList"].append(offer["OfferDetails"])

        else:
            # logger.info("Offer product Id {offer['OfferDetails']['ProductId']}
 not found. Skipping offer details update")
            pass

    return product_list
```

```
def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieving products and public offer information....")
    print("-" * 88)

    # Builds a list of products and their details
    product_list = get_enhanced_product_list("AmiProduct")

    # Queries offer information and attaches it to the product list
    product_offer_list = attach_offer_details(product_list)

    pretty_print(product_offer_list)
    return product_offer_list


if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeEntity
  - ListEntities

Genehmigung zum Weiterverkauf für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Erstellen Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp mithilfe eines AWS SDK
- Beschreiben Sie eine Wiederverkaufsautorisierung mit einem AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis mit einem privaten Angebot unter Verwendung eines AWS SDK

- Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum mithilfe eines AWS SDK
- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufserlaubnis mit einem Ablaufdatum und einer EULA, die dem Käufer unter Verwendung eines AWS SDK
- Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie die mehrfach verwendbare Wiederverkaufsgenehmigung mit Ablauf und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum mithilfe eines AWS SDK
- Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie eine benutzerdefinierte EULA hinzu, indem Sie AWS SDK
- Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie mithilfe eines AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung und fügen Sie einen flexiblen Zahlungsplan hinzu, indem Sie einen AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis für einen beliebigen Produkttyp und fügen Sie eine EULA hinzu, indem Sie eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK
- Veröffentlichen Sie eine einmalige Wiederverkaufsautorisierung für und fügen Sie hinzu, ob es sich um eine Verlängerung handelt. Verwenden Sie dazu AWS SDK
- Beschränken Sie die Wiederverkaufsautorisierung mithilfe eines AWS SDK
- Aktualisieren Sie den Namen und die Beschreibung der einmaligen oder mehrfach verwendbaren Wiederverkaufsautorisierung mithilfe eines AWS SDK

Erstellen Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn bei einem Channel-Partner veröffentlichen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
 can use that to create Channel Partner Private Offer (CPPO)
```

```
CAPI-41
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "draft resale auth")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Beschreiben Sie eine Wiederverkaufsautorisierung mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine Wiederverkaufsgenehmigung beschreiben.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();
```

```
  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- Einzelheiten zur API finden Sie DescribeEntityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

resaleAuthorizationId = "resaleauthz-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
```

```
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, resaleAuthorizationId))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DescribeEntityin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis mit einem privaten Angebot unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung mit einem privaten Angebot veröffentlichten sodass ein Channel-Partner damit ein Channel Partner Private Offer (CPPO) erstellen kann.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
can use that to create Channel Partner Private Offer (CPPO)
CAPI-42
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set, "onetime resale auth with private offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlicht wird, sodass ein Channel-Partner damit ein CPPO erstellen kann.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Contract",
            "Terms": [
                {
                    "Type": "ResaleConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI product
 so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-48
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multi-use resale auth with expiry date")
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine Mehrzweck-Wiederverkaufserlaubnis mit einem Ablaufdatum und einer EULA, die dem Käufer unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
CAPI-56
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multiuse resale auth with expiry date and custom
 EULA")
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
```

```
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleUsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/
Container product
and add reseller contract documentation between the ISV and channel partner
CAPI-57
"""

import os

import utils.start_changeset as sc
```

```
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "multi use resale auth with contract doc",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie die mehrfach verwendbare Wiederverkaufsgenehmigung mit Ablauf und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Publish multi-use resale authorization with expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-82
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Publish multi-use resale authorization with expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne ein Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlichen, sodass ein Channel-Partner damit ein CPPO erstellen kann.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
```

```
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
```

```
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI
 product so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-52
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "multi use resale auth with no expiry date")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie eine benutzerdefinierte EULA hinzu, indem Sie AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
```

```
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
CAPI-58
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and custom EULA"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie die Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add reseller contract documentation between the ISV and
 channel partner
CAPI-59
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and contract doc"
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine Mehrzweck-Wiederverkaufsgenehmigung ohne Ablaufdatum und fügen Sie mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization without expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-83
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "publish multi-use resale authorization without expiry date and add
 specific buyer account",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung und fügen Sie einen flexiblen Zahlungsplan hinzu, indem Sie einen AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und einen flexiblen Zahlungsplan hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
```

```
                        {
                          "DimensionKey": "Users",
                          "MaxQuantity": 10
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ResalePaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
                            "ChargeDate": "2023-09-01",
                            "ChargeAmount": "200.00"
                        },
                        {
                            "ChargeDate": "2023-12-01",
                            "ChargeAmount": "250.00"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-06-30",
            "OffersMaxQuantity": 1
        }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
 and add Flexible payment schedule
CAPI-78
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
```

```
    sc.usage_demo(change_set, "onetime resale auth with flexible payment
 schedule")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis für einen beliebigen Produkttyp und fügen Sie eine EULA hinzu, indem Sie eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
```

```
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 custom EULA to be sent to the buyer
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with custom EULA")
```

```
if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufserlaubnis und fügen Sie ein bestimmtes Käuferkonto für den Weiterverkauf hinzu. Verwenden Sie dazu eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
```

```
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
```

```
                        }
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
 and add specific buyer account for the resale
CAPI-81
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale authorization for specific buyer
 account")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp und fügen Sie die Vertragsdokumentation für Wiederverkäufer hinzu. Verwenden Sie dazu eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 reseller contract documentation between the ISV and channel partner
CAPI-47
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with reseller contract doc")


if __name__ == "__main__":
```

```
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie eine einmalige Wiederverkaufsautorisierung für und fügen Sie hinzu, ob es sich um eine Verlängerung handelt. Verwenden Sie dazu AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und hinzufügen, ob es sich um eine Verlängerung handelt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
```

```
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
```

```
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
and add whether it is renewal or not
CAPI-90
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(stringified_change_set, "onetime resale auth
 renewal")

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Beschränken Sie die Wiederverkaufsautorisierung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine Wiederverkaufsgenehmigung beschränken.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Restrict a  authorization for any product type (AMI/SaaS/Container)
CAPI-84
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict resale authorization")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Aktualisieren Sie den Namen und die Beschreibung der einmaligen oder mehrfach verwendbaren Wiederverkaufsautorisierung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Namen und die Beschreibung einer einmaligen oder mehrfach verwendbaren Wiederverkaufsgenehmigung vor der Veröffentlichung für einen beliebigen Produkttyp aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Update name/description of one-time or multi-use resale authorization before
 publishing for any product type (AMI/SaaS/Container)
CAPI-77
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "update name and description of one-time or multi-use resale
 authorization before publishing",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

SaaS-Produkte für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Erstellen Sie einen Entwurf eines SaaS-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertragspreisen unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertrag und Pay-As-You-Go Preisgestaltung unter Verwendung eines AWS SDK
- Erstellen Sie ein öffentliches oder eingeschränktes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen unter Verwendung eines AWS SDK
- Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot mit einem AWS SDK
- Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot anhand eines vorhandenen Entwurfs mithilfe eines AWS SDK
- Aktualisieren Sie die Dimensionen eines AMI- oder SaaS-Produkts mithilfe eines AWS SDK

Erstellen Sie einen Entwurf eines SaaS-Produkts mit einem Entwurf eines öffentlichen Angebots unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen SaaS-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
              "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an SaaS draft product
with a draft public offer.
CAPI-04
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft saas product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertragspreisen unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit nutzungsabhängiger Preisberechnung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
```

```
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
```

```
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract pricing and
 standard EULA
CAPI-11
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create a limited saas product with a public offer with contract
 pricing",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Vertrag und Pay-As-You-Go Preisgestaltung unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein öffentliches oder begrenztes SaaS-Produkt und ein öffentliches Angebot mit einem Vertrag mit Pay-As-You-Go Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
```

```
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
```

```
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
```

```
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
```

```
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract with PAYG pricing
 and standard EULA
CAPI-10
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
```

```
        change_set,
        "Create limited SaaS product with public offer with contract with payg
 pricing",
    )


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie ein öffentliches oder eingeschränktes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen erstellen. In diesen Beispielen wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
```

```
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
```

```
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
```

```
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
```

```
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
```

```
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
```

```
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with subscription(usage) pricing
 and standard EULA
CAPI-09
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "public saas public offer with subscription
 pricing")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot veröffentlichen. Das Produkt befindet sich standardmäßig in einem eingeschränkten Zustand.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
```

```
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
```

```
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
```

```
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
```

```
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish my SaaS product and associated public offer (product will be in limited
 state by default)
CAPI-05A
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset1.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish saas product and associated public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Veröffentlichen Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot anhand eines vorhandenen Entwurfs mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot anhand eines bestehenden Entwurfs veröffentlichen. Das Produkt befindet sich standardmäßig in einem eingeschränkten Zustand.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Aktualisieren Sie die Dimensionen eines AMI- oder SaaS-Produkts mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Dimensionen auf einem AMI- oder SaaS-Produkt aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
```

```
    ]
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update (e.g name) dimensions
 on my AMI or SaaS product
CAPI-24
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update name dimensions on my AMI or SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Dienstprogramme für AWS Marketplace Catalog-API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Catalog API mit AWS SDKs verwendet wird.

Beispiele

- Hilfsprogramme zum Starten eines Changesets mit einem AWS SDK

Hilfsprogramme zum Starten eines Changesets mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Dienstprogramme zum Starten eines Changesets definieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Dienstprogramm zum Laden eines Changesets aus einer JSON-Datei und zum Starten der Verarbeitung.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);
```

```
 // process the changeset request
 try {
  StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
  ReferenceCodesUtils.formatOutput(result);
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public static StartChangeSetResponse getChangeSetRequestResult(String
changeSetsInput) throws IOException {

 //set up AWS credentials
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 //changeset list to save all the changesets in the changesets file
 List<Change> changeSetLists = new ArrayList<Change>();

 // read all changesets into object
 Root root = GSON.fromJson(changeSetsInput, Root.class);

 // process each changeset and add each changeset request to changesets list
 for (ChangeSet cs : root.changeSet) {

  ChangeSetEntity entity = cs.Entity;
  String entityType = entity.Type;
  String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
  Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

  Entity awsEntity =
    Entity.builder()
    .type(entityType)
    .identifier(entityIdentifier)
    .build();

  Change inputChangeRequest =
    Change.builder()
    .changeType(cs.ChangeType)
    .changeName(cs.ChangeName)
```

```
    .entity(awsEntity)
    .detailsDocument(detailsDocument)
    .build();

  changeSetLists.add(inputChangeRequest);
 }

 // process all changeset requests
 StartChangeSetRequest startChangeSetRequest =
   StartChangeSetRequest.builder()
   .catalog(root.catalog)
   .changeSet(changeSetLists)
   .build();

 StartChangeSetResponse result =
marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

 return result;
}

public static Document getDocumentFromObject(Object detailsObject) {

 String detailsString = "{}";
 try {
  detailsString = IOUtils.toString(new
ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 JsonNodeParser jsonNodeParser = JsonNodeParser.create();
 Document doc = jsonNodeParser.parse(detailsString).visit(new
DocumentUnmarshaller());
 return doc;
}


public static String readChangeSetToString (String inputChangeSetFile) {

 InputStream changesetInputStream =
RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

 String changeSetsInput = null;
```

```
  try {
   changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  return changeSetsInput;

 }
}
```

- Einzelheiten zur API finden Sie StartChangeSetin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Dienstprogramme zum Starten eines Changesets.

```
"""
Purpose:

Generic function to start a changeset
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def generate_changeset(mp_client, change_set, change_set_name):
```

```
    """
    Start changeset
    """
    try:
        response = mp_client.start_change_set(
            Catalog="AWSMarketplace",
            ChangeSet=change_set,
            ChangeSetName=change_set_name,
        )
        logger.info("Changeset created!")
        logger.info("ChangeSet ID: %s", response["ChangeSetId"])
        logger.info("ChangeSet ARN: %s", response["ChangeSetArn"])

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo(change_set, change_set_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Executing changeset: " + change_set_name)
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = generate_changeset(mp_client, change_set, change_set_name)

    return response

    print("-" * 88)
```

Dienstprogramm zum Laden eines Changesets aus einer JSON-Datei.

```
"""
Purpose:

This module will stringify the details sections of a changeset file.
"""
```

```
import json


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


def stringify_details_sections(json_object):
    """
    Loops through every change type in the changeset to look for non-empty
    details section and stringifies them
    """
    for change_type in json_object["ChangeSet"]:
        # Only stringify details section if it is not empty
        if "Details" in change_type and change_type["Details"] != "{}":
            string_details = json.dumps(change_type["Details"])
            change_type["Details"] = string_details
        else:
            pass

    return json_object["ChangeSet"]


def stringify_changeset(file_path):
    changeset_file = open_json_file(file_path)
    changeset_stringified = stringify_details_sections(changeset_file)

    return changeset_stringified
```

- Einzelheiten zur API finden Sie StartChangeSetin AWS SDK for Python (Boto3) API Reference.

Codebeispiele für AWS Marketplace Vereinbarungs-API unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie die AWS Marketplace Agreement API mit einem AWS Software Development Kit (SDK) verwenden.

Weitere -Quellen

- AWS Marketplace Verkäuferleitfaden zur Agreement API — Weitere Informationen zur AWS Marketplace Agreement API.
- AWS Marketplace API-Referenz zur Agreement API — Details zu allen verfügbaren AWS Marketplace Agreement API-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Vereinbarungen für AWS Marketplace Vereinbarung API
  - Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung mit einem AWS SDK
  - Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung mit einem AWS SDK
  - Ändern Sie die Verlängerungsdauer einer SaaS-Vertragsvereinbarung mithilfe eines AWS SDK
  - Ändern Sie ein Preismodell ConfigurableUpfrontPricingTerm für AMI for Usage mithilfe eines AWS SDK
  - Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion mit einem AWS SDK
  - Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung unter Verwendung eines AWS SDK
  - Rufen Sie alle Vereinbarungs-IDs mit einem ab AWS SDK
  - Rufen Sie alle Vereinbarungen mit einem ab AWS SDK
  - Rufen Sie die Kundennummer aus einer Vereinbarung mit einem ab AWS SDK
  - Einzelheiten zu einer Anfrage zur Kündigung eines Vertrags erhalten Sie unter AWS SDK
  - Rufen Sie Einzelheiten zu einer Zahlungsaufforderung für einen Vertrag ab, indem Sie AWS SDK
  - Holen Sie sich finanzielle Details aus einer Vereinbarung mithilfe eines AWS SDK
  - Einzelheiten zur kostenlosen Testversion finden Sie in einer Vereinbarung unter AWS SDK

- Informationen zu einer Vereinbarung erhalten Sie mit einem AWS SDK
- Rufen Sie Produkt- und Angebotsdetails aus einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die EULA einer Vereinbarung mit einem ab AWS SDK
- Holen Sie sich die Bedingungen für die auto Verlängerung einer Vereinbarung mithilfe eines AWS SDK
- Holen Sie sich die im Rahmen eines Vertrags erworbenen Abmessungen mithilfe eines AWS SDK
- Rufen Sie die Instanzen jeder Dimension ab, die im Rahmen einer Vereinbarung erworben wurden, mithilfe eines AWS SDK
- Rufen Sie den Zahlungsplan einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die Preise pro Dimension in einer Vereinbarung mit einem ab AWS SDK
- Ermitteln Sie den Preistyp einer Vereinbarung mithilfe eines AWS SDK
- Rufen Sie den Produkttyp einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie den Status einer Vereinbarung mit einem ab AWS SDK
- Rufen Sie die Support-Bedingungen einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die Bedingungen einer Vereinbarung mit einem ab AWS SDK
- Listen Sie Anfragen zur Kündigung von Verträgen mit einem auf AWS SDK
- Führen Sie Zahlungsanfragen für Vereinbarungen mit einem auf AWS SDK
- Lehnen Sie eine Anfrage zur Kündigung eines Vertrags ab, indem Sie AWS SDK
- Lehnen Sie eine Zahlungsaufforderung für einen Vertrag ab, indem Sie einen AWS SDK
- Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vereinbarungsbasiertes Angebot unter Verwendung eines AWS SDK
- Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing mithilfe eines AWS SDK
- Ersetzen Sie eine nutzungsabhängige SaaS-Preisvereinbarung und kündigen Sie die vorherige Vereinbarung mit einem AWS SDK
- Ersetzen Sie eine nutzungsbasierte AMI-Preisgestaltung, behalten Sie jedoch die bestehende ConfigurableUpfrontPricingTerm bei und verwenden Sie eine AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Konto-ID mithilfe eines AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Vereinbarungs-ID mit einem AWS SDK
- Suchen Sie nach Vereinbarungen anhand des Enddatums mithilfe eines AWS SDK

- Suchen Sie nach Vereinbarungen anhand der Angebots-ID mit einem AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Produkt-ID mit einem AWS SDK
- Suchen Sie mithilfe eines nach Status nach Vereinbarungen AWS SDK
- Suchen Sie mit einem benutzerdefinierten Filter nach Vereinbarungen mithilfe eines AWS SDK
- Suchen Sie mithilfe eines nach Vereinbarungen mit zwei benutzerdefinierten Filtern AWS SDK
- Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung mithilfe eines AWS SDK

Vereinbarungen für AWS Marketplace Vereinbarung API

Die folgenden Codebeispiele zeigen, wie die AWS Marketplace Agreement API mit AWS SDKs verwendet wird.

Beispiele

- Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung mit einem AWS SDK
- Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung mit einem AWS SDK
- Ändern Sie die Verlängerungsdauer einer SaaS-Vertragsvereinbarung mithilfe eines AWS SDK
- Ändern Sie ein Preismodell ConfigurableUpfrontPricingTerm für AMI for Usage mithilfe eines AWS SDK
- Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion mit einem AWS SDK
- Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung unter Verwendung eines AWS SDK
- Rufen Sie alle Vereinbarungs-IDs mit einem ab AWS SDK
- Rufen Sie alle Vereinbarungen mit einem ab AWS SDK
- Rufen Sie die Kundennummer aus einer Vereinbarung mit einem ab AWS SDK
- Einzelheiten zu einer Anfrage zur Kündigung eines Vertrags erhalten Sie unter AWS SDK
- Rufen Sie Einzelheiten zu einer Zahlungsaufforderung für einen Vertrag ab, indem Sie AWS SDK
- Holen Sie sich finanzielle Details aus einer Vereinbarung mithilfe eines AWS SDK
- Einzelheiten zur kostenlosen Testversion finden Sie in einer Vereinbarung unter AWS SDK
- Informationen zu einer Vereinbarung erhalten Sie mit einem AWS SDK
- Rufen Sie Produkt- und Angebotsdetails aus einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die EULA einer Vereinbarung mit einem ab AWS SDK

- Holen Sie sich die Bedingungen für die auto Verlängerung einer Vereinbarung mithilfe eines AWS SDK
- Holen Sie sich die im Rahmen eines Vertrags erworbenen Abmessungen mithilfe eines AWS SDK
- Rufen Sie die Instanzen jeder Dimension ab, die im Rahmen einer Vereinbarung erworben wurden, mithilfe eines AWS SDK
- Rufen Sie den Zahlungsplan einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die Preise pro Dimension in einer Vereinbarung mit einem ab AWS SDK
- Ermitteln Sie den Preistyp einer Vereinbarung mithilfe eines AWS SDK
- Rufen Sie den Produkttyp einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie den Status einer Vereinbarung mit einem ab AWS SDK
- Rufen Sie die Support-Bedingungen einer Vereinbarung mithilfe eines ab AWS SDK
- Rufen Sie die Bedingungen einer Vereinbarung mit einem ab AWS SDK
- Listen Sie Anfragen zur Kündigung von Verträgen mit einem auf AWS SDK
- Führen Sie Zahlungsanfragen für Vereinbarungen mit einem auf AWS SDK
- Lehnen Sie eine Anfrage zur Kündigung eines Vertrags ab, indem Sie AWS SDK
- Lehnen Sie eine Zahlungsaufforderung für einen Vertrag ab, indem Sie einen AWS SDK
- Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vereinbarungsbasiertes Angebot unter Verwendung eines AWS SDK
- Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing mithilfe eines AWS SDK
- Ersetzen Sie eine nutzungsabhängige SaaS-Preisvereinbarung und kündigen Sie die vorherige Vereinbarung mit einem AWS SDK
- Ersetzen Sie eine nutzungsbasierte AMI-Preisgestaltung, behalten Sie jedoch die bestehende ConfigurableUpfrontPricingTerm bei und verwenden Sie eine AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Konto-ID mithilfe eines AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Vereinbarungs-ID mit einem AWS SDK
- Suchen Sie nach Vereinbarungen anhand des Enddatums mithilfe eines AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Angebots-ID mit einem AWS SDK
- Suchen Sie nach Vereinbarungen anhand der Produkt-ID mit einem AWS SDK
- Suchen Sie mithilfe eines nach Status nach Vereinbarungen AWS SDK
- Suchen Sie mit einem benutzerdefinierten Filter nach Vereinbarungen mithilfe eines AWS SDK

- Suchen Sie mithilfe eines nach Vereinbarungen mit zwei benutzerdefinierten Filtern AWS SDK
- Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung mithilfe eines AWS SDK

Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie eine vom Verkäufer initiierte Anfrage zur Kündigung eines Vertrags akzeptiert wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationRe
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationRe

public class AcceptAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        acceptAgreementCancellationRequest();
    }

    private static void acceptAgreementCancellationRequest() {
```

```
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementCancellationRequestRequest request =
                AcceptAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        AcceptAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.acceptAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Description: " + response.description());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie AcceptAgreementCancellationRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function acceptAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Description: " + response.description);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementCancellationRequest();
```

- Einzelheiten zur API finden Sie AcceptAgreementCancellationRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"

    @staticmethod
    def accept_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_cancellation_request(
            agreementId=AcceptAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=AcceptAgreementCancellationRequest.AGREEMENT_CANCELLATION
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Description: " + str(response.get("description", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
```

```
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementCancellationRequest.accept_agreement_cancellation_request()
```

- Einzelheiten zur API finden Sie AcceptAgreementCancellationRequestin AWS SDK for Python (Boto3) API Reference.

Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine vom Verkäufer initiierte Zahlungsaufforderung akzeptieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequest

public class AcceptAgreementPaymentRequest {
```

```
    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER
 REFERENCE HERE>";

    public static void main(String[] args) {
        acceptAgreementPaymentRequest();
    }

    private static void acceptAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementPaymentRequestRequest request =
                AcceptAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                        .build();

        AcceptAgreementPaymentRequestResponse response =

 marketplaceAgreementClient.acceptAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie AcceptAgreementPaymentRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>";

async function acceptAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementPaymentRequest();
```

- Einzelheiten zur API finden Sie AcceptAgreementPaymentRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>"

    @staticmethod
    def accept_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_payment_request(
            agreementId=AcceptAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=AcceptAgreementPaymentRequest.PAYMENT_REQUEST_ID,

 purchaseOrderReference=AcceptAgreementPaymentRequest.PURCHASE_ORDER_REFERENCE,
        )
```

```
        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementPaymentRequest.accept_agreement_payment_request()
```

- Einzelheiten zur API finden Sie AcceptAgreementPaymentRequestin AWS SDK for Python (Boto3) API Reference.

Ändern Sie die Verlängerungsdauer einer SaaS-Vertragsvereinbarung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine SaaS-Vertragsvereinbarung ändern, um ihre Verlängerungsdauer hinzuzufügen oder zu aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RenewalTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and
 then turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P1M}
 for 1 month).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer.</li>
 * </ul>
```

```
 */
public class AmendSaaSContractRenewalTerm {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the RenewalTerm in your offer.
    private static final String RENEWAL_TERM_ID = "<your-renewal-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    public static void main(String[] args) {
        amendSaaSContractAgreementRenewalTerm();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
     * 2. Wait for entitlements to become active.
     * 3. Amend the agreement to enable auto-renewal.
     */
    private static void amendSaaSContractAgreementRenewalTerm() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
```

```
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        // Initial agreement: auto-renewal disabled.
        RequestedTerm renewalTerm = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(false)
                                .build()))
                .build();

        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm supportTerm =
RequestedTerm.builder().id(SUPPORT_TERM_ID).build();

        // --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
renewalTerm, legalTerm, supportTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
```

```
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Amend: enable auto-renewal ---
        RequestedTerm renewalTermAmended = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(true)
                                .build()))
                .build();

        // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(configurableUpfrontPricingTerm,
renewalTermAmended, legalTerm, supportTerm)
```

```
 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Amend agreement request created. AgreementRequestId:
 " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Amendment accepted. Auto-renewal enabled. New
 AgreementId: " + aarResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");
```

```
/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and
 then turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P1M" for 1 month).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the RenewalTerm in your offer.
const RENEWAL_TERM_ID = "<your-renewal-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the SupportTerm in your offer.
```

```
const SUPPORT_TERM_ID = "<your-support-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
 * 2. Wait for entitlements to become active.
 * 3. Amend the agreement to enable auto-renewal.
 */
async function amendSaaSContractAgreementRenewalTerm() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };

    // Initial agreement: auto-renewal disabled.
    const renewalTerm = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: false,
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };

    // --- Create and accept the initial SaaS agreement request with CONTRACT
 pricing model ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
```

```
            requestedTerms: [configurableUpfrontPricingTerm, renewalTerm,
legalTerm, supportTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before amending.
    console.log("Waiting for entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("Entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Amend: enable auto-renewal ---
    const renewalTermAmended = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: true,
            },
        },
    };

    // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTermAmended,
legalTerm, supportTerm],
            sourceAgreementIdentifier:
acceptAgreementRequestResponse.agreementId,
```

```
        })
    );
    console.log("Amend agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aarResponse.agreementId);
}

amendSaaSContractAgreementRenewalTerm();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
the auto-renewal setting using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a public offer that supports
auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
auto-renewal via the RenewalTerm configuration.
```

```
Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P1M for 1 month).
  - DIMENSION_1_KEY — the dimension key defined in the offer.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendSaaSContractRenewalTerm:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the RenewalTerm in your offer.
    RENEWAL_TERM_ID = "<your-renewal-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"
```

```
    @staticmethod
    def amend_saas_contract_agreement_renewal_term():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
disabled.
        2. Wait for entitlements to become active.
        3. Amend the agreement to enable auto-renewal.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendSaaSContractRenewalTerm

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        # Initial agreement: auto-renewal disabled.
        renewal_term = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": False,
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        support_term = {"id": cls.SUPPORT_TERM_ID}

        # --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        create_response = client.create_agreement_request(
```

```
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term,
legal_term, support_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Amend: enable auto-renewal ---
        renewal_term_amended = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": True,
                }
            },
        }

        # Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[configurable_upfront_pricing_term,
renewal_term_amended, legal_term, support_term],
            sourceAgreementIdentifier=agreement_id,
        )
        print("Amend agreement request created. AgreementRequestId: " +
car_response["agreementRequestId"])
```

```
        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aar_response["agreementId"])


if __name__ == "__main__":
    AmendSaaSContractRenewalTerm.amend_saas_contract_agreement_renewal_term()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Ändern Sie ein Preismodell ConfigurableUpfrontPricingTerm für AMI for Usage mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine AMI-Vereinbarung ändern können, ConfigurableUpfrontPricingTerm indem Sie das Preismodell Dimension Quantity for Usage aktualisieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI agreement with
 ConfigurableUpfrontPricingTerm and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm (UBPT)</b> — accepted first
 to establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> —
 accepted after the UBPT agreement entitlements are active.</li>
 * </ol>
 * Once both agreement entitlements are available, this sample shows how to
 <b>amend</b> the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} —  duration for the agreement
```

```
 *       (e.g., {@code P365D} for one year).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer (e.g.,
 instance type).</li>
 *   <li>{@code DIMENSION_1_VALUE} — initial quantity; {@code
 NEW_DIMENSION_1_VALUE} — amended quantity.</li>
 * </ul>
 */
public class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Initial quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in your offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // New quantity to use when amending the dimension of CUPT.
    private static final int NEW_DIMENSION_1_VALUE = 5;

    public static void main(String[] args) {
        amendAmiCUPTAgreement();
    }

    /**
```

```
    * Full end-to-end flow:
    * 1. Create and accept an agreement request with usageBasedPricingTerm.
    * 2. Wait for entitlements to become active, then create and accept an
agreement request with configurableUpfrontPricingTerm (CUPT).
    * 3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
    */
   private static void amendAmiCUPTAgreement() {
       MarketplaceAgreementClient marketplaceAgreementClient =
               MarketplaceAgreementClient.builder()
                       .httpClient(ApacheHttpClient.builder().build())
                       .credentialsProvider(ProfileCredentialsProvider.create())
                       .build();

       RequestedTerm usageTerm = RequestedTerm.builder()
               .id(USAGE_TERM_ID)
               .build();
       RequestedTerm legalTerm = RequestedTerm.builder()
               .id(LEGAL_TERM_ID)
               .build();
       RequestedTerm validityTerm = RequestedTerm.builder()
               .id(VALIDITY_TERM_ID)
               .build();

       // --- Agreement with UBPT ---
       CreateAgreementRequestRequest createAgreementRequestRequest =
               CreateAgreementRequestRequest.builder()
                       .clientToken(AgreementApiUtils.generateClientToken())
                       .intent(Intent.NEW)
                       .requestedTerms(usageTerm, legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                       .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                       .build();
```

```
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before creating the agreement
with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
```

```
        System.out.println("Agreement request with CUPT created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: "
+ cuptAgreementId);

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Amend Agreement with CUPT ---
        // Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        RequestedTerm newConfig = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(NEW_DIMENSION_1_VALUE) // Increase quantity for this dimension
key
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carAmendRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
```

```
                        .intent(Intent.AMEND)
                        .requestedTerms(newConfig, legalTerm, validityTerm)
                        .sourceAgreementIdentifier(cuptAgreementId)
                        .build();
        CreateAgreementRequestResponse carAmendResponse =

 marketplaceAgreementClient.createAgreementRequest(carAmendRequest);
        System.out.println("Amendment of CUPT agreement request created.
 AgreementRequestId: " + carAmendResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarAmendRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(carAmendResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarAmendResponse =

 marketplaceAgreementClient.acceptAgreementRequest(aarAmendRequest);
        System.out.println("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
```

```
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI agreement with
 ConfigurableUpfrontPricingTerm and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to
 establish the base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after
 the UBPT agreement entitlements are active.
 *
 * Once both agreement entitlements are available, this sample shows how to amend
 the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D" for one year).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
 *   - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
```

```
const DIMENSION_1_KEY = "<your-dimension-key>";

// Initial quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in your offer.
const USAGE_TERM_ID = "<your-usage-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// New quantity to use when amending the dimension of CUPT.
const NEW_DIMENSION_1_VALUE = 5;

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm.
 * 2. Wait for entitlements to become active, then create and accept an agreement
 request with configurableUpfrontPricingTerm (CUPT).
 * 3. Wait for CUPT entitlements to become active, then amend the dimension
 quantity.
 */
async function amendAmiCUPTAgreement() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);
```

```
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before creating the agreement with
CUPT.
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

    const aarResponse = await client.send(
```

```
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const cuptAgreementId = aarResponse.agreementId;
    console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

    // Wait for entitlements to become active before amending.
    console.log("Waiting for CUPT agreement entitlements to become active...");
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Amend Agreement with CUPT ---
    // Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
    const newConfig = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
NEW_DIMENSION_1_VALUE }, // Increase quantity for this dimension key
                ],
            },
        },
    };

    const carAmendResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [newConfig, legalTerm, validityTerm],
            sourceAgreementIdentifier: cuptAgreementId,
        })
    );
    console.log("Amendment of CUPT agreement request created. AgreementRequestId:
" + carAmendResponse.agreementRequestId);

    const aarAmendResponse = await client.send(
        new AcceptAgreementRequestCommand({
```

```
            agreementRequestId: carAmendResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId);
}

amendAmiCUPTAgreement();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend
the dimension quantity using the AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement
     entitlements are active.
Once both agreement entitlements are available, this sample shows how to amend
 the agreement
with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.

Before running this sample, replace the placeholder constants below with values
 from
```

```
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D for one year).
  - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
  - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Initial quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in your offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"
```

```
    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # New quantity to use when amending the dimension of CUPT.
    NEW_DIMENSION_1_VALUE = 5

    @staticmethod
    def amend_ami_cupt_agreement():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm.
        2. Wait for entitlements to become active, then create and accept an
agreement request
           with configurableUpfrontPricingTerm (CUPT).
        3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
agreement_id)

        # Wait for entitlements to become active before creating the agreement
with CUPT.
```

```
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
```

```
        format_output(cupt_entitlements_response)

        # --- Amend Agreement with CUPT ---
        # Increase the dimension quantity using Intent.AMEND and
 sourceAgreementIdentifier.
        new_config = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.NEW_DIMENSION_1_VALUE,  #
 Increase quantity for this dimension key
                        },
                    ],
                }
            },
        }

        car_amend_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[new_config, legal_term, validity_term],
            sourceAgreementIdentifier=cupt_agreement_id,
        )
        print(
            "Amendment of CUPT agreement request created. AgreementRequestId: "
            + car_amend_response["agreementRequestId"]
        )

        aar_amend_response = client.accept_agreement_request(
            agreementRequestId=car_amend_response["agreementRequestId"]
        )
        print("Amendment accepted. New AgreementId: " +
 aar_amend_response["agreementId"])


if __name__ == "__main__":

 AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel.amend_ami_cupt_agreement()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a
 UsageBasedPricingTerm.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 * </ul>
 */
public class NewAmiFreeTrial {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in your offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the UsageBasedPricingTerm in your offer (applies after the
 trial ends).
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    public static void main(String[] args) {
        createAndAcceptAmiFreeTrialAgreementRequest();
    }

    /**
     * Creates an AMI Free Trial agreement.
     * The FreeTrialPricingTerm grants access at no cost for the trial period.
```

```
    * The UsageBasedPricingTerm defines the charges that apply once the trial
ends.
    */
   private static void createAndAcceptAmiFreeTrialAgreementRequest() {
       MarketplaceAgreementClient marketplaceAgreementClient =
               MarketplaceAgreementClient.builder()
                       .httpClient(ApacheHttpClient.builder().build())
                       .credentialsProvider(ProfileCredentialsProvider.create())
                       .build();

       RequestedTerm freeTrialPricingTerm = RequestedTerm.builder()
               .id(FREE_TRIAL_PRICING_TERM_ID)
               .build();
       RequestedTerm usageBasedPricingTerm = RequestedTerm.builder()
               .id(USAGE_BASED_PRICING_TERM_ID)
               .build();
       RequestedTerm supportTerm = RequestedTerm.builder()
               .id(SUPPORT_TERM_ID)
               .build();
       RequestedTerm legalTerm = RequestedTerm.builder()
               .id(LEGAL_TERM_ID)
               .build();

       CreateAgreementRequestRequest createAgreementRequestRequest =
               CreateAgreementRequestRequest.builder()
                       .clientToken(AgreementApiUtils.generateClientToken())
                       .intent(Intent.NEW)
                       .requestedTerms(freeTrialPricingTerm,
usageBasedPricingTerm, supportTerm, legalTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                       .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                       .build();
       AcceptAgreementRequestResponse acceptAgreementRequestResponse =
```

```
 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a
 UsageBasedPricingTerm.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
```

```
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the FreeTrialPricingTerm in your offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

/**
 * Creates an AMI Free Trial agreement.
 * The FreeTrialPricingTerm grants access at no cost for the trial period.
 * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
 */
async function createAndAcceptAmiFreeTrialAgreementRequest() {
    const client = new MarketplaceAgreementClient();

    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);
```

```
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createAndAcceptAmiFreeTrialAgreementRequest();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create an AMI Free Trial agreement
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to an AMI product that offers a free trial period.
The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
"""
```

```
import boto3

from utils.agreement_api_utils import generate_client_token


class NewAmiFreeTrial:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the FreeTrialPricingTerm in your offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the UsageBasedPricingTerm in your offer (applies after the
 trial ends).
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    @staticmethod
    def create_and_accept_ami_free_trial_agreement_request():
        """
        Create an AMI Free Trial agreement.

        The FreeTrialPricingTerm grants access at no cost for the trial period.
        The UsageBasedPricingTerm defines the charges that apply once the trial
 ends.
        """
        client = boto3.client("marketplace-agreement")

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[
                {"id": NewAmiFreeTrial.FREE_TRIAL_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.USAGE_BASED_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.SUPPORT_TERM_ID},
                {"id": NewAmiFreeTrial.LEGAL_TERM_ID},
            ],
```

```
 agreementProposalIdentifier=NewAmiFreeTrial.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "Agreement request with freeTrialPricingTerm accepted. AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewAmiFreeTrial.create_and_accept_ami_free_trial_agreement_request()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung erstellen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.TaxConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
```

```
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P12M}
 for 12 months).</li>
 *   <li>{@code DIMENSION_1_KEY}, {@code DIMENSION_2_KEY} — dimension keys
 defined in the offer.</li>
 *   <li>{@code DIMENSION_1_VALUE}, {@code DIMENSION_2_VALUE} — quantities for
 each dimension.</li>
 * </ul>
 */
public class NewSaaSContractWithUpfrontPayment {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P12M" for 12 months).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // First dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-1-key>";
    private static final int DIMENSION_1_VALUE = 10;

    // Second dimension key and quantity defined in your offer.
    private static final String DIMENSION_2_KEY = "<your-dimension-2-key>";
    private static final int DIMENSION_2_VALUE = 20;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Tax estimation setting: "ENABLED" to include estimated taxes in the
 agreement.
    private static final String TAX_ESTIMATION = "ENABLED";

    public static void main(String[] args) {
        createSaaSContractAgreement();
    }

    /**
     * Creates a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
     * for multiple dimensions and tax estimation.
```

```
     */
    private static void createSaaSContractAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build(),
                                                Dimension.builder()

.dimensionKey(DIMENSION_2_KEY)

.dimensionValue(DIMENSION_2_VALUE)
                                                        .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm)

.taxConfiguration(TaxConfiguration.builder().taxEstimation(TAX_ESTIMATION).build())

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
```

```
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("SaaS agreement request with CONTRACT pricing model
 accepted. AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");
```

```
/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P12M" for 12 months).
 *   - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
 *   - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P12M" for 12 months).
const SELECTOR_VALUE = "<your-selector-value>";

// First dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-1-key>";
const DIMENSION_1_VALUE = 10;

// Second dimension key and quantity defined in your offer.
const DIMENSION_2_KEY = "<your-dimension-2-key>";
const DIMENSION_2_VALUE = 20;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
const TAX_ESTIMATION = "ENABLED";
```

```
/**
 * Creates a SaaS agreement with CONTRACT pricing model with configurable upfront
 pricing
 * for multiple dimensions and tax estimation.
 */
async function createSaaSContractAgreement() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                    { dimensionKey: DIMENSION_2_KEY, dimensionValue:
 DIMENSION_2_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm],
            taxConfiguration: { taxEstimation: TAX_ESTIMATION },
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
```

```
    console.log("SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createSaaSContractAgreement();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
selecting an agreement duration and specifying quantities for multiple
 dimensions.
Tax estimation is enabled at the time of agreement creation.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P12M for 12 months).
  - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
  - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
"""
```

```
import boto3

from utils.agreement_api_utils import generate_client_token


class NewSaaSContractWithUpfrontPayment:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P12M" for 12 months).
    SELECTOR_VALUE = "<your-selector-value>"

    # First dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-1-key>"
    DIMENSION_1_VALUE = 10

    # Second dimension key and quantity defined in your offer.
    DIMENSION_2_KEY = "<your-dimension-2-key>"
    DIMENSION_2_VALUE = 20

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Tax estimation setting: "ENABLED" to include estimated taxes in the
 agreement.
    TAX_ESTIMATION = "ENABLED"

    @staticmethod
    def create_saas_contract_agreement():
        """
        Create a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
        for multiple dimensions and tax estimation.
        """
        client = boto3.client("marketplace-agreement")

        configurable_upfront_pricing_term = {
```

```
            "id":
NewSaaSContractWithUpfrontPayment.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue":
NewSaaSContractWithUpfrontPayment.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey":
NewSaaSContractWithUpfrontPayment.DIMENSION_1_KEY,
                            "dimensionValue":
NewSaaSContractWithUpfrontPayment.DIMENSION_1_VALUE,
                        },
                        {
                            "dimensionKey":
NewSaaSContractWithUpfrontPayment.DIMENSION_2_KEY,
                            "dimensionValue":
NewSaaSContractWithUpfrontPayment.DIMENSION_2_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": NewSaaSContractWithUpfrontPayment.LEGAL_TERM_ID}

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term],
            taxConfiguration={"taxEstimation":
NewSaaSContractWithUpfrontPayment.TAX_ESTIMATION},

agreementProposalIdentifier=NewSaaSContractWithUpfrontPayment.AGREEMENT_PROPOSAL_IDENTIF
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
```

```
            "SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewSaaSContractWithUpfrontPayment.create_saas_contract_agreement()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Rufen Sie alle Vereinbarungs-IDs mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle Vereinbarungs-IDs abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some
 time to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();

  ReferenceCodesUtils.formatOutput(agreementIds);

 }

 public static List<String> getAllAgreementIds() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
 agreements where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
 are the acceptor.
  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  List<Filter> searchFilters = new ArrayList<Filter>();

  searchFilters.addAll(Arrays.asList(partyType, agreementType));
```

```
  // Save all results in a list array
  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters)
     .build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }

  List<String> agreementIds = new ArrayList<String>();
  for (AgreementViewSummary summary : agreementSummaryList) {
   agreementIds.add(summary.agreementId());
  }
  return agreementIds;
 }

}
```

- Einzelheiten zur API finden Sie SearchAgreementsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement ids
AG-09
"""

import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements():
    AgreementSummaryList = []
    agreement_id_list = []

    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
```

```
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    for agreement in AgreementSummaryList:
        agreement_id_list.append(agreement["agreementId"])

    return agreement_id_list


if __name__ == "__main__":
    agreement_id_list = get_agreements()

    print(agreement_id_list)
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie alle Vereinbarungen mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie alle Vereinbarungen abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {
```

```
/*
 * Get all purchase agreements for your PartyType (Proposer, or Acceptor)
 * Depend on the number of agreements in your account, this code may take some
time to finish.
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAllAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);
}

public static List<AgreementViewSummary> getAllAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

       // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return
agreements where you are the acceptor.
       // Change to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements where
you are the proposer.
 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_ACCEPTOR).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();
```

```
  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters).build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- Einzelheiten zur API finden Sie SearchAgreementsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreements
AG-01
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

party_type_list = ["Acceptor"]
agreement_type_list = ["PurchaseAgreement"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e
```

```
    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
    helper.pretty_print_datetime(agreements_list)
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die Kundennummer aus einer Vereinbarung mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Kunden-ID aus einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {

 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();
```

```
  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get customer AWS account id
from a given agreement
AG-08
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    print(f"Customer account: {response['acceptor']['accountId']}")
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Einzelheiten zu einer Anfrage zur Kündigung eines Vertrags erhalten Sie unter AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Details zu einer bestimmten Anfrage zur Kündigung eines Vertrags abrufen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationReque

public class GetAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementCancellationRequest();
    }

    private static void getAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementCancellationRequestRequest request =
                GetAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
```

```
 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        GetAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.getAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie GetAgreementCancellationRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    GetAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
```

```
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function getAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementCancellationRequest();
```

- Einzelheiten zur API finden Sie GetAgreementCancellationRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
```

```
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"

    @staticmethod
    def get_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_cancellation_request(
            agreementId=GetAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=GetAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementCancellationRequest.get_agreement_cancellation_request()
```

- Einzelheiten zur API finden Sie GetAgreementCancellationRequestin AWS SDK for Python (Boto3) API Reference.

Rufen Sie Einzelheiten zu einer Zahlungsaufforderung für einen Vertrag ab, indem Sie AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Details zu einer bestimmten Zahlungsaufforderung für eine Vereinbarung abrufen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestRes

public class GetAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementPaymentRequest();
    }

    private static void getAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
```

```
                        .build();

        GetAgreementPaymentRequestRequest request =
                GetAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .build();

        GetAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.getAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge ID: " + response.chargeId());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie GetAgreementPaymentRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
```

```
    GetAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

async function getAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge ID: " + response.chargeId);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementPaymentRequest();
```

- Einzelheiten zur API finden Sie GetAgreementPaymentRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"

    @staticmethod
    def get_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_payment_request(
            agreementId=GetAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=GetAgreementPaymentRequest.PAYMENT_REQUEST_ID,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge ID: " + str(response.get("chargeId", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
```

```
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementPaymentRequest.get_agreement_payment_request()
```

- Einzelheiten zur API finden Sie GetAgreementPaymentRequestin AWS SDK for Python (Boto3) API Reference.

Holen Sie sich finanzielle Details aus einer Vereinbarung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie finanzielle Angaben aus einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
```

```
  * Obtain financial details, such as Total Contract Value of the agreement from
 a given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
   totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain financial details, such as Total Contract Value of the agreementfrom a
 given agreement
AG-14

Example Usage: python3 get_agreement_financial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_information(agreement_id):
    try:
        agreement = mp_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_information(args.agreement_id)

    if agreement is not None:
        print(f"Agreement Id: {args.agreement_id}")
        print(
            f"Agreement Value: {agreement['estimatedCharges']['currencyCode']}
 {agreement['estimatedCharges']['agreementValue']}"
        )

    else:
        print(f"Agreement with ID {args.agreement_id} is not found")
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Einzelheiten zur kostenlosen Testversion finden Sie in einer Vereinbarung unter AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Einzelheiten zur kostenlosen Testversion aus einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;
```

```
  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
  }
  return freeTrialPricingTerms;
 }
}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the details from an agreement of a free trial I have provided to the
 customer
AG-20

Example Usage: python3 get_agreement_free_trial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
```

```
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        freetrial_found = False

        for term in agreement["acceptedTerms"]:
            if "freeTrialPricingTerm" in term.keys():
                helper.pretty_print_datetime(term)
                freetrial_found = True

        if not freetrial_found:
            print(f"No free trial term found for agreement: {args.agreement_id}")
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Informationen zu einer Vereinbarung erhalten Sie mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Informationen über eine Vereinbarung abgerufen werden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

  ReferenceCodesUtils.formatOutput(describeAgreementResponse);

 }

 public static DescribeAgreementResponse getResponse(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
```

```
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement information
AG-07
"""

import argparse
import logging

import boto3
```

```
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(response)
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Rufen Sie Produkt- und Angebotsdetails aus einer Vereinbarung mithilfe eines ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Produkt- und Angebotsdetails aus einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {

  // call Agreement API to get offer and product information for the agreement

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);

  for (DescribeEntityResponse response : entityResponseList) {
   ReferenceCodesUtils.formatOutput(response);
  }
 }

 public static List<DescribeEntityResponse> getEntities(String agreementId) {
  List<DescribeEntityResponse> entityResponseList = new
 ArrayList<DescribeEntityResponse> ();

  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  // get offer id for the given agreement

  String offerId = describeAgreementResponse.proposalSummary().offerId();

  // get all the product ids for this agreement
```

```
  List<String> productIds = new ArrayList<String>();
  for (Resource resource :
 describeAgreementResponse.proposalSummary().resources()) {
   productIds.add(resource.id());
  }

  // call Catalog API to get the details of the offer and products

  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId).build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get product and offer details
 in a given agreement
AG-10
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mpa_client = boto3.client("marketplace-agreement")
mpc_client = boto3.client("marketplace-catalog")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    """
    Returns information about a given agreement
    Args: agreement_id str: Entity to return
```

```
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mpa_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mpc_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_components(agreement_id):
    agreement_component_list = []

    agreement = get_agreement_information(agreement_id)

    if agreement is not None:
        productIds = []
```

```
        for resource in agreement["proposalSummary"]["resources"]:
            productIds.append(resource["id"])

        for product_id in productIds:
            product_document = get_entity_information(product_id)

            product_document_dict = {}
            product_document_dict["product_id"] = product_id
            product_document_dict["document"] = product_document
            agreement_component_list.append(product_document_dict)

        offerId = agreement["proposalSummary"]["offerId"]

        offer_document = get_entity_information(offerId)

        offer_document_dict = {}
        offer_document_dict["offer_id"] = offerId
        offer_document_dict["document"] = offer_document
        agreement_component_list.append(offer_document_dict)

        return agreement_component_list

    else:
        print("Agreement with ID " + args.agreement_id + " is not found")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement_id",
        "-aid",
        help="Provide agreement ID to search for product and offer detail",
        required=True,
    )
    args = parser.parse_args()

    product_offer_detail =
 get_agreement_components(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(product_offer_detail)
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die EULA einer Vereinbarung mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die EULA einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
```

```
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
      .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
      .filter(docItem -> docItem.type() != null)
      .forEach(legalEulaArray::add);
  return legalEulaArray;
 }

}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the EULA I have entered into with my customer via the agreement
AG-18
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
```

```
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "legalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "documents"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        legalEulaArray = []
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term and ATTRIBUTE_NAME in term[TERM_NAME]:
                docs = term[TERM_NAME][ATTRIBUTE_NAME]
                for doc in docs:
                    if "type" in doc:
                        legalEulaArray.append(doc)
        return legalEulaArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)
```

```
    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Holen Sie sich die Bedingungen für die auto Verlängerung einer Vereinbarung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Bedingungen für die automatische Verlängerung einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {

 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
```

```
    if (acceptedTerm.renewalTerm() != null &&
  acceptedTerm.renewalTerm().configuration() != null
      && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
     autoRenewal =
  String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue
     break;
    }
   }
   return autoRenewal;
  }

 }
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the auto-renewal status of the agreement
AG-15
"""

import json
import logging
import os
import sys
import os
```

```
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper


import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-11111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "renewalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "enableAutoRenew"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        auto_renewal = "No Auto Renewal"
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
```

```
                    auto_renewal = term[TERM_NAME][CONFIG_ELEM][ATTRIBUTE_NAME]
                    break
        return auto_renewal

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        print(f"Auto Renewal is {agreement}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Holen Sie sich die im Rahmen eines Vertrags erworbenen Abmessungen mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die im Rahmen einer Vereinbarung gekauften Dimensionen abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
  */
```

```
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> dimensionKeys = getDimensionKeys(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionKeys);
 }

 public static List<String> getDimensionKeys(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<String> dimensionKeys = new ArrayList<String>();
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !
= null) {
     List<Dimension> dimensions =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     for (Dimension dimension : dimensions) {
      dimensionKeys.add(dimension.dimensionKey());
     }
    }

   }
  }
  return dimensionKeys;
 }
}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the dimensions the buyer has purchased from me via the agreement
AG-28
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-11111111111111111111111111"

# to use sample file or not
```

```
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimensionKeys = []

        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        if "dimensions" in confParam:
                            for dimension in confParam["dimensions"]:
                                if "dimensionKey" in dimension:
                                    dimensionKey = dimension["dimensionKey"]
                                    print(f"Dimension Key: {dimensionKey}")
                                    dimensionKeys.append(dimensionKey)
        return dimensionKeys

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
```

```
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die Instanzen jeder Dimension ab, die im Rahmen einer Vereinbarung erworben wurden, mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die im Rahmen einer Vereinbarung gekauften Instances jeder Dimensionen abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionInstances {

 /*
  * get instances of each dimension that buyer has purchased in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;
```

```
  Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionMap);
 }

 public static Map<String, List<Dimension>> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  Map<String, List<Dimension>> dimensionMap = new HashMap<String,
 List<Dimension>>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Dimension> dimensionsList = new ArrayList<Dimension>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    String selectorValue = "";
    if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !
= null) {
      selectorValue =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
     }
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions())
 {
      dimensionsList =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     }
    }
    if (selectorValue.length() > 0) {
     dimensionMap.put(selectorValue, dimensionsList);
    }
   }
```

```
  }
  return dimensionMap;
 }
}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain instances of each dimension that buyer has purchased in the agreement
AG-30
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

logger = logging.getLogger(__name__)


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        terms = mp_client.get_agreement_terms(agreementId=entity_id)
        dimensionKeyValueMap = {}
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        selectValue = confParam["selectorValue"]
                        dimensionKeyValueMap["selectorValue"] = selectValue
                        if "dimensions" in confParam:
                            dimensionKeyValueMap["dimensions"] =
 confParam["dimensions"]
                            """
                            for dimension in confParam['dimensions']:
                                if 'dimensionKey' in dimension:

                                    dimensionValue = dimension['dimensionValue']
                                    dimensionKey = dimension['dimensionKey']
                                    print(f"Selector: {selectValue}, Dimension
 Key: {dimensionKey}, Dimension Value: {dimensionValue}")
                                    dimensionKeyValueMap[dimensionKey] =
 dimensionValue
                            """
        return dimensionKeyValueMap
```

```
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie den Zahlungsplan einer Vereinbarung mithilfe eines ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Zahlungsplan einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including
 the invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray =
 getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId)
 {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();

  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
       scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
       scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
       scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
       paymentScheduleArray.add(scheduleMap);
      }
     }
    }
   }
  }
  return paymentScheduleArray;
 }
}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
AG-17
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"
```

```
# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "paymentScheduleTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        paymentScheduleArray = []
        currencyCode = ""
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                paymentSchedule = term[TERM_NAME]
                if "currencyCode" in paymentSchedule:
                    currencyCode = paymentSchedule["currencyCode"]
                if "schedule" in paymentSchedule:
                    for sch in paymentSchedule["schedule"]:
                        if "chargeDate" in sch:
                            chargeDate = sch["chargeDate"]
                            chargeAmount = sch["chargeAmount"]
                            # print(f"chargeDate: {chargeDate}, chargeAmount:
 {chargeAmount}")
                            schedule = {
                                "currencyCode": currencyCode,
                                "chargeDate": chargeDate,
                                "chargeAmount": chargeAmount,
                            }
                            paymentScheduleArray.append(schedule)

        return paymentScheduleArray
```

```
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die Preise pro Dimension in einer Vereinbarung mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Preise pro Dimension in einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
```

```
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<Object> dimensions = new ArrayList<Object>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
    }
    dimensions.add(rateInfo);
   }
  }
  return dimensions;
 }
}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain pricing per each dimension in the agreement
AG-29
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
```

```
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

TERMS_TO_SEARCH = [
    "configurableUpfrontPricingTerm",
    "usageBasedPricingTerm",
    "fixedUpfrontPricingTerm",
]


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__),
 SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimentions = []
        for term in terms[ROOT_ELEM]:
            for t in TERMS_TO_SEARCH:
                rateInfo = []
                if t in term:
                    if "type" in term[t]:
                        rateInfo.append(term[t]["type"])
                    if "currencyCode" in term[t]:
                        rateInfo.append(term[t]["currencyCode"])
                    if "rateCards" in term[t]:
                        rateInfo.append(term[t]["rateCards"])
                    dimentions.append(rateInfo)
        return dimentions

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
```

```
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client,
 AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Ermitteln Sie den Preistyp einer Vereinbarung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Preistyp einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;
```

```
import org.apache.commons.lang3.tuple.Triple;

import
 software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
```

```
private static final String CFPS = "Contract FPS";
private static final String CCPFPS = "CCP with FPS";
private static final String BYOL = "BYOL";
private static final String FREE = "Free";
private static final String FTH = "Free Trials and Hourly";

// Agreement term pricing types
private static final Set<String> LEGAL = Set.of("LegalTerm");
private static final Set<String> CONFIGURABLE_UPFRONT =
Set.of("ConfigurableUpfrontPricingTerm");
private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
private static final Set<String> RECURRING_PAYMENT =
Set.of("RecurringPaymentTerm");
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String>
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
```

```
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and
offer types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();

 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
new HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
new HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()),
HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new
HashSet<>()), MONTHLY);
```

```
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
```

```
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
```

```
 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).b

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PUR

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());

 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
```

```
    return Objects.hash(first, second);
  }

  @Override
  public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null || getClass() != obj.getClass())
      return false;
    KeyPair other = (KeyPair) obj;
    return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
  }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {

  Set<String> offerTermTypes = new HashSet<String>();

  DescribeEntityRequest request =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse result =
marketplaceCatalogClient.describeEntity(request);

  String details = result.details();

  try {
    ObjectMapper objectMapper = new ObjectMapper();
    JsonNode rootNode = objectMapper.readTree(details);
    JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

    for (JsonNode termNode : termsNode) {
      if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
        offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
      }
    }
  } catch (Exception e) {
```

```
    e.printStackTrace();
   }

   return offerTermTypes;

  }

  /*
   * Get all the agreement term types
   */
  public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
  agreementTerm) {
   Set<String> agreementTermTypes = new HashSet<String>();
   try {
    for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
     ObjectMapper objectMapper = new ObjectMapper();
     JsonNode termNode = objectMapper.readTree(getJson(term));
     Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
     while (fieldsIterator.hasNext()) {
      Map.Entry<String, JsonNode> entry = fieldsIterator.next();
      JsonNode value = entry.getValue();
      if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
       agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
      }
     }
    }
   } catch (Exception e) {
    e.printStackTrace();
   }
   return agreementTermTypes;

  }

  /*
   * make sure all elements in array2 exist in array1
   */
  public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
   for (String element : array2) {
    boolean found = false;
    for (String str : array1) {
     if (element.equals(str)) {
      found = true;
      break;
     }
```

```
    }
    if (!found) {
     return false;
    }
   }
   return true;
  }

  /*
   * Find the combinations of the agreement term types for the agreement
   */
  public static Set<String> getMatchedTermTypesCombination(Set<String>
  agreementTermTypes) {
   Set<String> matchedCombination = new HashSet<String>();
   for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
    if (allElementsExist(agreementTermTypes, element)) {
     matchedCombination = element;
    }
   }
   return matchedCombination;
  }

  public static void main(String[] args) {

   List<AgreementViewSummary> agreements = getAgreementsById();

   for (AgreementViewSummary summary : agreements) {
    String pricingType = "";
    String agreementId = summary.agreementId();
    System.out.println(agreementId);
    String offerId = summary.proposalSummary().offerId();

    //get all pricing term types for the offer in the agreement
    Set<String> offerTermTypes = getOfferTermTypes(offerId);
    String productType = summary.proposalSummary().resources().get(0).type();

    //get all pricing term types for the agreement
    GetAgreementTermsRequest getAgreementTermsRequest =
      GetAgreementTermsRequest.builder().agreementId(agreementId)
      .build();
    GetAgreementTermsResponse getAgreementTermsResponse =
  marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
    Set<String> agreementTermTypes =
  getAgreementTermTypes(getAgreementTermsResponse);
```

```
    //get matched pricing term type combination set
    Set<String> agreementMatchedTermType =
  getMatchedTermTypesCombination(agreementTermTypes);

    //check to see if this agreement pricing term combination needs additional
  check on offer pricing terms
    String needToCheckOfferType = needToCheckOfferTermsType(productType,
  agreementMatchedTermType);

    // get the pricing type for the agreement based on the product type, agreement
  term types and offer term types if needed
    if (needToCheckOfferType != null) {
     Set<String> offerMatchedTermType =
  getMatchedTermTypesCombination(offerTermTypes);
     pricingType = getPricingType(productType, agreementMatchedTermType,
  offerMatchedTermType);
    } else if (agreementMatchedTermType == LEGAL) {
     pricingType = FREE;
    } else {
     pricingType = getPricingType(productType, agreementMatchedTermType, new
  HashSet());
    }
    System.out.println("Pricing type is " + pricingType);
   }
  }

  private static String getJson(Object result) {
   String json = "";

   try {
    ObjectMapper om = new ObjectMapper();
    om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
    om.registerModule(new JavaTimeModule());
    ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

    json = ow.writeValueAsString(result);
   } catch (JsonProcessingException e) {
    e.printStackTrace();
   }
   return json;
  }

 }
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
AG-16
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "OfferId"

# replace id value as needed
idValue = "offer-1111111111111"

MAX_PAGE_RESULTS = 10

# catalog; switch to AWSMarketplace for release
AWSMPCATALOG = "AWSMarketplace"
```

```
# product types

SaaSProduct = "SaaSProduct"
AmiProduct = "AmiProduct"
MLProduct = "MachineLearningProduct"
ContainerProduct = "ContainerProduct"
DataProduct = "DataProduct"
ProServiceProduct = "ProfessionalServicesProduct"
AiqProduct = "AiqProduct"

# Define pricing types
CCP = "CCP"
Annual = "Annual"
Contract = "Contract"
SFT = "SaaS Free Trial"
HMA = "Hourly and Monthly Agreements"
Hourly = "Hourly"
Monthly = "Monthly"
AFPS = "Annual FPS"
CFPS = "Contract FPS"
CCPFPS = "CCP with FPS"
BYOL = "BYOL"
Free = "Free"
FTH = "Free Trials and Hourly"

# Define Agreement Term Types
legal = ["LegalTerm"]
config = ["ConfigurableUpfrontPricingTerm"]
usage = ["UsageBasedPricingTerm"]
config_usage = ["ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm"]
freeTrial = ["FreeTrialPricingTerm"]
recur = ["RecurringPaymentTerm"]
usage_recur = ("UsageBasedPricingTerm", "RecurringPaymentTerm")
fixed_payment = ["FixedUpfrontPricingTerm", "PaymentScheduleTerm"]
fixed_payment_usage = [
    "FixedUpfrontPricingTerm",
    "PaymentScheduleTerm",
    "UsageBasedPricingTerm",
]
byol = ["ByolPricingTerm"]
freeTrial_usage = ("FreeTrialPricingTerm", "UsageBasedPricingTerm")
all_agreement_types_combination = (
    legal,
```

```
    config,
    usage,
    config_usage,
    freeTrial,
    recur,
    usage_recur,
    fixed_payment,
    fixed_payment_usage,
    byol,
    freeTrial_usage,
)


# get pricing type method given product type, agreement temr type and offer type
 if needed
def get_pricing_type(product_type, agreement_term_type, offer_type):
    pricing_types = {
        (SaaSProduct, frozenset(config_usage), frozenset("")): CCP,
        (DataProduct, frozenset(config_usage), frozenset("")): CCP,
        (ContainerProduct, frozenset(config), frozenset(config_usage)): Annual,
        (AmiProduct, frozenset(config), frozenset(config_usage)): Annual,
        (MLProduct, frozenset(config), frozenset(config_usage)): Annual,
        (ContainerProduct, frozenset(config), frozenset(config)): Contract,
        (AmiProduct, frozenset(config), frozenset(config)): Contract,
        (SaaSProduct, frozenset(config), frozenset("")): Contract,
        (DataProduct, frozenset(config), frozenset("")): Contract,
        (AiqProduct, frozenset(config), frozenset("")): Contract,
        (ProServiceProduct, frozenset(config), frozenset("")): Contract,
        (SaaSProduct, frozenset(freeTrial), frozenset("")): SFT,
        (AmiProduct, frozenset(usage_recur), frozenset("")): HMA,
        (SaaSProduct, frozenset(usage), frozenset("")): Hourly,
        (AmiProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(usage), frozenset("")): Hourly,
        (MLProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(recur), frozenset("")): Monthly,
        (AmiProduct, frozenset(recur), frozenset("")): Monthly,
        (
            ContainerProduct,
            frozenset(fixed_payment),
            frozenset(fixed_payment_usage),
        ): AFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment_usage)):
 AFPS,
        (MLProduct, frozenset(fixed_payment), frozenset("")): AFPS,
```

```
        (ContainerProduct, frozenset(fixed_payment), frozenset(fixed_payment)):
 CFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): CFPS,
        (SaaSProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (DataProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (AiqProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (ProServiceProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (SaaSProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (DataProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AiqProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (ProServiceProduct, frozenset(fixed_payment_usage), frozenset("")):
 CCPFPS,
        (AmiProduct, frozenset(byol), frozenset("")): BYOL,
        (SaaSProduct, frozenset(byol), frozenset("")): BYOL,
        (ProServiceProduct, frozenset(byol), frozenset("")): BYOL,
        (AiqProduct, frozenset(byol), frozenset("")): BYOL,
        (MLProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(byol), frozenset("")): BYOL,
        (DataProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(legal), frozenset("")): Free,
        (AmiProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (ContainerProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (MLProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
    }

    key = (product_type, agreement_term_type, offer_type)
    if key in pricing_types:
        return pricing_types[key]
    else:
        return "Unknown"


# Example usage for testing purpose
"""
product_type = SaaSProduct
agreement_term_type = frozenset(config_usage)
offer_type = frozenset('')
pricing_type = get_pricing_type(product_type, agreement_term_type, offer_type)
print("pricing type = " + pricing_type)  # Output: CCP
"""


# check if offer term types are needed; if Y, needed
def get_offer_term_type(product_type, agreement_term_type):
```

```
    offer_term_types = {
        (ContainerProduct, frozenset(config)): "Y",
        (AmiProduct, frozenset(config)): "Y",
        (ContainerProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): "Y",
    }

    key = (product_type, agreement_term_type)
    if key in offer_term_types:
        return offer_term_types[key]
    else:
        return


logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog=AWSMPCATALOG,
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
```

```
                    catalog=AWSMPCATALOG,
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    # find all agreements matching the specified idType and idValue
    agreements = get_agreements(mp_client)

    for item in agreements:
        pricingType = ""
        agreement_id = item["agreementId"]

        # get term types inside offer
        offer_term_types = get_offer_term_types(item)

        # even though multiple product types are allowed for one agreement, only
 need the first one
        productType = item["proposalSummary"]["resources"][0]["type"]

        # get agreement terms types
        agreementTerm = mp_client.get_agreement_terms(agreementId=agreement_id)
```

```
        agreementTermTypes = get_agreement_term_types(agreementTerm)

        # match with agreement term type group
        matchedTermType = getMatchedTermTypesCombination(agreementTermTypes)

        # check if offer term type is needed.
        offer_term_type_needed = get_offer_term_type(
            productType, frozenset(matchedTermType)
        )

        # get pricing type given product type, agreement term types and offer
 type if needed;
        # one excpetion is Container with Legal term. LegalTerm needs to be the
 only term present
        if offer_term_type_needed is not None:
            matchedOfferTermTypes =
 getMatchedTermTypesCombination(offer_term_types)
            print(f"matchedOfferTermType = {matchedOfferTermTypes}")
            pricingType = get_pricing_type(
                productType,
                frozenset(matchedTermType),
                frozenset(matchedOfferTermTypes),
            )
        elif set(matchedTermType) == set(legal):
            pricingType = Free
        else:
            pricingType = get_pricing_type(
                productType, frozenset(matchedTermType), frozenset("")
            )

        print(
            f"agreementId={agreement_id};productType={productType};
 agreementTermTypes={agreementTermTypes}; matchedTermType={matchedTermType};
 offerTermTypeNeeded={offer_term_type_needed};
 offer_term_types={offer_term_types}"
        )
        print(f"pricing type={pricingType}")


def getMatchedTermTypesCombination(agreementTermTypes):
    matchedCombination = ()
    for element in all_agreement_types_combination:
        if check_elements(agreementTermTypes, element):
```

```
            matchedCombination = element
    return matchedCombination


def get_offer_term_types(item):
    offer_id = item["proposalSummary"]["offerId"]
    mp_catalogAPI_client = boto3.client("marketplace-catalog")
    offer_document = get_entity_information(mp_catalogAPI_client, offer_id)
    offerDetail = offer_document["Details"]
    offerDetail_json_object = json.loads(offerDetail)
    offer_term_types = [term["Type"] for term in
 offerDetail_json_object["Terms"]]
    return offer_term_types


# make sure all elements in array2 exist in array1
def check_elements(array1, array2):
    for element in array2:
        if element not in array1:
            return False
    return True


def get_entity_information(mp_client, entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)
```

```
def get_agreement_term_types(agreementTerm):
    types = []
    for term in agreementTerm["acceptedTerms"]:
        for value in term.values():
            if isinstance(value, dict) and "type" in value:
                types.append(value["type"])
    return types


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Rufen Sie den Produkttyp einer Vereinbarung mithilfe eines ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Produkttyp einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  List<String> productIds = new ArrayList<String>();
  for (Resource resource :
 describeAgreementResponse.proposalSummary().resources()) {
   productIds.add(resource.id() + ":" + resource.type());
```

```
  }
  return productIds;
 }
}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the Product Type of the product the agreement was created on
AG-11
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"


def get_agreement_information(mp_client, entity_id):
    """
```

```
    Returns information about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mp_client.describe_agreement(agreementId=entity_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for offer and product details in a given agreement by
 agreement id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        for resource in agreement["proposalSummary"]["resources"]:
            print(f"Product ID: {resource['id']}  |  Product Type:
 {resource['type']}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Rufen Sie den Status einer Vereinbarung mit einem ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie den Status einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementStatus {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Agreement status is " +
 describeAgreementResponse.status());

 }
```

```
 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- Einzelheiten zur API finden Sie DescribeAgreementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement status
```

```
AG-13

Example Usage: python3 get_agreement_status.py --agreement-id <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
        return response
    except ClientError as e:
        logger.error(f"Could not complete search_agreements request. {e}")

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement(agreement_id=args.agreement_id)

    if response is not None:
        print(f"Agreement status: {response['status']}")
    else:
        print(f"No agreement found for {args.agreement_id}")
```

- Einzelheiten zur API finden Sie DescribeAgreementin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die Support-Bedingungen einer Vereinbarung mithilfe eines ab AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Support-Bedingungen einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static
 com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
```

```
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }

 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the support and refund policy I have provided to the customer for an
 agreement
AG-19

Example Usage: python3 get_agreement_support_terms.py --agreement-id <agreement-
id>
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
```

```
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        support_found = False

        for term in agreement["acceptedTerms"]:
            if "supportTerm" in term.keys():
                helper.pretty_print_datetime(term)
                support_found = True

        if not support_found:
            print(f"No support term found for agreement: {args.agreement_id}")
```

- Einzelheiten zur API finden Sie GetAgreementTermsin AWS SDK for Python (Boto3) API Reference.

Rufen Sie die Bedingungen einer Vereinbarung mit einem ab AWS SDK

Das folgende Codebeispiel zeigt, wie Sie die Bedingungen einer Vereinbarung abrufen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
 }

}
```

- Einzelheiten zur API finden Sie GetAgreementTermsin der AWS SDK for Java 2.x API-Referenz.

Listen Sie Anfragen zur Kündigung von Verträgen mit einem auf AWS SDK

Die folgenden Codebeispiele zeigen, wie Anfragen zur Kündigung von Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementCancellationRequestS
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequ
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequ

public class ListAgreementCancellationRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementCancellationRequests();
    }

    private static void listAgreementCancellationRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementCancellationRequestsRequest request =
                    ListAgreementCancellationRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementCancellationRequestsResponse response =

 marketplaceAgreementClient.listAgreementCancellationRequests(request);

            for (AgreementCancellationRequestSummary summary : response.items())
 {
                System.out.println("Cancellation Request ID: " +
 summary.agreementCancellationRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
```

```
                System.out.println("Reason Code: " +
 summary.reasonCodeAsString());
                System.out.println("Agreement Type: " + summary.agreementType());
                System.out.println("Catalog: " + summary.catalog());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Einzelheiten zur API finden Sie ListAgreementCancellationRequestsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    ListAgreementCancellationRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementCancellationRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;
```

```
    do {
        const response = await client.send(
            new ListAgreementCancellationRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Cancellation Request ID: " +
 summary.agreementCancellationRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Reason Code: " + summary.reasonCode);
            console.log("Agreement Type: " + summary.agreementType);
            console.log("Catalog: " + summary.catalog);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementCancellationRequests();
```

- Einzelheiten zur API finden Sie ListAgreementCancellationRequestsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementCancellationRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_cancellation_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementCancellationRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_cancellation_requests(**kwargs)

            for summary in response.get("items", []):
                print("Cancellation Request ID: " +
 summary["agreementCancellationRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Reason Code: " + str(summary.get("reasonCode", "")))
                print("Agreement Type: " + str(summary.get("agreementType", "")))
                print("Catalog: " + str(summary.get("catalog", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
```

```
    ListAgreementCancellationRequests.list_agreement_cancellation_requests()
```

- Einzelheiten zur API finden Sie ListAgreementCancellationRequestsin AWS SDK for Python (Boto3) API Reference.

Führen Sie Zahlungsanfragen für Vereinbarungen mit einem auf AWS SDK

Die folgenden Codebeispiele zeigen, wie Zahlungsaufforderungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsR
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsR
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentRequestSummary;

public class ListAgreementPaymentRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementPaymentRequests();
```

```
    }

    private static void listAgreementPaymentRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementPaymentRequestsRequest request =
                    ListAgreementPaymentRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementPaymentRequestsResponse response =

 marketplaceAgreementClient.listAgreementPaymentRequests(request);

            for (PaymentRequestSummary summary : response.items()) {
                System.out.println("Payment Request ID: " +
 summary.paymentRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Name: " + summary.name());
                System.out.println("Charge ID: " + summary.chargeId());
                System.out.println("Charge Amount: " + summary.chargeAmount());
                System.out.println("Currency Code: " + summary.currencyCode());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- Einzelheiten zur API finden Sie ListAgreementPaymentRequestsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    ListAgreementPaymentRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementPaymentRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;

    do {
        const response = await client.send(
            new ListAgreementPaymentRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Payment Request ID: " + summary.paymentRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Name: " + summary.name);
            console.log("Charge ID: " + summary.chargeId);
            console.log("Charge Amount: " + summary.chargeAmount);
```

```
            console.log("Currency Code: " + summary.currencyCode);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementPaymentRequests();
```

- Einzelheiten zur API finden Sie ListAgreementPaymentRequestsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementPaymentRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_payment_requests():
```

```
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementPaymentRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_payment_requests(**kwargs)

            for summary in response.get("items", []):
                print("Payment Request ID: " + summary["paymentRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Name: " + str(summary.get("name", "")))
                print("Charge ID: " + str(summary.get("chargeId", "")))
                print("Charge Amount: " + str(summary.get("chargeAmount", "")))
                print("Currency Code: " + str(summary.get("currencyCode", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementPaymentRequests.list_agreement_payment_requests()
```

- Einzelheiten zur API finden Sie ListAgreementPaymentRequestsin AWS SDK for Python (Boto3) API Reference.

Lehnen Sie eine Anfrage zur Kündigung eines Vertrags ab, indem Sie AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine vom Verkäufer initiierte Anfrage zur Kündigung eines Vertrags ablehnen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationRe
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationRe

public class RejectAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementCancellationRequest();
    }

    private static void rejectAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementCancellationRequestRequest request =
                RejectAgreementCancellationRequestRequest.builder()
```

```
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.rejectAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie RejectAgreementCancellationRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
```

```
const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementCancellationRequest();
```

- Einzelheiten zur API finden Sie RejectAgreementCancellationRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_cancellation_request(
            agreementId=RejectAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=RejectAgreementCancellationRequest.AGREEMENT_CANCELLATION
            rejectionReason=RejectAgreementCancellationRequest.REJECTION_REASON,
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementCancellationRequest.reject_agreement_cancellation_request()
```

- Einzelheiten zur API finden Sie RejectAgreementCancellationRequestin AWS SDK for Python (Boto3) API Reference.

Lehnen Sie eine Zahlungsaufforderung für einen Vertrag ab, indem Sie einen AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine vom Verkäufer initiierte Zahlungsaufforderung für einen Vertrag ablehnen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequest

public class RejectAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementPaymentRequest();
    }

    private static void rejectAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
```

```
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementPaymentRequestRequest request =
                RejectAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementPaymentRequestResponse response =

 marketplaceAgreementClient.rejectAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- Einzelheiten zur API finden Sie RejectAgreementPaymentRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementPaymentRequest();
```

- Einzelheiten zur API finden Sie RejectAgreementPaymentRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_payment_request(
            agreementId=RejectAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=RejectAgreementPaymentRequest.PAYMENT_REQUEST_ID,
            rejectionReason=RejectAgreementPaymentRequest.REJECTION_REASON,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
```

```
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementPaymentRequest.reject_agreement_payment_request()
```

- Einzelheiten zur API finden Sie RejectAgreementPaymentRequestin AWS SDK for Python (Boto3) API Reference.

Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vereinbarungsbasiertes Angebot unter Verwendung eines AWS SDK

Die folgenden Codebeispiele zeigen, wie eine SaaS-Vertragsvereinbarung durch ein neues vertragsbasiertes Angebot ersetzt wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model
 and receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing
 agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * <p>Note: Unlike other Replace samples, this example starts from an already-
active
 * {@code EXISTING_AGREEMENT_ID} rather than creating a new agreement first. Use
 this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code EXISTING_AGREEMENT_ID} — the agreement ID of the active agreement
 to replace (starts with {@code agmt-}).</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new ABO offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the new offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSContractWithAgreementBasedOffer {

    // The agreement ID of the active SaaS agreement with CONTRACT pricing model
 to replace (starts with "agmt-").
    private static final String EXISTING_AGREEMENT_ID = "<your-existing-
agreement-id>";

    // The agreementProposalId from the new Agreement-Based Offer.
```

```
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the LegalTerm in the new offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the FixedUpfrontPricingTerm in the new offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the new offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-
schedule-term-id>";

    public static void main(String[] args) {
        replaceExistingAgreement();
    }

    /**
     * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
     * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
     */
    private static void replaceExistingAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm =
 RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
 RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
 RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();

        // Replace the agreement with the new offer
```

```
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(fixedUpfrontPricingTerm,
 paymentScheduleTerm, legalTerm, validityTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(EXISTING_AGREEMENT_ID)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Replace agreement request created.
 AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing
 agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * Note: Unlike other Replace samples, this example starts from an already-active
 * EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to
 replace (starts with "agmt-").
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 ABO offer.
 *   - Term IDs (starting with "term-") — found in the new offer's term list.
```

```
 */

// The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
const EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>";

// The agreementProposalId from the new Agreement-Based Offer.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the LegalTerm in the new offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the FixedUpfrontPricingTerm in the new offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the new offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

/**
 * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
 * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
 */
async function replaceExistingAgreement() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };

    // Replace the agreement with the new offer
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [fixedUpfrontPricingTerm, paymentScheduleTerm,
 legalTerm, validityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
```

```
            sourceAgreementIdentifier: EXISTING_AGREEMENT_ID,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId);
}

replaceExistingAgreement();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
Agreement-Based Offer from the seller. The buyer replaces the existing agreement
```

```
with the new ABO offer, which may include updated pricing terms and payment
 schedule.

Note: Unlike other Replace samples, this example starts from an already-active
EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
pattern when you already have an agreement ID and want to replace it directly.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with agmt-).
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
  - Term IDs (starting with term-) — found in the new offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class ReplaceSaaSContractWithAgreementBasedOffer:

    # The agreement ID of the active SaaS agreement with CONTRACT pricing model
 to replace (starts with "agmt-").
    EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>"

    # The agreementProposalId from the new Agreement-Based Offer.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the LegalTerm in the new offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the FixedUpfrontPricingTerm in the new offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the new offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"
```

```
    @staticmethod
    def replace_existing_agreement():
        """
        Replace an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.

        Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSContractWithAgreementBasedOffer

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}

        # Replace the agreement with the new offer
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[fixed_upfront_pricing_term, payment_schedule_term,
 legal_term, validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=cls.EXISTING_AGREEMENT_ID,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Replace agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print("Agreement replaced with ABO. New AgreementId: " +
 accept_response["agreementId"])


if __name__ == "__main__":
    ReplaceSaaSContractWithAgreementBasedOffer.replace_existing_agreement()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing (CCP) ersetzt wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it
 with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer first starts a free trial on a SaaS product. Once the
 trial is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create a SaaS free trial agreement with freeTrialPricingTerm.</li>
 *   <li>Wait for freeTrialPricingTerm agreement entitlements to become active.</
li>
 *   <li>Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 free trial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the paid CCP offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSFreeTrialWithCCP {

    // The agreementProposalId from the free trial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-
trial-agreement-proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in the free trial offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the LegalTerm in the free trial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the paid CCP offer to convert to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-
agreement-proposal-identifier>";
```

```
    // Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the CCP offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-
schedule-term-id>";

    // Term ID for the UsageBasedPricingTerm in the CCP offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the CCP offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the CCP offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        createSaaSFreeTrialAndReplaceWithCCP();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
     * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
     * 3. Replace the free trial agreement with the paid CCP offer.
     */
    private static void createSaaSFreeTrialAndReplaceWithCCP() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm =
 RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm freeTrialPricingTerm =
 RequestedTerm.builder().id(FREE_TRIAL_PRICING_TERM_ID).build();

        // --- Step 1: Agreement with freeTrialPricingTerm ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
```

```
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, legalTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
       AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
       System.out.println("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId());

       // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
       System.out.println("Waiting for freeTrialPricingTerm agreement
entitlements to become active...");
       GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
               marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
       System.out.println("freeTrialPricingTerm agreement entitlements are now
active.");
       AgreementApiUtils.formatOutput(entitlementsResponse);

       // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
       // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
       RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
       RequestedTerm fixedUpfrontPricingTerm =
RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
       RequestedTerm paymentScheduleTerm =
RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();
```

```
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
 RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(usageBasedPricingTerm,
 fixedUpfrontPricingTerm, paymentScheduleTerm, validityTerm, newLegalTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created.
 AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Agreement with freeTrialPricingTerm replaced with
 paid CCP offer. New AgreementId: " + aarResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it
 with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * Scenario: A buyer first starts a free trial on a SaaS product. Once the trial
 is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * Flow:
 *   1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 *   2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 *   3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free
 trial offer.
```

```
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid
 CCP offer.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the free trial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>";

// Term ID for the FreeTrialPricingTerm in the free trial offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the LegalTerm in the free trial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the paid CCP offer to convert to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>";

// Term ID for the FixedUpfrontPricingTerm in the CCP offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the CCP offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

// Term ID for the UsageBasedPricingTerm in the CCP offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the CCP offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the CCP offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 * 3. Replace the free trial agreement with the paid CCP offer.
 */
async function createSaaSFreeTrialAndReplaceWithCCP() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
```

```
    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };

    // --- Step 1: Agreement with freeTrialPricingTerm ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId);

    // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
    console.log("Waiting for freeTrialPricingTerm agreement entitlements to
become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("freeTrialPricingTerm agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
    // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
agreement.
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
```

```
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [usageBasedPricingTerm, fixedUpfrontPricingTerm,
 paymentScheduleTerm, validityTerm, newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier:
 acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Agreement with freeTrialPricingTerm replaced with paid CCP
 offer. New AgreementId: " + aarResponse.agreementId);
}

createSaaSFreeTrialAndReplaceWithCCP();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS free trial agreement and then replace it with a
```

```
paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.

Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
they decide to convert by replacing it with a paid offer that includes a
FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.

Flow:
  1. Create a SaaS free trial agreement with freeTrialPricingTerm.
  2. Wait for freeTrialPricingTerm agreement entitlements to become active.
  3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceSaaSFreeTrialWithCCP:

    # The agreementProposalId from the free trial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>"

    # Term ID for the FreeTrialPricingTerm in the free trial offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the LegalTerm in the free trial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"
```

```
    # The agreementProposalId from the paid CCP offer to convert to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>"

    # Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the CCP offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    # Term ID for the UsageBasedPricingTerm in the CCP offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the CCP offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the CCP offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def create_saas_free_trial_and_replace_with_ccp():
        """
        Full end-to-end flow:
        1. Create a SaaS free trial agreement with freeTrialPricingTerm.
        2. Wait for freeTrialPricingTerm agreement entitlements to become active.
        3. Replace the free trial agreement with the paid CCP offer.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSFreeTrialWithCCP

        legal_term = {"id": cls.LEGAL_TERM_ID}
        free_trial_pricing_term = {"id": cls.FREE_TRIAL_PRICING_TERM_ID}

        # --- Step 1: Agreement with freeTrialPricingTerm ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[free_trial_pricing_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with freeTrialPricingTerm created.
 AgreementRequestId: " + agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with freeTrialPricingTerm accepted. AgreementId:
" + agreement_id)

        # Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        print("Waiting for freeTrialPricingTerm agreement entitlements to become
active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("freeTrialPricingTerm agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[
                usage_based_pricing_term, fixed_upfront_pricing_term,
                payment_schedule_term, validity_term, new_legal_term,
            ],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print(
```

```
            "Agreement with freeTrialPricingTerm replaced with paid CCP offer.
 New AgreementId: "
            + aar_response["agreementId"]
        )


if __name__ == "__main__":
    ReplaceSaaSFreeTrialWithCCP.create_saas_free_trial_and_replace_with_ccp()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Ersetzen Sie eine nutzungsabhängige SaaS-Preisvereinbarung und kündigen Sie die vorherige Vereinbarung mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie eine nutzungsabhängige SaaS-Preisvereinbarung ersetzt und die vorherige Vereinbarung gekündigt wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.CancelAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT)
 and then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing
 agreement.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with UBPT.</li>
 *   <li>Wait for entitlements to become active.</li>
 *   <li>Replace the agreement with a new offer using Intent.REPLACE.</li>
 *   <li>Cancel the new agreement using CancelAgreement.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSUsageBasedPricingTermAndCancel {

    // The agreementProposalId from the initial offer.
```

```
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-
usage-based-pricing-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        replaceSaaSUbptAndCancel();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
     * 2. Wait for entitlements to become active.
     * 3. Replace the agreement with a new offer.
     * 4. Cancel the new agreement.
     */
    private static void replaceSaaSUbptAndCancel() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
```

```
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();

        // --- Step 1: Create and accept the initial UBPT agreement request ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageBasedPricingTerm, validityTerm,
legalTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before replacing.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
```

```
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace the UBPT agreement with a new offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the
existing agreement.
        RequestedTerm newUsageBasedPricingTerm =
RequestedTerm.builder().id(NEW_USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageBasedPricingTerm,
newValidityTerm, newLegalTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

.sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String agreementIdWithTheNewOffer = aarResponse.agreementId();
        System.out.println("UBPT agreement replaced. New AgreementId: " +
agreementIdWithTheNewOffer);

        // --- Step 3: Cancel the new agreement ---
        CancelAgreementRequest cancelAgreementRequest =
                CancelAgreementRequest.builder()
                        .agreementId(agreementIdWithTheNewOffer)
                        .build();
        marketplaceAgreementClient.cancelAgreement(cancelAgreementRequest);
```

```
        System.out.println("The new agreement has been cancelled. AgreementId: "
 + agreementIdWithTheNewOffer);
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    CancelAgreementCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT)
 and then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing
 agreement.
 *
 * Flow:
 *   1. Create and accept the initial agreement request with UBPT.
 *   2. Wait for entitlements to become active.
```

```
 *   3. Replace the agreement with a new offer using Intent.REPLACE.
 *   4. Cancel the new agreement using CancelAgreement.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept the initial agreement request with UsageBasedPricingTerm.
 * 2. Wait for entitlements to become active.
 * 3. Replace the agreement with a new offer.
 * 4. Cancel the new agreement.
```

```
 */
async function replaceSaaSUbptAndCancel() {
    const client = new MarketplaceAgreementClient();

    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    // --- Step 1: Create and accept the initial UBPT agreement request ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageBasedPricingTerm, validityTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
 createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before replacing.
    console.log("Waiting for entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
 acceptAgreementRequestResponse.agreementId);
    console.log("Entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Replace the UBPT agreement with a new offer ---
    // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
 agreement.
    const newUsageBasedPricingTerm = { id: NEW_USAGE_BASED_PRICING_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

    const carResponse = await client.send(
```

```
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier:
 acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const agreementIdWithTheNewOffer = aarResponse.agreementId;
    console.log("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

    // --- Step 3: Cancel the new agreement ---
    await client.send(
        new CancelAgreementCommand({
            agreementId: agreementIdWithTheNewOffer,
        })
    );
    console.log("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
}

replaceSaaSUbptAndCancel();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
with a new offer using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
The buyer then converts to a different offer by replacing the existing agreement.

Flow:
  1. Create and accept the initial agreement request with UBPT.
  2. Wait for entitlements to become active.
  3. Replace the agreement with a new offer using Intent.REPLACE.
  4. Cancel the new agreement using CancelAgreement.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)
```

```
class ReplaceSaaSUsageBasedPricingTermAndCancel:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def replace_saas_ubpt_and_cancel():
        """
        Full end-to-end flow:
        1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
        2. Wait for entitlements to become active.
        3. Replace the agreement with a new offer.
        4. Cancel the new agreement.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSUsageBasedPricingTermAndCancel

        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
```

```
        legal_term = {"id": cls.LEGAL_TERM_ID}

        # --- Step 1: Create and accept the initial UBPT agreement request ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_based_pricing_term, validity_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
agreement_id)

        # Wait for entitlements to become active before replacing.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace the UBPT agreement with a new offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the
existing agreement.
        new_usage_based_pricing_term = {"id":
cls.NEW_USAGE_BASED_PRICING_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_based_pricing_term, new_validity_term,
new_legal_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
```

```
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        agreement_id_with_new_offer = aar_response["agreementId"]
        print("UBPT agreement replaced. New AgreementId: " +
 agreement_id_with_new_offer)

        # --- Step 3: Cancel the new agreement ---
        client.cancel_agreement(agreementId=agreement_id_with_new_offer)
        print("The new agreement has been cancelled. AgreementId: " +
 agreement_id_with_new_offer)


if __name__ == "__main__":
    ReplaceSaaSUsageBasedPricingTermAndCancel.replace_saas_ubpt_and_cancel()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Ersetzen Sie eine nutzungsbasierte AMI-Preisgestaltung, behalten Sie jedoch die bestehende ConfigurableUpfrontPricingTerm bei und verwenden Sie eine AWS SDK

Die folgenden Codebeispiele zeigen, wie eine nutzungsbasierte AMI-Preisgestaltung ersetzt und gleichzeitig die bestehende beibehalten wird. ConfigurableUpfrontPricingTerm

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsRespo
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active,
 using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> —
 accepted after the usageBasedPricingTerm agreement entitlements are active.</li>
 * </ol>
 * <p> This sample shows how to replace only the agreement with
 usageBasedPricingTerm with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * <p>Flow:
 * <ol>
```

```
 *   <li>Create and accept the initial agreement request with
 usageBasedPricingTerm.</li>
 *   <li>Create and accept the agreement request with
 configurableUpfrontPricingTerm (CUPT) (after usageBasedPricingTerm agreement
 entitlements are active).</li>
 *   <li>Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code
 P365D}).</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the CUPT term.</li>
 * </ul>
 */
public class ReplaceAmiUsageBasedPricingTermButNotCupt {

    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key defined in the CUPT term.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in the initial offer.
```

```
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    public static void main(String[] args) {
        replaceAmiUsageWhenCuptExists();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept an agreement request with usageBasedPricingTerm, then
 wait for entitlements.
     * 2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
     * 3. Replace the agreement with usageBasedPricingTerm with a new offer
 (agreement with CUPT is unaffected).
     */
    private static void replaceAmiUsageWhenCuptExists() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageTerm =
 RequestedTerm.builder().id(USAGE_TERM_ID).build();
```

```
        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Step 1: Agreement with UBPT ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageTerm, legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String usageAgreementId =
acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request with UBPT accepted. AgreementId: "
+ usageAgreementId);

        // Wait for UBPT agreement entitlements to become active before creating
the agreement with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, usageAgreementId);
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
```

```
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()

.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created.
AgreementRequestId: " + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: "
+ cuptAgreementId);

        // Wait for CUPT agreement entitlements to become active before replacing
usage.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
```

```
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new
offer ---
        RequestedTerm newUsageTerm =
RequestedTerm.builder().id(NEW_USAGE_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();

        // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT
affected by this replacement.
        CreateAgreementRequestRequest carReplaceRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageTerm, newLegalTerm,
newValidityTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(usageAgreementId)
                        .build();
        CreateAgreementRequestResponse carReplaceResponse =

marketplaceAgreementClient.createAgreementRequest(carReplaceRequest);
        System.out.println("Replace agreement request created.
AgreementRequestId: " + carReplaceResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarReplaceRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(carReplaceResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarReplaceResponse =

marketplaceAgreementClient.acceptAgreementRequest(aarReplaceRequest);
```

```
        System.out.println("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId());
    }
}
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active,
 using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after
 the usageBasedPricingTerm agreement entitlements are active.
 *
```

```
 * This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * Flow:
 *   1. Create and accept the initial agreement request with
 usageBasedPricingTerm.
 *   2. Create and accept the agreement request with
 configurableUpfrontPricingTerm (CUPT) (after usageBasedPricingTerm agreement
 entitlements are active).
 *   3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D").
 *   - DIMENSION_1_KEY — dimension key defined in the CUPT term.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key defined in the CUPT term.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_TERM_ID = "<your-usage-term-id>";
```

```
// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm, then
 wait for entitlements.
 * 2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
 * 3. Replace the agreement with usageBasedPricingTerm with a new offer
 (agreement with CUPT is unaffected).
 */
async function replaceAmiUsageWhenCuptExists() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Step 1: Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
```

```
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
        })
    );
    const usageAgreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request with UBPT accepted. AgreementId: " +
usageAgreementId);

    // Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
usageAgreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
```

```
    );
    console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const cuptAgreementId = aarResponse.agreementId;
    console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

    // Wait for CUPT agreement entitlements to become active before replacing
usage.
    console.log("Waiting for CUPT agreement entitlements to become active...");
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
---
    const newUsageTerm = { id: NEW_USAGE_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };

    // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
    // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
by this replacement.
    const carReplaceResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageTerm, newLegalTerm, newValidityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: usageAgreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
carReplaceResponse.agreementRequestId);

    const aarReplaceResponse = await client.send(
```

```
        new AcceptAgreementRequestCommand({
            agreementRequestId: carReplaceResponse.agreementRequestId,
        })
    );
    console.log("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId);
}

replaceAmiUsageWhenCuptExists();
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the
AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
     usageBasedPricingTerm agreement entitlements are active.
This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
```

```
Flow:
  1. Create and accept the initial agreement request with usageBasedPricingTerm.
  2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT)
     (after usageBasedPricingTerm agreement entitlements are active).
  3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D).
  - DIMENSION_1_KEY — dimension key defined in the CUPT term.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceAmiUsageBasedPricingTermButNotCupt:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key defined in the CUPT term.
    DIMENSION_1_KEY = "<your-dimension-key>"
```

```
    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_TERM_ID = "<your-new-usage-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    @staticmethod
    def replace_ami_usage_when_cupt_exists():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm,
 then wait for entitlements.
        2. Create and accept an agreement request with CUPT, then wait for
 entitlements.
        3. Replace the agreement with usageBasedPricingTerm with a new offer
           (agreement with CUPT is unaffected).
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceAmiUsageBasedPricingTermButNotCupt

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
```

```
        # --- Step 1: Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        usage_agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
usage_agreement_id)

        # Wait for UBPT agreement entitlements to become active before creating
the agreement with CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
usage_agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
```

```
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for CUPT agreement entitlements to become active before replacing
usage.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Step 3: Replace Agreement with usageBasedPricingTerm with a new
offer ---
        new_usage_term = {"id": cls.NEW_USAGE_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}

        # Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        # The agreement with configurableUpfrontPricingTerm (CUPT) is NOT
affected by this replacement.
        car_replace_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_term, new_legal_term, new_validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=usage_agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
car_replace_response["agreementRequestId"])

        aar_replace_response = client.accept_agreement_request(
```

```
            agreementRequestId=car_replace_response["agreementRequestId"]
        )
        print("Agreement with UBPT replaced. New AgreementId: " +
 aar_replace_response["agreementId"])


if __name__ == "__main__":

 ReplaceAmiUsageBasedPricingTermButNotCupt.replace_ami_usage_when_cupt_exists()
```

- Einzelheiten zur API finden Sie CreateAgreementRequestin AWS SDK for Python (Boto3) API Reference.

Suchen Sie nach Vereinbarungen anhand der Konto-ID mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Konto-ID suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement by customer AWS
 account ID
AG-02
"""

import argparse
import logging
```

```
import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements(account_id):
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AcceptorId", "values": [account_id]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement and agreement["nextToken"] is not None:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
```

```
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AcceptorId", "values": [account_id]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--account_id",
        "-aid",
        help="Provide accepting account ID to search for agreements",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreements(account_id=args.account_id)

    helper.pretty_print_datetime(response)
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie nach Vereinbarungen anhand der Vereinbarungs-ID mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Vereinbarungs-ID suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements give id
 information
AG-02-A
"""


import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "ResourceType"

# replace id value as needed
idValue = "SaaSProduct"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)
```

```
def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise e

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])
```

```
    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie nach Vereinbarungen anhand des Enddatums mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie nach Vereinbarungen anhand des Enddatums suchen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
 agreements where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
 are the acceptor.
```

```
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter =
Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 // search agreement with filters

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}
```

```
}
```

- Einzelheiten zur API finden Sie SearchAgreementsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement
 information before or after end date
AG-03
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

# change to 'AfterEndTime' if after endtime is desired
```

```
beforeOrAfterEndtimeFilterName = "BeforeEndTime"

# Make sure to use the same date format as below
cutoffDate = "2322-11-18T00:00:00Z"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements():
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you
 are the proposer.
            # Change to "Acceptor" to return agreements where you are the
 acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": beforeOrAfterEndtimeFilterName, "values": [cutoffDate]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {
                        "name": beforeOrAfterEndtimeFilterName,
                        "values": [cutoffDate],
```

```
                    },
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    agreements = get_agreements()
    helper.pretty_print_datetime(agreements)
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie nach Vereinbarungen anhand der Angebots-ID mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Angebots-ID suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to search for agreements by offer
 id
AG-0
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# offer id to search by
offerId = "11111111111111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "OfferId", "values": [offerId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise
```

```
        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "OfferId", "values": [offerId]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement by offer id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie nach Vereinbarungen anhand der Produkt-ID mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Produkt-ID suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement by
 product id
AG-02
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# product id to search by
resourceId = "prod-1111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)
```

```
def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the
 proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "ResourceIdentifier", "values": [resourceId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "ResourceIdentifier", "values": [resourceId]},
                        {"name": "AgreementType", "values":
 ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList
```

```
def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie mithilfe eines nach Status nach Vereinbarungen AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand des Status suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to filter agreements by status
```

```
AG-04

Example Usage: python3 search_agreements_by_status.py
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

# Set PartyType to "Proposer" to return agreements where you are the proposer.
# Change to "Acceptor" to return agreements where you are the acceptor.
party_type_list = ["Proposer"]
agreement_type_list = ["PurchaseAgreement"]

# Accepted values: "ACTIVE", "TERMINATED", "CANCELED", "EXPIRED", "REPLACED",
 "RENEWED"
status_list = ["ACTIVE"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
    {"name": "Status", "values": status_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
```

```
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    helper.pretty_print_datetime(agreement_results_list)
    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
```

- Einzelheiten zur API finden Sie SearchAgreementsin AWS SDK for Python (Boto3) API Reference.

Suchen Sie mit einem benutzerdefinierten Filter nach Vereinbarungen mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen mit einem benutzerdefinierten Filter suchen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";
```

```
private static final String FILTER_VALUE = "SaaSProduct";

/*
 * search agreements by one customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);
}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();
```

```
 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Einzelheiten zur API finden Sie SearchAgreementsin der AWS SDK for Java 2.x API-Referenz.

Suchen Sie mithilfe eines nach Vereinbarungen mit zwei benutzerdefinierten Filtern AWS SDK

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen mit zwei benutzerdefinierten Filtern suchen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByTwoFilters {
```

```
public static final String FILTER_1_NAME = "ResourceType";

public static final String FILTER_1_VALUE = "SaaSProduct";

public static final String FILTER_2_NAME = "Status";

public static final String FILTER_2_VALUE = "ACTIVE";

/*
 * search agreements by two customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);

}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return
agreements where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you
are the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter1 =
Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

 Filter customizeFilter2 =
Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();


 List<Filter> filters = new ArrayList<Filter>();
```

```
 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- Einzelheiten zur API finden Sie SearchAgreementsin der AWS SDK for Java 2.x API-Referenz.

Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Bestellungen aktualisiert werden, nachdem die Vereinbarung akzeptiert wurde.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Charge;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTer
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRespons
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.PurchaseOrder;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.UpdatePurchaseOrdersRequest;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing
 model and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference
 with a
 * specific charge via UpdatePurchaseOrders.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term
 list.</li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement.</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the offer.</li>
 *   <li>{@code PURCHASE_ORDER_REFERENCE} — your internal purchase order number
 (e.g., {@code po-123456}).</li>
 * </ul>
 */
public class UpdatePurchaseOrdersAfterAgreementAcceptance {

    // Your internal purchase order reference number (e.g., "po-123456").
    private static final String PURCHASE_ORDER_REFERENCE = "po-123456";

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
```

```
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P366D" for 366 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    public static void main(String[] args) {
        listAgreementChargesAndUpdatePurchaseOrders();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase
 order reference.
     * 2. List charges to retrieve charge IDs and revisions.
     * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
     * 4. List charges again to confirm the update.
     */
    private static void listAgreementChargesAndUpdatePurchaseOrders() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

 .configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguratio
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
```

```
.dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        RequestedTerm legalTerm =
RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Create Agreement ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm,
legalTerm, validityTerm)

.agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        // --- Accept Agreement Request with Purchase Order ---
        // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        String chargeId =
createAgreementRequestResponse.chargeSummary().expectedCharges().get(0).id();
        PurchaseOrder purchaseOrderAtAcceptance = PurchaseOrder.builder()
                .chargeId(chargeId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .build();
        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .purchaseOrders(purchaseOrderAtAcceptance)
                        .build();
```

```
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String agreementId = acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request accepted with purchase order
reference '" + PURCHASE_ORDER_REFERENCE
                + "'. AgreementId: " + agreementId);

        // --- List Agreement Charges ---
        ListAgreementChargesRequest listAgreementChargesRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse listAgreementChargesResponse =

marketplaceAgreementClient.listAgreementCharges(listAgreementChargesRequest);

        System.out.println("All charges for agreement " + agreementId + ":");
        AgreementApiUtils.formatOutput(listAgreementChargesResponse);

        // --- Update Purchase Order ---
        Charge firstCharge = listAgreementChargesResponse.items().get(0);
        PurchaseOrder purchaseOrder = PurchaseOrder.builder()
                .agreementId(agreementId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .chargeRevision(firstCharge.revision())
                .chargeId(firstCharge.id())
                .build();
        UpdatePurchaseOrdersRequest updatePurchaseOrdersRequest =
                UpdatePurchaseOrdersRequest.builder()
                        .purchaseOrders(purchaseOrder)
                        .build();

marketplaceAgreementClient.updatePurchaseOrders(updatePurchaseOrdersRequest);
        System.out.println("Purchase order reference '" +
PURCHASE_ORDER_REFERENCE
                + "' updated for ChargeId: " + firstCharge.id());

        // --- Verify Update ---
        ListAgreementChargesRequest lacRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse lacResponse =
```

```
                marketplaceAgreementClient.listAgreementCharges(lacRequest);
        System.out.println("Verified updated charge:");
        AgreementApiUtils.formatOutput(lacResponse.items().get(0));
    }
}
```

- Einzelheiten zur API finden Sie UpdatePurchaseOrdersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    ListAgreementChargesCommand,
    UpdatePurchaseOrdersCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput } = require("./utils/
AgreementApiUtils");

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
```

```
 * charges via ListAgreementCharges and associates the purchase order reference
 with a
 * specific charge via UpdatePurchaseOrders.
 *
 * Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement.
 *   - DIMENSION_1_KEY — dimension key defined in the offer.
 *   - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 "po-123456").
 */

// Your internal purchase order reference number (e.g., "po-123456").
const PURCHASE_ORDER_REFERENCE = "po-123456";

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P366D" for 366 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";
const DIMENSION_1_VALUE = 1;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
 reference.
 * 2. List charges to retrieve charge IDs and revisions.
```

```
 * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
 * 4. List charges again to confirm the update.
 */
async function listAgreementChargesAndUpdatePurchaseOrders() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Create Agreement ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
 validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    // --- Accept Agreement Request with Purchase Order ---
    // The chargeId is available from the CAR response's
 chargeSummary.expectedCharges.
    const chargeId =
 createAgreementRequestResponse.chargeSummary.expectedCharges[0].id;
    const purchaseOrderAtAcceptance = {
        chargeId: chargeId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
```

```
    };
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId:
createAgreementRequestResponse.agreementRequestId,
            purchaseOrders: [purchaseOrderAtAcceptance],
        })
    );
    const agreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request accepted with purchase order reference '" +
PURCHASE_ORDER_REFERENCE
        + "'. AgreementId: " + agreementId);

    // --- List Agreement Charges ---
    const listAgreementChargesResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );

    console.log("All charges for agreement " + agreementId + ":");
    formatOutput(listAgreementChargesResponse);

    // --- Update Purchase Order ---
    const firstCharge = listAgreementChargesResponse.items[0];
    const purchaseOrder = {
        agreementId: agreementId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        chargeRevision: firstCharge.revision,
        chargeId: firstCharge.id,
    };
    await client.send(
        new UpdatePurchaseOrdersCommand({
            purchaseOrders: [purchaseOrder],
        })
    );
    console.log("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
        + "' updated for ChargeId: " + firstCharge.id);

    // --- Verify Update ---
    const lacResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
```

```
    );
    console.log("Verified updated charge:");
    formatOutput(lacResponse.items[0]);
}

listAgreementChargesAndUpdatePurchaseOrders();
```

- Einzelheiten zur API finden Sie UpdatePurchaseOrdersin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT
pricing model using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides
a purchase order reference in AcceptAgreementRequest. After acceptance, the buyer
 lists the
resulting charges via ListAgreementCharges and associates the purchase order
 reference with a
specific charge via UpdatePurchaseOrders.

Before running this sample, replace the placeholder constants below with values
 from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement.
```

```
  - DIMENSION_1_KEY — dimension key defined in the offer.
  - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 po-123456).
"""

import boto3

from utils.agreement_api_utils import format_output, generate_client_token


class UpdatePurchaseOrdersAfterAgreementAcceptance:

    # Your internal purchase order reference number (e.g., "po-123456").
    PURCHASE_ORDER_REFERENCE = "po-123456"

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>"

    # Duration for the agreement (e.g., "P366D" for 366 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"
    DIMENSION_1_VALUE = 1

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    @staticmethod
    def list_agreement_charges_and_update_purchase_orders():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with a purchase
 order reference.
        2. List charges to retrieve charge IDs and revisions.
        3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
```

```
        4. List charges again to confirm the update.
        """
        client = boto3.client("marketplace-agreement")
        cls = UpdatePurchaseOrdersAfterAgreementAcceptance

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Create Agreement ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        # --- Accept Agreement Request with Purchase Order ---
        # The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        charge_id = create_response["chargeSummary"]["expectedCharges"][0]["id"]
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id,
            purchaseOrders=[
                {
                    "chargeId": charge_id,
```

```
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                }
            ],
        )
        agreement_id = accept_response["agreementId"]
        print(
            "Agreement request accepted with purchase order reference '"
            + cls.PURCHASE_ORDER_REFERENCE + "'. AgreementId: " + agreement_id
        )

        # --- List Agreement Charges ---
        list_charges_response =
 client.list_agreement_charges(agreementId=agreement_id)
        print("All charges for agreement " + agreement_id + ":")
        format_output(list_charges_response)

        # --- Update Purchase Order ---
        first_charge = list_charges_response["items"][0]
        client.update_purchase_orders(
            purchaseOrders=[
                {
                    "agreementId": agreement_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                    "chargeRevision": first_charge["revision"],
                    "chargeId": first_charge["id"],
                }
            ],
        )
        print(
            "Purchase order reference '" + cls.PURCHASE_ORDER_REFERENCE
            + "' updated for ChargeId: " + first_charge["id"]
        )

        # --- Verify Update ---
        lac_response = client.list_agreement_charges(agreementId=agreement_id)
        print("Verified updated charge:")
        format_output(lac_response["items"][0])


if __name__ == "__main__":

 UpdatePurchaseOrdersAfterAgreementAcceptance.list_agreement_charges_and_update_purchase_
```

- Einzelheiten zur API finden Sie UpdatePurchaseOrdersin AWS SDK for Python (Boto3) API Reference.

Codebeispiele für die Verwendung MediaConnect AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elemental MediaConnect mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- MediaConnect Entwicklerhandbuch — Weitere Informationen zu MediaConnect.
- MediaConnect API-Referenz — Details zu allen verfügbaren MediaConnect Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung MediaConnect AWS SDKs
  - Erste Schritte mit Videotransport-Streaming

Szenarien für die Verwendung MediaConnect AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien MediaConnect mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb MediaConnect oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Videotransport-Streaming

Erste Schritte mit Videotransport-Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Zugriff auf Elemental MediaConnect
- Erstellen Sie einen Flow
- Fügen Sie eine Ausgabe hinzu
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
```

```
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"'
 -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-
zA-Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
```

```
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output"
 == "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
```

```
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
```

```
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/
null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1
 | cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi
```

```
# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/
dev/null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
```

```
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o
 '"EntitlementArn": "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN"
 --query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Codebeispiele für die MediaConvert Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elemental MediaConvert mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- MediaConvert Benutzerhandbuch — Weitere Informationen zu MediaConvert.
- MediaConvert API-Referenz — Details zu allen verfügbaren MediaConvert Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung MediaConvert AWS SDKs
  - Hallo MediaConvert

- Aktionen zur MediaConvert Verwendung AWS SDKs
  - Verwenden Sie CreateJobmit einem AWS SDK oder CLI
  - Verwenden Sie GetJobmit einem AWS SDK oder CLI
  - Verwenden Sie ListJobsmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung MediaConvert AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Elemental MediaConvert mit AWS SDKs verwendet werden.

Beispiele

- Hallo MediaConvert
- Aktionen zur MediaConvert Verwendung AWS SDKs
  - Verwenden Sie CreateJobmit einem AWS SDK oder CLI
  - Verwenden Sie GetJobmit einem AWS SDK oder CLI
  - Verwenden Sie ListJobsmit einem AWS SDK oder CLI

Hallo MediaConvert

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Elemental MediaConvert beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.MediaConvert;
using Amazon.MediaConvert.Model;

namespace MediaConvertActions;
```

```
public static class HelloMediaConvert
{
    static async Task Main(string[] args)
    {
        // Create the client using the default profile.
        var mediaConvertClient = new AmazonMediaConvertClient();

        Console.WriteLine($"Hello AWS Elemental MediaConvert! Your MediaConvert
 Jobs are:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get some MediaConvert jobs.
        var response = await mediaConvertClient.ListJobsAsync(
            new ListJobsRequest()
            {
                MaxResults = 10
            }
            );

        foreach (var job in response.Jobs)
        {
            Console.WriteLine($"\tJob: {job.Id} status {job.Status}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeEndpointsin der AWS SDK für .NET API-Referenz.

Aktionen zur MediaConvert Verwendung AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne MediaConvert Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Elemental MediaConvert -API-Referenz.

Beispiele

- Verwenden Sie CreateJobmit einem AWS SDK oder CLI
- Verwenden Sie GetJobmit einem AWS SDK oder CLI
- Verwenden Sie ListJobsmit einem AWS SDK oder CLI

Verwenden Sie **`CreateJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateJob` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Creating job for input file {fileInput}.");
        var jobId = await wrapper.CreateJob(mediaConvertRole!, fileInput!,
 fileOutput!);
```

```
        Console.WriteLine($"Created job with Job ID: {jobId}");
        Console.WriteLine(new string('-', 80));
```

Erstellen Sie den Auftrag mit der Wrapper-Methode und geben Sie die Auftrags-ID zurück.

```
    /// <summary>
    /// Create a job to convert a media file.
    /// </summary>
    /// <param name="mediaConvertRole">The Amazon Resource Name (ARN) of the
 media convert role, as specified here:
    /// https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html</param>
    /// <param name="fileInput">The Amazon Simple Storage Service (Amazon S3)
 location of the input media file.</param>
    /// <param name="fileOutput">The Amazon S3 location for the output media
 file.</param>
    /// <returns>The ID of the new job.</returns>
    public async Task<string> CreateJob(string mediaConvertRole, string
 fileInput,
        string fileOutput)
    {
        CreateJobRequest createJobRequest = new CreateJobRequest
        {
            Role = mediaConvertRole
        };

        createJobRequest.UserMetadata.Add("Customer", "Amazon");

        JobSettings jobSettings = new JobSettings
        {
            AdAvailOffset = 0,
            TimecodeConfig = new TimecodeConfig
            {
                Source = TimecodeSource.EMBEDDED
            }
        };
        createJobRequest.Settings = jobSettings;

        #region OutputGroup

        OutputGroup ofg = new OutputGroup
```

```
        {
            Name = "File Group",
            OutputGroupSettings = new OutputGroupSettings
            {
                Type = OutputGroupType.FILE_GROUP_SETTINGS,
                FileGroupSettings = new FileGroupSettings
                {
                    Destination = fileOutput
                }
            }
        };

        Output output = new Output
        {
            NameModifier = "_1"
        };

        #region VideoDescription

        VideoDescription vdes = new VideoDescription
        {
            ScalingBehavior = ScalingBehavior.DEFAULT,
            TimecodeInsertion = VideoTimecodeInsertion.DISABLED,
            AntiAlias = AntiAlias.ENABLED,
            Sharpness = 50,
            AfdSignaling = AfdSignaling.NONE,
            DropFrameTimecode = DropFrameTimecode.ENABLED,
            RespondToAfd = RespondToAfd.NONE,
            ColorMetadata = ColorMetadata.INSERT,
            CodecSettings = new VideoCodecSettings
            {
                Codec = VideoCodec.H_264
            }
        };
        output.VideoDescription = vdes;

        H264Settings h264 = new H264Settings
        {
            InterlaceMode = H264InterlaceMode.PROGRESSIVE,
            NumberReferenceFrames = 3,
            Syntax = H264Syntax.DEFAULT,
            Softness = 0,
            GopClosedCadence = 1,
            GopSize = 90,
```

```
            Slices = 1,
            GopBReference = H264GopBReference.DISABLED,
            SlowPal = H264SlowPal.DISABLED,
            SpatialAdaptiveQuantization =
H264SpatialAdaptiveQuantization.ENABLED,
            TemporalAdaptiveQuantization =
H264TemporalAdaptiveQuantization.ENABLED,
            FlickerAdaptiveQuantization =
H264FlickerAdaptiveQuantization.DISABLED,
            EntropyEncoding = H264EntropyEncoding.CABAC,
            Bitrate = 5000000,
            FramerateControl = H264FramerateControl.SPECIFIED,
            RateControlMode = H264RateControlMode.CBR,
            CodecProfile = H264CodecProfile.MAIN,
            Telecine = H264Telecine.NONE,
            MinIInterval = 0,
            AdaptiveQuantization = H264AdaptiveQuantization.HIGH,
            CodecLevel = H264CodecLevel.AUTO,
            FieldEncoding = H264FieldEncoding.PAFF,
            SceneChangeDetect = H264SceneChangeDetect.ENABLED,
            QualityTuningLevel = H264QualityTuningLevel.SINGLE_PASS,
            FramerateConversionAlgorithm =
                H264FramerateConversionAlgorithm.DUPLICATE_DROP,
            UnregisteredSeiTimecode = H264UnregisteredSeiTimecode.DISABLED,
            GopSizeUnits = H264GopSizeUnits.FRAMES,
            ParControl = H264ParControl.SPECIFIED,
            NumberBFramesBetweenReferenceFrames = 2,
            RepeatPps = H264RepeatPps.DISABLED,
            FramerateNumerator = 30,
            FramerateDenominator = 1,
            ParNumerator = 1,
            ParDenominator = 1
        };
        output.VideoDescription.CodecSettings.H264Settings = h264;

        #endregion VideoDescription

        #region AudioDescription

        AudioDescription ades = new AudioDescription
        {
            LanguageCodeControl = AudioLanguageCodeControl.FOLLOW_INPUT,
            // This name matches one specified in the following Inputs.
            AudioSourceName = "Audio Selector 1",
```

```
            CodecSettings = new AudioCodecSettings
            {
                Codec = AudioCodec.AAC
            }
        };

        AacSettings aac = new AacSettings
        {
            AudioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.NORMAL,
            RateControlMode = AacRateControlMode.CBR,
            CodecProfile = AacCodecProfile.LC,
            CodingMode = AacCodingMode.CODING_MODE_2_0,
            RawFormat = AacRawFormat.NONE,
            SampleRate = 48000,
            Specification = AacSpecification.MPEG4,
            Bitrate = 64000
        };
        ades.CodecSettings.AacSettings = aac;
        output.AudioDescriptions.Add(ades);

        #endregion AudioDescription

        #region Mp4 Container

        output.ContainerSettings = new ContainerSettings
        {
            Container = ContainerType.MP4
        };
        Mp4Settings mp4 = new Mp4Settings
        {
            CslgAtom = Mp4CslgAtom.INCLUDE,
            FreeSpaceBox = Mp4FreeSpaceBox.EXCLUDE,
            MoovPlacement = Mp4MoovPlacement.PROGRESSIVE_DOWNLOAD
        };
        output.ContainerSettings.Mp4Settings = mp4;

        #endregion Mp4 Container

        ofg.Outputs.Add(output);
        createJobRequest.Settings.OutputGroups.Add(ofg);

        #endregion OutputGroup
```

```
        #region Input

        Input input = new Input
        {
            FilterEnable = InputFilterEnable.AUTO,
            PsiControl = InputPsiControl.USE_PSI,
            FilterStrength = 0,
            DeblockFilter = InputDeblockFilter.DISABLED,
            DenoiseFilter = InputDenoiseFilter.DISABLED,
            TimecodeSource = InputTimecodeSource.EMBEDDED,
            FileInput = fileInput
        };

        AudioSelector audsel = new AudioSelector
        {
            Offset = 0,
            DefaultSelection = AudioDefaultSelection.NOT_DEFAULT,
            ProgramSelection = 1,
            SelectorType = AudioSelectorType.TRACK
        };
        audsel.Tracks.Add(1);
        input.AudioSelectors.Add("Audio Selector 1", audsel);

        input.VideoSelector = new VideoSelector
        {
            ColorSpace = ColorSpace.FOLLOW
        };

        createJobRequest.Settings.Inputs.Add(input);

        #endregion Input

        CreateJobResponse createJobResponse =
            await _amazonMediaConvert.CreateJobAsync(createJobRequest);

        var jobId = createJobResponse.Job.Id;

        return jobId;
    }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS Elemental MediaConvert job.
/*!
  \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                           Access Management (IAM) role for the job.
  \param fileInput: A URI to an input file that is stored in Amazon Simple
 Storage Service
                    (Amazon S3) or on an HTTP(S) server.
  \param fileOutput: A URI for an Amazon S3 output location and the output file
 name base.
  \param jobSettingsFile: An optional JSON settings file.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
                                     const Aws::String &jobSettingsFile,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

    createJobRequest.SetRole(mediaConvertRole);
    Aws::Http::HeaderValueCollection hvc;
    hvc.emplace("Customer", "Amazon");
    createJobRequest.SetUserMetadata(hvc);

    if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
    {
        std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
        if (!jobSettingsStream) {
            std::cerr << "Unable to open the job template file." << std::endl;
```

```
            return false;
        }
        std::vector<char> buffer(jobSettingsStream.tellg());
        jobSettingsStream.seekg(0);
        jobSettingsStream.read(buffer.data(), buffer.size());
        std::string jobSettingsJSON(buffer.data(), buffer.size());
        size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER),
fileInput);
        }

        pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
        }
        Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
        Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);

        createJobRequest.SetSettings(jobSettings);
    }
    else { // Configure the job settings programmatically.
        Aws::MediaConvert::Model::JobSettings jobSettings;
        jobSettings.SetAdAvailOffset(0);
        Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
        jobSettings.SetTimecodeConfig(timecodeConfig);

        // Configure the output group.
        Aws::MediaConvert::Model::OutputGroup outputGroup;
        outputGroup.SetName("File Group");
        Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
        outputGroupSettings.SetType(
                Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
        Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
        fileGroupSettings.SetDestination(fileOutput);
        outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
        outputGroup.SetOutputGroupSettings(outputGroupSettings);

        Aws::MediaConvert::Model::Output output;
        output.SetNameModifier("_1");
```

```
        Aws::MediaConvert::Model::VideoDescription videoDescription;
        videoDescription.SetScalingBehavior(
                Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);
        videoDescription.SetTimecodeInsertion(
                Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);

videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
        videoDescription.SetSharpness(50);

videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
        videoDescription.SetDropFrameTimecode(
                Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
        videoDescription.SetColorMetadata(
                Aws::MediaConvert::Model::ColorMetadata::INSERT);

        Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
        videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
        Aws::MediaConvert::Model::H264Settings h264Settings;
        h264Settings.SetNumberReferenceFrames(3);
        h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
        h264Settings.SetSoftness(0);
        h264Settings.SetGopClosedCadence(1);
        h264Settings.SetGopSize(90);
        h264Settings.SetSlices(1);
        h264Settings.SetGopBReference(
                Aws::MediaConvert::Model::H264GopBReference::DISABLED);
        h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
        h264Settings.SetSpatialAdaptiveQuantization(

Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
        h264Settings.SetTemporalAdaptiveQuantization(

Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
        h264Settings.SetFlickerAdaptiveQuantization(

Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
        h264Settings.SetEntropyEncoding(
                Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
        h264Settings.SetBitrate(5000000);
        h264Settings.SetFramerateControl(
                Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
        h264Settings.SetRateControlMode(
```

```
                Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
        h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
        h264Settings.SetMinIInterval(0);
        h264Settings.SetAdaptiveQuantization(
                Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);

h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
        h264Settings.SetFieldEncoding(
                Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
        h264Settings.SetSceneChangeDetect(
                Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
        h264Settings.SetQualityTuningLevel(
                Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
        h264Settings.SetFramerateConversionAlgorithm(

Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
        h264Settings.SetUnregisteredSeiTimecode(
                Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
        h264Settings.SetGopSizeUnits(
                Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
        h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
        h264Settings.SetFramerateNumerator(30);
        h264Settings.SetFramerateDenominator(1);
        h264Settings.SetParNumerator(1);
        h264Settings.SetParDenominator(1);
        videoCodecSettings.SetH264Settings(h264Settings);
        videoDescription.SetCodecSettings(videoCodecSettings);
        output.SetVideoDescription(videoDescription);

        Aws::MediaConvert::Model::AudioDescription audioDescription;
        audioDescription.SetLanguageCodeControl(

Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
        audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
        Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
        audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
        Aws::MediaConvert::Model::AacSettings aacSettings;
        aacSettings.SetAudioDescriptionBroadcasterMix(
```

```
Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
       aacSettings.SetRateControlMode(
               Aws::MediaConvert::Model::AacRateControlMode::CBR);

aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
       aacSettings.SetCodingMode(
               Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
       aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
       aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
       aacSettings.SetBitrate(64000);
       audioCodecSettings.SetAacSettings(aacSettings);
       audioDescription.SetCodecSettings(audioCodecSettings);
       Aws::Vector<Aws::MediaConvert::Model::AudioDescription>
audioDescriptions;
       audioDescriptions.emplace_back(audioDescription);
       output.SetAudioDescriptions(audioDescriptions);

       Aws::MediaConvert::Model::ContainerSettings mp4container;
       mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
       Aws::MediaConvert::Model::Mp4Settings mp4Settings;
       mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
       mp4Settings.SetMoovPlacement(

Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
       mp4container.SetMp4Settings(mp4Settings);
       output.SetContainerSettings(mp4container);

       outputGroup.AddOutputs(output);
       jobSettings.AddOutputGroups(outputGroup);

       // Configure inputs.
       Aws::MediaConvert::Model::Input input;
       input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
       input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
       input.SetFilterStrength(0);

input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
```

```
        input.SetTimecodeSource(
                Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
        input.SetFileInput(fileInput);

        Aws::MediaConvert::Model::AudioSelector audioSelector;
        audioSelector.SetOffset(0);
        audioSelector.SetDefaultSelection(
                Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
        audioSelector.SetProgramSelection(1);
        audioSelector.SetSelectorType(
                Aws::MediaConvert::Model::AudioSelectorType::TRACK);
        audioSelector.AddTracks(1);
        input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

        Aws::MediaConvert::Model::VideoSelector videoSelector;

 videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
        input.SetVideoSelector(videoSelector);

        jobSettings.AddInputs(input);

        createJobRequest.SetSettings(jobSettings);
    }

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
    Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(
            createJobRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Job successfully created with ID - "
                  << outcome.GetResult().GetJob().GetId() << std::endl;
    }
    else {
        std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um einen Job zu erstellen

Im folgenden Beispiel für `create-job` wird ein Transkodierungsauftrag mit den Einstellungen gemäß einer `job.json`-Datei erstellt, die sich auf dem System befindet, von dem der Befehl ausging. In dieser JSON-Auftragsspezifikation kann jede Einstellung einzeln angegeben werden, auf eine Auftragsvorlage oder auf Ausgabevorgaben verweisen.

```
aws mediaconvert create-job \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1 \
    --cli-input-json file://~/job.json
```

Sie können die AWS MediaConvert Elemental-Konsole verwenden, um die JSON-Jobspezifikation zu generieren, indem Sie Ihre Job-Einstellungen auswählen und dann unten im Job-Bereich die Option Job-JSON anzeigen auswählen.

Verwenden Sie `describe-endpoints` zum Abrufen des kontospezifischen Endpunkts oder senden Sie den Befehl ohne den Endpunkt. Der Service gibt einen Fehler und den Endpunkt zurück.

Wenn die Anfrage erfolgreich ist, gibt der Service die JSON-Auftragsspezifikation zurück, die Sie mit der Anfrage gesendet haben.

Weitere Informationen finden Sie unter Arbeiten mit AWS Elemental MediaConvert Jobs im AWS Elemental User Guide MediaConvert .

- Einzelheiten zur API finden Sie CreateJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import
 software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
```

```
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import
 software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                    Usage:
```

```
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
where the input file is located.\s
                                """;

               if (args.length != 2) {
                       System.out.println(usage);
                       System.exit(1);
               }

               String mcRoleARN = args[0];
               String fileInput = args[1];
               Region region = Region.US_WEST_2;
               MediaConvertClient mc = MediaConvertClient.builder()
                               .region(region)
                               .build();

               String id = createMediaJob(mc, mcRoleARN, fileInput);
               System.out.println("MediaConvert job created. Job Id = " + id);
               mc.close();
       }

       public static String createMediaJob(MediaConvertClient mc, String
mcRoleARN, String fileInput) {

               String s3path = fileInput.substring(0, fileInput.lastIndexOf('/')
+ 1) + "javasdk/out/";
               String fileOutput = s3path + "index";
               String thumbsOutput = s3path + "thumbs/";
               String mp4Output = s3path + "mp4/";

               try {
                       System.out.println("MediaConvert role arn: " +
mcRoleARN);
                       System.out.println("MediaConvert input file: " +
fileInput);
                       System.out.println("MediaConvert output path: " +
s3path);

                       // output group Preset HLS low profile
```

```
                        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                        // output group Preset HLS media profile
                        Output hlsMedium = createOutput("hls_medium", "_medium",
 "_$dt$", 1200000, 7, 1920, 1080, 1280);
                        // output group Preset HLS high profole
                        Output hlsHigh = createOutput("hls_high", "_high", "_$dt
$", 3500000, 8, 1920, 1080, 1920);

                        OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

       HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

       HlsManifestDurationFormat.INTEGER)

 .streamInfResolution(

       HlsStreamInfResolution.INCLUDE)

 .clientCache(HlsClientCache.ENABLED)

 .captionLanguageSetting(

       HlsCaptionLanguageSetting.OMIT)

 .manifestCompression(

       HlsManifestCompression.NONE)

 .codecSpecification(

       HlsCodecSpecification.RFC_4281)
```

```
.outputSelection(

      HlsOutputSelection.MANIFESTS_AND_SEGMENTS)

.programDateTime(HlsProgramDateTime.EXCLUDE)

.programDateTimePeriod(600)

.timedMetadataId3Frame(

      HlsTimedMetadataId3Frame.PRIV)

.timedMetadataId3Period(10)

.destination(fileOutput)

.segmentControl(HlsSegmentControl.SEGMENTED_FILES)

.minFinalSegmentLength((double) 0)

.segmentLength(4).minSegmentLength(0).build())
                                                        .build())
                                          .outputs(hlsLow, hlsMedium,
hlsHigh).build();

                       OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                        .build())

.outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())
```

```
.videoDescription(VideoDescription.builder().width(1280)
.height(720)
.scalingBehavior(ScalingBehavior.DEFAULT)
.sharpness(50).antiAlias(AntiAlias.ENABLED)
.timecodeInsertion(
      VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
      .codec(VideoCodec.H_264)
      .h264Settings(H264Settings
                      .builder()
                      .rateControlMode(
                                      H264RateControlMode.QVBR)
                      .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                      .qualityTuningLevel(
                                      H264QualityTuningLevel.SINGLE_PASS)
                      .qvbrSettings(
                                      H264QvbrSettings.builder()
                                                      .qvbrQualityLevel(
```

```
                                                                    8)
                                                        .build())
                        .codecLevel(H264CodecLevel.AUTO)
                        .codecProfile(H264CodecProfile.MAIN)
                        .maxBitrate(2400000)
                        .framerateControl(

H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                        2)
                        .gopClosedCadence(
                                        1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                        3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
```

```
                                        H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
                        .telecine(H264Telecine.NONE)
                        .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                        .entropyEncoding(
                                        H264EntropyEncoding.CABAC)
                        .slices(1)
                        .unregisteredSeiTimecode(
                                        H264UnregisteredSeiTimecode.DISABLED)
                        .repeatPps(H264RepeatPps.DISABLED)
                        .adaptiveQuantization(
                                        H264AdaptiveQuantization.HIGH)
                        .spatialAdaptiveQuantization(
                                        H264SpatialAdaptiveQuantization.ENABLED)
                        .temporalAdaptiveQuantization(
                                        H264TemporalAdaptiveQuantization.ENABLED)
                        .flickerAdaptiveQuantization(
                                        H264FlickerAdaptiveQuantization.DISABLED)
                        .softness(0)
                        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
```

```
                        .build())

      .build())
                                                                                .build())

.audioDescriptions(AudioDescription.builder()

.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

.languageCodeControl(

      AudioLanguageCodeControl.FOLLOW_INPUT)

.codecSettings(AudioCodecSettings.builder()

      .codec(AudioCodec.AAC)

      .aacSettings(AacSettings

                        .builder()

                        .codecProfile(AacCodecProfile.LC)

                        .rateControlMode(

                                        AacRateControlMode.CBR)

                        .codingMode(AacCodingMode.CODING_MODE_2_0)

                        .sampleRate(44100)

                        .bitrate(160000)

                        .rawFormat(AacRawFormat.NONE)

                        .specification(AacSpecification.MPEG4)

                        .audioDescriptionBroadcasterMix(

                                        AacAudioDescriptionBroadcasterMix.NORMAL)

                        .build())
```

```
      .build())
                                                                                .build())
                                                                .build())
                                                .build();
                        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                                .build())

.outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

      VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

      .codec(VideoCodec.FRAME_CAPTURE)

      .frameCaptureSettings(

                        FrameCaptureSettings
```

```
                                        .builder()

                                        .framerateNumerator(

                                                        1)

                                        .framerateDenominator(

                                                        1)

                                        .maxCaptures(10000000)

                                        .quality(80)

                                        .build())

      .build())
                                                                                .build())
                                                        .build())
                                         .build();

                       Map<String, AudioSelector> audioSelectors = new
HashMap<>();
                       audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                        .offset(0).build());

                       JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                         .audioSelectors(audioSelectors)
                                         .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)

.deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)
```

```
.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                        .outputGroups(appleHLS, thumbs,
fileMp4).build();

                        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
mc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
```

```
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

      489, 490, 491, 492)
                                                                                .build())
                                                                  .build())
                                        .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

      VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)
```

```
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
      .codec(VideoCodec.H_264)
      .h264Settings(H264Settings
                      .builder()
                      .rateControlMode(
                                      H264RateControlMode.QVBR)
                      .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                      .qualityTuningLevel(
                                      H264QualityTuningLevel.SINGLE_PASS)
                      .qvbrSettings(H264QvbrSettings
                                      .builder()
                                      .qvbrQualityLevel(
                                                      qvbrQualityLevel)
                                      .build())
                      .codecLevel(H264CodecLevel.AUTO)
                      .codecProfile((targetHeight > 720
                                      && targetWidth > 1280)
                                                      ? H264CodecProfile.HIGH
                                                      : H264CodecProfile.MAIN)
                      .maxBitrate(qvbrMaxBitrate)
                      .framerateControl(
```

```
H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                        2)
                        .gopClosedCadence(
                                        1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                        3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
                                        H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
                        .telecine(H264Telecine.NONE)
                        .framerateConversionAlgorithm(

H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                        .entropyEncoding(
```

```
                                        H264EntropyEncoding.CABAC)
                        .slices(1)
                        .unregisteredSeiTimecode(
                                        H264UnregisteredSeiTimecode.DISABLED)
                        .repeatPps(H264RepeatPps.DISABLED)
                        .adaptiveQuantization(
                                        H264AdaptiveQuantization.HIGH)
                        .spatialAdaptiveQuantization(
                                        H264SpatialAdaptiveQuantization.ENABLED)
                        .temporalAdaptiveQuantization(
                                        H264TemporalAdaptiveQuantization.ENABLED)
                        .flickerAdaptiveQuantization(
                                        H264FlickerAdaptiveQuantization.DISABLED)
                        .softness(0)
                        .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                        .build())
        .build())
                                                                        .build())
.audioDescriptions(AudioDescription.builder()
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
.languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
```

```
 .codec(AudioCodec.AAC).aacSettings(AacSettings

       .builder()

       .codecProfile(AacCodecProfile.LC)

       .rateControlMode(

                       AacRateControlMode.CBR)

       .codingMode(AacCodingMode.CODING_MODE_2_0)

       .sampleRate(44100)

       .bitrate(96000)

       .rawFormat(AacRawFormat.NONE)

       .specification(AacSpecification.MPEG4)

       .audioDescriptionBroadcasterMix(

                       AacAudioDescriptionBroadcasterMix.NORMAL)

       .build())
                                                                    .build())
                                                        .build())
                                     .build();
               } catch (MediaConvertException e) {
                       e.printStackTrace();
                       System.exit(0);
               }
               return output;
        }
}
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
```

```
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode =
H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                    }
                                }
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)
```

```
    // Return the job ID or null if not found
    return response.job?.id
}
```

- API-Details finden Sie CreateJobin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **GetJob**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetJob` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Rufen Sie einen Job anhand seiner ID ab.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting job information for Job ID {jobId}");
        var job = await wrapper.GetJobById(jobId);
        Console.WriteLine($"Job {job.Id} created on {job.CreatedAt:d} has status
 {job.Status}.");
        Console.WriteLine(new string('-', 80));
```

```
    /// <summary>
    /// Get the job information for a job by its ID.
    /// </summary>
    /// <param name="jobId">The ID of the job.</param>
    /// <returns>The Job object.</returns>
    public async Task<Job> GetJobById(string jobId)
    {
        var jobResponse = await _amazonMediaConvert.GetJobAsync(
                new GetJobRequest
                {
                    Id = jobId
                });

        return jobResponse.Job;
    }
```

- Einzelheiten zur API finden Sie GetJobin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the information for a specific completed transcoding job.
/*!
  \param jobID: A job ID.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
    const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
            request);
    if (outcome.IsSuccess()) {
        std::cout <<
 outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
                  << std::endl;
    }
    else {
        std::cerr << "DescribeEndpoints error - " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }


    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetJobin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Details für einen bestimmten Auftrag ab

Im folgenden Beispiel werden die Informationen für den Auftrag mit der ID `1234567890987-1ab2c3` abgerufen, was in diesem Beispiel mit einem Fehler endete.

```
aws mediaconvert get-job \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1 \
```

```
    --id 1234567890987-1ab2c3
```

Verwenden Sie `describe-endpoints` zum Abrufen des kontospezifischen Endpunkts oder senden Sie den Befehl ohne den Endpunkt. Der Service gibt einen Fehler und den Endpunkt zurück.

Wenn die Anfrage erfolgreich ist, gibt der Service, wie im Folgenden angezeigt, eine JSON-Datei mit Auftragsinformationen, einschließlich Auftragseinstellungen, allen zurückgegebenen Fehlern und anderen Auftragsdaten, zurück:

```
{
    "Job": {
        "Status": "ERROR",
        "Queue": "arn:aws:mediaconvert:region-name-1:012345678998:queues/Queue1",
        "Settings": {
            ...<truncated for brevity>...
        },
        "ErrorMessage": "Unable to open input file [s3://my-input-bucket/file-
name.mp4]: [Failed probe/open: [Failed to read data: AssumeRole failed]]",
        "ErrorCode": 1434,
        "Role": "arn:aws:iam::012345678998:role/MediaConvertServiceRole",
        "Arn": "arn:aws:mediaconvert:us-
west-1:012345678998:jobs/1234567890987-1ab2c3",
        "UserMetadata": {},
        "Timing": {
            "FinishTime": 1517442131,
            "SubmitTime": 1517442103,
            "StartTime": 1517442104
        },
        "Id": "1234567890987-1ab2c3",
        "CreatedAt": 1517442103
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit AWS Elemental MediaConvert Jobs im AWS Elemental MediaConvert User Guide.

- Einzelheiten zur API finden Sie GetJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetJobin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
```

```
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- API-Details finden Sie GetJobin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListJobs` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
```

```
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-
role-in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Listen Sie die Jobs mit einem bestimmten Status auf.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Listing all complete jobs.");
        var completeJobs = await wrapper.ListAllJobsByStatus(JobStatus.COMPLETE);
        completeJobs.ForEach(j =>
        {
            Console.WriteLine($"Job {j.Id} created on {j.CreatedAt:d} has status
 {j.Status}.");
        });
```

Listen Sie die Jobs mithilfe eines Paginators auf.

```
    /// <summary>
    /// List all of the jobs with a particular status using a paginator.
    /// </summary>
    /// <param name="status">The status to use when listing jobs.</param>
    /// <returns>The list of jobs matching the status.</returns>
    public async Task<List<Job>> ListAllJobsByStatus(JobStatus? status = null)
    {
        var returnedJobs = new List<Job>();

        var paginatedJobs = _amazonMediaConvert.Paginators.ListJobs(
                new ListJobsRequest
                {
                    Status = status
                });
```

```
        // Get the entire list using the paginator.
        await foreach (var job in paginatedJobs.Jobs)
        {
            returnedJobs.Add(job);
        }

        return returnedJobs;
    }
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of created jobs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::listJobs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;
    Aws::String nextToken; // Used to handle paginated results.
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome =
 client.ListJobs(
```

```
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                    outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
                std::cout << "  " << job.Jsonize().View().WriteReadable() <<
 std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;

        }
    } while (!nextToken.empty());


    return result;
}
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die Details für alle Aufträge in einer Region ab

Im folgenden Beispiel werden die Informationen für alle Aufträge in der angegebenen Region angefordert.

```
aws mediaconvert list-jobs \
    --endpoint-url https://abcd1234.mediaconvert.region-name-1.amazonaws.com \
    --region region-name-1
```

Verwenden Sie `describe-endpoints` zum Abrufen des kontospezifischen Endpunkts oder senden Sie den Befehl ohne den Endpunkt. Der Service gibt einen Fehler und den Endpunkt zurück.

Weitere Informationen finden Sie unter Arbeiten mit AWS Elemental MediaConvert Jobs im AWS Elemental MediaConvert User Guide.

- Einzelheiten zur API finden Sie ListJobsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListJobs {
```

```
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
```

```
        }
    }
}
```

- API-Details finden Sie ListJobsin der API-Referenz zum AWS SDK für Kotlin.

Codebeispiele für die Verwendung MediaLive AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elemental MediaLive mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- MediaLive Benutzerhandbuch — Weitere Informationen zu MediaLive.
- MediaLive API-Referenz — Details zu allen verfügbaren MediaLive Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung MediaLive AWS SDKs
  - Aktionen zur Verwendung MediaLive AWS SDKs
    - Verwenden Sie ListInputsmit einem AWS SDK

Grundlegende Beispiele für die Verwendung MediaLive AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Elemental MediaLive mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung MediaLive AWS SDKs

- Verwenden Sie ListInputsmit einem AWS SDK

Aktionen zur Verwendung MediaLive AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne MediaLive Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Elemental MediaLive -API-Referenz.

Beispiele

- Verwenden Sie ListInputsmit einem AWS SDK

Verwenden Sie **`ListInputs`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListInputs`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

MediaLive Geben Sie Ihre Eingabenamen und ARNs in der Region an.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
```

```
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListInputsin der API-Referenz zum AWS SDK für Rust.

Codebeispiele für die MediaPackage Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Elemental MediaPackage mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- MediaPackage Benutzerhandbuch — Weitere Informationen zu MediaPackage.
- MediaPackage API-Referenz — Details zu allen verfügbaren MediaPackage Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung MediaPackage AWS SDKs
  - Aktionen zur Verwendung MediaPackage AWS SDKs
    - Verwenden Sie ListChannelsmit einem AWS SDK oder CLI
    - Verwenden Sie ListOriginEndpointsmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung MediaPackage AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Elemental MediaPackage mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung MediaPackage AWS SDKs
  - Verwenden Sie ListChannelsmit einem AWS SDK oder CLI
  - Verwenden Sie ListOriginEndpointsmit einem AWS SDK oder CLI

Aktionen zur Verwendung MediaPackage AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne MediaPackage Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Elemental MediaPackage -API-Referenz.

Beispiele

- Verwenden Sie ListChannelsmit einem AWS SDK oder CLI
- Verwenden Sie ListOriginEndpointsmit einem AWS SDK oder CLI

Verwenden Sie **`ListChannels`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListChannels` verwendet wird.

CLI

AWS CLI

So listen Sie alle Kanäle auf

Der folgende `list-channels` Befehl listet alle Kanäle auf, die auf dem aktuellen AWS Konto konfiguriert sind.

```
aws mediapackage list-channels
```

Ausgabe:

```
{
    "Channels": [
        {
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:channels/584797f1740548c389a273585dd22a63",
```

```
            "HlsIngest": {
                "IngestEndpoints": [
                    {
                        "Id": "584797f1740548c389a273585dd22a63",
                        "Password": "webdavgeneratedpassword1",
                        "Url": "https://9be9c4405c474882.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/584797f1740548c389a273585dd22a63/channel",
                        "Username": "webdavgeneratedusername1"
                    },
                    {
                        "Id": "7d187c8616fd455f88aaa5a9fcf74442",
                        "Password": "webdavgeneratedpassword2",
                        "Url": "https://7bf454c57220328d.mediapackage.us-
west-2.amazonaws.com/in/
v2/584797f1740548c389a273585dd22a63/7d187c8616fd455f88aaa5a9fcf74442/channel",
                        "Username": "webdavgeneratedusername2"
                    }
                ]
            },
            "Id": "test",
            "Tags": {}
        }
    ]
}
```

Weitere Informationen finden Sie unter Kanaldetails anzeigen im AWS Elemental MediaPackage User Guide.

- Einzelheiten zur API finden Sie ListChannelsin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Kanal-ARNs und Beschreibungen auf.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();

        println!("  Description : {}", description);
        println!("  ARN :         {}", arn);
        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListChannelsin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`ListOriginEndpoints`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListOriginEndpoints` verwendet wird.

CLI

AWS CLI

So listen Sie Ursprungsendpunkte in einem Kanal auf

Mit dem folgenden `list-origin-endpoints`-Befehl werden alle Ursprungsendpunkte aufgelistet, die für den aktuellen Kanal mit dem Namen `test` konfiguriert sind.

```
aws mediapackage list-origin-endpoints \
    --channel-id test
```

Ausgabe:

```
{
    "OriginEndpoints": [
        {
```

```
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/247cff871f2845d3805129be22f2c0a2",
            "ChannelId": "test",
            "DashPackage": {
                "ManifestLayout": "FULL",
                "ManifestWindowSeconds": 60,
                "MinBufferTimeSeconds": 30,
                "MinUpdatePeriodSeconds": 15,
                "PeriodTriggers": [],
                "Profile": "NONE",
                "SegmentDurationSeconds": 2,
                "SegmentTemplateFormat": "NUMBER_WITH_TIMELINE",
                "StreamSelection": {
                    "MaxVideoBitsPerSecond": 2147483647,
                    "MinVideoBitsPerSecond": 0,
                    "StreamOrder": "ORIGINAL"
                },
                "SuggestedPresentationDelaySeconds": 25
            },
            "Id": "tester2",
            "ManifestName": "index",
            "StartoverWindowSeconds": 0,
            "Tags": {},
            "TimeDelaySeconds": 0,
            "Url": "https://8343f7014c0ea438.mediapackage.us-
west-2.amazonaws.com/out/v1/247cff871f2845d3805129be22f2c0a2/index.mpd",
            "Whitelist": []
        },
        {
            "Arn": "arn:aws:mediapackage:us-
west-2:111222333:origin_endpoints/869e237f851549e9bcf10e3bc2830839",
            "ChannelId": "test",
            "HlsPackage": {
                "AdMarkers": "NONE",
                "IncludeIframeOnlyStream": false,
                "PlaylistType": "EVENT",
                "PlaylistWindowSeconds": 60,
                "ProgramDateTimeIntervalSeconds": 0,
                "SegmentDurationSeconds": 6,
                "StreamSelection": {
                    "MaxVideoBitsPerSecond": 2147483647,
                    "MinVideoBitsPerSecond": 0,
                    "StreamOrder": "ORIGINAL"
                },
```

```
                "UseAudioRenditionGroup": false
            },
            "Id": "tester",
            "ManifestName": "index",
            "StartoverWindowSeconds": 0,
            "Tags": {},
            "TimeDelaySeconds": 0,
            "Url": "https://8343f7014c0ea438.mediapackage.us-
west-2.amazonaws.com/out/v1/869e237f851549e9bcf10e3bc2830839/index.m3u8",
            "Whitelist": []
        }
    ]
}
```

Weitere Informationen finden Sie im AWS Elemental MediaPackage User Guide unter Alle Endpoints anzeigen, die einem Kanal zugeordnet sind.

- Einzelheiten zur API finden Sie ListOriginEndpointsin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Endpunktbeschreibungen und URLs auf.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
        println!("  URL :        {}", endpoint_url);
        println!();
```

```
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListOriginEndpointsin der API-Referenz zum AWS SDK für Rust.

Codebeispiele für Migration Hub mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Migration Hub mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Migration-Hub-Benutzerhandbuch – Weitere Informationen zu Migration Hub.
- API-Referenz für Migration Hub – Details zu allen verfügbaren Aktionen von Migration Hub.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Migration Hub AWS SDKs
  - Aktionen für Migration Hub mit AWS SDKs
    - Verwenden Sie DeleteProgressUpdateStreammit einem AWS SDK
    - Verwenden Sie DescribeApplicationStatemit einem AWS SDK
    - Verwenden Sie DescribeMigrationTaskmit einem AWS SDK
    - Verwenden Sie ImportMigrationTaskmit einem AWS SDK
    - Verwenden Sie ListApplicationsmit einem AWS SDK
    - Verwenden Sie ListCreatedArtifactsmit einem AWS SDK

- Verwenden Sie ListMigrationTasksmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Migration Hub AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von AWS Migration Hub mit AWS SDKs verwenden können.

Beispiele

- Aktionen für Migration Hub mit AWS SDKs
  - Verwenden Sie DeleteProgressUpdateStreammit einem AWS SDK
  - Verwenden Sie DescribeApplicationStatemit einem AWS SDK
  - Verwenden Sie DescribeMigrationTaskmit einem AWS SDK
  - Verwenden Sie ImportMigrationTaskmit einem AWS SDK
  - Verwenden Sie ListApplicationsmit einem AWS SDK
  - Verwenden Sie ListCreatedArtifactsmit einem AWS SDK
  - Verwenden Sie ListMigrationTasksmit einem AWS SDK

Aktionen für Migration Hub mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Migration Hub Hub-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Migration Hub -API-Referenz.

Beispiele

- Verwenden Sie DeleteProgressUpdateStreammit einem AWS SDK
- Verwenden Sie DescribeApplicationStatemit einem AWS SDK
- Verwenden Sie DescribeMigrationTaskmit einem AWS SDK
- Verwenden Sie ImportMigrationTaskmit einem AWS SDK
- Verwenden Sie ListApplicationsmit einem AWS SDK
- Verwenden Sie ListCreatedArtifactsmit einem AWS SDK
- Verwenden Sie ListMigrationTasksmit einem AWS SDK

Verwenden Sie **`DeleteProgressUpdateStream`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteProgressUpdateStream`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
 streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
 DeleteProgressUpdateStreamRequest
                    .builder()
                    .progressUpdateStreamName(streamName)
                    .build();


 migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteProgressUpdateStreamin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeApplicationState`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeApplicationState`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeAppState <appId>\s

                Where:
                    appId -  the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }

    public static void describeApplicationState(MigrationHubClient
 migrationClient, String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
 DescribeApplicationStateRequest.builder()
                    .applicationId(appId)
                    .build();

            DescribeApplicationStateResponse applicationStateResponse =
 migrationClient
                    .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
 applicationStateResponse.applicationStatusAsString());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeApplicationStatein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeMigrationTask`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeMigrationTask`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeMigrationTask <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
 migrationTask,
            String progressStream) {
        try {
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
 DescribeMigrationTaskRequest.builder()
                    .progressUpdateStream(progressStream)
                    .migrationTaskName(migrationTask)
                    .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                    .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
 migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeMigrationTaskin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ImportMigrationTask`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ImportMigrationTask`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
 migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
 CreateProgressUpdateStreamRequest.builder()
                    .progressUpdateStreamName(progressStream)
                    .dryRun(false)
                    .build();


 migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
            ImportMigrationTaskRequest migrationTaskRequest =
 ImportMigrationTaskRequest.builder()
                    .migrationTaskName(migrationTask)
                    .progressUpdateStream(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.importMigrationTask(migrationTaskRequest);

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ImportMigrationTaskin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListApplications`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListApplications`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listApps(migrationClient);
        migrationClient.close();
    }
```

```
    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
 ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
 migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
 appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListApplicationsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListCreatedArtifacts`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListCreatedArtifacts`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
```

```
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCreatedArtifactsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListMigrationTasks`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListMigrationTasks`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
 ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();

            ListMigrationTasksResponse response =
 migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
 response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
 migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
 migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListMigrationTasksin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon MSK mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Managed Streaming for Apache Kafka mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-MSK-Entwicklerhandbuch – Weitere Informationen zu Amazon MSK.
- API-Referenz für Amazon MSK – Details zu allen verfügbaren Aktionen von Amazon MSK.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von Amazon MSK AWS SDKs
  - Erste Schritte mit verwaltetem Streaming
- Serverless-Beispiele für Amazon MSK
  - Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Szenarien für die Verwendung von Amazon MSK AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon MSK mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon MSK oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit verwaltetem Streaming

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
```

```
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
```

```
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)
```

```
                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
```

```
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
 removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
```

```
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
```

```
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)
```

```
            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)
```

```
if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
```

```
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
```

```
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
```

```
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"
```

```
# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi
```

```
echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"
```

```
# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
```

```
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
```

```
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
```

```
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
```

```
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
```

```
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
```

```
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
```

```
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy

- AuthorizeSecurityGroupIngress
- CreateCluster
- CreateInstanceProfile
- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile

- RevokeSecurityGroupEgress

- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Serverless-Beispiele für Amazon MSK

Die folgenden Codebeispiele zeigen, wie Amazon MSK mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Die folgenden Codebeispiele zeigen, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
```

```
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to
 process.</param>
    /// <param name="context">The ILambdaContext that provides methods for
 logging and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Ein Amazon MSK-Ereignis mit Lambda verwenden. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Ein Amazon MSK-Ereignis mit Lambda verwenden. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
```

```
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity
```

```
use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error>
 {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
```

```
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Codebeispiele für Neptune mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Neptune mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Neptun-Benutzerhandbuch – Weitere Informationen zu Neptune.
- Neptun-API-Referenz – Details zu allen verfügbaren Neptun-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Neptune AWS SDKs
  - Hello Amazon Neptune
  - Erlernen Sie die Grundlagen von Neptune mit einem AWS SDK
  - Aktionen für Neptune mit AWS SDKs

    - Verwenden Sie CreateDBCluster mit einem AWS SDK
    - Verwenden Sie CreateDBInstance mit einem AWS SDK
    - Verwenden Sie CreateDBSubnetGroupmit einem AWS SDK
    - Verwenden Sie mit CreateGrapheinem AWS SDK
    - Verwenden Sie DeleteDBCluster mit einem AWS SDK
    - Verwenden Sie DeleteDBInstance mit einem AWS SDK
    - Verwenden Sie DeleteDBSubnetGroupmit einem AWS SDK
    - Verwenden Sie DescribeDBClusters mit einem AWS SDK
    - Verwenden Sie DescribeDBInstances mit einem AWS SDK
    - Verwenden Sie ExecuteGremlinProfileQuerymit einem AWS SDK
    - Verwenden Sie mit ExecuteGremlinQueryeinem AWS SDK
    - Verwenden Sie mit ExecuteOpenCypherExplainQueryeinem AWS SDK
    - Verwenden Sie mit ExecuteQueryeinem AWS SDK
    - Verwenden Sie StartDBCluster mit einem AWS SDK
    - Verwenden Sie StopDBCluster mit einem AWS SDK
- Szenarien für die Verwendung von Neptune AWS SDKs
  - Erste Schritte mit Graphdatenbanken
  - Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt

Grundlegende Beispiele für die Verwendung von Neptune AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Neptune mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Neptune
- Erlernen Sie die Grundlagen von Neptune mit einem AWS SDK
- Aktionen für Neptune mit AWS SDKs
  - Verwenden Sie CreateDBCluster mit einem AWS SDK
  - Verwenden Sie CreateDBInstance mit einem AWS SDK
  - Verwenden Sie CreateDBSubnetGroupmit einem AWS SDK

- Verwenden Sie mit CreateGrapheinem AWS SDK
- Verwenden Sie DeleteDBCluster mit einem AWS SDK
- Verwenden Sie DeleteDBInstance mit einem AWS SDK
- Verwenden Sie DeleteDBSubnetGroupmit einem AWS SDK
- Verwenden Sie DescribeDBClusters mit einem AWS SDK
- Verwenden Sie DescribeDBInstances mit einem AWS SDK
- Verwenden Sie ExecuteGremlinProfileQuerymit einem AWS SDK
- Verwenden Sie mit ExecuteGremlinQueryeinem AWS SDK
- Verwenden Sie mit ExecuteOpenCypherExplainQueryeinem AWS SDK
- Verwenden Sie mit ExecuteQueryeinem AWS SDK
- Verwenden Sie StartDBCluster mit einem AWS SDK
- Verwenden Sie StopDBCluster mit einem AWS SDK

Hello Amazon Neptune

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Neptune beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloNeptune {
```

```
   public static void main(String[] args) {
       NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
       describeDbCluster(neptuneClient).join(); // This ensures the async code
runs to completion
   }

   /**
    * Describes the Amazon Neptune DB clusters.
    *
    * @param neptuneClient the Neptune asynchronous client used to make the
request
    * @return a {@link CompletableFuture} that completes when the operation is
finished
    */
   public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
neptuneClient) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .maxRecords(20)
               .build();

       SdkPublisher<DescribeDbClustersResponse> paginator =
neptuneClient.describeDBClustersPaginator(request);
       CompletableFuture<Void> future = new CompletableFuture<>();

       paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
           private Subscription subscription;

           @Override
           public void onSubscribe(Subscription s) {
               this.subscription = s;
               s.request(Long.MAX_VALUE); // request all items
           }

           @Override
           public void onNext(DescribeDbClustersResponse response) {
               response.dbClusters().forEach(cluster -> {
                   System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                   System.out.println("Status: " + cluster.status());
               });
           }

           @Override
           public void onError(Throwable t) {
```

```
                future.completeExceptionally(t);
            }

            @Override
            public void onComplete() {
                future.complete(null);
            }
        });

        return future.whenComplete((result, throwable) -> {
            neptuneClient.close();
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
 throwable.getMessage());
            }
        });
    }
```

- Einzelheiten zur API finden Sie DescribeDBClustersPaginatorin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError


def describe_db_clusters(neptune_client):
    """
    Describes the Amazon Neptune DB clusters using a paginator to handle multiple
 pages.
    Raises ClientError with 'ResourceNotFoundException' if no clusters are found.
```

```
    """
    paginator = neptune_client.get_paginator("describe_db_clusters")
    clusters_found = False

    for page in paginator.paginate():
        for cluster in page.get("DBClusters", []):
            clusters_found = True
            print(f"Cluster Identifier: {cluster['DBClusterIdentifier']}")
            print(f"Status: {cluster['Status']}")

    if not clusters_found:
        raise ClientError(
            {
                "Error": {
                    "Code": "ResourceNotFoundException",
                    "Message": "No Neptune DB clusters found."
                }
            },
            operation_name="DescribeDBClusters"
        )

def main():
    """
    Main entry point: creates the Neptune client and calls the describe
 operation.
    """
    neptune_client = boto3.client("neptune")
    try:
        describe_db_clusters(neptune_client)
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "ResourceNotFoundException":
            print(f"Resource not found: {e.response['Error']['Message']}")
        else:
            print(f"Unexpected ClientError: {e.response['Error']['Message']}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie DescribeDBClustersPaginatorin AWS SDK for Python (Boto3) API Reference.

Erlernen Sie die Grundlagen von Neptune mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine Subnetzgruppe von Amazon Neptune.
- Erstellen Sie einen Neptun-Cluster.
- Erstellen Sie eine Neptun-Instance.
- Prüfen des Status einer Neptune-Instance.
- Zeigen Sie Details zum Neptun-Cluster an.
- Stoppen Sie den Neptune-Cluster.
- Starten Sie den Neptune-Cluster.
- Löschen Sie die Neptune-Komponenten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Neptune-Features gezeigt werden.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
```

```
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";

        logger.info("""
                   Amazon Neptune is a fully managed graph
                   database service by AWS, designed specifically
                   for handling complex relationships and connected
                   datasets at scale. It supports two popular graph models:
                   property graphs (via openCypher and Gremlin) and RDF
                   graphs (via SPARQL). This makes Neptune ideal for
                   use cases such as knowledge graphs, fraud detection,
                   social networking, recommendation engines, and
                   network management, where relationships between
                   entities are central to the data.

                   Being fully managed, Neptune handles database
                   provisioning, patching, backups, and replication,
                   while also offering high availability and durability
                   within AWS's infrastructure.

                   For developers, programming with Neptune allows
                   for building intelligent, relationship-aware
                   applications that go beyond traditional tabular
                   databases. Developers can use the AWS SDK for Java
                   to automate infrastructure operations (via NeptuneClient).

                    Let's get started...
                    """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }
```

```
   public static void runScenario(String subnetGroupName, String dbInstanceId,
String clusterName) {
       logger.info(DASHES);
       logger.info("1. Create a Neptune DB Subnet Group");
       logger.info("The Neptune DB subnet group is used when launching a Neptune
cluster");
       waitForInputToContinue(scanner);
       try {
           neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ServiceQuotaExceededException) {
               logger.error("The request failed due to service quota exceeded:
{}", cause.getMessage());
           } else {
               logger.error("An unexpected error occurred.", cause);
           }
           return;
       }
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("2. Create a Neptune Cluster");
       logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
       waitForInputToContinue(scanner);
       String dbClusterId;
       try {
           dbClusterId =
neptuneActions.createDBClusterAsync(clusterName).join();
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ServiceQuotaExceededException) {
               logger.error("The request failed due to service quota exceeded:
{}", cause.getMessage());
           } else {
               logger.error("An unexpected error occurred.", cause);
           }
           return;
       }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
 cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded:
 {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
```

```
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
```

```
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found:
 {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
 clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not
 found: {}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
```

```
                """
                Thank you for checking out the Amazon Neptune Service Use demo.
 We hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Neptune-SDK-Methoden.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneActions.class);
    private final NeptuneClient neptuneClient =
 NeptuneClient.builder().region(region).build();

    /**
     * Retrieves an instance of the NeptuneAsyncClient.
```

```
    * <p>
    * This method initializes and returns a singleton instance of the
NeptuneAsyncClient. The client
    * is configured with the following settings:
    * <ul>
    *     <li>Maximum concurrency: 100</li>
    *     <li>Connection timeout: 60 seconds</li>
    *     <li>Read timeout: 60 seconds</li>
    *     <li>Write timeout: 60 seconds</li>
    *     <li>API call timeout: 2 minutes</li>
    *     <li>API call attempt timeout: 90 seconds</li>
    *     <li>Retry strategy: STANDARD</li>
    * </ul>
    * The client is built using the NettyNioAsyncHttpClient.
    *
    * @return the singleton instance of the NeptuneAsyncClient
    */
   private static NeptuneAsyncClient getAsyncClient() {
       if (neptuneAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                   .maxConcurrency(100)
                   .connectionTimeout(Duration.ofSeconds(60))
                   .readTimeout(Duration.ofSeconds(60))
                   .writeTimeout(Duration.ofSeconds(60))
                   .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                   .apiCallTimeout(Duration.ofMinutes(2))
                   .apiCallAttemptTimeout(Duration.ofSeconds(90))
                   .retryStrategy(RetryMode.STANDARD)
                   .build();

           neptuneAsyncClient = NeptuneAsyncClient.builder()
                   .httpClient(httpClient)
                   .overrideConfiguration(overrideConfig)
                   .build();
       }
       return neptuneAsyncClient;
   }

   /**
    * Asynchronously deletes a set of Amazon Neptune resources in a defined
order.
```

```
    * <p>
    * The method performs the following operations in sequence:
    * <ol>
    *     <li>Deletes the Neptune DB instance identified by {@code
dbInstanceId}.</li>
    *     <li>Waits until the DB instance is fully deleted.</li>
    *     <li>Deletes the Neptune DB cluster identified by {@code
dbClusterId}.</li>
    *     <li>Deletes the Neptune DB subnet group identified by {@code
subnetGroupName}.</li>
    * </ol>
    * <p>
    * If any step fails, the subsequent operations are not performed, and the
exception
    * is logged. This method blocks the calling thread until all operations
complete.
    *
    * @param dbInstanceId      the ID of the Neptune DB instance to delete
    * @param dbClusterId       the ID of the Neptune DB cluster to delete
    * @param subnetGroupName   the name of the Neptune DB subnet group to delete
    */
   public void deleteNeptuneResourcesAsync(String dbInstanceId, String
dbClusterId, String subnetGroupName) {
       deleteDBInstanceAsync(dbInstanceId)
               .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
               .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
               .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
               .whenComplete((v, ex) -> {
                   if (ex != null) {
                       logger.info("Failed to delete Neptune resources: " +
ex.getMessage());
                   } else {
                       logger.info("Neptune resources deleted successfully.");
                   }
               })
               .join(); // Waits for the entire async chain to complete
   }

   /**
    * Deletes a subnet group.
    *
    * @param subnetGroupName the identifier of the subnet group to delete
    * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
```

```
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String
subnetGroupName) {
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: "
+ subnetGroupName));
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Cluster: " + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String
instanceId) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance
has been deleted
```

```
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Instance: " + instanceId));
    }


    private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof NeptuneException &&
                                ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                            long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                            logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                            future.complete(null);
                            return;
                        }
                        future.completeExceptionally(new
CompletionException("Error polling DB instance", cause));
                        return;
                    }

                    String status =
response.dbInstances().get(0).dbInstanceStatus();
                    long elapsed = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r  Waiting: Instance %s status: %-10s
(%ds elapsed)", instanceId, status, elapsed);
```

```
                    System.out.flush();

                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() ->
checkInstanceDeletedRecursive(instanceId, startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String
desiredStatus, long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new
RuntimeException("Cluster not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
```

```
                    long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
   }


   /**
    * Starts an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to start DB cluster: "
+ cause.getMessage(), cause);
```

```
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }

    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: "
+ cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
```

```
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " +
cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " +
cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
```

```
                    }

                    throw new RuntimeException("Failed to describe the DB
cluster: " + cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired
status is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate
the elapsed time
     * @param future         a {@link CompletableFuture} that will be completed
when the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
                    }
```

```
                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new
RuntimeException("Instance not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : "
secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
```

```
    * Creates a new Amazon Neptune DB instance asynchronously.
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new
instance will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of
the newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of
either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune
DB instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }


   /**
```

```
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason,
including if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune
DB cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId =
response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
               });
   }


   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon
Resource Name (ARN) of the created subnet group
```

```
     * @throws CompletionException if the operation fails, with a cause that may
be a ServiceQuotaExceededException if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster
and resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
```

```
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBCluster

- CreateDBInstance
- CreateDBSubnetGroup
- CreateGraph
- DeleteDBCluster
- DeleteDBInstance
- DeleteDBSubnetGroup
- DescribeDBClusters
- DescribeDBInstances
- ExecuteGremlinProfileQuery
- ExecuteGremlinQuery
- ExecuteOpenCypherExplainQuery
- ExecuteQuery
- StartDBCluster
- StopDBCluster

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
import time
from botocore.exceptions import ClientError

# Constants used in this scenario
POLL_INTERVAL_SECONDS = 10
TIMEOUT_SECONDS = 1200  # 20 minutes

def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.
```

```
    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise

def format_elapsed_time(seconds: int) -> str:
    mins, secs = divmod(seconds, 60)
    hours, mins = divmod(mins, 60)
    return f"{hours:02}:{mins:02}:{secs:02}"


def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
```

```
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise

def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }
```

```
    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
        raise

def wait_for_cluster_status(
        neptune_client,
        cluster_id: str,
        desired_status: str,
        timeout_seconds: int = TIMEOUT_SECONDS,
        poll_interval_seconds: int = POLL_INTERVAL_SECONDS
):
    """
    Waits for a Neptune DB cluster to reach a desired status.

    Args:
        neptune_client (boto3.client): The Amazon Neptune client.
        cluster_id (str): The identifier of the Neptune DB cluster.
        desired_status (str): The target status (e.g., "available", "stopped").
        timeout_seconds (int): Max time to wait in seconds (default: 1200).
        poll_interval_seconds (int): Polling interval in seconds (default: 10).

    Raises:
        RuntimeError: If the desired status is not reached before timeout.
    """
    print(f"Waiting for cluster '{cluster_id}' to reach status
 '{desired_status}'...")
    start_time = time.time()

    while True:
        # Prepare request object
```

```
        describe_cluster_request = {
            'DBClusterIdentifier': cluster_id
        }

        # Call the Neptune API
        response =
 neptune_client.describe_db_clusters(**describe_cluster_request)
        clusters = response.get('DBClusters', [])
        current_status = clusters[0].get('Status') if clusters else None
        elapsed_seconds = int(time.time() - start_time)

        status_str = current_status if current_status else "Unknown"
        print(
            f"\r Elapsed: {format_elapsed_time(elapsed_seconds):<20}  Cluster
 status: {status_str:<20}",
            end="", flush=True
        )

        if current_status and current_status.lower() == desired_status.lower():
            print(
                f"\nNeptune cluster reached desired status '{desired_status}'
 after {format_elapsed_time(elapsed_seconds)}."
            )
            return

        if elapsed_seconds > timeout_seconds:
            raise RuntimeError(f"Timeout waiting for Neptune cluster to reach
 status: {desired_status}")

        time.sleep(poll_interval_seconds)


def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
```

```
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
```

```
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
 '{cluster_identifier}' to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)


def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
```

```
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
 It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
 {message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
 flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
 '{cluster_identifier}' to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)



def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)
```

```
        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier:
{cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
{cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
{cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
{cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
{cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        elif code == "DBClusterNotFound":
```

```
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
 ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise

def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else
 None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
 {current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
```

```
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
 {format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
 '{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)


def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) -
> str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become
 available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
```

```
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
 '{db_instance_id}': {e}") from e


def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id
```

```
    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException",
 "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
 clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}':
 {e}") from e

def get_subnet_ids(vpc_id: str) -> list[str]:
    ec2_client = boto3.client('ec2')

    describe_subnets_request = {
        'Filters': [{'Name': 'vpc-id', 'Values': [vpc_id]}]
    }

    response = ec2_client.describe_subnets(**describe_subnets_request)
    subnets = response.get('Subnets', [])
    subnet_ids = [subnet['SubnetId'] for subnet in subnets if 'SubnetId' in
 subnet]
    return subnet_ids


def get_default_vpc_id() -> str:
    ec2_client = boto3.client('ec2')
    describe_vpcs_request = {
        'Filters': [{'Name': 'isDefault', 'Values': ['true']}]
    }

    response = ec2_client.describe_vpcs(**describe_vpcs_request)
    vpcs = response.get('Vpcs', [])
    if not vpcs:
        raise RuntimeError("No default VPC found in this region.")

    default_vpc_id = vpcs[0]['VpcId']
    print(f"Default VPC ID: {default_vpc_id}")
    return default_vpc_id
```

```
def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
```

```
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group
 '{group_name}': {e}") from e

def wait_for_input_to_continue():
    input("\nPress <ENTER> to continue...")
    print("Continuing with the program...\n")


def run_scenario(neptune_client, subnet_group_name: str, db_instance_id: str,
 cluster_name: str):
    print("-" * 88)
    print("1. Create a Neptune DB Subnet Group")
    wait_for_input_to_continue()

    try:
        name, arn = create_subnet_group(neptune_client, subnet_group_name)
        print(f"Subnet group successfully created: {name}")

        print("-" * 88)
        print("2. Create a Neptune Cluster")
        wait_for_input_to_continue()
        db_cluster_id = create_db_cluster(neptune_client, cluster_name)

        print("-" * 88)
        print("3. Create a Neptune DB Instance")
        wait_for_input_to_continue()
        create_db_instance(neptune_client, db_instance_id, cluster_name)

        print("-" * 88)
        print("4. Check the status of the Neptune DB Instance")
        print("""
        Even though you're targeting a single DB instance,
        describe_db_instances supports pagination and can return multiple pages.

        Handling paginated responses ensures your method continues to work
 reliably
        even if AWS returns large or paged results.
```

```
        """)
        wait_for_input_to_continue()
        check_instance_status(neptune_client, db_instance_id, "available")

        print("-" * 88)
        print("5. Show Neptune Cluster details")
        wait_for_input_to_continue()
        describe_db_clusters(neptune_client, db_cluster_id)

        print("-" * 88)
        print("6. Stop the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for stop_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in a stopped state.
        """)
        wait_for_input_to_continue()
        stop_db_cluster(neptune_client, db_cluster_id)
        check_instance_status(neptune_client, db_instance_id, "stopped")

        print("-" * 88)
        print("7. Start the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for start_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in an available state.
        """)
        wait_for_input_to_continue()
        start_db_cluster(neptune_client, db_cluster_id)
        wait_for_cluster_status(neptune_client, db_cluster_id, "available")
        check_instance_status(neptune_client, db_instance_id, "available")

        print("All Neptune resources are now available.")
        print("-" * 88)

        print("-" * 88)
        print("8. Delete the Neptune Assets")
        print("Would you like to delete the Neptune Assets? (y/n)")
        del_ans = input().strip().lower()

        if del_ans == "y":
            print("You selected to delete the Neptune assets.")
```

```
            delete_db_instance(neptune_client, db_instance_id)
            delete_db_cluster(neptune_client, db_cluster_id)
            delete_db_subnet_group(neptune_client, subnet_group_name)

            print("Neptune resources deleted successfully")

    except ClientError as ce:
        code = ce.response["Error"]["Code"]

        if code in ("DBInstanceNotFound", "DBInstanceNotFoundFault",
 "ResourceNotFound"):
            print(f"Instance '{db_instance_id}' not found.")
        elif code in ("DBClusterNotFound", "DBClusterNotFoundFault",
 "ResourceNotFoundFault"):
            print(f"Cluster '{cluster_name}' not found.")
        elif code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"AWS error [{code}]: {ce.response['Error']['Message']}")
            raise  # re-raise unexpected errors

    except RuntimeError as re:
        print(f"Runtime error or timeout: {re}")


def main():
    neptune_client = boto3.client('neptune')

    # Customize the following names to match your Neptune setup
    # (You must change these to unique values for your environment)
    subnet_group_name = "neptuneSubnetGroup111"
    cluster_name = "neptuneCluster111"
    db_instance_id = "neptuneDB111"

    print("""
    Amazon Neptune is a fully managed graph database service by AWS...
    Let's get started!
    """)
    wait_for_input_to_continue()
    run_scenario(neptune_client, subnet_group_name, db_instance_id, cluster_name)
```

```
    print("""
    Thank you for checking out the Amazon Neptune Service Use demo.
    For more AWS code examples, visit:
    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
    """)

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance
  - DeleteDBSubnetGroup
  - DescribeDBClusters
  - DescribeDBInstances
  - ExecuteGremlinProfileQuery
  - ExecuteGremlinQuery
  - ExecuteOpenCypherExplainQuery
  - ExecuteQuery
  - StartDBCluster
  - StopDBCluster

Aktionen für Neptune mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Neptune-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Neptune API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Neptune AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Neptune.

Beispiele

- Verwenden Sie CreateDBCluster mit einem AWS SDK
- Verwenden Sie CreateDBInstance mit einem AWS SDK
- Verwenden Sie CreateDBSubnetGroupmit einem AWS SDK
- Verwenden Sie mit CreateGrapheinem AWS SDK
- Verwenden Sie DeleteDBCluster mit einem AWS SDK
- Verwenden Sie DeleteDBInstance mit einem AWS SDK
- Verwenden Sie DeleteDBSubnetGroupmit einem AWS SDK
- Verwenden Sie DescribeDBClusters mit einem AWS SDK
- Verwenden Sie DescribeDBInstances mit einem AWS SDK
- Verwenden Sie ExecuteGremlinProfileQuerymit einem AWS SDK
- Verwenden Sie mit ExecuteGremlinQueryeinem AWS SDK
- Verwenden Sie mit ExecuteOpenCypherExplainQueryeinem AWS SDK
- Verwenden Sie mit ExecuteQueryeinem AWS SDK
- Verwenden Sie StartDBCluster mit einem AWS SDK
- Verwenden Sie StopDBCluster mit einem AWS SDK

Verwenden Sie **`CreateDBCluster mit einem`** AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason,
including if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune
DB cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
```

```
                    String clusterId =
 response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK for Java 2.x - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }
```

```
    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException",
"DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}':
{e}") from e
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

Verwenden Sie **`CreateDBInstance mit einem`** AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new Amazon Neptune DB instance asynchronously.
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new
instance will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of
the newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of
either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was
denied because the request would exceed the maximum quota.", cause);
                       }
```

```
                        throw new CompletionException("Failed to create Neptune
 DB instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
 response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) -
> str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
```

```
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become
available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
'{db_instance_id}': {e}") from e
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Verwenden Sie **`CreateDBSubnetGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDBSubnetGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon
Resource Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may
be a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster
and resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();

       return getAsyncClient().createDBSubnetGroup(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
```

```
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was
 denied because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
 group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }
```

- Einzelheiten zur API finden Sie CreateDBSubnetGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)
```

```
    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group
'{group_name}': {e}") from e
```

- Einzelheiten zur API finden Sie CreateDBSubnetGroupin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateGraph`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateGraph` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes the process of creating a new Neptune graph.
    *
    * @param client        the Neptune graph client used to interact with the
Neptune service
    * @param graphName     the name of the graph to be created
    * @throws NeptuneGraphException if an error occurs while creating the graph
    */
   public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
       try {
           // Create the graph request
           CreateGraphRequest request = CreateGraphRequest.builder()
                   .graphName(graphName)
                   .provisionedMemory(16)
                   .build();

           // Create the graph
           CreateGraphResponse response = client.createGraph(request);
```

```
            // Extract the graph name and ARN
            String createdGraphName = response.name();
            String graphArn = response.arn();
            String graphEndpoint = response.endpoint();

            System.out.println("Graph created successfully!");
            System.out.println("Graph Name: " + createdGraphName);
            System.out.println("Graph ARN: " + graphArn);
            System.out.println("Graph Endpoint: " +graphEndpoint );

        } catch (NeptuneGraphException e) {
            System.err.println("Failed to create graph: " +
 e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
    }
```

- Einzelheiten zur API finden Sie CreateGraphin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

-------------------------------------------------------------------------------
VPC Networking Requirement:
-------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:
```

```
  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**

"""

GRAPH_NAME = "sample-analytics-graph"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)
    execute_create_graph(client, GRAPH_NAME)

def execute_create_graph(client, graph_name):
    try:
        print("Creating Neptune graph...")
        response = client.create_graph(
            graphName=graph_name,
            provisionedMemory = 16
        )

        created_graph_name = response.get("name")
        graph_arn = response.get("arn")
        graph_endpoint = response.get("endpoint")

        print("Graph created successfully!")
        print(f"Graph Name: {created_graph_name}")
        print(f"Graph ARN: {graph_arn}")
        print(f"Graph Endpoint: {graph_endpoint}")

    except ClientError as e:
        print(f"Failed to create graph: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Failed to create graph: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie CreateGraphin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie `DeleteDBCluster mit` einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a DB instance asynchronously.
    *
    * @param clusterId the identifier of the cluster to delete
    * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
    */
   public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
       DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
               .dbClusterIdentifier(clusterId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBCluster(request)
               .thenAccept(response -> System.out.println("## Deleting DB
Cluster: " + clusterId));
   }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK for Java 2.x - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

Verwenden Sie **`DeleteDBInstance mit`** einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance
has been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB
Instance: " + instanceId));
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
```

```
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Verwenden Sie **`DeleteDBSubnetGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteDBSubnetGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has
been deleted
     */
```

```
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String
 subnetGroupName) {
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: "
 + subnetGroupName));
    }
```

- Einzelheiten zur API finden Sie DeleteDBSubnetGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
```

```
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
        raise
```

- Einzelheiten zur API finden Sie DeleteDBSubnetGroupin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie **`DescribeDBClusters`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBClusters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " +
cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " +
cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB
 cluster: " + cause.getMessage(), cause);
                });
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
```

```
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier:
{cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
{cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
{cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
{cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
{cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
```

```
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

Verwenden Sie **`DescribeDBInstances`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeDBInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Checks the status of a Neptune instance recursively until the desired
status is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate
the elapsed time
     * @param future         a {@link CompletableFuture} that will be completed
when the desired status is reached
     */
```

```
   private void checkStatusRecursive(String instanceId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new
RuntimeException("Instance not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() -
startTime) / 1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                       System.out.printf("\r Neptune instance reached
desired status '%s' after %s.\n", desiredStatus, formatElapsedTime((int)
elapsedSeconds));
                       future.complete(null);
                   } else {
                       CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                               .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                   }
               });
   }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
```

```
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else
None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
{current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
{format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
'{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

Verwenden Sie **`ExecuteGremlinProfileQuery`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteGremlinProfileQuery` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
    *
    * @param client the {@link NeptunedataClient} instance to use for executing
the Gremlin query
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Querying Neptune...");
           ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                   .gremlinQuery("g.V().has('code', 'ANC')")
                   .build();

           ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

           System.out.println("Full Response:");
           System.out.println(response);

           // Retrieve and print the result
           if (response.result() != null) {
               System.out.println("Query Result:");
               System.out.println(response.result().toString());
           } else {
               System.out.println("No result returned from the query.");
           }
       } catch (NeptunedataException e) {
           System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
       } catch (Exception e) {
```

```
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteGremlinProfileQueryin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Replace this with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and executes the
 Gremlin query.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_gremlin_query(neptune_client)


def execute_gremlin_query(neptune_client):
```

```
    """
    Executes a Gremlin query against an Amazon Neptune database.
    """
    try:
        print("Querying Neptune...")

        response = neptune_client.execute_gremlin_explain_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )

        print("Full Response:")
        print(response['output'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Error calling Neptune: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ExecuteGremlinProfileQueryin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ExecuteGremlinQuery`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteGremlinQuery` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
     *
     * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
     */
    private static void executeGremlinProfileQuery(NeptunedataClient client) {
        System.out.println("Executing Gremlin PROFILE query...");

        ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
                .gremlinQuery("g.V().has('code', 'ANC')")
                .build();

        ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
        if (response.output() != null) {
            System.out.println("Query Profile Output:");
            System.out.println(response.output());
        } else {
            System.out.println("No output returned from the profile query.");
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteGremlinQueryin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**

"""

# Replace with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify-Your-Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and runs both
 EXPLAIN and PROFILE queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
```

```
    )

    try:
        run_profile_query(neptune_client)
    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

def run_profile_query(neptune_client):
    """
    Runs a PROFILE query on the Neptune graph database.
    """
    print("Running Gremlin PROFILE query...")

    try:
        response = neptune_client.execute_gremlin_profile_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )
        print("Profile Query Result:")
        output = response.get("output")
        if output:
            print(output.read().decode('utf-8'))
        else:
            print("No explain output returned.")
    except Exception as e:
        print(f"Failed to execute PROFILE query: {str(e)}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ExecuteGremlinQueryin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ExecuteOpenCypherExplainQuery`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteOpenCypherExplainQuery` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes an OpenCypher EXPLAIN query using the provided Neptune data
client.
    *
    * @param client The Neptune data client to use for the query execution.
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Executing OpenCypher EXPLAIN query...");
           ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                   .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                   .explainMode("debug")
                   .build();

           ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

           if (response.results() != null) {
               System.out.println("Explain Results:");
               System.out.println(response.results().asUtf8String());
           } else {
               System.out.println("No explain results returned.");
           }

       } catch (NeptunedataException e) {
```

```
            System.err.println("Neptune error: " +
 e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteOpenCypherExplainQueryin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Replace with your actual Neptune endpoint URL
NEPTUNE_ENDPOINT = "https://<your-neptune-endpoint>:8182"

def main():
    """
    Entry point: Create Neptune client and execute different OpenCypher queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_open_cypher_query_without_params(neptune_client)
```

```
    execute_open_cypher_query_with_params(neptune_client)
    execute_open_cypher_explain_query(neptune_client)

def execute_open_cypher_query_without_params(client):
    """
    Executes a simple OpenCypher query without parameters.
    """
    try:
        print("\nRunning OpenCypher query without parameters...")
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n"
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in simple OpenCypher query: {str(e)}")


def execute_open_cypher_query_with_params(client):
    """
    Executes an OpenCypher query using parameters.
    """
    try:
        print("\nRunning OpenCypher query with parameters...")
        parameters = {'code': 'ANC'}
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: $code}) RETURN n",
            parameters=json.dumps(parameters)
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in parameterized OpenCypher query: {str(e)}")

def execute_open_cypher_explain_query(client):
    """
    Runs an OpenCypher EXPLAIN query in debug mode.
    """
    try:
        print("\nRunning OpenCypher EXPLAIN query (debug mode)...")
        resp = client.execute_open_cypher_explain_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n",
```

```
            explainMode="details"
        )
        results = resp.get('results')
        if results is None:
            print("No explain results returned.")
        else:
            try:
                print("Explain Results:")
                print(results.read().decode('UTF-8'))
            except Exception as e:
                print(f"Error in OpenCypher EXPLAIN query: {str(e)}")

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ExecuteOpenCypherExplainQueryin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ExecuteQuery`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteQuery` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes a Gremlin profile query on the Neptune Analytics graph.
    *
    * @param client       the {@link NeptuneGraphClient} instance to use for the
query
    * @param graphId      the identifier of the graph to execute the query on
    *
    * @throws NeptuneGraphException if an error occurs while executing the query
on the Neptune Graph
    * @throws Exception if an unexpected error occurs
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client,
String graphId) {

       try {
           System.out.println("Running openCypher query on Neptune
Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
```

```
                System.err.println("Error reading response: " + e.getMessage());
            }

        } catch (NeptuneGraphException e) {
            System.err.println("NeptuneGraph error: " +
 e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteQueryin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune
 cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a
 **peered VPC**
"""
```

```
GRAPH_ID = "<your-graph-id>"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)

    try:
        print("\n--- Running OpenCypher query without parameters ---")
        run_open_cypher_query(client, GRAPH_ID)

        print("\n--- Running OpenCypher query with parameters ---")
        run_open_cypher_query_with_params(client, GRAPH_ID)

        print("\n--- Running OpenCypher explain query ---")
        run_open_cypher_explain_query(client, GRAPH_ID)

    except Exception as e:
        print(f"Unexpected error in main: {e}")

def run_open_cypher_query(client, graph_id):
    """
    Run an OpenCypher query without parameters.
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_query_with_params(client, graph_id):
    """
    Run an OpenCypher query with parameters.
    """
    try:
```

```
        parameters = {'code': 'ANC'}
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: $code}) RETURN n",
            language='OPEN_CYPHER',
            parameters=parameters
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_explain_query(client, graph_id):
    """
    Run an OpenCypher explain query (explainMode = "debug").
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER',
            explainMode='DETAILS'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Unexpected Boto3 error: {str(e)}")
    except Exception as e:  # <-- Add this generic catch
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie ExecuteQueryin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie **`StartDBCluster`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
```

```
                        throw new RuntimeException("Failed to start DB cluster: "
 + cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }
```

- Weitere API-Informationen finden Sie unter StartDBCluster in der AWS SDK for Java 2.x - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]
```

```
        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
'{cluster_identifier}' to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter StartDBCluster in der API-Referenz zum AWS - SDK für Python (Boto3).

Verwenden Sie **`StopDBCluster`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StopDBCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Stops an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
```

```
                        Throwable cause = error.getCause() != null ?
 error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: "
 + cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }
```

- Weitere API-Informationen finden Sie unter StopDBCluster in der AWS SDK for Java 2.x - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
```

```
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling.
It may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s - Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return
```

```
        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster
'{cluster_identifier}' to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter StopDBCluster in der API-Referenz zum AWS - SDK für Python (Boto3).

Szenarien für die Verwendung von Neptune AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Neptune mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Neptune oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Graphdatenbanken
- Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt

Erste Schritte mit Graphdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihre Neptune-Datenbank
- Erstellen Sie Subnetze in mehreren Verfügbarkeitszonen
- Konfigurieren Sie die Sicherheit für Ihre Neptune-Datenbank
- Erstellen Sie eine Neptune DB-Subnetzgruppe
- Erstellen Sie einen Neptune-DB-Cluster und eine Neptune-DB-Instanz

- Fügen Sie Daten zu Ihrer Graph-Datenbank hinzu
- Fragen Sie Ihre Graphdatenbank ab

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
```

```
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
```

```
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -
f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"

log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"
```

```
# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2
 | tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3
 | tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a
 "$LOG_FILE"
```

```
# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' |
 cut -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-
cidr-block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-
id $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID
 --protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
```

```
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a
 "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
```

```
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name
\\\", \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or
 setting up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a
 "$LOG_FILE"
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID
 --skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so
 we'll sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a
 "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-
id $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
```

```
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later."
 | tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup

- DeleteInternetGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Verwenden der Amazon Neptune-API zur Entwicklung einer Lambda-Funktion, die Graphdaten abfragt

Das folgende Codebeispiel zeigt, wie Sie mit der Neptune-API Grafikdaten abfragen.

Java

SDK für Java 2.x

Zeigt, wie die Java-API von Amazon Neptune verwendet wird, um eine Lambda-Funktion zu erstellen, die Grafikdaten innerhalb der VPC abfragt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Neptune

Codebeispiele für Organizations, die AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Organizations mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zu Organizations – Weitere Informationen zu Organizations.
- API-Referenz für Organizations – Details zu allen verfügbaren Organizations-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Organizations, die AWS SDKs
  - Aktionen für Organizations, die AWS SDKs
    - Verwenden Sie mit AttachPolicyeinem AWS SDK oder CLI
    - Verwenden Sie CreateAccountmit einem AWS SDK oder CLI
    - Verwenden Sie CreateOrganizationmit einem AWS SDK oder CLI
    - Verwenden Sie CreateOrganizationalUnitmit einem AWS SDK oder CLI
    - Verwenden Sie CreatePolicymit einem AWS SDK oder CLI
    - Verwenden Sie DeleteOrganizationmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteOrganizationalUnitmit einem AWS SDK oder CLI
    - Verwenden Sie DeletePolicymit einem AWS SDK oder CLI
    - Verwenden Sie DescribePolicymit einem AWS SDK oder CLI
    - Verwenden Sie DetachPolicymit einem AWS SDK oder CLI
    - Verwenden Sie ListAccountsmit einem AWS SDK oder CLI
    - Verwenden Sie ListOrganizationalUnitsForParentmit einem AWS SDK oder CLI
    - Verwenden Sie ListPoliciesmit einem AWS SDK oder CLI
- Szenarien für Organizations, die AWS SDKs

- Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen
- Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation
- Richtlinie zur Aktivierung der Automatisierung für Ihr Konto
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation

Grundlegende Beispiele für Organizations, die AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Organizations mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Organizations, die AWS SDKs
  - Verwenden Sie mit AttachPolicyeinem AWS SDK oder CLI
  - Verwenden Sie CreateAccountmit einem AWS SDK oder CLI
  - Verwenden Sie CreateOrganizationmit einem AWS SDK oder CLI
  - Verwenden Sie CreateOrganizationalUnitmit einem AWS SDK oder CLI
  - Verwenden Sie CreatePolicymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteOrganizationmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteOrganizationalUnitmit einem AWS SDK oder CLI
  - Verwenden Sie DeletePolicymit einem AWS SDK oder CLI
  - Verwenden Sie DescribePolicymit einem AWS SDK oder CLI
  - Verwenden Sie DetachPolicymit einem AWS SDK oder CLI
  - Verwenden Sie ListAccountsmit einem AWS SDK oder CLI

- Verwenden Sie ListOrganizationalUnitsForParentmit einem AWS SDK oder CLI
- Verwenden Sie ListPoliciesmit einem AWS SDK oder CLI

Aktionen für Organizations, die AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Organisationsaktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Organizations-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Organizations, die AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Organizations -API-Referenz.

Beispiele

- Verwenden Sie mit AttachPolicyeinem AWS SDK oder CLI
- Verwenden Sie CreateAccountmit einem AWS SDK oder CLI
- Verwenden Sie CreateOrganizationmit einem AWS SDK oder CLI
- Verwenden Sie CreateOrganizationalUnitmit einem AWS SDK oder CLI
- Verwenden Sie CreatePolicymit einem AWS SDK oder CLI
- Verwenden Sie DeleteOrganizationmit einem AWS SDK oder CLI
- Verwenden Sie DeleteOrganizationalUnitmit einem AWS SDK oder CLI
- Verwenden Sie DeletePolicymit einem AWS SDK oder CLI
- Verwenden Sie DescribePolicymit einem AWS SDK oder CLI
- Verwenden Sie DetachPolicymit einem AWS SDK oder CLI
- Verwenden Sie ListAccountsmit einem AWS SDK oder CLI
- Verwenden Sie ListOrganizationalUnitsForParentmit einem AWS SDK oder CLI
- Verwenden Sie ListPoliciesmit einem AWS SDK oder CLI

Verwenden Sie mit **`AttachPolicy`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AttachPolicy` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new AttachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };

        var response = await client.AttachPolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
 Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie AttachPolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So hängen Sie eine Richtlinie an ein Stammverzeichnis, eine Organisationsrichtlinie oder ein Konto an

Beispiel 1

Das folgende Beispiel zeigt, wie Sie eine Service-Kontrollrichtlinie (SCP) an eine Organisationseinheit anfügen:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id ou-examplerootid111-exampleouid111
```

Beispiel 2

Das folgende Beispiel zeigt, wie Sie eine Service-Kontrollrichtlinie direkt an ein Konto anfügen:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id 333333333333
```

- Einzelheiten zur API finden Sie AttachPolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account,
 or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- Einzelheiten zur API finden Sie AttachPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_org->attachpolicy(
      iv_policyid = iv_policy_id
      iv_targetid = iv_target_id ).
    MESSAGE 'Policy attached to target.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to attach the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgtargetnotfoundex.
    MESSAGE 'The specified target does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgduplicateplyatta00.
    MESSAGE 'The policy is already attached to the target.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie AttachPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateAccount`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAccount` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations account.
/// </summary>
public class CreateAccount
{
    /// <summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
        };

        var response = await client.CreateAccountAsync(request);
        var status = response.CreateAccountStatus;

        Console.WriteLine($"The staus of {status.AccountName} is
{status.State}.");
    }
```

```
    }
```

- Einzelheiten zur API finden Sie CreateAccountin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Mitgliedskonto, das automatisch Teil Ihrer Organisation ist

Das folgende Beispiel zeigt, wie ein Mitgliedskonto in einer Organisation erstellt wird. Das Mitgliedskonto ist mit dem Namen Production Account und der E-Mail-Adresse susan@example.com konfiguriert. Organizations erstellt automatisch eine IAM-Rolle mit dem Standardnamen von, OrganizationAccountAccessRole da der RoleName-Parameter nicht angegeben ist. Außerdem ist die Einstellung, die IAM-Benutzern oder -Rollen mit ausreichenden Berechtigungen den Zugriff auf Kontoabrechnungsdaten ermöglicht, auf den Standardwert ALLOW gesetzt, da der IamUserAccessToBilling Parameter nicht angegeben ist. Organizations sendet Susan automatisch eine „Willkommen bei AWS“ -E-Mail:

```
aws organizations create-account --email susan@example.com --account-
name "Production Account"
```

Die Ausgabe umfasst ein Anforderungsobjekt, aus dem hervorgeht, dass der Status jetzt `IN_PROGRESS` lautet:

```
{
        "CreateAccountStatus": {
                "State": "IN_PROGRESS",
                "Id": "car-examplecreateaccountrequestid111"
        }
}
```

Sie können später den aktuellen Status der Anforderung abfragen, indem Sie den ID-Antwortwert für den Befehl describe-create-account-status als Wert für den Parameter create-account-request-id angeben.

Weitere Informationen finden Sie unter Erstellen eines AWS Kontos in Ihrer Organisation im Benutzerhandbuch für AWS Organizations.

- Einzelheiten zur API finden Sie CreateAccountunter AWS CLI Befehlsreferenz.

Verwenden Sie **`CreateOrganization`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOrganization` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;
```

```
            Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
 {newOrg.MasterAccountId}");
        }
    }
```

- Einzelheiten zur API finden Sie CreateOrganizationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine neue Organisation

Bill möchte eine Organisation mit den Anmeldeinformationen des Kontos 111111111111 erstellen. Das folgende Beispiel zeigt, dass das Konto zum Hauptkonto in der neuen Organisation wird. Da er keinen Funktionsumfang angibt, sind in der neuen Organisation standardmäßig alle Funktionen aktiviert, und die Richtlinien zur Servicekontrollrichtlinien sind im Stammverzeichnis aktiviert.

```
aws organizations create-organization
```

Die Ausgabe umfasst ein Organisationsobjekt mit Details zur neuen Organisation:

```
{
        "Organization": {
                "AvailablePolicyTypes": [
                        {
                                "Status": "ENABLED",
                                "Type": "SERVICE_CONTROL_POLICY"
                        }
                ],
                "MasterAccountId": "111111111111",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "FeatureSet": "ALL",
                "Id": "o-exampleorgid",
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid"
```

```
        }
}
```

Beispiel 2: So erstellen Sie eine neue Organisation, für die nur konsolidierte Rechnungsstellungsfunktionen aktiviert sind

Im folgenden Beispiel wird eine Organisation erstellt, die nur konsolidierte Rechnungsstellungsfunktionen unterstützt:

```
aws organizations create-organization --feature-set CONSOLIDATED_BILLING
```

Die Ausgabe umfasst ein Organisationsobjekt mit Details zur neuen Organisation:

```
{
        "Organization": {
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
                "AvailablePolicyTypes": [],
                "Id": "o-exampleorgid",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "MasterAccountId": "111111111111",
                "FeatureSet": "CONSOLIDATED_BILLING"
        }
}
```

Weitere Informationen finden Sie unter Erstellen einer Organisation im Benutzerhandbuch für AWS Organizations.

- Einzelheiten zur API finden Sie CreateOrganizationin der AWS CLI Befehlsreferenz.

Verwenden Sie **`CreateOrganizationalUnit`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOrganizationalUnit` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };

        var response = await client.CreateOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
                Console.WriteLine($"Successfully created organizational unit:
 {orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
 {response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateOrganizationalUnitin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Organisationseinheit in einer Stamm- oder übergeordneten Organisationseinheit

Das folgende Beispiel zeigt, wie eine Organisationseinheit mit dem Namen AccountingOU erstellt wird:

```
aws organizations create-organizational-unit --parent-id r-examplerootid111 --
name AccountingOU
```

Die Ausgabe umfasst ein organizationalUnit-Objekt mit Details zur neuen Organisationseinheit:

```
{
        "OrganizationalUnit": {
                "Id": "ou-examplerootid111-exampleouid111",
                "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
examplerootid111-exampleouid111",
                "Name": "AccountingOU",
                "Path": "o-exampleorgid/r-examplerootid111/ou-examplerootid111-
exampleouid111/"
```

```
        }
}
```

- Einzelheiten zur API finden Sie CreateOrganizationalUnitin der AWS CLI Befehlsreferenz.

Verwenden Sie **`CreatePolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePolicy` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations Policy.
/// </summary>
public class CreatePolicy
{
    /// <summary>
    /// Initializes the AWS Organizations client object, uses it to
    /// create a new Organizations Policy, and then displays information
    /// about the newly created Policy.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyContent = "{" +
            "   \"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:*\"]," +
```

```
                    " \"Effect\" : \"Allow\"," +
                    " \"Resource\" : \"*\"" +
                "}]" +
            "}";

            try
            {
                var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
                {
                    Content = policyContent,
                    Description = "Enables admins of attached accounts to
delegate all Amazon S3 permissions",
                    Name = "AllowAllS3Actions",
                    Type = "SERVICE_CONTROL_POLICY",
                });

                Policy policy = response.Policy;
                Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Richtlinie mit einer Textquelldatei für die JSON-Richtlinie

Das folgende Beispiel zeigt, wie Sie eine Service-Kontrollrichtlinie (SCP) namens `AllowAllS3Actions` erstellen: Der Richtlinieninhalt stammt aus einer Datei auf dem lokalen Computer namens `policy.json`.

```
aws organizations create-policy --content file://policy.json --
name AllowAllS3Actions, --type SERVICE_CONTROL_POLICY --description "Allows
 delegation of all S3 actions"
```

Die Ausgabe umfasst ein Richtlinienobjekt mit Details zur neuen Richtlinie:

```
{
        "Policy": {
                "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource\":[\"*\"]}]}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::o-exampleorgid:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Allows delegation of all S3 actions",
                        "Name": "AllowAllS3Actions",
                        "Type":"SERVICE_CONTROL_POLICY"
                }
        }
}
```

Beispiel 2: So erstellen Sie eine Richtlinie mit einer JSON-Richtlinie als Parameter

Das folgende Beispiel zeigt, wie Sie dasselbe SCP erstellen, diesmal indem Sie den Richtlinieninhalt als JSON-Zeichenfolge in den Parameter einbetten. Die Zeichenfolge muss mit Backslashes vor den doppelten Anführungszeichen maskiert werden, um sicherzustellen, dass sie im Parameter, der selbst von doppelten Anführungszeichen umgeben ist, als Literale behandelt werden:

```
aws organizations create-policy --content "{\"Version\":\"2012-10-17\",
\"Statement\":[{\"Effect\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource
\":[\"*\"]}]}" --name AllowAllS3Actions --type SERVICE_CONTROL_POLICY --
description "Allows delegation of all S3 actions"
```

Weitere Informationen zum Erstellen und Verwenden von Richtlinien in Ihrer Organisation finden Sie unter „Verwalten von Organisationsrichtlinien“ im Benutzerhandbuch für AWS Organizations.

- Einzelheiten zur API finden Sie CreatePolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON
 before
                    it is sent to AWS. The specific format depends on the policy
 type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
    :return: The newly created policy.
    """
    try:
        response = orgs_client.create_policy(
            Name=name,
            Description=description,
            Content=json.dumps(content),
            Type=policy_type,
        )
        policy = response["Policy"]
        logger.info("Created policy %s.", name)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- Einzelheiten zur API finden Sie CreatePolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_org->createpolicy(       " oo_result is returned for
testing purposes. "
         iv_name        = iv_policy_name
         iv_description = iv_policy_description
         iv_content     = iv_policy_content
         iv_type        = iv_policy_type ).
       MESSAGE 'Policy created.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to create a policy.' TYPE 'E'.
     CATCH /aws1/cx_orgduplicatepolicyex.
       MESSAGE 'A policy with this name already exists.' TYPE 'E'.
     CATCH /aws1/cx_orgmalformedplydocex.
       MESSAGE 'The policy content is malformed.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreatePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteOrganization`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteOrganization` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully deleted organization.");
        }
        else
        {
            Console.WriteLine("Could not delete organization.");
        }
    }
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteOrganizationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Organisation zu löschen

Im folgenden Beispiel wird gezeigt, wie eine Organisation gelöscht wird. Um diesen Vorgang ausführen zu können, müssen Sie Administrator des Hauptkontos in der Organisation sein. Im Beispiel wird davon ausgegangen, dass Sie zuvor alle Mitgliedskonten, Organisationseinheiten und Richtlinien aus der Organisation entfernt haben:

```
aws organizations delete-organization
```

- Einzelheiten zur API finden Sie DeleteOrganizationin der AWS CLI Befehlsreferenz.

Verwenden Sie **`DeleteOrganizationalUnit`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteOrganizationalUnit` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;
```

```
    /// <summary>
    /// Shows how to delete an existing AWS Organizations organizational unit.
    /// </summary>
    public class DeleteOrganizationalUnit
    {
        /// <summary>
        /// Initializes the Organizations client object and calls
        /// DeleteOrganizationalUnitAsync to delete the organizational unit
        /// with the selected ID.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var orgUnitId = "ou-0000-00000000";

            var request = new DeleteOrganizationalUnitRequest
            {
                OrganizationalUnitId = orgUnitId,
            };

            var response = await client.DeleteOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DeleteOrganizationalUnitin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um eine Organisationseinheit zu löschen

Im folgenden Beispiel wird gezeigt, wie eine Organisationseinheit gelöscht wird. Im Beispiel wird davon ausgegangen, dass Sie zuvor alle Konten und andere Organisationseinheiten aus der Organisationseinheit entfernt haben:

```
aws organizations delete-organizational-unit --organizational-unit-id ou-
examplerootid111-exampleouid111
```

- Einzelheiten zur API finden Sie DeleteOrganizationalUnitin der AWS CLI Befehlsreferenz.

Verwenden Sie **`DeletePolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePolicy` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
```

```
        /// delete the policy with the specified policyId.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var policyId = "p-00000000";

            var request = new DeletePolicyRequest
            {
                PolicyId = policyId,
            };

            var response = await client.DeletePolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted Policy: {policyId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie eine Richtlinie

Das folgende Codebeispiel zeigt, wie Sie eine Richtlinie aus einer Organisation löschen. Im Beispiel wird davon ausgegangen, dass Sie die Richtlinie zuvor von allen Entitäten getrennt haben:

```
aws organizations delete-policy --policy-id p-examplepolicyid111
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- Einzelheiten zur API finden Sie DeletePolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_org->deletepolicy(
          iv_policyid = iv_policy_id ).
        MESSAGE 'Policy deleted.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to delete the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicyinuseex.
        MESSAGE 'The policy is still attached to one or more targets.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeletePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribePolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePolicy` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Richtlinie ab

Das folgende Beispiel zeigt, wie Sie Informationen zu einer Richtlinie anfordern:

```
aws organizations describe-policy --policy-id p-examplepolicyid111
```

Die Ausgabe umfasst ein Richtlinienobjekt, das Details zur Richtlinie enthält:

```
{
        "Policy": {
                "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n
 \"Resource\": \"*\"\n    }\n  ]\n}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::111111111111:policy/o-
exampleorgid/service_control_policy/p-examplepolicyid111",
                        "Type": "SERVICE_CONTROL_POLICY",
```

```
                        "Id": "p-examplepolicyid111",
                        "AwsManaged": false,
                        "Name": "AllowAllS3Actions",
                        "Description": "Enables admins to delegate S3
 permissions"
                }
        }
}
```

- Einzelheiten zur API finden Sie DescribePolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- Einzelheiten zur API finden Sie DescribePolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_org->describepolicy(     " oo_result is returned for
testing purposes. "
         iv_policyid = iv_policy_id ).
       DATA(lo_policy) = oo_result->get_policy( ).
       MESSAGE 'Retrieved policy details.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to describe the policy.' TYPE 'E'.
     CATCH /aws1/cx_orgpolicynotfoundex.
       MESSAGE 'The specified policy does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribePolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetachPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetachPolicy` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new DetachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };

        var response = await client.DetachPolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
 {policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetachPolicyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So trennen Sie eine Richtlinie von einem Root, einer Organisationseinheit oder einem Konto

Das folgende Codebeispiel zeigt, wie Sie eine Richtlinie von einer Organisationseinheit trennen.

```
aws organizations  detach-policy  --target-id ou-examplerootid111-exampleouid111
 --policy-id p-examplepolicyid111
```

- Einzelheiten zur API finden Sie DetachPolicyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def detach_policy(policy_id, target_id, orgs_client):
```

```
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently
attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- Einzelheiten zur API finden Sie DetachPolicyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_org->detachpolicy(
          iv_policyid = iv_policy_id
          iv_targetid = iv_target_id ).
        MESSAGE 'Policy detached from target.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to detach the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgtargetnotfoundex.
```

```
      MESSAGE 'The specified target does not exist.' TYPE 'E'.
    CATCH /aws1/cx_orgpolicynotattex.
      MESSAGE 'The policy is not attached to the target.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie DetachPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAccounts`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListAccounts` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account.
/// </summary>
public class ListAccounts
{
    /// <summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
```

```
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var request = new ListAccountsRequest
            {
                MaxResults = 5,
            };

            var response = new ListAccountsResponse();
            try
            {
                do
                {
                    response = await client.ListAccountsAsync(request);
                    response.Accounts.ForEach(a => DisplayAccounts(a));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations account.
        /// </summary>
        /// <param name="account">An Organizations account for which to display
        /// information on the console.</param>
        private static void DisplayAccounts(Account account)
        {
            string accountInfo = $"{account.Id}
{account.Name}\t{account.Status}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Einzelheiten zur API finden Sie ListAccountsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste aller Konten in einer Organisation ab

Das folgende Beispiel zeigt, wie Sie eine Liste der Konten in einer Organisation anfordern:

```
aws organizations list-accounts
```

Die Ausgabe enthält eine Liste von Kontozusammenfassungsobjekten.

```
{
        "Accounts": [
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481830215.45,
                        "Id": "111111111111",
                        "Name": "Master Account",
                        "Email": "bill@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/222222222222",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835741.044,
                        "Id": "222222222222",
                        "Name": "Production Account",
                        "Email": "alice@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
```

```
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835795.536,
                        "Id": "333333333333",
                        "Name": "Development Account",
                        "Email": "juan@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/444444444444",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835812.143,
                        "Id": "444444444444",
                        "Name": "Test Account",
                        "Email": "anika@example.com",
                        "Paths": [
                                "o-exampleorgid/r-examplerootid111/111111111111/"
                        ],
                        "State": "ACTIVE",
                        "Status": "ACTIVE"
                }
        ]
}
```

- Einzelheiten zur API finden Sie ListAccountsin der AWS CLI Befehlsreferenz.

Verwenden Sie **`ListOrganizationalUnitsForParent`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListOrganizationalUnitsForParent` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Lists the AWS Organizations organizational units that belong to an
/// organization.
/// </summary>
public class ListOrganizationalUnitsForParent
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// call the ListOrganizationalUnitsForParentAsync method to retrieve
    /// the list of organizational units.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var parentId = "r-0000";

        var request = new ListOrganizationalUnitsForParentRequest
        {
            ParentId = parentId,
            MaxResults = 5,
        };

        var response = new ListOrganizationalUnitsForParentResponse();
        try
        {
```

```
                do
                {
                    response = await
client.ListOrganizationalUnitsForParentAsync(request);
                    response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations organizational unit.
        /// </summary>
        /// <param name="unit">The OrganizationalUnit for which to display
        /// information.</param>
        public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
        {
            string accountInfo = $"{unit.Id} {unit.Name}\t{unit.Arn}";

            Console.WriteLine(accountInfo);
        }
    }
```

- Einzelheiten zur API finden Sie ListOrganizationalUnitsForParentin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste der OUs in einer übergeordneten OU oder im Stammverzeichnis ab

Das folgende Beispiel zeigt, wie Sie eine Liste von OUs in einem angegebenen Root abrufen:

```
aws organizations list-organizational-units-for-parent --parent-id r-
examplerootid111
```

Die Ausgabe zeigt, dass das angegebene Stammverzeichnis zwei Organisationseinheiten enthält, und es werden Details zu jeder Organisationseinheit angezeigt:

```
{
        "OrganizationalUnits": [
                {
                        "Name": "AccountingDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid111",
                        "Id": "ou-examplerootid111-exampleouid111",
                        "Path": "o-exampleorgid/r-examplerootid111/ou-
examplerootid111-exampleouid111/"
                },
                {
                        "Name": "ProductionDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid222",
                        "Id": "ou-examplerootid111-exampleouid222",
                        "Path": "o-exampleorgid/r-examplerootid111/ou-
examplerootid111-exampleouid222/"
                }
        ]
}
```

- Einzelheiten zur API finden Sie ListOrganizationalUnitsForParentin der AWS CLI Befehlsreferenz.

Verwenden Sie **`ListPolicies`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListPolicies` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        // The value for the Filter parameter is required and must must be
        // one of the following:
        //     AISERVICES_OPT_OUT_POLICY
        //     BACKUP_POLICY
        //     SERVICE_CONTROL_POLICY
        //     TAG_POLICY
        var request = new ListPoliciesRequest
        {
            Filter = "SERVICE_CONTROL_POLICY",
            MaxResults = 5,
        };
```

```
            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id}
{policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
    }
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste aller Richtlinien in einer Organisation eines bestimmten Typs ab

Das folgende Beispiel zeigt, wie Sie eine Liste von SCPs abrufen, wie sie im Filter-Parameter angegeben wird:

```
aws organizations list-policies --filter SERVICE_CONTROL_POLICY
```

Die Ausgabe umfasst eine Liste von Richtlinien mit zusammenfassenden Informationen:

```
{
        "Policies": [
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllS3Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid111",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Enables account admins to delegate
 permissions for any S3 actions to users and roles in their accounts."
                },
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllEC2Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid222",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid222",
                        "Description": "Enables account admins to delegate
 permissions for any EC2 actions to users and roles in their accounts."
                },
                {
                        "AwsManaged": true,
                        "Description": "Allows access to every operation",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-FullAWSAccess",
                        "Arn": "arn:aws:organizations::aws:policy/
service_control_policy/p-FullAWSAccess",
                        "Name": "FullAWSAccess"
```

```
                }
        ]
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- Einzelheiten zur API finden Sie ListPoliciesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_org->listpolicies(       " oo_result is returned for
testing purposes. "
         iv_filter = iv_filter ).
       DATA(lt_policies) = oo_result->get_policies( ).
       MESSAGE 'Retrieved list of policies.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to list policies.' TYPE 'E'.
     CATCH /aws1/cx_orgawsorgsnotinuseex.
       MESSAGE 'Your account is not a member of an organization.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ListPoliciesin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für Organizations, die AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Organizations mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von Organizations aufrufen oder mit anderen kombinieren AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen
- Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation
- Richtlinie zur Aktivierung der Automatisierung für Ihr Konto
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation
- Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten
- Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation

Erlaubt die AWS Compute Optimizer Automatisierungsfunktion zur Anwendung empfohlener Aktionen

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie ermöglicht es der AWS Compute Optimizer Automatisierungsfunktion, empfohlene Aktionen anzuwenden

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "aco-automation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Richtlinie zur Aktivierung der Automatisierung in Ihrer gesamten Organisation

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie die Automatisierung in Ihrer gesamten Organisation aktivieren können

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:AssociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:DisassociateAccounts",
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "aco-automation:ListAccounts",
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Aktivierung der Automatisierung für Ihr Konto

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie die Automatisierung für Ihr Konto aktivieren

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie gewährt einem Verwaltungskonto einer Organisation vollen Zugriff auf Compute Optimizer Automation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
               "ec2:DescribeVolumes",
               "organizations:ListAccounts",
               "organizations:DescribeOrganization",
               "organizations:DescribeAccount",
               "organizations:EnableAWSServiceAccess",
               "organizations:ListDelegatedAdministrators",
               "organizations:RegisterDelegatedAdministrator",
               "organizations:DeregisterDelegatedAdministrator"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung des vollen Zugriffs auf Compute Optimizer Automation for Standalone AWS Konten

Das folgende Codebeispiel zeigt, wie Sie mit dieser berechtigungsbasierten Richtlinie eigenständigen Konten vollen Zugriff auf Compute Optimizer Automation gewähren AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
            "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation für ein Verwaltungskonto einer Organisation

Das folgende Codebeispiel zeigt, wie: Diese berechtigungsbasierte Richtlinie gewährt einem Verwaltungskonto einer Organisation schreibgeschützten Zugriff auf Compute Optimizer Automation.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAccounts",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
```

```
                "aco-automation:ListRecommendedActions",
                "aco-automation:ListRecommendedActionSummaries",
                "aco-automation:ListTagsForResource",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

Richtlinie zur Gewährung von schreibgeschütztem Zugriff auf Compute Optimizer Automation for Standalone AWS Konten

Das folgende Codebeispiel zeigt, wie das geht: Diese berechtigungsbasierte Richtlinie gewährt eigenständigen Konten schreibgeschützten Zugriff auf Compute Optimizer Automation AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aco-automation:GetEnrollmentConfiguration",
                "aco-automation:GetAutomationEvent",
                "aco-automation:GetAutomationRule",
                "aco-automation:ListAutomationEvents",
                "aco-automation:ListAutomationEventSteps",
                "aco-automation:ListAutomationEventSummaries",
                "aco-automation:ListAutomationRules",
                "aco-automation:ListAutomationRulePreview",
                "aco-automation:ListAutomationRulePreviewSummaries",
                "aco-automation:ListRecommendedActions",
                "aco-automation:ListRecommendedActionSummaries",
                "aco-automation:ListTagsForResource",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Richtlinie zur Gewährung von dienstbezogenen Rollenberechtigungen für Compute Optimization Automation

Das folgende Codebeispiel zeigt, wie diese berechtigungsbasierte Richtlinie dienstbezogene Rollenberechtigungen für Compute Optimization Automation gewährt

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        }
    ]
}
```

Codebeispiele für die Verwendung von Partner Central AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS Partner Central mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Erste Schritte mit Partner Central – Weitere Informationen zu Partner Central.
- API-Referenz für Partner Central – Details zu allen verfügbaren Partner-Central-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Partner Central AWS SDKs
  - Aktionen für Partner Central unter Verwendung AWS SDKs
    - Verwenden Sie AssignOpportunitymit einem AWS SDK
    - Verwenden Sie mit AssociateOpportunityeinem AWS SDK
    - Verwenden Sie mit CreateOpportunityeinem AWS SDK
    - Verwenden Sie mit DisassociateOpportunityeinem AWS SDK
    - Verwenden Sie mit GetAwsOpportunitySummaryeinem AWS SDK
    - Verwenden Sie mit GetEngagementInvitationeinem AWS SDK
    - Verwenden Sie mit GetOpportunityeinem AWS SDK
    - Verwenden Sie mit ListEngagementInvitationseinem AWS SDK
    - Verwenden Sie mit ListOpportunitieseinem AWS SDK
    - Verwenden Sie mit ListSolutionseinem AWS SDK
    - Verwenden Sie mit RejectEngagementInvitationeinem AWS SDK
    - Verwenden Sie mit StartEngagementByAcceptingInvitationTaskeinem AWS SDK
    - Verwenden Sie mit StartEngagementFromOpportunityTaskeinem AWS SDK
    - Verwenden Sie mit UpdateOpportunityeinem AWS SDK
- Szenarien für die Verwendung von Partner Central AWS SDKs
  - Aktualisieren der zugehörigen Entität einer Opportunity

Grundlegende Beispiele für die Verwendung von Partner Central AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von AWS Partner Central mit AWS SDKs verwenden können.

Beispiele

- Aktionen für Partner Central unter Verwendung AWS SDKs
  - Verwenden Sie AssignOpportunitymit einem AWS SDK
  - Verwenden Sie mit AssociateOpportunityeinem AWS SDK
  - Verwenden Sie mit CreateOpportunityeinem AWS SDK
  - Verwenden Sie mit DisassociateOpportunityeinem AWS SDK
  - Verwenden Sie mit GetAwsOpportunitySummaryeinem AWS SDK
  - Verwenden Sie mit GetEngagementInvitationeinem AWS SDK
  - Verwenden Sie mit GetOpportunityeinem AWS SDK
  - Verwenden Sie mit ListEngagementInvitationseinem AWS SDK
  - Verwenden Sie mit ListOpportunitieseinem AWS SDK
  - Verwenden Sie mit ListSolutionseinem AWS SDK
  - Verwenden Sie mit RejectEngagementInvitationeinem AWS SDK
  - Verwenden Sie mit StartEngagementByAcceptingInvitationTaskeinem AWS SDK
  - Verwenden Sie mit StartEngagementFromOpportunityTaskeinem AWS SDK
  - Verwenden Sie mit UpdateOpportunityeinem AWS SDK

Aktionen für Partner Central unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Partner Central-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Partner-Central-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Partner Central AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für AWS -Partner-Central.

Beispiele

- Verwenden Sie AssignOpportunitymit einem AWS SDK
- Verwenden Sie mit AssociateOpportunityeinem AWS SDK
- Verwenden Sie mit CreateOpportunityeinem AWS SDK
- Verwenden Sie mit DisassociateOpportunityeinem AWS SDK
- Verwenden Sie mit GetAwsOpportunitySummaryeinem AWS SDK
- Verwenden Sie mit GetEngagementInvitationeinem AWS SDK
- Verwenden Sie mit GetOpportunityeinem AWS SDK
- Verwenden Sie mit ListEngagementInvitationseinem AWS SDK
- Verwenden Sie mit ListOpportunitieseinem AWS SDK
- Verwenden Sie mit ListSolutionseinem AWS SDK
- Verwenden Sie mit RejectEngagementInvitationeinem AWS SDK
- Verwenden Sie mit StartEngagementByAcceptingInvitationTaskeinem AWS SDK
- Verwenden Sie mit StartEngagementFromOpportunityTaskeinem AWS SDK
- Verwenden Sie mit UpdateOpportunityeinem AWS SDK

Verwenden Sie **`AssignOpportunity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `AssignOpportunity` verwendet wird.

Java

SDK für Java 2.x

Weisen Sie eine bestehende Opportunity einem anderen Benutzer neu zu.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {
```

```
  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
          .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie AssignOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Weisen Sie eine bestehende Opportunity einem anderen Benutzer neu zu.

```
#!/usr/bin/env python

"""
Purpose
PC-API-07 Assigning a new owner
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE
```

```
serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def assign_opportunity(identifier):
    assign_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "Assignee": {
            "BusinessTitle": "OpportunityOwner",
            "Email": "test@test.com",
            "FirstName": "John",
            "LastName": "Doe"
        }
    }
    try:
        # Perform an API call
        response =
 partner_central_client.assign_opportunity(**assign_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O4236468"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Assigning a new owner to an opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(assign_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie AssignOpportunityin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`AssociateOpportunity`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `AssociateOpportunity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Aktualisieren der zugehörigen Entität einer Opportunity

Java

SDK für Java 2.x

Erstellen Sie eine formelle Zuordnung zwischen einer Opportunity und verschiedenen verwandten Entitäten.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
```

```
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie AssociateOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Erstellen Sie eine formelle Zuordnung zwischen einer Opportunity und verschiedenen verwandten Entitäten.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def associate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0059717"
    opportunityIdentifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Associate Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(associate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie AssociateOpportunityin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateOpportunity`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateOpportunity` verwendet wird.

.NET

SDK für .NET

Erstellen Sie eine Opportunity.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0
```

```
using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new CreateOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Origin = "Partner Referral",
                    Customer = new Customer
                    {
                        Account = new Account
                        {
                            Address = new Address
                            {
                                CountryCode = "US",
                                PostalCode = "99502",
                                StateOrRegion = "Alaska"
                            },
                            CompanyName = "TestCompanyName",
                            Duns = "123456789",
                            WebsiteUrl = "www.test.io",
                            Industry = "Automotive"
                        },
```

```
                        Contacts = new List<Contact>
                        {
                            new Contact
                            {
                                Email = "test@test.io",
                                FirstName = "John  ",
                                LastName = "Doe",
                                Phone = "+14444444444",
                                BusinessTitle = "test title"
                            }
                        }
                    },
                    LifeCycle = new LifeCycle
                    {
                        ReviewStatus = "Submitted",
                        TargetCloseDate = "2024-12-30"
                    },
                    Marketing = new Marketing
                    {
                        Source = "None"
                    },
                    OpportunityType = "Net New Business",
                    PrimaryNeedsFromAws = new List<string> { "Co-Sell -
Architectural Validation" },
                    Project = new Project
                    {
                        Title = "Moin Test UUID",
                        CustomerBusinessProblem = "Sandbox is not working as
expected",
                        CustomerUseCase = "AI Machine Learning and Analytics",
                        DeliveryModels = new List<string> { "SaaS or PaaS" },
                        ExpectedCustomerSpend = new List<ExpectedCustomerSpend>
                        {
                            new ExpectedCustomerSpend
                            {
                                Amount = "2000.0",
                                CurrencyCode = "USD",
                                Frequency = "Monthly",
                                TargetCompany = "Ibexlabs"
                            }
                        },
                        SalesActivities = new List<string> { "Initialized
discussions with customer" }
                    }
```

```
                };

                try
                {
                    var response = await client.CreateOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                }
                catch (ValidationException ex)
                {
                    Console.WriteLine("Validation error: " + ex.Message);
                }
                catch (AmazonPartnerCentralSellingException e)
                {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateOpportunityin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

Erstellen Sie eine Opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;
```

```
import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import
 software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
```

```
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

public static void main(String[] args) {

 String inputFile = "CreateOpportunity2.json";

 if (args.length > 0)
  inputFile = args[0];

 CreateOpportunityResponse response = createOpportunity(inputFile);

 client.close();
}

static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
```

```
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

  }

  Address address = null;
  if ( root.customer != null && root.customer.account != null &&
 root.customer.account.address != null ) {
   address = Address.builder()
    .city(root.customer.account.address.city)
                .postalCode(root.customer.account.address.postalCode)
                .stateOrRegion(root.customer.account.address.stateOrRegion)
                .countryCode(root.customer.account.address.countryCode)
                .streetAddress(root.customer.account.address.streetAddress)
                .build();
  }

  Account account = null;
  if ( root.customer != null && root.customer.account!= null) {
   account = Account.builder()
             .address(address)
             .awsAccountId(root.customer.account.awsAccountId)
                .duns(root.customer.account.duns)
                .industry(root.customer.account.industry)
                .otherIndustry(root.customer.account.otherIndustry)
                .companyName(root.customer.account.companyName)
                .websiteUrl(root.customer.account.websiteUrl)
                .build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
```

```
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
   Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
   contacts.add(contact);
  }
 }

 Customer customer = Customer.builder()
   .account(account)
   .contacts(contacts)
   .build();

 Contact oportunityTeamContact = null;
 if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
  oportunityTeamContact = Contact.builder()
               .firstName(root.opportunityTeam.get(0).firstName)
               .lastName(root.opportunityTeam.get(0).lastName)
               .email(root.opportunityTeam.get(0).email)
               .phone(root.opportunityTeam.get(0).phone)
               .businessTitle(root.opportunityTeam.get(0).businessTitle)
               .build();
 }

 List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
 if ( root.project != null && root.project.expectedCustomerSpend != null) {
  for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
   ExpectedCustomerSpend expectedCustomerSpend = null;
   expectedCustomerSpend = ExpectedCustomerSpend.builder()
     .amount(expectedCustomerSpendJson.amount)
     .currencyCode(expectedCustomerSpendJson.currencyCode)
     .frequency(expectedCustomerSpendJson.frequency)
     .targetCompany(expectedCustomerSpendJson.targetCompany)
     .build();
   expectedCustomerSpends.add(expectedCustomerSpend);
  }
       }
```

```
        Project project = null;
        if ( root.project != null) {
         project = Project.builder()
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
            .amount(root.softwareRevenue.value.amount)
            .currencyCode(root.softwareRevenue.value.currencyCode)
            .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

 // Building the Actual CreateOpportunity Request
 CreateOpportunityRequest createOpportunityRequest =
CreateOpportunityRequest.builder()
   .catalog(CATALOG_TO_USE)
   .clientToken(root.clientToken)
   .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
   .opportunityType(root.opportunityType)
   .lifeCycle(lifeCycle)
   .marketing(marketing)
   .nationalSecurity(root.nationalSecurity)
   .origin(root.origin)
   .customer(customer)
   .project(project)
```

```
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- Einzelheiten zur API finden Sie CreateOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Erstellen Sie eine Opportunity.

```
#!/usr/bin/env python
import boto3
import logging
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
import utils.stringify_details as sd
from botocore.client import ClientError
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

def create_opportunity(partner_central_client):
    create_opportunity_request = helper.remove_nulls(sd.stringify_json("src/
create_opportunity/createOpportunity.json"))
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.create_opportunity(**create_opportunity_request)

        helper.pretty_print_datetime(response)

        # Retrieve the opportunity details
        get_response = partner_central_client.get_opportunity(
            Identifier=response["Id"],
            Catalog=CATALOG_TO_USE
        )
        helper.pretty_print_datetime(get_response)
        return response
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Create Opportunity.")
    print("-" * 88)

    partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
    )

    create_opportunity(partner_central_client)

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie CreateOpportunityin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DisassociateOpportunity`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DisassociateOpportunity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Aktualisieren der zugehörigen Entität einer Opportunity

Java

SDK für Java 2.x

Entfernen Sie eine bestehende Verknüpfung zwischen einer Opportunity und verwandten Entitäten.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie DisassociateOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Entfernen Sie eine bestehende Verknüpfung zwischen einer Opportunity und verwandten Entitäten.

```
#!/usr/bin/env python

"""
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def disassociate_opportunity(entity_type, entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
```

```
    entity_type = "Solutions"
    entity_identifier = "S-0049999"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(disassociate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie DisassociateOpportunityin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetAwsOpportunitySummary`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetAwsOpportunitySummary` verwendet wird.

Java

SDK für Java 2.x

Ruft eine Zusammenfassung einer AWS Opportunity ab.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequ
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryResp

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId)
 {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie GetAwsOpportunitySummaryunter AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Ruft eine Zusammenfassung einer AWS Opportunity ab.

```
#!/usr/bin/env python

"""
Purpose
PC-API-25 Retrieves a summary of an AWS Opportunity.
 LifeCycle.ReviewStatus=Approved
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "RelatedOpportunityIdentifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_aws_opportunity_summary(**get_opportunity_request)
        return response
```

```
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "05465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get AWS Opportunity summary.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetAwsOpportunitySummaryin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetEngagementInvitation`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetEngagementInvitation` verwendet wird.

Java

SDK für Java 2.x

Ruft die Details einer Einladung zur Interaktion ab, die von AWS einem Partner geteilt wurde.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie GetEngagementInvitationunter AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Ruft die Details einer Einladung zur Interaktion ab, die von AWS einem Partner geteilt wurde.

```
#!/usr/bin/env python

"""
Purpose
PC-API-22  GetOpportunityEngagementInvitation - Retrieves details of a specific
 engagement invitation.
This operation allows partners to view the invitation and its associated
 information,
such as the customer, project, and lifecycle details.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity_engagement_invitation(identifier):
    get_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_engagement_invitation
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral-selling:us-east-1:aws:catalog/Sandbox/
engagement-invitation/engi-0000000IS0Qga"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier, retrieve details of Opportunity Engagement
 Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(get_opportunity_engagement_invitation(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetEngagementInvitationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`GetOpportunity`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetOpportunity` verwendet wird.

.NET

SDK für .NET

Rufen Sie eine Opportunity ab.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
```

```
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static readonly string identifier = "O1111111";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new GetOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Identifier = identifier
                };

                try {
                    var response = await client.GetOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
```

```
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie GetOpportunityin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

Rufen Sie eine Opportunity ab.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)
```

```
 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- Einzelheiten zur API finden Sie GetOpportunityin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Rufen Sie eine Opportunity ab.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
```

```
/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie GetOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Rufen Sie eine Opportunity ab.

```
#!/usr/bin/env python

"""
Purpose
PC-API -08 Get updated Opportunity given opportunity id
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie GetOpportunityin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListEngagementInvitations`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListEngagementInvitations` verwendet wird.

Java

SDK für Java 2.x

Ruft eine Liste mit Einladungen zur Interaktion ab, die an den Partner gesendet wurden.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReq
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRes
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();

        return opportunitySummaries;
 }
}
```

- Einzelheiten zur API finden Sie ListEngagementInvitationsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Ruft eine Liste mit Einladungen zur Interaktion ab, die an den Partner gesendet wurden.

```
#!/usr/bin/env python

"""
Purpose
PC-API-21 ListEngagementInvitations - Retrieves a list of engagement invitations
 based on specified criteria.
This operation allows partners to view all invitations to engagement.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def list_engagement_invitations():
    list_engagement_invitations_request ={
        "Catalog": CATALOG_TO_USE,
        "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_engagement_invitations(**list_engagement_invitations_request
        return response
```

```
    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieve list of Engagement Invitations.")
    print("-" * 88)

    helper.pretty_print_datetime(list_engagement_invitations())

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie ListEngagementInvitationsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListOpportunities`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListOpportunities` verwendet wird.

.NET

SDK für .NET

Auflisten von Opportunities.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
```

```
    {
        static readonly string catalogToUse = "Sandbox";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                //var config = new AmazonPartnerCentralSellingConfig()
                //{
                //    ServiceURL = "https://partnercentral-selling.us-
east-1.api.aws",
                //};
                //var client = new
 AmazonPartnerCentralSellingClient(awsCredentials, config);
                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);
                var request = new ListOpportunitiesRequest
                {
                    Catalog = catalogToUse,
                    MaxResults = 2
                };

                try {
                    var response = await client.ListOpportunitiesAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    foreach (var opportunity in response.OpportunitySummaries)
                    {
                        Console.WriteLine("Opportunity id: " + opportunity.Id);
                    }
                    string formattedJson =
 JsonConvert.SerializeObject(response.OpportunitySummaries, Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
```

```
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie ListOpportunitiesin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

Listen Sie Opportunitys auf.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)
```

```
 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- Einzelheiten zur API finden Sie ListOpportunitiesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Listen Sie Opportunitys auf.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;
```

```
/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
     List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

     opportunitySummaries.addAll(response.opportunitySummaries());

     while (response.nextToken() != null && response.nextToken().length() > 0) {
      listOpportunityRequest = ListOpportunitiesRequest.builder()
                    .catalog(CATALOG_TO_USE)
              .maxResults(5)
              .nextToken(response.nextToken())
              .build();
      response = client.listOpportunities(listOpportunityRequest);
      opportunitySummaries.addAll(response.opportunitySummaries());
     }
```

```
     client.close();

        return opportunitySummaries;
 }
}
```

- Einzelheiten zur API finden Sie ListOpportunitiesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Listen Sie Opportunitys auf.

```
#!/usr/bin/env python

"""
Purpose
PC-API -18 Getting list of Opportunities
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_opportunities():

    opportunity_list = []

    list_opportunities_request ={
        "Catalog": CATALOG_TO_USE,
```

```
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_opportunities(**list_opportunities_request)
        opportunity_list.extend(response["OpportunitySummaries"])

        while "NextToken" in response and response["NextToken"] is not None:
            list_opportunities_request["NextToken"] = response["NextToken"]
            response =
 partner_central_client.list_opportunities(**list_opportunities_request)
            opportunity_list.extend(response["OpportunitySummaries"])

        return opportunity_list

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of Opportunities.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_opportunities())

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie ListOpportunitiesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`ListSolutions`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListSolutions` verwendet wird.

Java

SDK für Java 2.x

Ruft eine Liste von Partner Solutions ab, die der Partner bei Partner Central registriert hat.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }
```

```
    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- Einzelheiten zur API finden Sie ListSolutionsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Ruft eine Liste von Partner Solutions ab, die der Partner bei Partner Central registriert hat.

```
#!/usr/bin/env python

"""
Purpose
PC-API-10 Getting list of solutions
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
```

```
        region_name='us-east-1'
)

def get_list_of_solutions():
    list_solutions_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_solutions(**list_solutions_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of solutions.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_solutions())

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie ListSolutionsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`RejectEngagementInvitation`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `RejectEngagementInvitation` verwendet wird.

Java

SDK für Java 2.x

Lehnt jedes AWS geteilte ab. EngagementInvitation

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {
```

```
        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- Einzelheiten zur API finden Sie RejectEngagementInvitationunter AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Lehnt jeden ab EngagementInvitation , der AWS geteilt wurde.

```
#!/usr/bin/env python

"""
Purpose
PC-API-05 AWS Originated AO rejection - RejectOpportunityEngagementInvitation -
 Rejects a engagement invitation.
This action indicates that the partner does not wish to participate in the
 engagement and
provides a reason for the rejection.
Upon rejection, a OpportunityEngagementInvitationRejected event is triggered.
Subsequently, the invitation will no longer be available for the partner to act
 on.
"""
import json
import logging
import boto3
import utils.helpers as helper
```

```
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def reject_opportunity_engagement_invitation(identifier, reject_reason):
    reject_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "RejectionReason": reject_reason
    }
    try:
        # Perform an API call
        response =
 partner_central_client.reject_engagement_invitation(**reject_opportunity_engagement_invi
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isviga"
    reject_reason = "Customer problem unclear"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier and reject reason, reject the Opportunity
 Engagement Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(reject_opportunity_engagement_invitation(identifier,
 reject_reason))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie RejectEngagementInvitationin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`StartEngagementByAcceptingInvitationTask`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartEngagementByAcceptingInvitationTask` verwendet wird.

Java

SDK für Java 2.x

Beginnt das Engagement mit der Annahme eines EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo
import
 software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
```

```
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

     if ( response == null) {
      System.out.println("Opportunity is not AWS Originated.");
     } else {
      ReferenceCodesUtils.formatOutput(response);
     }
    }

    private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {

     GetEngagementInvitationRequest getRequest =
 GetEngagementInvitationRequest.builder()
          .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getRequest);

        return response;
 }

 static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
 invitationId) {
```

```
  if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
   StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
     StartEngagementByAcceptingInvitationTaskRequest.builder()
     .catalog(Constants.CATALOG_TO_USE)
           .identifier(invitationId)
           .clientToken(clientToken)
           .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
         return response;
  }
  return null;
 }
}
```

- Einzelheiten zur API finden Sie StartEngagementByAcceptingInvitationTaskin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Beginnt das Engagement mit der Annahme eines EngagementInvitation.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE
```

```
serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def start_engagement_by_accepting_invitation_task(identifier):

    response = get_opportunity(identifier)

    if ( response['Status'] == 'PENDING') :
        accept_opportunity_engagement_invitation_request ={
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-123456"
        }
        try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_by_accepting_invitation_task(**accept_opportunit
            return response

        except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None
    else:
        return None
```

```
def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isusga"
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_by_accepting_invitation_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie StartEngagementByAcceptingInvitationTaskin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`StartEngagementFromOpportunityTask`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartEngagementFromOpportunityTask` verwendet wird.

Java

SDK für Java 2.x

Initiiert den Interaktionsprozess von einer bestehenden Opportunity aus, indem die Einladung zur Interaktion angenommen und eine entsprechende Opportunity im System des Partners erstellt wird.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String
 opportunityId) {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
```

```
            .clientToken("test-annjqwesdsd99")

  .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).vis
            .build();

   StartEngagementFromOpportunityTaskResponse response =
  client.startEngagementFromOpportunityTask(submitOpportunityRequest);

         return response;
  }
 }
```

- Einzelheiten zur API finden Sie StartEngagementFromOpportunityTaskin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Initiiert den Interaktionsprozess von einer bestehenden Opportunity aus, indem die Einladung zur Interaktion angenommen und eine entsprechende Opportunity im System des Partners erstellt wird.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"
```

```
partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def start_engagement_from_opportunity_task(identifier):

    start_engagement_from_opportunity_task_request ={
            "AwsSubmission": {
                "InvolvementType": "Co-Sell",
                "Visibility": "Full"
            },
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-annjqwesdsd99"
    }
    try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_from_opportunity_task(**start_engagement_from_op
            return response

    except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Start Engagement from Opportunity Task.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_from_opportunity_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie StartEngagementFromOpportunityTaskin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`UpdateOpportunity`**einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `UpdateOpportunity` verwendet wird.

Java

SDK für Java 2.x

Aktualisieren Sie eine Opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted
 or In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

 public static void main(String[] args) {

  String inputFile = "updateOpportunity.json";

  if (args.length > 0)
   inputFile = args[0];

  UpdateOpportunityResponse response = updateOpportunity(inputFile);
```

```
 client.close();
}

public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
```

```
    lifeCycle = LifeCycle.builder()
     .closedLostReason(root.lifeCycle.closedLostReason)
     .nextSteps(root.lifeCycle.nextSteps)
     .nextStepsHistory(nextStepsHistories)
     .reviewComments(root.lifeCycle.reviewComments)
     .reviewStatus(root.lifeCycle.reviewStatus)
     .reviewStatusReason(root.lifeCycle.reviewStatusReason)
     .stage(root.lifeCycle.stage)
     .targetCloseDate(root.lifeCycle.targetCloseDate)
     .build();
   }

   Marketing marketing = null;
   if ( root.marketing != null ) {
    marketing = Marketing.builder()
       .awsFundingUsed(root.marketing.awsFundingUsed)
       .campaignName(root.marketing.campaignName)
       .channels(root.marketing.channels)
       .source(root.marketing.source)
       .useCases(root.marketing.useCases)
       .build();

   }

   Address address = null;
   if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
    address =
Address.builder().postalCode(root.customer.account.address.postalCode)
       .stateOrRegion(root.customer.account.address.stateOrRegion)
       .countryCode(root.customer.account.address.countryCode).build();
   }

   Account account = null;
   if (root.customer != null && root.customer.account != null) {
    account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
       .websiteUrl(root.customer.account.websiteUrl).build();
   }

   List<Contact> contacts = new ArrayList<Contact>();
   if ( root.customer != null && root.customer.contacts != null) {
    for (org.example.entity.Contact jsonContact : root.customer.contacts) {
```

```
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModel
     .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorNam
     .otherSolutionDescription(root.project.otherSolutionDescription).build();
  }

  // Building the Actual CreateOpportunity Request
```

```
    UpdateOpportunityRequest updateOpportunityRequest =
  UpdateOpportunityRequest.builder().catalog(root.catalog)

  .identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

  .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportuni
      .lifeCycle(lifeCycle)
      .customer(customer)
      .project(project)
      .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
      .marketing(marketing)
      .nationalSecurity(root.nationalSecurity)
      .opportunityType(root.opportunityType)
      .build();

    UpdateOpportunityResponse updateResponse =
  client.updateOpportunity(updateOpportunityRequest);
    System.out.println("Successfully updated opportunity: " + updateResponse);

    return updateResponse;
   } else {
    System.out.println("Opportunity cannot be updated.");
    return null;
   }
     }
}
```

- Einzelheiten zur API finden Sie UpdateOpportunityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Aktualisieren Sie eine Opportunity.

```
#!/usr/bin/env python

"""
Purpose
PC-API-2  Updating Partner Originated Opportunity
```

```
"""
import logging
import boto3
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
from botocore.client import ClientError
import utils.stringify_details as sd
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity():
    update_opportunity_request_orig = sd.stringify_json("src/update_opportunity/
update_opportunity_technical_validation.json")
    update_opportunity_request =
 helper.remove_nulls(update_opportunity_request_orig)

    try:
        # Perform an API call
        response =
 partner_central_client.update_opportunity(**update_opportunity_request)
        return response
```

```
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity_if_eligible(identifier):
    response = get_opportunity(identifier)
    if response is not None:
        return update_opportunity()
    else:
        print("Failed to retrieve opportunity details")

def usage_demo():
    identifier = "05465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Updating opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(update_opportunity_if_eligible(identifier))

if __name__ == "__main__":
    usage_demo()
```

- Einzelheiten zur API finden Sie UpdateOpportunityin AWS SDK for Python (Boto3) API Reference.

Szenarien für die Verwendung von Partner Central AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Partner Central mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Partner Central oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Aktualisieren der zugehörigen Entität einer Opportunity

Aktualisieren der zugehörigen Entität einer Opportunity

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Trennen Sie die Zuordnung einer alten Entität.
- Ordnen Sie eine neue Entität zu.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie es im Scenarios-Repository eingerichtet und ausgeführt wird.

Aktualisieren der zugehörigen Entität einer Opportunity

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo
```

```
/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";

     disassociateOppornitityResponse(opportunityId, entityType,
 originalEntityIdentifier );
     AssociateOpportunityResponse associateOpportunityResponse =
 associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

     ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
    }

 private static AssociateOpportunityResponse associateOpportunityResponse(String
 opportunityId, String entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);
```

```
        return response;
 }

 private static DisassociateOpportunityResponse
 disassociateOppornititityResponse(String opportunityId, String entityType, String
 entityIdentifier) {
  PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

        DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AssociateOpportunity
  - DisassociateOpportunity

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren der zugehörigen Entität einer Opportunity

```
#!/usr/bin/env python

"""
Purpose
PC-API -17 Replacing a solution
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def replace_solution(original_entity_identifier, new_entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : original_entity_identifier
    }

    associate_opportunity_request ={
```

```
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : new_entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    original_entity_identifier = "S-0049999"
    new_entity_identifier = "S-0050014"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Replacing a solution.")
    print("-" * 88)

    helper.pretty_print_datetime(replace_solution(original_entity_identifier,
 new_entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AssociateOpportunity
  - DisassociateOpportunity

Codebeispiele für AWS Payment Cryptography verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Payment Cryptography mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS Payment Cryptography Benutzerhandbuch — Weitere Informationen zu AWS Payment Cryptography.
- AWS Payment Cryptography API-Referenz — Details zu allen verfügbaren AWS Payment Cryptography Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für AWS Payment Cryptography verwenden AWS SDKs
  - Erste Schritte mit Kryptografie im Bereich Zahlungssicherheit

Szenarien für AWS Payment Cryptography verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS Payment Cryptography mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS Payment Cryptography oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Kryptografie im Bereich Zahlungssicherheit

Erste Schritte mit Kryptografie im Bereich Zahlungssicherheit

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines -Schlüssels
- Überprüfen Sie den CVV2-Wert
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Payment Cryptography Getting Started Script
# This script demonstrates how to use AWS Payment Cryptography to create a key,
# generate and verify CVV2 values, and clean up resources.

set -euo pipefail

# Initialize log file with secure permissions
LOG_FILE="payment-cryptography-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "AWS Payment Cryptography Tutorial - $(date)" > "$LOG_FILE"

# Function to log messages
log() {
    local message="$1"
    echo "$(date +"%Y-%m-%d %H:%M:%S") - $message" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
```

```
    local error_message="$1"
    log "ERROR: $error_message"
    log "Script failed. Please check the log file: $LOG_FILE"

    echo ""
    echo "==========================================="
    echo "ERROR ENCOUNTERED"
    echo "==========================================="
    echo "The script encountered an error: $error_message"
    echo "Resources created will be listed below."
    echo ""

    if [ -n "${KEY_ARN:-}" ]; then
        echo "Key ARN: $KEY_ARN"
    fi

    exit 1
}

# Function to check command output for errors
check_error() {
    local output="$1"
    local command="$2"

    if echo "$output" | grep -iq "error\|exception\|fail"; then
        handle_error "Command failed: $command"
    fi
}

# Validate AWS CLI is available and credentials are configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not properly configured"
fi

log "Starting AWS Payment Cryptography tutorial"

# Step 1: Create a key
log "Step 1: Creating a card verification key (CVK)"
if ! KEY_OUTPUT=$(aws payment-cryptography create-key \
  --exportable \
```

```
  --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,Key
 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-payment-cryptography-
gs 2>&1); then
    handle_error "Failed to create key"
fi

echo "$KEY_OUTPUT"
check_error "$KEY_OUTPUT" "create-key"

# Extract the Key ARN from the output
KEY_ARN=$(echo "$KEY_OUTPUT" | grep -o '"KeyArn": "[^"]*' | cut -d'"' -f4)

if [ -z "$KEY_ARN" ]; then
    handle_error "Failed to extract Key ARN from output"
fi

log "Successfully created key with ARN: $KEY_ARN"

# Step 2: Generate a CVV2 value
log "Step 2: Generating a CVV2 value"
if ! CVV2_OUTPUT=$(aws payment-cryptography-data generate-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --generation-attributes CardVerificationValue2={CardExpiryDate=0123} 2>&1);
 then
    handle_error "Failed to generate CVV2 value"
fi

echo "$CVV2_OUTPUT"
check_error "$CVV2_OUTPUT" "generate-card-validation-data"

# Extract the CVV2 value from the output - updated to use ValidationData instead
 of CardDataValue
CVV2_VALUE=$(echo "$CVV2_OUTPUT" | grep -o '"ValidationData": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CVV2_VALUE" ]; then
    handle_error "Failed to extract CVV2 value from output"
fi

log "Successfully generated CVV2 value"
```

```
# Step 3: Verify the CVV2 value
log "Step 3: Verifying the CVV2 value"
if ! VERIFY_OUTPUT=$(aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data "$CVV2_VALUE" 2>&1); then
    handle_error "Failed to verify CVV2 value"
fi

echo "$VERIFY_OUTPUT"
check_error "$VERIFY_OUTPUT" "verify-card-validation-data"

log "Successfully verified CVV2 value"

# Step 4: Perform a negative test
log "Step 4: Performing a negative test with incorrect CVV2"
if aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 999 2>&1; then
    handle_error "Negative test did not fail as expected"
fi

log "Negative test completed successfully (verification failed as expected)"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Key ARN: $KEY_ARN"
echo ""

# Auto-confirm cleanup
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Proceeding with cleanup of all created resources..."

log "Step 5: Cleaning up resources"

# Delete the key
```

```
log "Deleting key: $KEY_ARN"
if ! DELETE_OUTPUT=$(aws payment-cryptography delete-key \
  --key-identifier "$KEY_ARN" 2>&1); then
    handle_error "Failed to delete key"
fi

echo "$DELETE_OUTPUT"
check_error "$DELETE_OUTPUT" "delete-key"

log "Key scheduled for deletion. Default waiting period is 7 days."
log "To cancel deletion before the waiting period ends, use:"
log "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

echo ""
echo "==========================================="
echo "CLEANUP COMPLETE"
echo "==========================================="
echo "The key has been scheduled for deletion after the default waiting period (7
 days)."
echo "To cancel deletion before the waiting period ends, use:"
echo "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

log "Tutorial completed successfully"
echo ""
echo "Tutorial completed successfully. See $LOG_FILE for details."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateKey
  - DeleteKey
  - RestoreKey

Codebeispiele für Amazon Personalize mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Personalize mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Amazon-Personalize-Entwicklerhandbuch – Weitere Informationen zu Amazon Personalize.
- API-Referenz für Amazon Personalize – Details zu allen verfügbaren Aktionen von Amazon Personalize.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Personalize AWS SDKs
  - Aktionen für Amazon Personalize mit AWS SDKs
    - Verwenden Sie CreateBatchInferenceJobmit einem AWS SDK
    - Verwenden Sie CreateBatchSegmentJobmit einem AWS SDK
    - Verwenden Sie CreateCampaignmit einem AWS SDK
    - Verwenden Sie CreateDatasetmit einem AWS SDK
    - Verwenden Sie CreateDatasetExportJobmit einem AWS SDK
    - Verwenden Sie CreateDatasetGroupmit einem AWS SDK
    - Verwenden Sie CreateDatasetImportJobmit einem AWS SDK
    - Verwenden Sie CreateEventTrackermit einem AWS SDK
    - Verwenden Sie CreateFiltermit einem AWS SDK
    - Verwenden Sie CreateRecommendermit einem AWS SDK
    - Verwenden Sie CreateSchemamit einem AWS SDK
    - Verwenden Sie CreateSolutionmit einem AWS SDK
    - Verwenden Sie CreateSolutionVersionmit einem AWS SDK
    - Verwenden Sie DeleteCampaignmit einem AWS SDK
    - Verwenden Sie DeleteEventTrackermit einem AWS SDK
    - Verwenden Sie DeleteSolutionmit einem AWS SDK
    - Verwenden Sie DescribeCampaignmit einem AWS SDK
    - Verwenden Sie DescribeRecipemit einem AWS SDK
    - Verwenden Sie DescribeSolutionmit einem AWS SDK

- Verwenden Sie ListCampaignsmit einem AWS SDK
- Verwenden Sie ListDatasetGroupsmit einem AWS SDK
- Verwenden Sie ListRecipesmit einem AWS SDK
- Verwenden Sie ListSolutionsmit einem AWS SDK
- Verwenden Sie UpdateCampaignmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Personalize AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Personalize mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Personalize mit AWS SDKs
  - Verwenden Sie CreateBatchInferenceJobmit einem AWS SDK
  - Verwenden Sie CreateBatchSegmentJobmit einem AWS SDK
  - Verwenden Sie CreateCampaignmit einem AWS SDK
  - Verwenden Sie CreateDatasetmit einem AWS SDK
  - Verwenden Sie CreateDatasetExportJobmit einem AWS SDK
  - Verwenden Sie CreateDatasetGroupmit einem AWS SDK
  - Verwenden Sie CreateDatasetImportJobmit einem AWS SDK
  - Verwenden Sie CreateEventTrackermit einem AWS SDK
  - Verwenden Sie CreateFiltermit einem AWS SDK
  - Verwenden Sie CreateRecommendermit einem AWS SDK
  - Verwenden Sie CreateSchemamit einem AWS SDK
  - Verwenden Sie CreateSolutionmit einem AWS SDK
  - Verwenden Sie CreateSolutionVersionmit einem AWS SDK
  - Verwenden Sie DeleteCampaignmit einem AWS SDK
  - Verwenden Sie DeleteEventTrackermit einem AWS SDK
  - Verwenden Sie DeleteSolutionmit einem AWS SDK
  - Verwenden Sie DescribeCampaignmit einem AWS SDK

- Verwenden Sie DescribeRecipemit einem AWS SDK
- Verwenden Sie DescribeSolutionmit einem AWS SDK
- Verwenden Sie ListCampaignsmit einem AWS SDK
- Verwenden Sie ListDatasetGroupsmit einem AWS SDK
- Verwenden Sie ListRecipesmit einem AWS SDK
- Verwenden Sie ListSolutionsmit einem AWS SDK
- Verwenden Sie UpdateCampaignmit einem AWS SDK

Aktionen für Amazon Personalize mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Personalize-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Personalize.

Beispiele

- Verwenden Sie CreateBatchInferenceJobmit einem AWS SDK
- Verwenden Sie CreateBatchSegmentJobmit einem AWS SDK
- Verwenden Sie CreateCampaignmit einem AWS SDK
- Verwenden Sie CreateDatasetmit einem AWS SDK
- Verwenden Sie CreateDatasetExportJobmit einem AWS SDK
- Verwenden Sie CreateDatasetGroupmit einem AWS SDK
- Verwenden Sie CreateDatasetImportJobmit einem AWS SDK
- Verwenden Sie CreateEventTrackermit einem AWS SDK
- Verwenden Sie CreateFiltermit einem AWS SDK
- Verwenden Sie CreateRecommendermit einem AWS SDK
- Verwenden Sie CreateSchemamit einem AWS SDK
- Verwenden Sie CreateSolutionmit einem AWS SDK
- Verwenden Sie CreateSolutionVersionmit einem AWS SDK
- Verwenden Sie DeleteCampaignmit einem AWS SDK
- Verwenden Sie DeleteEventTrackermit einem AWS SDK

- Verwenden Sie DeleteSolutionmit einem AWS SDK
- Verwenden Sie DescribeCampaignmit einem AWS SDK
- Verwenden Sie DescribeRecipemit einem AWS SDK
- Verwenden Sie DescribeSolutionmit einem AWS SDK
- Verwenden Sie ListCampaignsmit einem AWS SDK
- Verwenden Sie ListDatasetGroupsmit einem AWS SDK
- Verwenden Sie ListRecipesmit einem AWS SDK
- Verwenden Sie ListSolutionsmit einem AWS SDK
- Verwenden Sie UpdateCampaignmit einem AWS SDK

Verwenden Sie **`CreateBatchInferenceJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateBatchInferenceJob` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                        String solutionVersionArn,
                        String jobName,
                        String s3InputDataSourcePath,
                        String s3DataDestinationPath,
                        String roleArn,
                        String explorationWeight,
                        String explorationItemAgeCutOff) {

                long waitInMilliseconds = 60 * 1000;
                String status;
                String batchInferenceJobArn;
```

```
                try {

                        // Set up data input and output parameters.
                        S3DataConfig inputSource = S3DataConfig.builder()
                                        .path(s3InputDataSourcePath)
                                        .build();

                        S3DataConfig outputDestination = S3DataConfig.builder()
                                        .path(s3DataDestinationPath)
                                        .build();

                        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
                                        .s3DataSource(inputSource)
                                        .build();

                        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                                        .s3DataDestination(outputDestination)
                                        .build();

                        // Optional code to build the User-Personalization
specific item exploration
                        // config.
                        HashMap<String, String> explorationConfig = new
HashMap<>();

                        explorationConfig.put("explorationWeight",
explorationWeight);
                        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

                        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
                                        .itemExplorationConfig(explorationConfig)
                                        .build();

                        // End optional User-Personalization recipe specific
code.

                        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                                        .builder()
                                        .solutionVersionArn(solutionVersionArn)
```

```
                                        .jobInput(jobInput)
                                        .jobOutput(jobOutputLocation)
                                        .jobName(jobName)
                                        .roleArn(roleArn)
                                        .batchInferenceJobConfig(jobConfig) //
Optional
                                        .build();

                        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                                        .batchInferenceJobArn();

                        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                                        .builder()

.batchInferenceJobArn(batchInferenceJobArn)
                                        .build();

                        long maxTime = Instant.now().getEpochSecond() + 3 * 60 *
60;
                        while (Instant.now().getEpochSecond() < maxTime) {

                                BatchInferenceJob batchInferenceJob =
personalizeClient

.describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                                .batchInferenceJob();

                                status = batchInferenceJob.status();
                                System.out.println("Batch inference job status: "
+ status);

                                if (status.equals("ACTIVE") ||
status.equals("CREATE FAILED")) {
                                        break;
                                }
                                try {
                                        Thread.sleep(waitInMilliseconds);
                                } catch (InterruptedException e) {
                                        System.out.println(e.getMessage());
                                }
                        }
                        return batchInferenceJobArn;
```

```
                } catch (PersonalizeException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return "";
        }
```

- Einzelheiten zur API finden Sie CreateBatchInferenceJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: "JOB_NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
```

```
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchInferenceJobCommand(createBatchInferenceJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateBatchInferenceJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateBatchSegmentJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateBatchSegmentJob`.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
```

```
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: "NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchSegmentJobCommand(createBatchSegmentJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateBatchSegmentJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateCampaign`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateCampaign` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalCompaign(PersonalizeClient
personalizeClient, String solutionVersionArn,
           String name) {

       try {
           CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                   .minProvisionedTPS(1)
                   .solutionVersionArn(solutionVersionArn)
                   .name(name)
                   .build();

           CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
           System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
           return campaignResponse.campaignArn();
       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }
```

- Einzelheiten zur API finden Sie CreateCampaignin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: "SOLUTION_VERSION_ARN" /* required */,
  name: "NAME" /* required */,
  minProvisionedTPS: 1 /* optional integer */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateCampaignCommand(createCampaignParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateCampaignin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateDataset`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDataset` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static String createDataset(PersonalizeClient personalizeClient,
        String datasetName,
        String datasetGroupArn,
        String datasetType,
        String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
                .name(datasetName)
                .datasetGroupArn(datasetGroupArn)
                .datasetType(datasetType)
                .schemaArn(schemaArn)
                .build();

        String datasetArn = personalizeClient.createDataset(request)
                .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Einzelheiten zur API finden Sie CreateDataset in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  datasetType: "DATASET_TYPE" /* required */,
  name: "NAME" /* required */,
  schemaArn: "SCHEMA_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetCommand(createDatasetParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateDatasetin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateDatasetExportJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDatasetExportJob` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDatasetExportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           IngestionMode ingestionMode,
           String roleArn,
           String s3BucketPath,
           String kmsKeyArn) {

       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String status = null;

       try {

           S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
           DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
                   .build();

           CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                   .jobName(jobName)
                   .datasetArn(datasetArn)
                   .ingestionMode(ingestionMode)
                   .jobOutput(jobOutput)
                   .roleArn(roleArn)
                   .build();
```

```
            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                    .datasetExportJobArn(datasetExportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob = personalizeClient

.describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateDatasetExportJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  jobOutput: {
    s3DataDestination: {
      path: "S3_DESTINATION_PATH" /* required */,
      //kmsKeyArn: 'ARN'  /* include if your bucket uses AWS KMS for encryption
    },
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetExportJobCommand(datasetExportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateDatasetExportJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateDatasetGroup`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDatasetGroup` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

Erstellen Sie eine Domain-Datensatzgruppe.

```
   public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
           String datasetGroupName,
```

```
            String domain) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
 CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .domain(domain)
                    .build();
            return
 personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateDatasetGroupin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.

import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset group parameters.
export const createDatasetGroupParam = {
  name: "NAME" /* required */,
```

```
};

export const run = async (createDatasetGroupParam) => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(createDatasetGroupParam),
    );
    console.log("Success", response);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run(createDatasetGroupParam);
```

Erstellen Sie eine Domain-Datensatzgruppe.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: "NAME" /* required */,
  domain:
    "DOMAIN" /* required for a domain dsg, specify ECOMMERCE or VIDEO_ON_DEMAND
 */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(domainDatasetGroupParams),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Einzelheiten zur API finden Sie CreateDatasetGroupin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateDatasetImportJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateDatasetImportJob` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
           String jobName,
           String datasetArn,
           String s3BucketPath,
           String roleArn) {

       long waitInMilliseconds = 60 * 1000;
       String status;
       String datasetImportJobArn;

       try {
           DataSource importDataSource = DataSource.builder()
                   .dataLocation(s3BucketPath)
                   .build();

           CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                   .datasetArn(datasetArn)
                   .dataSource(importDataSource)
```

```
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient

.describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;

        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateDatasetImportJobin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetImportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  dataSource: {
    /* required */
    dataLocation: "S3_PATH",
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetImportJobCommand(datasetImportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- Einzelheiten zur API finden Sie CreateDatasetImportJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateEventTracker`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateEventTracker` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
           String datasetGroupArn) {

       String eventTrackerId = "";
       String eventTrackerArn;
       long maxTime = 3 * 60 * 60; // 3 hours
       long waitInMilliseconds = 20 * 1000; // 20 seconds
       String status;

       try {

           CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                   .name(eventTrackerName)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
```

```
                    .createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- Einzelheiten zur API finden Sie CreateEventTrackerin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateEventTrackerCommand(createEventTrackerParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateEventTrackerin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateFilter`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateFilter` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static String createFilter(PersonalizeClient personalizeClient,
        String filterName,
        String datasetGroupArn,
        String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .datasetGroupArn(datasetGroupArn)
                .filterExpression(filterExpression)
                .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- Einzelheiten zur API finden Sie CreateFilterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
  filterExpression: "FILTER_EXPRESSION" /*required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateFilterCommand(createFilterParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateFilterin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateRecommender`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateRecommender` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
```

```
            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateRecommenderin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
```

```
import { CreateRecommenderCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: "NAME" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateRecommenderCommand(createRecommenderParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateRecommenderin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateSchema`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSchema` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest =
CreateSchemaRequest.builder()
                   .name(schemaName)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

Erstellen Sie ein Schema mit einer Domain.

```
   public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
           String filePath) {

       String schema = null;
       try {
           schema = new String(Files.readAllBytes(Paths.get(filePath)));
       } catch (IOException e) {
           System.out.println(e.getMessage());
       }

       try {
           CreateSchemaRequest createSchemaRequest =
CreateSchemaRequest.builder()
                   .name(schemaName)
                   .domain(domain)
                   .schema(schema)
                   .build();

           String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

           System.out.println("Schema arn: " + schemaArn);

           return schemaArn;

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie CreateSchemain der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

Erstellen Sie ein Schema mit einer Domain.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
  domain:
    "DOMAIN" /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createDomainSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
```

```
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateSchemain der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateSolution`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSolution` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
           String datasetGroupArn,
           String solutionName,
           String recipeArn) {

       try {
           CreateSolutionRequest solutionRequest =
CreateSolutionRequest.builder()
                   .name(solutionName)
                   .datasetGroupArn(datasetGroupArn)
                   .recipeArn(recipeArn)
                   .build();

           CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
           return solutionResponse.solutionArn();

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateSolutionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(createSolutionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateSolutionin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`CreateSolutionVersion`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSolutionVersion` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
       long maxTime = 0;
       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String solutionStatus = "";
       String solutionVersionStatus = "";
       String solutionVersionArn = "";

       try {
           DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

           // Wait until solution is active.
           while (Instant.now().getEpochSecond() < maxTime) {

               solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
               System.out.println("Solution status: " + solutionStatus);
```

```
                if (solutionStatus.equals("ACTIVE") ||
solutionStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " +
solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
```

```
                    try {
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateSolutionVersionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: "SOLUTION_ARN" /* required */,
};

export const run = async () => {
```

```
  try {
    const response = await personalizeClient.send(
      new CreateSolutionVersionCommand(solutionVersionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateSolutionVersionin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DeleteCampaign`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteCampaign`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteSpecificCampaign(PersonalizeClient
personalizeClient, String campaignArn) {
       try {
           DeleteCampaignRequest campaignRequest =
DeleteCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .build();

           personalizeClient.deleteCampaign(campaignRequest);
           System.out.println("Delete request sent successfully.");
       } catch (PersonalizeException e) {
```

```
            System.err.println("Error deleting campaign: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteCampaignin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteEventTracker`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteEventTracker`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
       try {
           DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
                   .eventTrackerArn(eventTrackerArn)
                   .build();

           int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().status

           System.out.println("Status code:" + status);

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteEventTrackerin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteSolution`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteSolution`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           DeleteSolutionRequest solutionRequest =
DeleteSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           personalizeClient.deleteSolution(solutionRequest);
           System.out.println("Done");

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DeleteSolutionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeCampaign`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeCampaign`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificCampaign(PersonalizeClient
personalizeClient, String campaignArn) {

       try {
           DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .build();

           DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
           Campaign myCampaign = campaignResponse.campaign();
           System.out.println("The Campaign name is " + myCampaign.name());
           System.out.println("The Campaign status is " + myCampaign.status());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeCampaignin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeRecipe`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeRecipe`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificRecipe(PersonalizeClient
personalizeClient, String recipeArn) {

       try {
           DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
                   .recipeArn(recipeArn)
                   .build();

           DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
           System.out.println("The recipe name is " +
recipeResponse.recipe().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeRecipein der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DescribeSolution`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeSolution`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificSolution(PersonalizeClient
personalizeClient, String solutionArn) {

       try {
           DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
           System.out.println("The Solution name is " +
response.solution().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeSolutionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListCampaigns`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListCampaigns`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listAllCampaigns(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           ListCampaignsRequest campaignsRequest =
ListCampaignsRequest.builder()
                   .maxResults(10)
                   .solutionArn(solutionArn)
                   .build();

           ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
           List<CampaignSummary> campaigns = response.campaigns();
           for (CampaignSummary campaign : campaigns) {
               System.out.println("Campaign name is : " + campaign.name());
               System.out.println("Campaign ARN is : " +
campaign.campaignArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListCampaignsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListDatasetGroups`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListDatasetGroups`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listDSGroups(PersonalizeClient personalizeClient) {

       try {
           ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
                   .maxResults(15)
                   .build();

           ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
           List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
           for (DatasetGroupSummary group : groups) {
               System.out.println("The DataSet name is : " + group.name());
               System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListDatasetGroupsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListRecipes`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListRecipes`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                    .maxResults(15)
                    .build();

            ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe : recipes) {
                System.out.println("The recipe ARN is: " + recipe.recipeArn());
                System.out.println("The recipe name is: " + recipe.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListRecipesin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListSolutions`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListSolutions`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listAllSolutions(PersonalizeClient personalizeClient,
String datasetGroupArn) {

       try {
           ListSolutionsRequest solutionsRequest =
ListSolutionsRequest.builder()
                   .maxResults(10)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
           List<SolutionSummary> solutions = response.solutions();
           for (SolutionSummary solution : solutions) {
               System.out.println("The solution ARN is: " +
solution.solutionArn());
               System.out.println("The solution name is: " + solution.name());
           }

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListSolutionsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UpdateCampaign`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateCampaign`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String updateCampaign(PersonalizeClient personalizeClient,
            String campaignArn,
            String solutionVersionArn,
            Integer minProvisionedTPS) {

        try {
            // build the updateCampaignRequest
            UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .solutionVersionArn(solutionVersionArn)
                    .minProvisionedTPS(minProvisionedTPS)
                    .build();

            // update the campaign
            personalizeClient.updateCampaign(updateCampaignRequest);

            DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign updatedCampaign = campaignResponse.campaign();

            System.out.println("The Campaign status is " +
updatedCampaign.status());
            return updatedCampaign.status();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie UpdateCampaignin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon Personalize Events mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Personalize Events mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Personalize Events – Weitere Informationen zu Amazon Personalize Events.
- API-Referenz für Amazon Personalize Events – Details zu allen verfügbaren Aktionen von Amazon Personalize Events.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Amazon Personalize Events mit AWS SDKs
  - Aktionen für Amazon Personalize Events mit AWS SDKs
    - Verwenden Sie PutEventsmit einem AWS SDK
    - Verwenden Sie PutItemsmit einem AWS SDK
    - Verwenden Sie PutUsersmit einem AWS SDK

Grundlegende Beispiele für Amazon Personalize Events mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Personalize Events mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Personalize Events mit AWS SDKs
  - Verwenden Sie PutEventsmit einem AWS SDK
  - Verwenden Sie PutItemsmit einem AWS SDK
  - Verwenden Sie PutUsersmit einem AWS SDK

Aktionen für Amazon Personalize Events mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Personalize Events-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie unter API-Referenz für Amazon Personalize.

Beispiele

- Verwenden Sie PutEventsmit einem AWS SDK
- Verwenden Sie PutItemsmit einem AWS SDK
- Verwenden Sie PutUsersmit einem AWS SDK

Verwenden Sie **`PutEvents`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutEvents` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static int putItems(PersonalizeEventsClient
personalizeEventsClient,
                        String datasetArn,
                        String item1Id,
                        String item1PropertyName,
                        String item1PropertyValue,
                        String item2Id,
                        String item2PropertyName,
                        String item2PropertyValue) {

                int responseCode = 0;
                ArrayList<Item> items = new ArrayList<>();

                try {
                        Item item1 = Item.builder()
                                        .itemId(item1Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        item1PropertyName,
item1PropertyValue))
                                        .build();

                        items.add(item1);

                        Item item2 = Item.builder()
                                        .itemId(item2Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        item2PropertyName,
item2PropertyValue))
                                        .build();

                        items.add(item2);

                        PutItemsRequest putItemsRequest =
PutItemsRequest.builder()
                                        .datasetArn(datasetArn)
                                        .items(items)
                                        .build();

                        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
```

```
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { PutEventsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000); // 1613443801 is a testing value.
 Replace it with your sentAt timestamp in UNIX format.

// Set put events parameters.
const putEventsParam = {
  eventList: [
    /* required */
    {
      eventType: "EVENT_TYPE" /* required */,
      sentAt: sentAtDate /* required, must be a Date with js */,
      eventId: "EVENT_ID" /* optional */,
      itemId: "ITEM_ID" /* optional */,
    },
```

```
  ],
  sessionId: "SESSION_ID" /* required */,
  trackingId: "TRACKING_ID" /* required */,
  userId: "USER_ID" /* required */,
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutEventsCommand(putEventsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`PutItems`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PutItems`.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});
```

```
// Set the put items parameters. For string properties and values, use the \
 character to escape quotes.
const putItemsParam = {
  datasetArn: "DATASET_ARN" /* required */,
  items: [
    /* required */
    {
      itemId: "ITEM_ID" /*  required */,
      properties:
        '{"PROPERTY1_NAME": "PROPERTY1_VALUE", "PROPERTY2_NAME":
 "PROPERTY2_VALUE", "PROPERTY3_NAME": "PROPERTY3_VALUE"}' /* optional */,
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie PutItemsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`PutUsers`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutUsers` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static int putUsers(PersonalizeEventsClient
personalizeEventsClient,
                        String datasetArn,
                        String user1Id,
                        String user1PropertyName,
                        String user1PropertyValue,
                        String user2Id,
                        String user2PropertyName,
                        String user2PropertyValue) {

                int responseCode = 0;
                ArrayList<User> users = new ArrayList<>();

                try {
                        User user1 = User.builder()
                                        .userId(user1Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        user1PropertyName,
user1PropertyValue))
                                        .build();

                        users.add(user1);

                        User user2 = User.builder()
                                        .userId(user2Id)
                                        .properties(String.format("{\"%1$s\":
\"%2$s\"}",
                                                        user2PropertyName,
user2PropertyValue))
                                        .build();

                        users.add(user2);

                        PutUsersRequest putUsersRequest =
PutUsersRequest.builder()
                                        .datasetArn(datasetArn)
                                        .users(users)
                                        .build();

                        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
```

```
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- Einzelheiten zur API finden Sie PutUsersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
 "REGION"});

// Set the put users parameters. For string properties and values, use the \
 character to escape quotes.
const putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      userId: "USER_ID",
      properties: '{"PROPERTY1_NAME": "PROPERTY1_VALUE"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
```

```
      new PutUsersCommand(putUsersParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie PutUsersin der AWS SDK für JavaScript API-Referenz.

Codebeispiele für Amazon Personalize Runtime mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Personalize Runtime mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Personalize Runtime – Weitere Informationen zu Amazon Personalize Runtime.
- API-Referenz für Amazon Personalize Runtime – Details zu allen verfügbaren Aktionen von Amazon Personalize Runtime.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Personalize Runtime AWS SDKs
  - Aktionen für Amazon Personalize Runtime mit AWS SDKs
    - Verwenden Sie GetPersonalizedRankingmit einem AWS SDK
    - Verwenden Sie GetRecommendationsmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Personalize Runtime AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Personalize Runtime mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Personalize Runtime mit AWS SDKs
  - Verwenden Sie GetPersonalizedRankingmit einem AWS SDK
  - Verwenden Sie GetRecommendationsmit einem AWS SDK

Aktionen für Amazon Personalize Runtime mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Personalize Runtime-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Personalize Runtime.

Beispiele

- Verwenden Sie GetPersonalizedRankingmit einem AWS SDK
- Verwenden Sie GetRecommendationsmit einem AWS SDK

Verwenden Sie **`GetPersonalizedRanking`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetPersonalizedRanking` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           ArrayList<String> items) {

       try {
           GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
                   .campaignArn(campaignArn)
                   .userId(userId)
                   .inputList(items)
                   .build();

           GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
                   .getPersonalizedRanking(rankingRecommendationsRequest);
           List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
           int rank = 1;
           for (PredictedItem item : rankedItems) {
               System.out.println("Item ranked at position " + rank + "
details");
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());

System.out.println("--------------------------------------------");
               rank++;
           }
           return rankedItems;
       } catch (PersonalizeRuntimeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }
```

- Einzelheiten zur API finden Sie GetPersonalizedRankingin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from "@aws-sdk/client-personalize-
runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetPersonalizedRankingCommand(getPersonalizedRankingParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie GetPersonalizedRankingin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`GetRecommendations`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetRecommendations` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Liste von empfohlenen Artikeln ab.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();
           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }

       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Rufen Sie eine Liste von empfohlenen Artikeln von einem Empfehler ab, der in einer Domain-Datensatzgruppe erstellt wurde.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
           String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .recommenderArn(recommenderArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();

           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }
       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Verwenden Sie einen Filter, wenn Sie Empfehlungen anfordern.

```
   public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           String filterArn,
           String parameter1Name,
           String parameter1Value1,
           String parameter1Value2,
           String parameter2Name,
           String parameter2Value) {
```

```
        try {

            Map<String, String> filterValues = new HashMap<>();

            filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                    parameter1Value1, parameter1Value2));
            filterValues.put(parameter2Name, String.format("\"%1$s\"",
                    parameter2Value));

            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
                    .userId(userId)
                    .filterArn(filterArn)
                    .filterValues(filterValues)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie GetRecommendationsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Holen Sie sich eine Empfehlung mit einem Filter (benutzerdefinierte Datensatzgruppe).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Holen Sie sich gefilterte Empfehlungen von einem Empfehler, der in einer Domain-Datensatzgruppe erstellt wurde.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
```

```
  filterArn: "FILTER_ARN" /* required to filter recommendations */,
  filterValues: {
    PROPERTY:
      '"VALUE"' /* Only required if your filter has a placeholder parameter */,
  },
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie GetRecommendationsin der AWS SDK für JavaScript API-Referenz.

Codebeispiele für Amazon Pinpoint mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Pinpoint mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Pinpoint-Entwicklerhandbuch – Weitere Informationen zu Amazon Pinpoint.
- API-Referenz für Amazon Pinpoint – Details zu allen verfügbaren Amazon-Pinpoint-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Pinpoint AWS SDKs
  - Aktionen für Amazon Pinpoint mit AWS SDKs
    - Verwenden Sie CreateAppmit einem AWS SDK oder CLI
    - Verwenden Sie CreateCampaignmit einem AWS SDK
    - Verwenden Sie CreateExportJobmit einem AWS SDK
    - Verwenden Sie CreateImportJobmit einem AWS SDK
    - Verwenden Sie CreateSegmentmit einem AWS SDK
    - Verwenden Sie DeleteAppmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteEndpointmit einem AWS SDK
    - Verwenden Sie GetEndpointmit einem AWS SDK oder CLI
    - Verwenden Sie GetSegmentsmit einem AWS SDK
    - Verwenden Sie GetSmsChannelmit einem AWS SDK oder CLI
    - Verwenden Sie GetUserEndpointsmit einem AWS SDK
    - Verwenden Sie SendMessagesmit einem AWS SDK oder CLI
    - Verwenden Sie UpdateEndpointmit einem AWS SDK
- Szenarien für die Verwendung von Amazon Pinpoint AWS SDKs
  - Erste Schritte mit Push-Benachrichtigungen

Grundlegende Beispiele für die Verwendung von Amazon Pinpoint AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Pinpoint mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Pinpoint mit AWS SDKs
  - Verwenden Sie CreateAppmit einem AWS SDK oder CLI
  - Verwenden Sie CreateCampaignmit einem AWS SDK

- Verwenden Sie CreateExportJobmit einem AWS SDK
- Verwenden Sie CreateImportJobmit einem AWS SDK
- Verwenden Sie CreateSegmentmit einem AWS SDK
- Verwenden Sie DeleteAppmit einem AWS SDK oder CLI
- Verwenden Sie DeleteEndpointmit einem AWS SDK
- Verwenden Sie GetEndpointmit einem AWS SDK oder CLI
- Verwenden Sie GetSegmentsmit einem AWS SDK
- Verwenden Sie GetSmsChannelmit einem AWS SDK oder CLI
- Verwenden Sie GetUserEndpointsmit einem AWS SDK
- Verwenden Sie SendMessagesmit einem AWS SDK oder CLI
- Verwenden Sie UpdateEndpointmit einem AWS SDK

Aktionen für Amazon Pinpoint mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Pinpoint Pinpoint-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Pinpoint-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Pinpoint AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon-Pinpoint-API-Referenz.

Beispiele

- Verwenden Sie CreateAppmit einem AWS SDK oder CLI
- Verwenden Sie CreateCampaignmit einem AWS SDK
- Verwenden Sie CreateExportJobmit einem AWS SDK
- Verwenden Sie CreateImportJobmit einem AWS SDK
- Verwenden Sie CreateSegmentmit einem AWS SDK
- Verwenden Sie DeleteAppmit einem AWS SDK oder CLI
- Verwenden Sie DeleteEndpointmit einem AWS SDK
- Verwenden Sie GetEndpointmit einem AWS SDK oder CLI

- Verwenden Sie GetSegmentsmit einem AWS SDK
- Verwenden Sie GetSmsChannelmit einem AWS SDK oder CLI
- Verwenden Sie GetUserEndpointsmit einem AWS SDK
- Verwenden Sie SendMessagesmit einem AWS SDK oder CLI
- Verwenden Sie UpdateEndpointmit einem AWS SDK

Verwenden Sie **`CreateApp`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateApp` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Push-Benachrichtigungen

CLI

AWS CLI

Beispiel 1: Erstellen einer Anwendung

Im folgenden Beispiel für `create-app` wird eine neue Anwendung (Projekt) erstellt.

```
aws pinpoint create-app \
    --create-application-request Name=ExampleCorp
```

Ausgabe:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {}
    }
}
```

Beispiel 2: Erstellen einer mit Tags versehenen Anwendung

Im folgenden Beispiel für `create-app` wird eine neue Anwendung (Projekt) erstellt und der Anwendung ein Tag (Schlüssel und Wert) zugeordnet.

```
aws pinpoint create-app \
    --create-application-request Name=ExampleCorp,tags={"Stack"="Test"}
```

Ausgabe:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {
            "Stack": "Test"
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateAppin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = """

                 Usage:  <appName>

                 Where:
                  appName - The name of the application to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String
 appName) {
        try {
            CreateApplicationRequest appRequest =
 CreateApplicationRequest.builder()
                    .name(appName)
                    .build();

            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
```

```
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateAppin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- API-Details finden Sie CreateAppin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateCampaign`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateCampaign` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Kampagne.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCampaign {
```

```
    public static void main(String[] args) {

        final String usage = """

                Usage:   <appId> <segmentId>

                Where:
                  appId - The ID of the application to create the campaign in.
                  segmentId - The ID of the segment to create the campaign from.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
 String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
    }

    public static CampaignResponse createCampaign(PinpointClient client, String
 appID, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
                    .action(Action.OPEN_APP)
                    .body("My message body.")
```

```
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration =
 MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
                    .name("MyCampaign")
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result =
 client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build());

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
}
```

- Einzelheiten zur API finden Sie CreateCampaignin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
```

```
            pinpoint.createCampaign(
                CreateCampaignRequest {
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- API-Details finden Sie CreateCampaignin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateExportJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateExportJob`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Exportieren Sie einen Endpunkt.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
```

```
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-
export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = """

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
                2. Downloads the exported endpoints files from Amazon S3.
                3. Parses the endpoints files to obtain the endpoint IDs and
 prints them.
                Usage: ExportEndpoints <applicationId> <s3BucketName>
 <iamExportRoleArn> <path>

                Where:
```

```
                  applicationId - The ID of the Amazon Pinpoint application that
has the endpoint.
                  s3BucketName - The name of the Amazon S3 bucket to export the
JSON file to.\s
                  iamExportRoleArn - The ARN of an IAM role that grants Amazon
Pinpoint write permissions to the S3 bucket.  path - The path where the files
downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).
               """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String applicationId = args[0];
       String s3BucketName = args[1];
       String iamExportRoleArn = args[2];
       String path = args[3];
       System.out.println("Deleting an application with ID: " + applicationId);

       Region region = Region.US_EAST_1;
       PinpointClient pinpoint = PinpointClient.builder()
               .region(region)
               .build();

       S3Client s3Client = S3Client.builder()
               .region(region)
               .build();

       exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
       pinpoint.close();
       s3Client.close();
   }

   public static void exportAllEndpoints(PinpointClient pinpoint,
           S3Client s3Client,
           String applicationId,
           String s3BucketName,
           String path,
           String iamExportRoleArn) {

       try {
```

```
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
 s3BucketName, iamExportRoleArn,
                    applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
 o.endsWith(".gz"))
                    .collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> exportEndpointsToS3(PinpointClient pinpoint,
 S3Client s3Client, String s3BucketName,
            String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
 "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                    .roleArn(iamExportRoleArn)
                    .s3UrlPrefix(s3UrlPrefix)
                    .build();

            CreateExportJobRequest exportJobRequest =
 CreateExportJobRequest.builder()
                    .applicationId(applicationId)
                    .exportJobRequest(jobRequest)
                    .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint
 application %s to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);
```

```
            CreateExportJobResponse exportResult =
 pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                    .bucket(s3BucketName)
                    .prefix(endpointsKeyPrefix)
                    .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
            List<S3Object> objects = v2Response.contents();
            for (S3Object object : objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
            String applicationId,
            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                    .jobId(jobId)
                    .applicationId(applicationId)
                    .build();

            do {
```

```
                getExportJobResult =
pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }

        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path
location.
    public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                        .bucket(s3BucketName)
                        .key(key)
                        .build();

                ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix + ".gz";
                File myFile = new File(newPath);
```

```
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateExportJobin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateImportJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateImportJob`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie ein Segment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <bucket> <key> <roleArn>\s

                Where:
                  appId - The application ID to create a segment for.
                  bucket - The name of the Amazon S3 bucket that contains the
 segment definitons.
                  key - The key of the S3 object.
                  roleArn - ARN of the role that allows Amazon
 Pinpoint to access S3. You need to set trust management for this
 to work. See https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_elements_principal.html
                  """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
 key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
 import job status.");
```

```
        System.out.println("See application " + response.jobStatus() + " for
 import job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
            String appId,
            String bucket,
            String key,
            String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                    .defineSegment(true)
                    .registerEndpoints(true)
                    .roleArn(roleArn)
                    .format(Format.JSON)
                    .s3Url("s3://" + bucket + "/" + key)
                    .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                    .importJobRequest(importRequest)
                    .applicationId(appId)
                    .build();

            CreateImportJobResponse jobResponse =
 client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Einzelheiten zur API finden Sie CreateImportJobin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateSegment`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateSegment` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateSegment {
        public static void main(String[] args) {
                final String usage = """

                                Usage:   <appId>
```

```
                                Where:
                                  appId - The application ID to create a segment
for.

                                """;

                if (args.length != 1) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String appId = args[0];
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                SegmentResponse result = createSegment(pinpoint, appId);
                System.out.println("Segment " + result.name() + " created.");
                System.out.println(result.segmentType());
                pinpoint.close();
        }

        public static SegmentResponse createSegment(PinpointClient client, String
appId) {
                try {
                        Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();
                        segmentAttributes.put("Team",
AttributeDimension.builder()
                                        .attributeType(AttributeType.INCLUSIVE)
                                        .values("Lakers")
                                        .build());

                        RecencyDimension recencyDimension =
RecencyDimension.builder()
                                        .duration("DAY_30")
                                        .recencyType("ACTIVE")
                                        .build();

                        SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                                        .recency(recencyDimension)
                                        .build();
```

```
                        SegmentDemographics segmentDemographics =
 SegmentDemographics
                                        .builder()
                                        .build();

                        SegmentLocation segmentLocation = SegmentLocation
                                        .builder()
                                        .build();

                        SegmentDimensions dimensions = SegmentDimensions
                                        .builder()
                                        .attributes(segmentAttributes)
                                        .behavior(segmentBehaviors)
                                        .demographic(segmentDemographics)
                                        .location(segmentLocation)
                                        .build();

                        WriteSegmentRequest writeSegmentRequest =
 WriteSegmentRequest.builder()
                                        .name("MySegment")
                                        .dimensions(dimensions)
                                        .build();

                        CreateSegmentRequest createSegmentRequest =
 CreateSegmentRequest.builder()
                                        .applicationId(appId)
                                        .writeSegmentRequest(writeSegmentRequest)
                                        .build();

                        CreateSegmentResponse createSegmentResult =
 client.createSegment(createSegmentRequest);
                        System.out.println("Segment ID: " +
 createSegmentResult.segmentResponse().id());
                        System.out.println("Done");
                        return createSegmentResult.segmentResponse();

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return null;
        }
}
```

- Einzelheiten zur API finden Sie CreateSegmentin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }

    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
```

```
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- API-Details finden Sie CreateSegmentin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteApp`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteApp` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Push-Benachrichtigungen

CLI

AWS CLI

So löschen Sie eine Anwendung

Im folgenden Beispiel für `delete-app` wird eine Anwendung (Projekt) gelöscht.

```
aws pinpoint delete-app \
    --application-id 810c7aab86d42fb2b56c8c966example
```

Ausgabe:

```
{
    "ApplicationResponse": {
        "Arn": "arn:aws:mobiletargeting:us-
west-2:AIDACKCEVSQ6C2EXAMPLE:apps/810c7aab86d42fb2b56c8c966example",
        "Id": "810c7aab86d42fb2b56c8c966example",
        "Name": "ExampleCorp",
        "tags": {}
    }
}
```

- Einzelheiten zur API finden Sie DeleteAppin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Anwendung.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                 appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinApp(pinpoint, appId);
        System.out.println("Done");
        pinpoint.close();
    }

    public static void deletePinApp(PinpointClient pinpoint, String appId) {
        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                    .applicationId(appId)
                    .build();

            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie DeleteAppin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- API-Details finden Sie DeleteAppin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteEndpoint`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteEndpoint` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Endpunktes

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appName> <endpointId >

                Where:
                  appId - The id of the application to delete.
                  endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Einzelheiten zur API finden Sie DeleteEndpointin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- API-Details finden Sie DeleteEndpointin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetEndpoint`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetEndpoint` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen über die Einstellungen und Attribute eines bestimmten Endpunkts für eine Anwendung ab

Das folgende `get-endpoint`-Beispiel ruft Informationen über die Einstellungen und Attribute eines bestimmten Endpunkts für eine Anwendung ab.

```
aws pinpoint get-endpoint \
    --application-id 611e3e3cdd47474c9c1399a505665b91 \
    --endpoint-id testendpoint \
    --region us-east-1
```

Ausgabe:

```
{
```

```
    "EndpointResponse": {
        "Address": "+11234567890",
        "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",
        "Attributes": {},
        "ChannelType": "SMS",
        "CohortId": "63",
        "CreationDate": "2019-01-28T23:55:11.534Z",
        "EffectiveDate": "2021-08-06T00:04:51.763Z",
        "EndpointStatus": "ACTIVE",
        "Id": "testendpoint",
        "Location": {
            "Country": "USA"
        },
        "Metrics": {
            "SmsDelivered": 1.0
        },
        "OptOut": "ALL",
        "RequestId": "a204b1f2-7e26-48a7-9c80-b49a2143489d",
        "User": {
            "UserAttributes": {
                "Age": [
                    "24"
                ]
            },
        "UserId": "testuser"
        }
    }
}
```

- Einzelheiten zur API finden Sie GetEndpointin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.google.gson.FieldNamingPolicy;
```

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <endpoint>

                Where:
                  appId - The ID of the application to delete.
                  endpoint - The ID of the endpoint.\s
                  """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }
```

```
    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String
 appId, String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpoint)
                    .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                    .setPrettyPrinting()
                    .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- Einzelheiten zur API finden Sie GetEndpointin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse

        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- API-Details finden Sie GetEndpointin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetSegments`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetSegments` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Segmente auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId>

                Where:
                  appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSegs(pinpoint, appId);
        pinpoint.close();
    }
```

```
    public static void listSegs(PinpointClient pinpoint, String appId) {
        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();

            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for (SegmentResponse segment : segments) {
                System.out
                        .println("Segement " + segment.id() + " " +
 segment.name() + " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetSegmentsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
```

```
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- API-Details finden Sie GetSegmentsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetSmsChannel`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSmsChannel` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen über den Status und die Einstellungen jedes Sprachkanals für eine Anwendung ab

Im folgenden Beispiel für `get-sms-channel` werden Status und Einstellungen des SMS-Kanals für eine Anwendung abgerufen.

```
aws pinpoint get-sms-channel \
    --application-id 6e0b7591a90841d2b5d93fa11143e5a7 \
    --region us-east-1
```

Ausgabe:

```
{
    "SMSChannelResponse": {
        "ApplicationId": "6e0b7591a90841d2b5d93fa11143e5a7",
        "CreationDate": "2019-10-08T18:39:18.511Z",
        "Enabled": true,
        "Id": "sms",
        "IsArchived": false,
        "LastModifiedDate": "2019-10-08T18:39:18.511Z",
        "Platform": "SMS",
        "PromotionalMessagesPerSecond": 20,
        "TransactionalMessagesPerSecond": 20,
        "Version": 1
    }
```

```
}
```

- Einzelheiten zur API finden Sie GetSmsChannelin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = """

                Usage: CreateChannel <appId>

                Where:
                  appId - The name of the application whose channel is updated.

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        SMSChannelResponse getResponse = getSMSChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSMSChannel(PinpointClient client, String
appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                    .applicationId(appId)
                    .build();

            SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
            System.out.println("Channel state is " + response.enabled());
            return response;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
        boolean enabled = !getResponse.enabled();
        try {
            SMSChannelRequest request = SMSChannelRequest.builder()
                    .enabled(enabled)
                    .build();
```

```
            UpdateSmsChannelRequest updateRequest =
 UpdateSmsChannelRequest.builder()
                    .smsChannelRequest(request)
                    .applicationId(appId)
                    .build();

            UpdateSmsChannelResponse result =
 client.updateSmsChannel(updateRequest);
            System.out.println("Channel state: " +
 result.smsChannelResponse().enabled());

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetSmsChannelin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetUserEndpoints`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetUserEndpoints`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <applicationId> <userId>

                Where:
                   applicationId - The ID of the Amazon Pinpoint application that
 has the endpoint.
                   userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
            String applicationId,
            String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
 GetUserEndpointsRequest.builder()
                    .userId(userId)
```

```
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
 pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints =
 response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
 endpoint.channelType());
                System.out.println("The address is  " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetUserEndpointsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`SendMessages`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendMessages` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Push-Benachrichtigungen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendEmailMessage;

public class SendEmailMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load test settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();

        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The "From" address. This address has to be verified in Amazon
 Pinpoint
        // in the region you're using to send email.
        string senderAddress = configuration["SenderAddress"]!;

        // The address on the "To" line. If your Amazon Pinpoint account is in
        // the sandbox, this address also has to be verified.
```

```
        string toAddress = configuration["ToAddress"]!;

        // The Amazon Pinpoint project/application ID to use when you send this
message.
        // Make sure that the SMS channel is enabled for the project or
application
        // that you choose.
        string appId = configuration["AppId"]!;

        try
        {
            await SendEmailMessage(region, appId, toAddress, senderAddress);
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
        }
    }

    public static async Task<MessageResponse> SendEmailMessage(
        string region, string appId, string toAddress, string senderAddress)
    {
        var client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        // The subject line of the email.
        string subject = "Amazon Pinpoint Email test";

        // The body of the email for recipients whose email clients don't
        // support HTML content.
        string textBody = @"Amazon Pinpoint Email Test (.NET)"
                          + "\n---------------------------------"
                          + "\nThis email was sent using the Amazon Pinpoint API
using the AWS SDK for .NET.";

        // The body of the email for recipients whose email clients support
        // HTML content.
        string htmlBody = @"<html>"
                          + "\n<head></head>"
                          + "\n<body>"
                          + "\n  <h1>Amazon Pinpoint Email Test (AWS SDK
for .NET)</h1>"
                          + "\n  <p>This email was sent using the "
```

```
                            + "\n    <a href='https://aws.amazon.com/
pinpoint/'>Amazon Pinpoint</a> API "
                            + "\n    using the <a href='https://aws.amazon.com/sdk-
for-net/'>AWS SDK for .NET</a>"
                            + "\n  </p>"
                            + "\n</body>"
                            + "\n</html>";

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        string charset = "UTF-8";

        var sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
                    {
                        toAddress,
                        new AddressConfiguration
                        {
                            ChannelType = ChannelType.EMAIL
                        }
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    EmailMessage = new EmailMessage
                    {
                        FromAddress = senderAddress,
                        SimpleEmail = new SimpleEmail
                        {
                            HtmlPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = htmlBody
                            },
                            TextPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = textBody
                            },
```

```
                            Subject = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = subject
                            }
                        }
                    }
                }
            }
        };
        Console.WriteLine("Sending message...");
        SendMessagesResponse response = await
 client.SendMessagesAsync(sendRequest);
        Console.WriteLine("Message sent!");
        return response.MessageResponse;
    }
}
```

Senden Sie eine SMS-Nachricht.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendSmsMessage;

public class SendSmsMessageMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // The AWS Region that you want to use to send the message. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
```

```
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The phone number or short code to send the message from. The phone
 number
        // or short code that you specify has to be associated with your Amazon
 Pinpoint
        // account. For best results, specify long codes in E.164 format.
        string originationNumber = configuration["OriginationNumber"]!;

        // The recipient's phone number.  For best results, you should specify
 the
        // phone number in E.164 format.
        string destinationNumber = configuration["DestinationNumber"]!;

        // The Pinpoint project/ application ID to use when you send this
 message.
        // Make sure that the SMS channel is enabled for the project or
 application
        // that you choose.
        string appId = configuration["AppId"]!;

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        MessageType messageType = MessageType.TRANSACTIONAL;

        // The registered keyword associated with the originating short code.
        string? registeredKeyword = configuration["RegisteredKeyword"];

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        string? senderId = configuration["SenderId"];

        try
        {
            var response = await SendSmsMessage(region, appId, destinationNumber,
                originationNumber, registeredKeyword, senderId, messageType);
            Console.WriteLine($"Message sent to
 {response.MessageResponse.Result.Count} recipient(s).");
            foreach (var messageResultValue in
                     response.MessageResponse.Result.Select(r => r.Value))
```

```
            {
                Console.WriteLine($"{messageResultValue.MessageId} Status:
{messageResultValue.DeliveryStatus}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
        }
    }

    public static async Task<SendMessagesResponse> SendSmsMessage(
        string region, string appId, string destinationNumber, string
originationNumber,
        string? keyword, string? senderId, MessageType messageType)
    {

        // The content of the SMS message.
        string message = "This message was sent through Amazon Pinpoint using" +
                         " the AWS SDK for .NET. Reply STOP to opt out.";


        var client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        SendMessagesRequest sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses =
                    new Dictionary<string, AddressConfiguration>
                    {
                        {
                            destinationNumber,
                            new AddressConfiguration { ChannelType =
ChannelType.SMS }
                        }
                    },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    SMSMessage = new SMSMessage
                    {
```

```
                        Body = message,
                        MessageType = MessageType.TRANSACTIONAL,
                        OriginationNumber = originationNumber,
                        SenderId = senderId,
                        Keyword = keyword
                    }
                }
            }
        };
        SendMessagesResponse response = await
 client.SendMessagesAsync(sendRequest);
        return response;
    }
}
```

- Einzelheiten zur API finden Sie SendMessagesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So senden Sie eine SMS-Nachricht über den Endpunkt einer Anwendung

Im folgenden Beispiel für `send-messages` wird eine Direktnachricht für eine Anwendung mit einem Endpunkt gesendet.

```
aws pinpoint send-messages \
    --application-id 611e3e3cdd47474c9c1399a505665b91 \
    --message-request file://myfile.json \
    --region us-west-2
```

Inhalt von `myfile.json`:

```
{
    "MessageConfiguration": {
        "SMSMessage": {
            "Body": "hello, how are you?"
        }
    },
    "Endpoints": {
        "testendpoint": {}
```

```
    }
}
```

Ausgabe:

```
{
    "MessageResponse": {
        "ApplicationId": "611e3e3cdd47474c9c1399a505665b91",
        "EndpointResult": {
            "testendpoint": {
                "Address": "+12345678900",
                "DeliveryStatus": "SUCCESSFUL",
                "MessageId": "itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0",
                "StatusCode": 200,
                "StatusMessage": "MessageId:
 itnuqhai5alf1n6ahv3udc05n7hhddr6gb3lq6g0"
            }
        },
        "RequestId": "c7e23264-04b2-4a46-b800-d24923f74753"
    }
}
```

Weitere Informationen finden Sie unter Amazon-Pinpoint-SMS-Kanal im Benutzerhandbuch zu Amazon Pinpoint.

- Einzelheiten zur API finden Sie SendMessagesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
```

```
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendEmailMessage {

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
 content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS
 SDK for Java 2.x

        """;
```

```
        public static void main(String[] args) {
                final String usage = """

                                Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified
in Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendEmail(pinpoint, subject, senderAddress, toAddress);
        System.out.println("Email was sent");
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
```

```
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Senden einer E-Mail-Nachricht mit CC-Werten.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import  software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
```

```
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS
 SDK for Java 2.x

        """;
    public static void main(String[] args) {
        final String usage = """

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified
 in Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
               ccAddress - The CC address.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        String ccAddress = args[3];

        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
```

```
            .region(Region.US_EAST_1)
            .build();

        ArrayList<String> ccList = new ArrayList<>();
        ccList.add(ccAddress);
        sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses)
{
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .ccAddresses(ccAddresses)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");
```

```
        } catch (PinpointException e) {
            // Handle exception
            e.printStackTrace();
        }
    }
}
```

Senden Sie eine SMS-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";
```

```
        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
                                  message - The body of the message to send.
                                  appId - The Amazon Pinpoint project/application
 ID to use when you send this message.
                                  originationNumber - The phone number or
 short code that you specify has to be associated with your Amazon Pinpoint
 account. For best results, specify long codes in E.164 format (for example,
 +1-555-555-5654).
                                  destinationNumber - The recipient's phone
 number.  For best results, you should specify the phone number in E.164 format
 (for example, +1-555-555-5654).\s
                                  """;

                if (args.length != 4) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
 destinationNumber);
                pinpoint.close();
        }
```

```
        public static void sendSMSMessage(PinpointClient pinpoint, String
message, String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request =
SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();
```

```
                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" +
 v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Senden Sie Batch-SMS-Nachrichten.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
```

```
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
    public static String senderId = "MySenderID";

    public static void main(String[] args) {
        final String usage = """

                Usage:   <message> <appId> <originationNumber>
 <destinationNumber> <destinationNumber1>\s

                Where:
                  message - The body of the message to send.
                  appId - The Amazon Pinpoint project/application ID to use when
 you send this message.
                  originationNumber - The phone number or short code that
 you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
 results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).
                  destinationNumber1 - The second recipient's phone number.  For
 best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
```

```
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                      String originationNumber,
                                      String destinationNumber, String
destinationNumber1) {
        try {
            Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
            AddressConfiguration addConfig = AddressConfiguration.builder()
                    .channelType(ChannelType.SMS)
                    .build();

            // Add an entry to the Map object for each number to whom you want to
send a
            // message.
            addressMap.put(destinationNumber, addConfig);
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                    .body(message)
                    .messageType(messageType)
                    .originationNumber(originationNumber)
                    .senderId(senderId)
                    .keyword(registeredKeyword)
                    .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                    .smsMessage(smsMessage)
                    .build();

            MessageRequest msgReq = MessageRequest.builder()
                    .addresses(addressMap)
                    .messageConfiguration(direct)
                    .build();
```

```
            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                    .applicationId(appId)
                    .messageRequest(msgReq)
                    .build();

            SendMessagesResponse response = pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SendMessagesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
export const pinClient = new PinpointClient({ region: REGION });
```

Senden Sie eine E-Mail-Nachricht.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
const subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
const body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
const body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a>
 using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
const charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
```

```
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
  try {
    const { MessageResponse } = await pinClient.send(
      new SendMessagesCommand(params),
    );

    if (!MessageResponse) {
      throw new Error("No message response.");
    }

    if (!MessageResponse.Result) {
      throw new Error("No message result.");
    }

    const recipientResult = MessageResponse.Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    }
    console.log(recipientResult.MessageId);
  } catch (err) {
```

```
    console.log(err.message);
  }
};

run();
```

Senden Sie eine SMS-Nachricht.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

/* The phone number or short code to send the message from. The phone number
 or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number.  For best results, you should specify the phone
 number in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
marketing-related content, specify PROMOTIONAL.*/
const messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
const registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
```

```
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html.*/

const senderId = "MySenderID";

// Specify the parameters to pass to the API.
const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));
    console.log(
      `Message sent!
 ${data.MessageResponse.Result[destinationNumber].StatusMessage}`,
    );
  } catch (err) {
    console.log(err);
  }
};
run();
```

- Einzelheiten zur API finden Sie SendMessagesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
"use strict";

const AWS = require("aws-sdk");

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2";

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https:\/\/aws.amazon.com/sdk-for-node-js/`;
```

```
// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using
 the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";

// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: senderAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
```

```
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

//Try to send the email.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
  } else {
    console.log(
      "Email sent! Message ID: ",
      data["MessageResponse"]["Result"][toAddress]["MessageId"]
    );
  }
});
```

Senden Sie eine SMS-Nachricht.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
```

```
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number.  For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
var senderId = "MySenderID";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
```

```
var params = {
  ApplicationId: applicationId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

//Try to send the message.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log(
      "Message sent! " +
        data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
    );
  }
});
```

- Einzelheiten zur API finden Sie SendMessagesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <subject> <appId> <senderAddress> <toAddress>

    Where:
        subject - The email subject to use.
        senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
```

```
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
        }

    val emailContent =
        EmailContent {
            simple = message
```

```
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient.fromEnvironment { region = "us-east-1" }.use
 { pinpointemail ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- API-Details finden Sie SendMessagesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_email_message(
    pinpoint_client,
    app_id,
```

```
    sender,
    to_addresses,
    char_set,
    subject,
    html_message,
    text_message,
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
 account
                         is in the sandbox, these addresses must be verified.
    :param char_set: The character encoding to use for the subject line and
 message
                      body of the email.
    :param subject: The subject line of the email.
    :param html_message: The body of the email for recipients whose email clients
 can
                         display HTML content.
    :param text_message: The body of the email for recipients whose email clients
                         don't support HTML content.
    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {
                    "EmailMessage": {
                        "FromAddress": sender,
                        "SimpleEmail": {
                            "Subject": {"Charset": char_set, "Data": subject},
                            "HtmlPart": {"Charset": char_set, "Data":
 html_message},
```

```
                            "TextPart": {"Charset": char_set, "Data":
 text_message},
                        },
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]
["Result"].items()
        }


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)
    -------------------------------------
    This email was sent with Amazon Pinpoint using the AWS SDK for Python
 (Boto3).
    For more information, see https://aws.amazon.com/sdk-for-python/
                """
    html_message = """<html>
    <head></head>
    <body>
      <h1>Amazon Pinpoint Test (SDK for Python (Boto3)</h1>
      <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
          AWS SDK for Python (Boto3)</a>.</p>
    </body>
    </html>
                """

    print("Sending email.")
    message_ids = send_email_message(
```

```
        boto3.client("pinpoint"),
        app_id,
        sender,
        [to_address],
        char_set,
        subject,
        html_message,
        text_message,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Senden Sie eine SMS-Nachricht.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_sms_message(
    pinpoint_client,
    app_id,
    origination_number,
    destination_number,
    message,
    message_type,
):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you
 send
                   this message. The SMS channel must be enabled for the project
 or
                   application.
```

```
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This
 phone
                               number must be associated with your Amazon
 Pinpoint
                               account and be in E.164 format.
    :param message: The content of the SMS message.
    :param message_type: The type of SMS message that you want to send. If you
 send
                         time-sensitive content, specify TRANSACTIONAL. If you
 send
                         marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "Body": message,
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK
 for "
        "Python (Boto 3)."
```

```
    )
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
    message_id = send_sms_message(
        boto3.client("pinpoint"),
        app_id,
        origination_number,
        destination_number,
        message,
        message_type,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

Senden Sie eine E-Mail-Nachricht mit einer vorhandenen E-Mail-Vorlage.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_email_message(
    pinpoint_client, project_id, sender, to_addresses, template_name,
 template_version
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
                         account is in the sandbox, these addresses must be
 verified.
```

```
    :param template_name: The name of the email template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {"EmailMessage": {"FromAddress":
 sender}},
                "TemplateConfiguration": {
                    "EmailTemplate": {
                        "Name": template_name,
                        "Version": template_version,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]
["Result"].items()
        }


def main():
    project_id = "296b04b342374fceb661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send_templated_email_message(
```

```
        boto3.client("pinpoint"),
        project_id,
        sender,
        to_addresses,
        template_name,
        template_version,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Senden Sie eine Textnachricht mit einer vorhandenen SMS-Vorlage.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version,
):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the
 message.
    :param template_version: The version number of the message template.
```

```
    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
                "TemplateConfiguration": {
                    "SMSTemplate": {"Name": template_name, "Version":
 template_version}
                },
            },
        )

    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    region = "us-east-1"
    origination_number = "+18555550001"
    destination_number = "+14255550142"
    project_id = "7353f53e6885409fa32d07cedexample"
    message_type = "TRANSACTIONAL"
    template_name = "My_SMS_Template"
    template_version = "1"
    message_id = send_templated_sms_message(
        boto3.client("pinpoint", region_name=region),
        project_id,
        destination_number,
        message_type,
        origination_number,
        template_name,
```

```
        template_version,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie SendMessagesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
    " Build the addresses map from the list of to_addresses
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    LOOP AT it_to_addresses INTO DATA(lo_address).
      INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
        key = lo_address->get_value( )
        value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
      ) INTO TABLE lt_addresses.
    ENDLOOP.

    " Send the email message
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_emailmessage = NEW /aws1/cl_pptemailmessage(
            iv_fromaddress = iv_sender
            io_simpleemail = NEW /aws1/cl_pptsimpleemail(
```

```
            io_subject = NEW /aws1/cl_pptsimpleemailpart(
              iv_charset = iv_char_set
              iv_data = iv_subject
            )
            io_htmlpart = NEW /aws1/cl_pptsimpleemailpart(
              iv_charset = iv_char_set
              iv_data = iv_html_message
            )
            io_textpart = NEW /aws1/cl_pptsimpleemailpart(
              iv_charset = iv_char_set
              iv_data = iv_text_message
            )
          )
        )
      )
    )
  ).

  " Extract message IDs from response
  DATA(lo_message_response) = lo_result->get_messageresponse( ).
  ot_message_ids = lo_message_response->get_result( ).

  MESSAGE 'Email message sent successfully.' TYPE 'I'.
```

Senden Sie eine SMS-Nachricht.

```
  " Build the addresses map for the destination number
  DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = iv_destination_number
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
  ) INTO TABLE lt_addresses.

  " Send the SMS message
  DATA(lo_result) = lo_ppt->sendmessages(
    iv_applicationid = iv_app_id
    io_messagerequest = NEW /aws1/cl_pptmessagerequest(
      it_addresses = lt_addresses
      io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
        io_smsmessage = NEW /aws1/cl_pptsmsmessage(
          iv_body = iv_message
          iv_messagetype = iv_message_type
```

```
            iv_originationnumber = iv_origination_number
          )
        )
      )
    ).

    " Extract message ID from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    DATA(lt_results) = lo_message_response->get_result( ).
    LOOP AT lt_results INTO DATA(ls_result).
      IF ls_result-key = iv_destination_number.
        ov_message_id = ls_result-value->get_messageid( ).
        EXIT.
      ENDIF.
    ENDLOOP.

    MESSAGE 'SMS message sent successfully.' TYPE 'I'.
```

Senden Sie eine E-Mail-Nachricht mit einer vorhandenen E-Mail-Vorlage.

```
    " Build the addresses map from the list of to_addresses
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    LOOP AT it_to_addresses INTO DATA(lo_address).
      INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
        key = lo_address->get_value( )
        value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
      ) INTO TABLE lt_addresses.
    ENDLOOP.

    " Send the email message using a template
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_emailmessage = NEW /aws1/cl_pptemailmessage(
            iv_fromaddress = iv_sender
          )
        )
        io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
          io_emailtemplate = NEW /aws1/cl_ppttemplate(
            iv_name = iv_template_name
```

```
            iv_version = iv_template_version
          )
        )
      )
    ).

    " Extract message IDs from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    ot_message_ids = lo_message_response->get_result( ).

    MESSAGE 'Templated email message sent successfully.' TYPE 'I'.
```

Senden Sie eine Textnachricht mit einer vorhandenen SMS-Vorlage.

```
    " Build the addresses map for the destination number
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = iv_destination_number
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
    ) INTO TABLE lt_addresses.

    " Send the SMS message using a template
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_smsmessage = NEW /aws1/cl_pptsmsmessage(
            iv_messagetype = iv_message_type
            iv_originationnumber = iv_origination_number
          )
        )
        io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
          io_smstemplate = NEW /aws1/cl_ppttemplate(
            iv_name = iv_template_name
            iv_version = iv_template_version
          )
        )
      )
    ).

    " Extract message ID from response
```

```
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    DATA(lt_results) = lo_message_response->get_result( ).
    LOOP AT lt_results INTO DATA(ls_result).
      IF ls_result-key = iv_destination_number.
        ov_message_id = ls_result-value->get_messageid( ).
        EXIT.
      ENDIF.
    ENDLOOP.

    MESSAGE 'Templated SMS message sent successfully.' TYPE 'I'.
```

- Einzelheiten zur API finden Sie SendMessagesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateEndpoint`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateEndpoint`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
```

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                  appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }
```

```
   public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
       String endpointId = UUID.randomUUID().toString();
       System.out.println("Endpoint ID: " + endpointId);

       try {
           EndpointRequest endpointRequest = createEndpointRequestData();
           UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                   .applicationId(appId)
                   .endpointId(endpointId)
                   .endpointRequest(endpointRequest)
                   .build();

           UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
           System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

           GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                   .applicationId(appId)
                   .endpointId(endpointId)
                   .build();

           GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
           System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());

System.out.println(getEndpointResponse.endpointResponse().requestId());
           System.out.println(getEndpointResponse.endpointResponse().user());

           return getEndpointResponse.endpointResponse();

       } catch (PinpointException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
```

```
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                    .appVersion("1.0")
                    .make("apple")
                    .model("iPhone")
                    .modelVersion("7")
                    .platform("ios")
                    .platformVersion("10.1.1")
                    .timezone("America/Los_Angeles")
                    .build();

            EndpointLocation location = EndpointLocation.builder()
                    .city("Los Angeles")
                    .country("US")
                    .latitude(34.0)
                    .longitude(-118.2)
                    .postalCode("90068")
                    .region("CA")
                    .build();

            Map<String, Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                    .userId(UUID.randomUUID().toString())
                    .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); //
Quoted "Z" to indicate UTC, no timezone
                                                                           //
offset
            String nowAsISO = df.format(new Date());

            return EndpointRequest.builder()
```

```
                    .address(UUID.randomUUID().toString())
                    .attributes(customAttributes)
                    .channelType("APNS")
                    .demographic(demographic)
                    .effectiveDate(nowAsISO)
                    .location(location)
                    .metrics(metrics)
                    .optOut("NONE")
                    .requestId(UUID.randomUUID().toString())
                    .user(user)
                    .build();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Einzelheiten zur API finden Sie UpdateEndpointin der AWS SDK for Java 2.x API-Referenz.

Szenarien für die Verwendung von Amazon Pinpoint AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Pinpoint mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen in Amazon Pinpoint oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Push-Benachrichtigungen

Erste Schritte mit Push-Benachrichtigungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Anwendung
- Aktivieren Sie Push-Benachrichtigungskanäle
- Push-Benachrichtigung senden
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
```

```
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
```

```
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""

    return 0
}
```

```
# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
```

```
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters"
 >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi

check_json_tools
validate_permissions
```

```
# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "============================================"
echo "ENABLING FCM (GCM) CHANNEL"
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
```

```
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-
demo-only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server
 Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real
 certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --
apns-channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
```

```
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-
channel command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "==========================================="
echo "CREATING MESSAGE FILES"
echo "==========================================="

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
```

```
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens
 with actual ones."
```

```
# Step 5: Demonstrate how to send messages (this will fail with placeholder
 tokens)
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
```

```
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
```

```
echo "2. Attempting to enable FCM and APNS channels (with placeholder
 credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Codebeispiele für Amazon Pinpoint SMS und Voice API unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Pinpoint SMS and Voice API mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Amazon-Pinpoint-SMS- und -Sprachnachrichten-API – Weitere Informationen zur Amazon-Pinpoint-SMS- und -Sprachnachrichten-API.
- API-Referenz zur Amazon-Pinpoint-SMS- und -Sprachnachrichten-API – Details zu allen verfügbaren Aktionen der Amazon-Pinpoint-SMS- und -Sprachnachrichten-API.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Pinpoint SMS und Voice API AWS SDKs
  - Aktionen für Amazon Pinpoint SMS und Voice API unter Verwendung AWS SDKs
    - Verwenden Sie CreateConfigurationSetmit einem AWS SDK
    - Verwenden Sie CreateConfigurationSetEventDestinationmit einem AWS SDK
    - Verwenden Sie DeleteConfigurationSetmit einem AWS SDK
    - Verwenden Sie DeleteConfigurationSetEventDestinationmit einem AWS SDK
    - Verwenden Sie GetConfigurationSetEventDestinationsmit einem AWS SDK
    - Verwenden Sie ListConfigurationSetsmit einem AWS SDK
    - Verwenden Sie SendVoiceMessagemit einem AWS SDK
    - Verwenden Sie UpdateConfigurationSetEventDestinationmit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Pinpoint SMS und Voice API AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen der Amazon-Pinpoint-SMS- und -Sprachnachrichten-API mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Pinpoint SMS und Voice API unter Verwendung AWS SDKs
  - Verwenden Sie CreateConfigurationSetmit einem AWS SDK
  - Verwenden Sie CreateConfigurationSetEventDestinationmit einem AWS SDK
  - Verwenden Sie DeleteConfigurationSetmit einem AWS SDK
  - Verwenden Sie DeleteConfigurationSetEventDestinationmit einem AWS SDK
  - Verwenden Sie GetConfigurationSetEventDestinationsmit einem AWS SDK
  - Verwenden Sie ListConfigurationSetsmit einem AWS SDK
  - Verwenden Sie SendVoiceMessagemit einem AWS SDK
  - Verwenden Sie UpdateConfigurationSetEventDestinationmit einem AWS SDK

Aktionen für Amazon Pinpoint SMS und Voice API unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Pinpoint SMS- und Voice API-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Referenz zur Amazon-Pinpoint-SMS- und -Sprachnachrichten-API.

Beispiele

- Verwenden Sie CreateConfigurationSetmit einem AWS SDK
- Verwenden Sie CreateConfigurationSetEventDestinationmit einem AWS SDK
- Verwenden Sie DeleteConfigurationSetmit einem AWS SDK
- Verwenden Sie DeleteConfigurationSetEventDestinationmit einem AWS SDK
- Verwenden Sie GetConfigurationSetEventDestinationsmit einem AWS SDK
- Verwenden Sie ListConfigurationSetsmit einem AWS SDK
- Verwenden Sie SendVoiceMessagemit einem AWS SDK

- Verwenden Sie UpdateConfigurationSetEventDestinationmit einem AWS SDK

Verwenden Sie **`CreateConfigurationSet`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateConfigurationSet`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create a new configuration set
        lo_pps->createconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        MESSAGE 'Configuration set created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateConfigurationSetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateConfigurationSetEventDestination`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateConfigurationSetEventDestination`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create event destination for the configuration set
        lo_pps->createconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
```

```
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateConfigurationSetEventDestinationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteConfigurationSet`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteConfigurationSet`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Delete the configuration set
        lo_pps->deleteconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        MESSAGE 'Configuration set deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
```

```
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteConfigurationSetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteConfigurationSetEventDestination`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteConfigurationSetEventDestination`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Delete the event destination
        lo_pps->deleteconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
        ).

        MESSAGE 'Event destination deleted successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteConfigurationSetEventDestinationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetConfigurationSetEventDestinations`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetConfigurationSetEventDestinations`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Get event destinations for the configuration set
        oo_result = lo_pps->getconfseteventdestinations(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
        ).

        " Process the event destinations
```

```
        LOOP AT oo_result->get_eventdestinations( ) INTO DATA(lo_event_dest).
          DATA(lv_dest_name) = lo_event_dest->get_name( ).
          DATA(lv_enabled) = lo_event_dest->get_enabled( ).

          MESSAGE |Event destination: { lv_dest_name }, Enabled: { lv_enabled }|
 TYPE 'I'.

          " Check for CloudWatch Logs destination
          DATA(lo_cloudwatch_dest) = lo_event_dest->get_cloudwatchlogsdst( ).
          IF lo_cloudwatch_dest IS NOT INITIAL.
            DATA(lv_log_group_arn) = lo_cloudwatch_dest->get_loggrouparn( ).
            MESSAGE |  CloudWatch Logs destination: { lv_log_group_arn }| TYPE
 'I'.
          ENDIF.

          " Check for Kinesis Firehose destination
          DATA(lo_firehose_dest) = lo_event_dest->get_kinesisfirehosedst( ).
          IF lo_firehose_dest IS NOT INITIAL.
            DATA(lv_delivery_stream) = lo_firehose_dest-
>get_deliverystreamarn( ).
            MESSAGE |  Kinesis Firehose destination: { lv_delivery_stream }| TYPE
 'I'.
          ENDIF.

          " Check for SNS destination
          DATA(lo_sns_dest) = lo_event_dest->get_snsdestination( ).
          IF lo_sns_dest IS NOT INITIAL.
            DATA(lv_topic_arn) = lo_sns_dest->get_topicarn( ).
            MESSAGE |  SNS destination: { lv_topic_arn }| TYPE 'I'.
          ENDIF.
        ENDLOOP.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetConfigurationSetEventDestinationsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListConfigurationSets`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListConfigurationSets`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " List all configuration sets
        oo_result = lo_pps->listconfigurationsets(
          iv_nexttoken = iv_next_token    " Optional: Token for pagination
          iv_pagesize = iv_page_size      " Optional: Number of results per page,
e.g., '10'
        ).

        " Process the configuration sets
        LOOP AT oo_result->get_configurationsets( ) INTO DATA(lo_config_set).
          DATA(lv_config_set_name) = lo_config_set->get_value( ).
          MESSAGE |Configuration set: { lv_config_set_name }| TYPE 'I'.
        ENDLOOP.

        " Check if there are more results
        DATA(lv_next_token) = oo_result->get_nexttoken( ).
        IF lv_next_token IS NOT INITIAL.
          MESSAGE |More results available. Next token: { lv_next_token }| TYPE
'I'.
        ENDIF.
```

```
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListConfigurationSetsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendVoiceMessage`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SendVoiceMessage` verwendet wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
 list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and
 control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
            + "<emphasis>Amazon Pinpoint</emphasis> "
            + "using the <break strength='weak'/>AWS "
            + "SDK for Java. "
            + "<amazon:effect phonation='soft'>Thank "
            + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = """
                Usage:   <originationNumber> <destinationNumber>\s

                Where:
```

```
                 originationNumber - The phone number or short code that
you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                 destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
               """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }
       String originationNumber = args[0];
       String destinationNumber = args[1];
       System.out.println("Sending a voice message");

       // Set the content type to application/json.
       List<String> listVal = new ArrayList<>();
       listVal.add("application/json");
       Map<String, List<String>> values = new HashMap<>();
       values.put("Content-Type", listVal);

       ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
               .headers(values)
               .build();

       PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
               .overrideConfiguration(config2)
               .region(Region.US_EAST_1)
               .build();

       sendVoiceMsg(client, originationNumber, destinationNumber);
       client.close();
   }

   public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
                                   String destinationNumber) {
       try {
           SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                   .languageCode(languageCode)
                   .text(ssmlMessage)
                   .voiceId(voiceName)
```

```
                    .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                    .ssmlMessage(ssmlMessageType)
                    .build();

            SendVoiceMessageRequest voiceMessageRequest =
 SendVoiceMessageRequest.builder()
                    .destinationPhoneNumber(destinationNumber)
                    .originationPhoneNumber(originationNumber)
                    .content(content)
                    .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SendVoiceMessagein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"use strict";

var AWS = require("aws-sdk");
```

```
// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best
 results, you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/
SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage =
  "<speak>" +
  "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> " +
  "using the <break strength='weak'/>AWS SDK for JavaScript in Node.js. " +
  "<amazon:effect phonation='soft'>Thank you for listening." +
  "</amazon:effect>" +
  "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint
 account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";
```

```
// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpointsmsvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId,
    },
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber,
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! Message ID: " + data["MessageId"]);
  }
});
```

- Einzelheiten zur API finden Sie SendVoiceMessagein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_voice_message(
    sms_voice_client,
    origination_number,
    caller_id,
    destination_number,
    language_code,
    voice_id,
    ssml_message,
):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
                               The phone number must be associated with your
 Amazon
                               Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's
                      device. The phone number must be associated with your
 Amazon
                      Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone
                               number in E.164 format.
    :param language_code: The language to use when sending the message.
```

```
    :param voice_id: The Amazon Polly voice that you want to use to send the
 message.
    :param ssml_message: The content of the message. This example uses SSML to
 control
                         certain aspects of the message, such as the volume and
 the
                         speech rate. The message must not contain line breaks.
    :return: The ID of the message.
    """
    try:
        response = sms_voice_client.send_voice_message(
            DestinationPhoneNumber=destination_number,
            OriginationPhoneNumber=origination_number,
            CallerId=caller_id,
            Content={
                "SSMLMessage": {
                    "LanguageCode": language_code,
                    "VoiceId": voice_id,
                    "Text": ssml_message,
                }
            },
        )
    except ClientError:
        logger.exception(
            "Couldn't send message from %s to %s.",
            origination_number,
            destination_number,
        )
        raise
    else:
        return response["MessageId"]


def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "using the <break strength='weak'/>AWS SDK for Python (Boto3). "
        "<amazon:effect phonation='soft'>Thank you for listening."
```

```
        "</amazon:effect>"
        "</speak>"
    )
    print(f"Sending voice message from {origination_number} to
 {destination_number}.")
    message_id = send_voice_message(
        boto3.client("pinpoint-sms-voice"),
        origination_number,
        caller_id,
        destination_number,
        language_code,
        voice_id,
        ssml_message,
    )
    print(f"Message sent!\nMessage ID: {message_id}")


if __name__ == "__main__":
    main()
```

- Einzelheiten zur API finden Sie SendVoiceMessagein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create SSML message type object with voice parameters
        DATA(lo_ssml_message) = NEW /aws1/cl_ppsssmlmessagetype(
          iv_languagecode = iv_language_code    " e.g., 'en-US'
          iv_voiceid = iv_voice_id              " e.g., 'Matthew'
          iv_text = iv_ssml_message             " SSML formatted message text
        ).
```

```
      " Create voice message content with the SSML message
      DATA(lo_content) = NEW /aws1/cl_ppsvoicemessagecont(
        io_ssmlmessage = lo_ssml_message
      ).

      " Send the voice message
      DATA(lo_result) = lo_pps->sendvoicemessage(
        iv_originationphonenumber = iv_origination_number  " e.g.,
'+12065550110'
        iv_callerid = iv_caller_id                         " e.g.,
'+12065550199'
        iv_destinationphonenumber = iv_destination_number  " e.g.,
'+12065550142'
        io_content = lo_content
      ).

      " Retrieve the message ID from the response
      ov_message_id = lo_result->get_messageid( ).

      MESSAGE 'Voice message sent successfully.' TYPE 'I'.

    CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
      MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_internal_error_ex.
    CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
      MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many_requests_ex.
  ENDTRY.
```

- Einzelheiten zur API finden Sie SendVoiceMessagein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateConfigurationSetEventDestination`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdateConfigurationSetEventDestination`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Update the event destination
        lo_pps->updateconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-
config-set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-
event-dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination updated successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateConfigurationSetEventDestinationin der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für Amazon Polly mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Polly mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Polly-Entwicklerhandbuch – Weitere Informationen zu Amazon Polly.
- API-Referenz für Amazon Polly – Details zu allen verfügbaren Amazon-Polly-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Polly AWS SDKs
  - Aktionen für Amazon Polly mit AWS SDKs
    - Verwenden Sie DeleteLexiconmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeVoicesmit einem AWS SDK
    - Verwenden Sie GetLexiconmit einem AWS SDK oder CLI
    - Verwenden Sie GetSpeechSynthesisTaskmit einem AWS SDK oder CLI
    - Verwenden Sie ListLexiconsmit einem AWS SDK oder CLI
    - Verwenden Sie ListSpeechSynthesisTasksmit einem AWS SDK oder CLI
    - Verwenden Sie PutLexiconmit einem AWS SDK oder CLI
    - Verwenden Sie StartSpeechSynthesisTaskmit einem AWS SDK oder CLI
    - Verwenden Sie SynthesizeSpeechmit einem AWS SDK
- Szenarien für die Verwendung von Amazon Polly AWS SDKs
  - Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK

- Erstellen Sie eine Lippensynchronisierungsanwendung mit Amazon Polly mithilfe eines AWS SDK
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erste Schritte mit der Text-zu-Sprache-Synthese

Grundlegende Beispiele für die Verwendung von Amazon Polly AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Polly mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Polly mit AWS SDKs
  - Verwenden Sie DeleteLexiconmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeVoicesmit einem AWS SDK
  - Verwenden Sie GetLexiconmit einem AWS SDK oder CLI
  - Verwenden Sie GetSpeechSynthesisTaskmit einem AWS SDK oder CLI
  - Verwenden Sie ListLexiconsmit einem AWS SDK oder CLI
  - Verwenden Sie ListSpeechSynthesisTasksmit einem AWS SDK oder CLI
  - Verwenden Sie PutLexiconmit einem AWS SDK oder CLI
  - Verwenden Sie StartSpeechSynthesisTaskmit einem AWS SDK oder CLI
  - Verwenden Sie SynthesizeSpeechmit einem AWS SDK

Aktionen für Amazon Polly mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Polly Polly-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Polly API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Polly AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Polly.

Beispiele

- Verwenden Sie DeleteLexiconmit einem AWS SDK oder CLI
- Verwenden Sie DescribeVoicesmit einem AWS SDK
- Verwenden Sie GetLexiconmit einem AWS SDK oder CLI
- Verwenden Sie GetSpeechSynthesisTaskmit einem AWS SDK oder CLI
- Verwenden Sie ListLexiconsmit einem AWS SDK oder CLI
- Verwenden Sie ListSpeechSynthesisTasksmit einem AWS SDK oder CLI
- Verwenden Sie PutLexiconmit einem AWS SDK oder CLI
- Verwenden Sie StartSpeechSynthesisTaskmit einem AWS SDK oder CLI
- Verwenden Sie SynthesizeSpeechmit einem AWS SDK

Verwenden Sie **`DeleteLexicon`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteLexicon` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Deletes an existing Amazon Polly lexicon using the AWS SDK for .NET.
```

```
    /// </summary>
    public class DeleteLexicon
    {
        public static async Task Main()
        {
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();

            var success = await DeletePollyLexiconAsync(client, lexiconName);

            if (success)
            {
                Console.WriteLine($"Successfully deleted {lexiconName}.");
            }
            else
            {
                Console.WriteLine($"Could not delete {lexiconName}.");
            }
        }

        /// <summary>
        /// Deletes the named Amazon Polly lexicon.
        /// </summary>
        /// <param name="client">The initialized Amazon Polly client object.</
param>
        /// <param name="lexiconName">The name of the Amazon Polly lexicon to
        /// delete.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeletePollyLexiconAsync(
            AmazonPollyClient client,
            string lexiconName)
        {
            var deleteLexiconRequest = new DeleteLexiconRequest()
            {
                Name = lexiconName,
            };

            var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteLexiconin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie ein Lexikon

Im folgenden Beispiel für `delete-lexicon` wird das angegebene Lexikon gelöscht.

```
aws polly delete-lexicon \
    --name w3c
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden des DeleteLexicon Vorgangs im Amazon Polly Developer Guide.

- Einzelheiten zur API finden Sie unter DeleteLexicon AWS CLIBefehlsreferenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ply->deletelexicon( iv_name ).
        MESSAGE 'Lexicon deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteLexiconin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeVoices`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeVoices` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };
```

```
            try
            {
                string nextToken;
                do
                {
                    var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                    nextToken = allVoicesResponse.NextToken;
                    allVoicesRequest.NextToken = nextToken;

                    Console.WriteLine("\nAll voices: ");
                    allVoicesResponse.Voices.ForEach(voice =>
                    {
                        DisplayVoiceInfo(voice);
                    });
                }
                while (nextToken is not null);

                do
                {
                    var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                    nextToken = enUsVoicesResponse.NextToken;
                    enUsVoicesRequest.NextToken = nextToken;

                    Console.WriteLine("\nen-US voices: ");
                    enUsVoicesResponse.Voices.ForEach(voice =>
                    {
                        DisplayVoiceInfo(voice);
                    });
                }
                while (nextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }

        public static void DisplayVoiceInfo(Voice voice)
        {
            Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
        }
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeVoicesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        describeVoice(polly);
        polly.close();
    }
```

```
    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                    .languageCode("en-US")
                    .build();

            DescribeVoicesResponse enUsVoicesResult =
 polly.describeVoices(voicesRequest);
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
 myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeVoicesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
```

```
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def describe_voices(self):
        """
        Gets metadata about available voices.

        :return: The list of voice metadata.
        """
        try:
            response = self.polly_client.describe_voices()
            self.voice_metadata = response["Voices"]
            logger.info("Got metadata about %s voices.",
len(self.voice_metadata))
        except ClientError:
            logger.exception("Couldn't get voice metadata.")
            raise
        else:
            return self.voice_metadata
```

- Einzelheiten zur API finden Sie DescribeVoicesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie DescribeVoicesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_voices().send().await?;

    println!("Voices:");

    let voices = resp.voices();
```

```
    for voice in voices {
        println!("  Name:     {}", voice.name().unwrap_or("No name!"));
        println!(
            "  Language: {}",
            voice.language_name().unwrap_or("No language!")
        );

        println!();
    }

    println!("Found {} voices", voices.len());

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeVoicesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_engine IS NOT INITIAL AND iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine
            iv_languagecode = iv_language ).
        ELSEIF iv_engine IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine ).
        ELSEIF iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_languagecode = iv_language ).
        ELSE.
          oo_result = lo_ply->describevoices( ).
```

```
      ENDIF.
      MESSAGE 'Retrieved voice metadata.' TYPE 'I'.
    CATCH /aws1/cx_plyinvalidnexttokenex.
      MESSAGE 'The NextToken is invalid.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeVoicesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetLexicon`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetLexicon` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Retrieves information about a specific Amazon Polly lexicon.
/// </summary>
public class GetLexicon
```

```
    {
        public static async Task Main(string[] args)
        {
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();

            await GetPollyLexiconAsync(client, lexiconName);
        }

        public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
 string lexiconName)
        {
            var getLexiconRequest = new GetLexiconRequest()
            {
                Name = lexiconName,
            };

            try
            {
                var response = await client.GetLexiconAsync(getLexiconRequest);
                Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
                Console.WriteLine($"Content: {response.Lexicon.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: " + ex.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetLexiconin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den Inhalt eines Lexikons ab

Im folgenden Beispiel für `get-lexicon` wird der Inhalt des angegebenen Aussprache-Lexikons abgerufen.

```
aws polly get-lexicon \
    --name w3c
```

Ausgabe:

```
{
    "Lexicon": {
        "Content": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<lexicon version=
\"1.0\" \n      xmlns=    \"http://www.w3.org/2005/01/pronunciation-lexicon
\"\n      xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" \n
 xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon \n
 http://www.w3.org/TR/2007/CR-pronunciation-    lexicon-20071212/pls.xsd\"\n
   alphabet=\"ipa\" \n      xml:lang=\"en-US\">\n  <lexeme>\n    <grapheme>W3C</
grapheme>\n        <alias>World Wide Web Consortium</alias>\n  </lexeme>\n</
lexicon>\n",
        "Name": "w3c"
    },
    "LexiconAttributes": {
        "Alphabet": "ipa",
        "LanguageCode": "en-US",
        "LastModified": 1603908910.99,
        "LexiconArn": "arn:aws:polly:us-west-2:880185128111:lexicon/w3c",
        "LexemesCount": 1,
        "Size": 492
    }
}
```

Weitere Informationen finden Sie unter Verwenden des GetLexicon Vorgangs im Amazon Polly Developer Guide.

- Einzelheiten zur API finden Sie unter GetLexicon AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie GetLexiconin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ply->getlexicon( iv_name ).
        DATA(lo_lexicon) = oo_result->get_lexicon( ).
        IF lo_lexicon IS BOUND.
          DATA(lv_lex_name) = lo_lexicon->get_name( ).
          MESSAGE |Retrieved lexicon: { lv_lex_name }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetLexiconin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetSpeechSynthesisTask`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSpeechSynthesisTask` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen über eine Sprachsyntheseaufgabe ab

Im folgenden Beispiel für `get-speech-synthesis-task` werden Informationen über die angegebene Sprachsyntheseaufgabe abgerufen.

```
aws polly get-speech-synthesis-task \
```

```
    --task-id 70b61c0f-57ce-4715-a247-cae8729dcce9
```

Ausgabe:

```
{
    "SynthesisTask": {
        "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
        "TaskStatus": "completed",
        "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
        "CreationTime": 1603911042.689,
        "RequestCharacters": 1311,
        "OutputFormat": "mp3",
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

Weitere Informationen finden Sie unter Erstellen von langen Audiodateien im Benutzerhandbuch für Amazon Polly.

- Einzelheiten zur API finden Sie GetSpeechSynthesisTaskin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
```

```
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response =
self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response["SynthesisTask"]
            logger.info("Got synthesis task. Status is %s.", task["TaskStatus"])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- Einzelheiten zur API finden Sie GetSpeechSynthesisTaskin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
      oo_result = lo_ply->getspeechsynthesistask( iv_task_id ).
      DATA(lo_task) = oo_result->get_synthesistask( ).
      IF lo_task IS BOUND.
        DATA(lv_status) = lo_task->get_taskstatus( ).
        MESSAGE |Task status: { lv_status }| TYPE 'I'.
      ENDIF.
    CATCH /aws1/cx_plyinvalidtaskidex.
      MESSAGE 'Invalid task ID.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
    CATCH /aws1/cx_plysynthesistsknotf00.
      MESSAGE 'Synthesis task not found.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie GetSpeechSynthesisTaskin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListLexicons`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListLexicons` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;
```

```
    /// <summary>
    /// Lists the Amazon Polly lexicons that have been defined. By default,
    /// lists the lexicons that are defined in the same AWS Region as the default
    /// user. To view Amazon Polly lexicons that are defined in a different AWS
    /// Region, supply it as a parameter to the Amazon Polly constructor.
    /// </summary>
    public class ListLexicons
    {
        public static async Task Main()
        {
            var client = new AmazonPollyClient();
            var request = new ListLexiconsRequest();

            try
            {
                Console.WriteLine("All voices: ");

                do
                {
                    var response = await client.ListLexiconsAsync(request);
                    request.NextToken = response.NextToken;

                    response.Lexicons.ForEach(lexicon =>
                    {
                        var attributes = lexicon.Attributes;
                        Console.WriteLine($"Name: {lexicon.Name}");
                        Console.WriteLine($"\tAlphabet: {attributes.Alphabet}");
                        Console.WriteLine($"\tLanguageCode:
{attributes.LanguageCode}");
                        Console.WriteLine($"\tLastModified:
{attributes.LastModified}");
                        Console.WriteLine($"\tLexemesCount:
{attributes.LexemesCount}");
                        Console.WriteLine($"\tLexiconArn:
{attributes.LexiconArn}");
                        Console.WriteLine($"\tSize: {attributes.Size}");
                    });
                }
                while (request.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
```

```
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListLexiconsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Ihre Lexika auf

Im folgenden Beispiel für `list-lexicons` werden Ihre Aussprachelexika aufgelistet.

```
aws polly list-lexicons
```

Ausgabe:

```
{
    "Lexicons": [
        {
            "Name": "w3c",
            "Attributes": {
                "Alphabet": "ipa",
                "LanguageCode": "en-US",
                "LastModified": 1603908910.99,
                "LexiconArn": "arn:aws:polly:us-east-2:123456789012:lexicon/w3c",
                "LexemesCount": 1,
                "Size": 492
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden des ListLexicons Vorgangs im Amazon Polly Developer Guide.

- Einzelheiten zur API finden Sie unter ListLexicons AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest =
 ListLexiconsRequest.builder()
                    .build();
```

```
            ListLexiconsResponse listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
 listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
 lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListLexiconsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None
```

```
    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response["Lexicons"]
            logger.info("Got %s lexicons.", len(lexicons))
        except ClientError:
            logger.exception(
                "Couldn't get  %s.",
            )
            raise
        else:
            return lexicons
```

- Einzelheiten zur API finden Sie ListLexiconsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
```

```
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie ListLexiconsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons();

    for lexicon in lexicons {
        println!("  Name:     {}", lexicon.name().unwrap_or_default());
        println!(
            "  Language: {:?}\n",
            lexicon
                .attributes()
```

```
                .as_ref()
                .map(|attrib| attrib
                    .language_code
                    .as_ref()
                    .expect("languages must have language codes"))
                .expect("languages must have attributes")
        );
    }

    println!();
    println!("Found {} lexicons.", lexicons.len());
    println!();

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListLexiconsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ply->listlexicons( ).
        DATA(lt_lexicons) = oo_result->get_lexicons( ).
        DATA(lv_count) = lines( lt_lexicons ).
        MESSAGE |Found { lv_count } lexicons| TYPE 'I'.
      CATCH /aws1/cx_plyinvalidnexttokenex.
        MESSAGE 'Invalid NextToken.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListLexiconsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListSpeechSynthesisTasks`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListSpeechSynthesisTasks` verwendet wird.

CLI

AWS CLI

So listen Sie Ihre Sprachsyntheseaufgaben auf

Im folgenden Beispiel für `list-speech-synthesis-tasks` werden Ihre Sprachsyntheseaufgaben aufgelistet.

```
aws polly list-speech-synthesis-tasks
```

Ausgabe:

```
{
    "SynthesisTasks": [
        {
            "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
            "TaskStatus": "completed",
            "OutputUri": "https://s3.us-west-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
            "CreationTime": 1603911042.689,
            "RequestCharacters": 1311,
            "OutputFormat": "mp3",
            "TextType": "text",
            "VoiceId": "Joanna"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen von langen Audiodateien im Benutzerhandbuch für Amazon Polly.

- Einzelheiten zur API finden Sie ListSpeechSynthesisTasksin der AWS CLI Befehlsreferenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Only pass optional parameters if they have values
    IF iv_max_results IS NOT INITIAL AND iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results
        iv_status = iv_status ).
    ELSEIF iv_max_results IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_maxresults = iv_max_results ).
    ELSEIF iv_status IS NOT INITIAL.
      oo_result = lo_ply->listspeechsynthesistasks(
        iv_status = iv_status ).
    ELSE.
      oo_result = lo_ply->listspeechsynthesistasks( ).
    ENDIF.
    DATA(lt_tasks) = oo_result->get_synthesistasks( ).
    DATA(lv_count) = lines( lt_tasks ).
    MESSAGE |Found { lv_count } synthesis tasks| TYPE 'I'.
  CATCH /aws1/cx_plyinvalidnexttokenex.
    MESSAGE 'Invalid NextToken.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListSpeechSynthesisTasksin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutLexicon`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutLexicon` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    /// <summary>
    /// Creates a new Amazon Polly lexicon using the AWS SDK for .NET.
    /// </summary>
    public class PutLexicon
    {
        public static async Task Main()
        {
            string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
 +
                "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/
pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
 +
                "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-
lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
                "alphabet=\"ipa\" xml:lang=\"en-US\">" +
                "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>"
 +
                "</lexicon>";
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();
```

```
            var putLexiconRequest = new PutLexiconRequest()
            {
                Name = lexiconName,
                Content = lexiconContent,
            };

            try
            {
                var response = await client.PutLexiconAsync(putLexiconRequest);
                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine($"Successfully created Lexicon:
 {lexiconName}.");
                }
                else
                {
                    Console.WriteLine($"Could not create Lexicon:
 {lexiconName}.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie PutLexiconin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So speichern Sie ein Lexikon

Im folgenden Beispiel für `put-lexicon` wird das angegebene Aussprachelexikon gespeichert. Die `example.pls` Datei spezifiziert ein PLS-compliant W3C-Lexikon.

```
aws polly put-lexicon \
    --name w3c \
```

```
    --content file://example.pls
```

Inhalt von `example.pls`

```
{
    <?xml version="1.0" encoding="UTF-8"?>
    <lexicon version="1.0"
        xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
            http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
        alphabet="ipa"
        xml:lang="en-US">
        <lexeme>
            <grapheme>W3C</grapheme>
            <alias>World Wide Web Consortium</alias>
        </lexeme>
    </lexicon>
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verwenden des PutLexicon Vorgangs im Amazon Polly Developer Guide.

- Einzelheiten zur API finden Sie unter PutLexicon AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
```

```
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def create_lexicon(self, name, content):
        """
        Creates a lexicon with the specified content. A lexicon contains custom
        pronunciations.

        :param name: The name of the lexicon.
        :param content: The content of the lexicon.
        """
        try:
            self.polly_client.put_lexicon(Name=name, Content=content)
            logger.info("Created lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't create lexicon %s.")
            raise
```

- Einzelheiten zur API finden Sie PutLexiconin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
 Result<(), Error> {
    let content = format!("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
```

```
    <lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-
lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
    xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://
www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\"
    alphabet=\"ipa\" xml:lang=\"en-US\">
    <lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
    </lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- Einzelheiten zur API finden Sie PutLexiconin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ply->putlexicon(
          iv_name = iv_name
          iv_content = iv_content ).
        MESSAGE 'Lexicon created successfully.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidlexiconex.
        MESSAGE 'Invalid lexicon.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconsizeexcdex.
        MESSAGE 'Lexicon size exceeded.' TYPE 'E'.
```

```
      CATCH /aws1/cx_plymaxlexemelengthe00.
        MESSAGE 'Maximum lexeme length exceeded.' TYPE 'E'.
      CATCH /aws1/cx_plymaxlexiconsnoexc00.
        MESSAGE 'Maximum number of lexicons exceeded.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plyunsuppedplsalpha00.
        MESSAGE 'Unsupported PLS alphabet.' TYPE 'E'.
      CATCH /aws1/cx_plyunsuppedplslangu00.
        MESSAGE 'Unsupported PLS language.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutLexiconin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartSpeechSynthesisTask`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartSpeechSynthesisTask` verwendet wird.

CLI

AWS CLI

So synthetisieren Sie Text

Im folgenden Beispiel für `start-speech-synthesis-task` wird der Text in `text_file.txt` synthetisiert und die resultierende MP3-Datei im angegebenen Bucket gespeichert .

```
aws polly start-speech-synthesis-task \
    --output-format mp3 \
    --output-s3-bucket-name amzn-s3-demo-bucket \
    --text  file://text_file.txt \
    --voice-id Joanna
```

Ausgabe:

```
{
    "SynthesisTask": {
        "TaskId": "70b61c0f-57ce-4715-a247-cae8729dcce9",
        "TaskStatus": "scheduled",
```

```
        "OutputUri": "https://s3.us-east-2.amazonaws.com/amzn-s3-demo-
bucket/70b61c0f-57ce-4715-a247-cae8729dcce9.mp3",
        "CreationTime": 1603911042.689,
        "RequestCharacters": 1311,
        "OutputFormat": "mp3",
        "TextType": "text",
        "VoiceId": "Joanna"
    }
}
```

Weitere Informationen finden Sie unter Erstellen von langen Audiodateien im Benutzerhandbuch für Amazon Polly.

- Einzelheiten zur API finden Sie StartSpeechSynthesisTaskin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def do_synthesis_task(
        self,
        text,
```

```
        engine,
        voice,
        audio_format,
        s3_bucket,
        lang_code=None,
        include_visemes=False,
        wait_callback=None,
    ):
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real
time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
When
                             speech marks are synthesized, the output format is
JSON.
        :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                          write access to. Synthesis output is written to this
bucket.
        :param lang_code: The language code of the voice to use. This has an
effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon
Polly
                                to synthesize a list of visemes, using the
specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :param wait_callback: A callback function that is called periodically
during
                              task processing, to give the caller an opportunity
to
                              take action, such as to display status.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
```

```
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "OutputS3BucketName": s3_bucket,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.start_speech_synthesis_task(**kwargs)
            speech_task = response["SynthesisTask"]
            logger.info("Started speech synthesis task %s.",
speech_task["TaskId"])

            viseme_task = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response =
self.polly_client.start_speech_synthesis_task(**kwargs)
                viseme_task = response["SynthesisTask"]
                logger.info("Started viseme synthesis task %s.",
viseme_task["TaskId"])
        except ClientError:
            logger.exception("Couldn't start synthesis task.")
            raise
        else:
            bucket = self.s3_resource.Bucket(s3_bucket)
            audio_stream = self._wait_for_task(
                10, speech_task["TaskId"], "speech", wait_callback, bucket
            )

            visemes = None
            if include_visemes:
                viseme_data = self._wait_for_task(
                    10, viseme_task["TaskId"], "viseme", wait_callback, bucket
                )
                visemes = [
                    json.loads(v) for v in viseme_data.read().decode().split() if
v
                ]
```

```
            return audio_stream, visemes
```

- Einzelheiten zur API finden Sie StartSpeechSynthesisTaskin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_lang_code IS NOT INITIAL AND iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
```

```
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ELSE.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesis task started.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalids3bucketex.
        MESSAGE 'Invalid S3 bucket.' TYPE 'E'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartSpeechSynthesisTaskin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SynthesizeSpeech`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SynthesizeSpeech` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Text-zu-Sprache-Synthese

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   using System;
   using System.IO;
   using System.Threading.Tasks;
   using Amazon.Polly;
   using Amazon.Polly.Model;

   public class SynthesizeSpeech
   {
       public static async Task Main()
       {
           string outputFileName = "speech.mp3";
           string text = "Twas brillig, and the slithy toves did gyre and gimbol
in the wabe";

           var client = new AmazonPollyClient();
           var response = await PollySynthesizeSpeech(client, text);

           WriteSpeechToStream(response.AudioStream, outputFileName);
       }

       /// <summary>
       /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
       /// to speech.
       /// </summary>
       /// <param name="client">The Amazon Polly client object used to connect
       /// to the Amazon Polly service.</param>
       /// <param name="text">The text to convert to speech.</param>
       /// <returns>A SynthesizeSpeechResponse object that includes an
AudioStream
       /// object with the converted text.</returns>
       private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
```

```
        {
            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Mp3,
                VoiceId = VoiceId.Joanna,
                Text = text,
            };

            var synthesizeSpeechResponse =
                await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

            return synthesizeSpeechResponse;
        }

        /// <summary>
        /// Writes the AudioStream returned from the call to
        /// SynthesizeSpeechAsync to a file in MP3 format.
        /// </summary>
        /// <param name="audioStream">The AudioStream returned from the
        /// call to the SynthesizeSpeechAsync method.</param>
        /// <param name="outputFileName">The full path to the file in which to
        /// save the audio stream.</param>
        private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
        {
            var outputStream = new FileStream(
                outputFileName,
                FileMode.Create,
                FileAccess.Write);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
            {
                outputStream.Write(buffer, 0, readBytes);
            }

            // Flushes the buffer to avoid losing the last second or so of
            // the synthesized text.
            outputStream.Flush();
            Console.WriteLine($"Saved {outputFileName} to disk.");
        }
    }
```

Synthetisieren Sie Sprache aus Text mithilfe von Sprachmarken mit Amazon Polly mithilfe eines AWS SDK.

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    public class SynthesizeSpeechMarks
    {
        public static async Task Main()
        {
            var client = new AmazonPollyClient();
            string outputFileName = "speechMarks.json";

            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Json,
                SpeechMarkTypes = new List<string>
                {
                    SpeechMarkType.Viseme,
                    SpeechMarkType.Word,
                },
                VoiceId = VoiceId.Joanna,
                Text = "This is a sample text to be synthesized.",
            };

            try
            {
                using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
                {
                    var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                    var buffer = new byte[2 * 1024];
                    int readBytes;

                    var inputStream = synthesizeSpeechResponse.AudioStream;
```

```
                    while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) >
 0)
                    {
                        outputStream.Write(buffer, 0, readBytes);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie unter SynthesizeSpeech AWS SDK für .NETAPI-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
```

```
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
 built this working demo " +
            " of Amazon Polly in Java Version 2. Have fun building voice enabled
 apps with Amazon Polly (that's me!), and always "
            +
            " look at the AWS website for tips and tricks on using Amazon Polly
 and other great services from AWS";

    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
 DescribeVoicesRequest.builder()
                    .engine("standard")
                    .build();

            DescribeVoicesResponse describeVoicesResult =
 polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                    .filter(v -> v.name().equals("Joanna"))
                    .findFirst()
                    .orElseThrow(() -> new RuntimeException("Voice not found"));
```

```
            InputStream stream = synthesize(polly, SAMPLE, voice,
 OutputFormat.MP3);
            AdvancedPlayer player = new AdvancedPlayer(stream,

 javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlayBackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }

                public void playbackFinished(PlaybackEvent evt) {
                    System.out.println("Playback finished");
                }
            });

            // play it!
            player.play();

        } catch (PollyException | JavaLayerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static InputStream synthesize(PollyClient polly, String text, Voice
 voice, OutputFormat format)
            throws IOException {
        SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
                .text(text)
                .voiceId(voice.id())
                .outputFormat(format)
                .build();

        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
 polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def synthesize(
        self, text, engine, voice, audio_format, lang_code=None,
 include_visemes=False
    ):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
 When
                                speech marks are synthesized, the output format is
 JSON.
        :param lang_code: The language code of the voice to use. This has an
 effect
                            only when a bilingual voice is selected.
```

```
        :param include_visemes: When True, a second request is made to Amazon
Polly
                                to synthesize a list of visemes, using the
specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.synthesize_speech(**kwargs)
            audio_stream = response["AudioStream"]
            logger.info("Got audio stream spoken by %s.", voice)
            visemes = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.synthesize_speech(**kwargs)
                visemes = [
                    json.loads(v)
                    for v in response["AudioStream"].read().decode().split()
                    if v
                ]
                logger.info("Got %s visemes.", len(visemes))
        except ClientError:
            logger.exception("Couldn't get audio stream.")
            raise
        else:
            return audio_stream, visemes
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/
config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                   output_format: 'mp3',
                                   text: contents,
                                   voice_id: 'Joanna'
                                 })
```

```
  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;
```

```
    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional language code if it has a value
        IF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
```

```
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSE.
          oo_result = lo_ply->synthesizespeech(
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesized successfully.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SynthesizeSpeechin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Polly AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Polly mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Polly oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK
- Erstellen Sie eine Lippensynchronisierungsanwendung mit Amazon Polly mithilfe eines AWS SDK

- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erste Schritte mit der Text-zu-Sprache-Synthese

Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

Rust

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, die Audiodatei in einen Amazon S3 S3-Bucket hochzuladen, Amazon Transcribe zu verwenden, um diese Audiodatei in Text zu konvertieren und den Text anzuzeigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Erstellen Sie eine Lippensynchronisierungsanwendung mit Amazon Polly mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine Lippensynchronisationsanwendung mit Amazon Polly erstellen.

Python

SDK für Python (Boto3)

Zeigt, wie Amazon Polly und Tkinter verwendet werden, um eine Lippensynchronisierungsanwendung zu erstellen, die ein animiertes sprechendes Gesicht zusammen mit der von Amazon Polly synthetisierten Sprache anzeigt. Lip-sync wird erreicht, indem bei Amazon Polly eine Liste von Visemen angefordert wird, die mit der synthetisierten Sprache übereinstimmen.

- Erhalten Sie Sprachmetadaten von Amazon Polly und zeigen Sie diese in einer Tkinter-Anwendung an.
- Erhalten Sie synthetisierte Sprachaufnahmen und passende Viseme-Sprachmarkierungen von Amazon Polly.
- Spielen Sie die Audioaufnahme mit synchronisierten Mundbewegungen in einem animierten Gesicht ab.
- Asynchrone Syntheseaufgaben für lange Texte übermitteln und die Ausgabe aus einem Bucket von Amazon Simple Storage Service (Amazon S3) abrufen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly

Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

Die folgenden Codebeispiele zeigen, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus ihrer Originalsprache übersetzt, ihre Stimmung ermittelt und aus dem übersetzten Text eine Audiodatei generiert.

.NET

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen.

Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie der innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
```

```
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
```

```
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
```

```
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Erste Schritte mit der Text-zu-Sprache-Synthese

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Polly Getting Started Script
# This script demonstrates how to use Amazon Polly with the AWS CLI

set -euo pipefail

# Set up logging
LOG_FILE="polly-tutorial.log"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
WORK_DIR=$(mktemp -d)
trap 'cleanup_temp' EXIT

cleanup_temp() {
    rm -rf "$WORK_DIR"
}

echo "Starting Amazon Polly tutorial at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    # Use bash array to safely handle arguments
    bash -c "$1" 2>&1 | tee -a "$LOG_FILE" || return $?
}

# Function to check for errors
check_error() {
    if echo "$1" | grep -iq "error"; then
        echo "ERROR detected in output. Exiting script." | tee -a "$LOG_FILE"
        echo "$1" | tee -a "$LOG_FILE"
```

```
        return 1
    fi
    return 0
}

# Function to handle errors and cleanup
handle_error() {
    local line_number=$1
    echo "Error occurred at line $line_number. Attempting cleanup..." | tee -a
 "$LOG_FILE"
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"

    # Delete lexicon if it exists
    if [[ -n "${LEXICON_NAME:-}" ]]; then
        echo "Deleting lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
        if aws polly delete-lexicon --name "$LEXICON_NAME" 2>&1 | tee -a
 "$LOG_FILE"; then
            echo "Lexicon deleted successfully." | tee -a "$LOG_FILE"
        else
            echo "Warning: Failed to delete lexicon." | tee -a "$LOG_FILE"
        fi
    fi

    # Remove audio files
    for file in output.mp3 ssml-output.mp3 lexicon-output.mp3 example.pls; do
        if [[ -f "$file" ]]; then
            rm -f "$file"
            echo "Removed $file" | tee -a "$LOG_FILE"
        fi
    done

    echo "Cleanup complete." | tee -a "$LOG_FILE"
}
```

```
# Trap errors with line number
trap 'handle_error ${LINENO}' ERR

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "AWS CLI is not installed. Please install it first." | tee -a
 "$LOG_FILE"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "AWS credentials are not configured. Please configure them first." | tee
 -a "$LOG_FILE"
    exit 1
fi

# Step 1: Verify Amazon Polly is available
echo "Step 1: Verifying Amazon Polly availability" | tee -a "$LOG_FILE"
if aws polly describe-voices --query 'Voices[0].Name' --output text &> /dev/null;
 then
    echo "Amazon Polly is available. Proceeding with tutorial." | tee -a
 "$LOG_FILE"
else
    echo "Amazon Polly is not available in your AWS CLI installation or region."
 | tee -a "$LOG_FILE"
    echo "Please update your AWS CLI to the latest version or check your region."
 | tee -a "$LOG_FILE"
    exit 1
fi

# Step 2: List available voices
echo "" | tee -a "$LOG_FILE"
echo "Step 2: Listing available voices" | tee -a "$LOG_FILE"
log_cmd "aws polly describe-voices --language-code en-US --output text --query
 'Voices[0:3].[Id, LanguageCode, Gender]'" || true

# Step 3: Basic text-to-speech conversion
echo "" | tee -a "$LOG_FILE"
echo "Step 3: Converting text to speech" | tee -a "$LOG_FILE"
OUTPUT_FILE="${WORK_DIR}/output.mp3"
POLLY_TEXT="Hello, welcome to Amazon Polly. This is a sample text to speech
 conversion."
```

```
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --text
 '$POLLY_TEXT' '$OUTPUT_FILE'" || true

if [[ -f "$OUTPUT_FILE" ]]; then
    echo "Successfully created output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$OUTPUT_FILE" output.mp3
else
    echo "Failed to create output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 4: Using SSML for enhanced speech
echo "" | tee -a "$LOG_FILE"
echo "Step 4: Using SSML for enhanced speech" | tee -a "$LOG_FILE"
SSML_OUTPUT="${WORK_DIR}/ssml-output.mp3"
SSML_TEXT='<speak>Hello! <break time="1s"/> This is a sample of <emphasis>SSML
 enhanced speech</emphasis>.</speak>'
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Matthew --
text-type ssml --text '$SSML_TEXT' '$SSML_OUTPUT'" || true

if [[ -f "$SSML_OUTPUT" ]]; then
    echo "Successfully created ssml-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$SSML_OUTPUT" ssml-output.mp3
else
    echo "Failed to create ssml-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 5: Working with lexicons
echo "" | tee -a "$LOG_FILE"
echo "Step 5: Working with lexicons" | tee -a "$LOG_FILE"

# Generate a random identifier for the lexicon (max 20 chars, alphanumeric only)
LEXICON_NAME="example$(openssl rand -hex 6 | cut -c 1-10)"
echo "Using lexicon name: $LEXICON_NAME" | tee -a "$LOG_FILE"

# Create a lexicon file
echo "Creating lexicon file..." | tee -a "$LOG_FILE"
LEXICON_FILE="${WORK_DIR}/example.pls"
cat > "$LEXICON_FILE" << 'EOF'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa"
      xml:lang="en-US">
  <lexeme>
    <grapheme>AWS</grapheme>
    <alias>Amazon Web Services</alias>
  </lexeme>
</lexicon>
EOF

# Upload the lexicon
echo "Uploading lexicon..." | tee -a "$LOG_FILE"
LEXICON_ARN=$(aws polly put-lexicon --name "$LEXICON_NAME" --content
 file://"$LEXICON_FILE" --query 'LexiconArn' --output text 2>&1) || true
if [[ -n "$LEXICON_ARN" && "$LEXICON_ARN" != "" && ! "$LEXICON_ARN" =~ error ]];
 then
    echo "Lexicon uploaded with ARN: $LEXICON_ARN" | tee -a "$LOG_FILE"
    aws polly tag-resource --resource-arn "$LEXICON_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-polly-gs 2>&1 | tee -a
 "$LOG_FILE" || true
else
    echo "Lexicon uploaded." | tee -a "$LOG_FILE"
fi

# List available lexicons
echo "Listing available lexicons..." | tee -a "$LOG_FILE"
log_cmd "aws polly list-lexicons --output text --query 'Lexicons[*].[Name]'" ||
 true

# Get details about the lexicon
echo "Getting details about the lexicon..." | tee -a "$LOG_FILE"
log_cmd "aws polly get-lexicon --name '$LEXICON_NAME' --output text --query
 'Lexicon.Name'" || true

# Use the lexicon when synthesizing speech
echo "Using the lexicon for speech synthesis..." | tee -a "$LOG_FILE"
LEXICON_OUTPUT="${WORK_DIR}/lexicon-output.mp3"
LEXICON_TEXT="I work with AWS every day."
```

```
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --
lexicon-names '$LEXICON_NAME' --text '$LEXICON_TEXT' '$LEXICON_OUTPUT'" || true

if [[ -f "$LEXICON_OUTPUT" ]]; then
    echo "Successfully created lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$LEXICON_OUTPUT" lexicon-output.mp3
else
    echo "Failed to create lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Summary of created resources
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "TUTORIAL SUMMARY" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Created resources:" | tee -a "$LOG_FILE"
echo "1. Lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
echo "2. Audio files:" | tee -a "$LOG_FILE"
echo "   - output.mp3" | tee -a "$LOG_FILE"
echo "   - ssml-output.mp3" | tee -a "$LOG_FILE"
echo "   - lexicon-output.mp3" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Cleanup with auto-confirmation
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "" | tee -a "$LOG_FILE"
echo "Tutorial completed successfully!" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - DeleteLexicon
  - DescribeVoices
  - GetLexicon
  - Hilfe
  - ListLexicons
  - PutLexicon
  - SynthesizeSpeech

Codebeispiele für AWS-Preisliste verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS-Preisliste mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- AWS-Preisliste Benutzerhandbuch — Weitere Informationen zu AWS-Preisliste.
- AWS-Preisliste API-Referenz — Details zu allen verfügbaren AWS-Preisliste Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS-Preisliste verwenden AWS SDKs
  - Aktionen für AWS-Preisliste verwenden AWS SDKs
    - Verwendung von DescribeServices mit einer CLI
    - Verwendung von GetAttributeValues mit einer CLI
    - Verwendung von GetProducts mit einer CLI

Grundlegende Beispiele für AWS-Preisliste verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS-Preisliste mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS-Preisliste verwenden AWS SDKs
  - Verwendung von DescribeServices mit einer CLI
  - Verwendung von GetAttributeValues mit einer CLI
  - Verwendung von GetProducts mit einer CLI

Aktionen für AWS-Preisliste verwenden AWS SDKs

Die folgenden Codebeispiele veranschaulichen, wie einzelne AWS-Preisliste Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS-Preisliste -API-Referenz.

Beispiele

- Verwendung von DescribeServices mit einer CLI
- Verwendung von GetAttributeValues mit einer CLI
- Verwendung von GetProducts mit einer CLI

Verwendung von **`DescribeServices`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeServices` verwendet wird.

CLI

AWS CLI

So rufen Sie Service-Metadaten ab

In diesem Beispiel werden die Metadaten für den Amazon-EC2-Servicecode abgerufen.

Befehl:

```
aws pricing describe-services --service-code AmazonEC2 --format-version aws_v1 --
max-items 1
```

Ausgabe:

```
{
  "Services": [
      {
          "ServiceCode": "AmazonEC2",
          "AttributeNames": [
              "volumeType",
              "maxIopsvolume",
              "instance",
              "instanceCapacity10xlarge",
              "locationType",
              "instanceFamily",
              "operatingSystem",
              "clockSpeed",
              "LeaseContractLength",
              "ecu",
              "networkPerformance",
              "instanceCapacity8xlarge",
              "group",
              "maxThroughputvolume",
              "gpuMemory",
              "ebsOptimized",
              "elasticGpuType",
              "maxVolumeSize",
              "gpu",
              "processorFeatures",
              "intelAvxAvailable",
              "instanceCapacity4xlarge",
              "servicecode",
              "groupDescription",
              "processorArchitecture",
              "physicalCores",
              "productFamily",
              "enhancedNetworkingSupported",
              "intelTurboAvailable",
              "memory",
              "dedicatedEbsThroughput",
              "vcpu",
              "OfferingClass",
```

```
                "instanceCapacityLarge",
                "capacitystatus",
                "termType",
                "storage",
                "intelAvx2Available",
                "storageMedia",
                "physicalProcessor",
                "provisioned",
                "servicename",
                "PurchaseOption",
                "instanceCapacity18xlarge",
                "instanceType",
                "tenancy",
                "usagetype",
                "normalizationSizeFactor",
                "instanceCapacity2xlarge",
                "instanceCapacity16xlarge",
                "maxIopsBurstPerformance",
                "instanceCapacity12xlarge",
                "instanceCapacity32xlarge",
                "instanceCapacityXlarge",
                "licenseModel",
                "currentGeneration",
                "preInstalledSw",
                "location",
                "instanceCapacity24xlarge",
                "instanceCapacity9xlarge",
                "instanceCapacityMedium",
                "operation"
            ]
        }
    ],
    "FormatVersion": "aws_v1"
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Metadaten für alle verfügbaren Servicecodes in der Region us-east-1 zurück.

```
Get-PLSService -Region us-east-1
```

Ausgabe:

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Beispiel 2: Gibt die Metadaten für den Amazon-EC2-Service in der Region us-east-1 zurück.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
AttributeNames
 ServiceCode
--------------
 -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Metadaten für alle verfügbaren Servicecodes in der Region us-east-1 zurück.

```
Get-PLSService -Region us-east-1
```

Ausgabe:

```
AttributeNames                                                  ServiceCode
--------------                                                  -----------
{productFamily, servicecode, groupDescription, termType...}     AWSBudgets
```

```
{productFamily, servicecode, termType, usagetype...}            AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}            AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}            AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}            AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}            AWSConfig
...
```

Beispiel 2: Gibt die Metadaten für den Amazon-EC2-Service in der Region us-east-1 zurück.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
AttributeNames
 ServiceCode
--------------
 -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetAttributeValues`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetAttributeValues` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste von Attributwerten ab

Im folgenden Beispiel für `get-attribute-values` wird eine Liste von Werten für das angegebene Attribut abgerufen.

```
aws pricing get-attribute-values \
    --service-code AmazonEC2 \
    --attribute-name volumeType \
    --max-items 2
```

Ausgabe:

```
{
    "NextToken":
 "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ==",
    "AttributeValues": [
        {
            "Value": "Cold HDD"
        },
        {
            "Value": "General Purpose"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetAttributeValuesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Werte für das Attribut „volumeType” für Amazon EC2 in der Region us-east-1 zurück.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region
 us-east-1
```

Ausgabe:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Einzelheiten zur API finden Sie unter GetAttributeValues AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Werte für das Attribut „volumeType” für Amazon EC2 in der Region us-east-1 zurück.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region
 us-east-1
```

Ausgabe:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- Einzelheiten zur API finden Sie unter GetAttributeValues AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetProducts`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetProducts` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste von Produkten ab

Dieses Beispiel ruft eine Liste von Produkten ab, die die angegebenen Kriterien erfüllen.

Befehl:

```
aws pricing get-products --filters file://filters.json --format-version aws_v1 --
max-results 1 --service-code AmazonEC2
```

filters.json:

```
        [
  {
    "Type": "TERM_MATCH",
    "Field": "ServiceCode",
    "Value": "AmazonEC2"
  },
```

```
  {
    "Type": "TERM_MATCH",
    "Field": "volumeType",
    "Value": "Provisioned IOPS"
  }
]
```

Ausgabe:

```
{
  "FormatVersion": "aws_v1",
  "NextToken": "WGDY7ko8fQXdlaUZVdasFQ==:RVSagyIFn770XQOzdUIcO9BY6ucBG9itXAZGZF/
zioUzOsUKh6PCcPWaOyPZRiMePb986TeoKYB9l55fw/
CyoMq5ymnGmT1Vj39TljbbAlhcqnVfTmPIilx8Uy5bdDaBYy/e/2Ofw9Edzsykbs8LTBuNbiDQ
+BBds5yeI9AQkUepruKk3aEahFPxJ55kx/zk",
  "PriceList": [
      "{\"product\":{\"productFamily\":\"Storage\",\"attributes\":{\"storageMedia
\":\"SSD-backed\",\"maxThroughputvolume\":\"320 MB/sec\",\"volumeType
\":\"Provisioned IOPS\",\"maxIopsvolume\":\"20000\",\"servicecode\":
\"AmazonEC2\",\"usagetype\":\"APS1-EBS:VolumeUsage.piops\",\"locationType
\":\"AWS Region\",\"location\":\"Asia Pacific (Singapore)\",\"servicename
\":\"Amazon Elastic Compute Cloud\",\"maxVolumeSize\":\"16 TiB\",\"operation
\":\"\"},\"sku\":\"3MKHN58N7RDDVGKJ\"},\"serviceCode\":\"AmazonEC2\",\"terms
\":{\"OnDemand\":{\"3MKHN58N7RDDVGKJ.JRTCKXETXF\":{\"priceDimensions\":
{\"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\":{\"unit\":\"GB-Mo\",\"endRange\":
\"Inf\",\"description\":\"$0.138 per GB-month of Provisioned IOPS SSD (io1)
 provisioned storage - Asia Pacific (Singapore)\",\"appliesTo\":[],\"rateCode\":
\"3MKHN58N7RDDVGKJ.JRTCKXETXF.6YS6EN2CT7\",\"beginRange\":\"0\",\"pricePerUnit
\":{\"USD\":\"0.1380000000\"}}},\"sku\":\"3MKHN58N7RDDVGKJ\",\"effectiveDate\":
\"2018-08-01T00:00:00Z\",\"offerTermCode\":\"JRTCKXETXF\",\"termAttributes\":
{}}}},\"version\":\"20180808005701\",\"publicationDate\":\"2018-08-08T00:57:01Z
\"}"
  ]
}
```

- Einzelheiten zur API finden Sie GetProductsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details zu allen Produkten für Amazon EC2 zurück.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Beispiel 2: Gibt Daten für Amazon EC2 in der Region us-east-1 zurück, gefiltert nach Volumetypen von „General Purpose“, d. h. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region
 us-east-1
```

Ausgabe:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Einzelheiten zur API finden Sie unter GetProductsCmdlet-Referenz (V4).AWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Gibt Details zu allen Produkten für Amazon EC2 zurück.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Beispiel 2: Gibt Daten für Amazon EC2 in der Region us-east-1 zurück, gefiltert nach Volumetypen von „General Purpose“, d. h. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region
 us-east-1
```

Ausgabe:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- Einzelheiten zur API finden Sie unter GetProductsCmdlet-Referenz (V5).AWS -Tools für PowerShell

Codebeispiele für Amazon RDS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Relational Database Service mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-RDS-Benutzerhandbuch – Weitere Informationen zu Amazon RDS.
- API-Referenz für Amazon RDS – Details zu allen verfügbaren Amazon-RDS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon RDS AWS SDKs
  - Hello Amazon RDS
  - Erlernen Sie die Grundlagen von Amazon RDS mit einem AWS SDK
  - Aktionen für Amazon RDS mit AWS SDKs
    - Verwenden Sie CreateDBInstance mit einem AWS SDK oder CLI
    - Verwenden Sie CreateDBParameterGroupmit einem AWS SDK oder CLI
    - Verwenden Sie CreateDBSnapshot mit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDBInstance mit einem AWS SDK oder CLI
    - Verwenden Sie DeleteDBParameterGroupmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeAccountAttributesmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeDBInstances mit einem AWS SDK oder CLI
    - Verwenden Sie DescribeDBParameterGroupsmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeDBParameters mit einem AWS SDK oder CLI
    - Verwenden Sie DescribeDBSnapshots mit einem AWS SDK oder CLI
    - Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
    - Verwenden Sie GenerateRDSAuthTokenmit einem AWS SDK

    - Verwenden Sie ModifyDBInstance mit einem AWS SDK oder CLI
    - Verwenden Sie ModifyDBParameterGroupmit einem AWS SDK oder CLI
    - Verwenden Sie RebootDBInstance mit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon RDS AWS SDKs
  - Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben
  - Eine relationale Datenbankinstanz erstellen
- Serverless-Beispiele für Amazon RDS
  - Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Grundlegende Beispiele für die Verwendung von Amazon RDS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Relational Database Service mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon RDS
- Erlernen Sie die Grundlagen von Amazon RDS mit einem AWS SDK
- Aktionen für Amazon RDS mit AWS SDKs
  - Verwenden Sie CreateDBInstance mit einem AWS SDK oder CLI
  - Verwenden Sie CreateDBParameterGroupmit einem AWS SDK oder CLI
  - Verwenden Sie CreateDBSnapshot mit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDBInstance mit einem AWS SDK oder CLI
  - Verwenden Sie DeleteDBParameterGroupmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeAccountAttributesmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDBInstances mit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDBParameterGroupsmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDBParameters mit einem AWS SDK oder CLI
  - Verwenden Sie DescribeDBSnapshots mit einem AWS SDK oder CLI
  - Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
  - Verwenden Sie GenerateRDSAuthTokenmit einem AWS SDK
  - Verwenden Sie ModifyDBInstance mit einem AWS SDK oder CLI

- Verwenden Sie ModifyDBParameterGroupmit einem AWS SDK oder CLI
- Verwenden Sie RebootDBInstance mit einem AWS SDK oder CLI

Hello Amazon RDS

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon RDS beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;

public static class HelloRds
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });
```

```
        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})
```

```
if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_rds.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
```

```
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());
```

```
        std::cout << instanceDBIDs.size() << " RDS instances found." <<
 std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database
 Service (Amazon RDS)
// client and list up to 20 DB instances in your account.
```

```
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n",
 *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
```

```
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational
 Database Service
(Amazon RDS) to list the databases in your account.
"""

import boto3
from botocore.exceptions import ClientError
```

```
# Create an RDS client
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
```

```
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Ruby -API-Referenz.

Erlernen Sie die Grundlagen von Amazon RDS mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Scenario for RDS DB instance example.
```

```
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
    2.  Selects an engine family and creates a custom DB parameter group using
 the CreateDBParameterGroupAsync method.
    3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync
 method.
    4.  Gets parameters in the group using the DescribeDBParameters method.
    5.  Parses and displays parameters in the group.
    6.  Modifies both the auto_increment_offset and auto_increment_increment
 parameters
        using the ModifyDBParameterGroupAsync method.
    7.  Gets and displays the updated parameters using the DescribeDBParameters
 method with a source of "user".
    8.  Gets a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Displays and selects from a list of micro instance classes available for
 the selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
 parameter group
        using the CreateDBInstanceAsync method.
    11. Waits for DB instance to be ready using the DescribeDBInstancesAsync
 method.
    12. Prints out the connection endpoint string for the new DB instance.
    13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
 method.
    14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
    15. Deletes the DB instance using the DeleteDBInstanceAsync method.
    16. Waits for DB instance to be deleted using the DescribeDbInstances method.
    17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
    */

    private static readonly string sepBar = new('-', 80);
    private static RDSWrapper rdsWrapper = null!;
    private static ILogger logger = null!;
    private static readonly string engine = "mysql";
    static async Task Main(string[] args)
```

```
    {
        // Set up dependency injection for the Amazon RDS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<RDSWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<RDSInstanceScenario>();

        rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
        Console.WriteLine(sepBar);

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await
CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParameters(parameterGroup.DBParameterGroupName,
parameters);
```

```
            await
DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);
```

```
        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");
```

```
        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
```

```
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
```

```
    public static async Task DescribeUserSourceParameters(string
parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. Engine: {version.Engine} Version
{version.EngineVersion}.");
```

```
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDbInstanceClass(string engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
```

```
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB
 instance.</param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"10. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
```

```
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }

        Console.WriteLine(sepBar);
```

```
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection
 string.</param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P
 {instance.Endpoint.Port} "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</
param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;

        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
```

```
            var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
```

```
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup
 parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
 {parameterGroup.DBParameterGroupName}.");
        await
 rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

        Console.WriteLine(sepBar);
    }
```

Wrapper-Methoden, die vom Szenario für DB-Instance-Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 DB instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
 engine,
```

```
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }



    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }



    /// <summary>
```

```
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }



    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
```

```
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }



    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

Wrapper-Methoden, die vom Szenario für DB-Parametergruppen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
```

```
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
```

```
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
 group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

Wrapper-Methoden, die vom Szenario für DB-Snapshot-Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
/// </summary>
public partial class RDSWrapper
{
```

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon RDS instance and demonstrates several
 operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
```

```
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

```
   if (!parameterGroupFound) {
       Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

       // 2. Get available engine versions for the specified engine.
       if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                engineVersions, client)) {
           return false;
       }

       std::cout << "Getting available database engine versions for " <<
DB_ENGINE
                 << "."
                 << std::endl;
       std::vector<Aws::String> families;
       for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
           Aws::String family = version.GetDBParameterGroupFamily();
           if (std::find(families.begin(), families.end(), family) ==
               families.end()) {
               families.push_back(family);
               std::cout << "  " << families.size() << ": " << family <<
std::endl;
           }
       }

       int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                           static_cast<int>(families.size()));
       dbParameterGroupFamily = families[choice - 1];
   }
   if (!parameterGroupFound) {
       // 3.  Create a DB parameter group.
       Aws::RDS::Model::CreateDBParameterGroupRequest request;
       request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
       request.SetDBParameterGroupFamily(dbParameterGroupFamily);
       request.SetDescription("Example parameter group.");

       Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
               client.CreateDBParameterGroup(request);

       if (outcome.IsSuccess()) {
           std::cout << "The DB parameter group was successfully created."
                     << std::endl;
       }
```

```
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your parameter group."
              << std::endl;

    Aws::String marker;
    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB parameter group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                         autoIncrementParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);
```

```
            }
            else {
                std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a
source of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }
```

```
    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

    Aws::RDS::Model::DBInstance dbInstance;
    // 7.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "  " << index << ": " <<
engineVersion.GetEngineVersion()
```

```
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                                1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
```

```
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
```

```
                        << dbInstance.GetDBInstanceStatus()
                        << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();

    // 12. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbInstance);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB instance (y/n)? ")) {
        Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 13. Create a snapshot of the DB instance.
            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
```

```
                return false;
            }
        }

        std::cout << "Waiting for snapshot to become available." << std::endl;

        Aws::RDS::Model::DBSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
```

```
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB instance and parameter group (y/n)? "))
 {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
```

```
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

```
//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
```

```
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
```

```
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the
parameter group."
                << std::endl;
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) ==
0) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
```

```
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "fmt"
 "log"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
 "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the
 AWS SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
```

```
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
 parameterGroup)
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(ctx, instance)
 scenario.Cleanup(ctx, instance, parameterGroup)
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB parameter group.")
  _, err = scenario.instances.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
```

```
  }
  parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
```

```
 if err != nil {
  panic(err)
 }
 log.Println("To get a list of parameters that you set previously, specify a
 source of 'user'.")
 userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context,
 instanceName string, dbEngine string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(ctx, instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
   *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  engineIndex := scenario.questioner.AskChoice(
```

```
   "The available engines for your parameter group are:\n", engineChoices)
  engineSelection := engineVersions[engineIndex]
  instOpts, err := scenario.instances.GetOrderableInstances(ctx,
*engineSelection.Engine,
   *engineSelection.EngineVersion)
  if err != nil {
   panic(err)
  }
  optSet := map[string]struct{}{}
  for _, opt := range instOpts {
   if strings.Contains(*opt.DBInstanceClass, "micro") {
    optSet[*opt.DBInstanceClass] = struct{}{}
   }
  }
  var optChoices []string
  for opt := range optSet {
   optChoices = append(optChoices, opt)
  }
  sort.Strings(optChoices)
  optIndex := scenario.questioner.AskChoice(
   "The available micro DB instance classes for your database engine are:\n",
optChoices)
  storageType := "standard"
  allocatedStorage := int32(5)
  log.Printf("Creating a DB instance named %v and database %v.\n"+
   "The DB instance is configured to use your custom parameter group %v,\n"+
   "selected engine %v,\n"+
   "selected DB instance class %v,"+
   "and %v GiB of %v storage.\n"+
   "This typically takes several minutes.",
   instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
   optChoices[optIndex], allocatedStorage, storageType)
  instance, err = scenario.instances.CreateInstance(
   ctx, instanceName, dbName, *engineSelection.Engine,
*engineSelection.EngineVersion,
   *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
   allocatedStorage, adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *instance.DBInstanceStatus != "available" {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx, instanceName)
```

```
    if err != nil {
     panic(err)
    }
   }
   log.Println("Instance created and available.")
  }
  log.Println("Instance data:")
  log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
  log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
  log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
  log.Printf("\tEngine: %v\n", *instance.Engine)
  log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
  log.Println(strings.Repeat("-", 88))
  return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance)
 {
 log.Println(
  "You can now connect to your database by using your favorite MySQL client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
   "port, and administrator username to 'mysql'. Then, enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
 log.Println("For more information, see the User Guide for RDS:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
 *types.DBInstance) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
```

```
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.instances.CreateSnapshot(ctx,
 *instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 ctx context.Context, instance *types.DBInstance, parameterGroup
 *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ",
 "y") {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance to delete. This typically takes several
 minutes.")
```

```
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx,
 *instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBParameterGroupName)
  err = scenario.instances.DeleteParameterGroup(ctx,
 *parameterGroup.DBParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Amazon-RDS-Aktionen zu verwalten.

```
import (
```

```
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}


// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
```

```
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
```

```
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
```

```
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
```

```
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
```

```
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie mehrere Operationen aus.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
```

```
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
```

```
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier> <secretName>

                Where:
                    dbGroupName - The database group name.\s
                    dbParameterGroupFamily - The database parameter group name
 (for example, mysql8.0).
                    dbInstanceIdentifier - The database instance identifier\s
                    dbName - The database name.\s
                    dbSnapshotIdentifier - The snapshot identifier.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
```

```
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon RDS example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql
database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
```

```
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();
```

```
            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
```

```
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Create a database instance and return the ARN of the database.
    public static String createDatabaseInstance(RdsClient rdsClient,
            String dbGroupName,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();
```

```
            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }

        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("mysql")
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient

.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption :
orderableDBInstances) {
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
```

```
   public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
       try {
           DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .engine("mysql")
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
           List<DBEngineVersion> dbEngines = response.dbEngineVersions();
           for (DBEngineVersion dbEngine : dbEngines) {
               System.out.println("The engine version is " +
dbEngine.engineVersion());
               System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   // Modify auto_increment_offset and auto_increment_increment parameters.
   public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
       try {
           Parameter parameter1 = Parameter.builder()
                   .parameterName("auto_increment_offset")
                   .applyMethod("immediate")
                   .parameterValue("5")
                   .build();

           List<Parameter> paraList = new ArrayList<>();
           paraList.add(parameter1);
           ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .parameters(paraList)
                   .build();

           ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
```

```
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
```

```
                    .build();

            CreateDbParameterGroupResponse response =
 rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
 response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking
 the createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment
 parameters by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the
 describeDbEngineVersions method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance
 that contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint
 value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
 <dbName> <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
```

```
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
   """

   if (args.size != 6) {
       println(usage)
       exitProcess(1)
   }

   val dbGroupName = args[0]
   val dbParameterGroupFamily = args[1]
   val dbInstanceIdentifier = args[2]
   val dbName = args[3]
   val dbSnapshotIdentifier = args[4]
   val secretName = args[5]

   val gson = Gson()
   val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
   val username = user.username
   val userPassword = user.password

   println("1. Return a list of the available DB engines")
   describeDBEngines()

   println("2. Create a custom parameter group")
   createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

   println("3. Get the parameter groups")
   describeDbParameterGroups(dbGroupName)

   println("4. Get the parameters in the group")
   describeDbParameters(dbGroupName, 0)

   println("5. Modify the auto_increment_offset parameter")
   modifyDBParas(dbGroupName)

   println("6. Display the updated value")
   describeDbParameters(dbGroupName, -1)

   println("7. Get a list of allowed engine versions")
   getAllowedEngines(dbParameterGroupFamily)
```

```
    println("8. Get a list of micro instance classes available for the selected
 engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database
 and uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
```

```
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

```
// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
```

```
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
```

```
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is
 ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
```

```
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is
 ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
```

```
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset
 or auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
```

```
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
```

```
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the
 database engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance

- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
        self.instance_wrapper = instance_wrapper

    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database
 engine and
        create a DB parameter group that is compatible with a selected engine
 family.
```

```
        :param parameter_group_name: The name given to the newly created
parameter group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
{parameter_group_name}."
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions =
self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
```

```
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine,
parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.

        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the
DB instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
```

```
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
                "Which micro DB instance class do you want to use? ",
inst_choices
            )
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
```

```
                f"The DB instance is configured to use your custom parameter
 group {group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
```

```
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P
 {db_inst['Endpoint']['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
 instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances
 must first
        be deleted.
```

```
        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)?
",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes
several minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name,
db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(parameter_group_name,
db_engine)
        self.update_parameters(parameter_group_name)
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)
```

```
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Amazon-RDS-Aktionen zu verwalten.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.
```

```
        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
```

```
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
```

```
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
```

```
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
```

```
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
```

```
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
```

```
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "rds-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "rds-scenario",
```

```
            dependencies: [
                .product(name: "AWSRDS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Relational Database Service (RDS).
//

import ArgumentParser
import Foundation
import AWSRDS

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"
    @Option(help: "The username to use for the database administrator.")
    var dbUsername = "admin"
    @Option(help: "The password to use for the database administrator.")
    var dbPassword: String

    static var configuration = CommandConfiguration(
        commandName: "rds-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon RDS
 Instances
        using the AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
```

```
        let example = try await Example(region: awsRegion, username: dbUsername,
 password: dbPassword)

        await example.run()
    }
}

class Example {
    let rdsClient: RDSClient

    // Storage for AWS RDS properties

    let dbUsername: String
    let dbPassword: String
    var dbInstanceIdentifier: String
    var dbSnapshotIdentifier: String
    var dbParameterGroupName: String
    var dbParameterGroup: RDSClientTypes.DBParameterGroup?
    var selectedEngineVersion: String?

    init(region: String, username: String, password: String) async throws{
        let rdsConfig = try await RDSClient.RDSClientConfiguration(region:
 region)
        rdsClient = RDSClient(config: rdsConfig)

        dbUsername = username
        dbPassword = password
        dbParameterGroupName = ""
        dbInstanceIdentifier = ""
        dbSnapshotIdentifier = ""
    }

    /// The example's main body.
    func run() async {
        var parameterGroupFamilies: Set<String> = []

        //=====================================================================
        // 1. Get available database engine families for MySQL.
        //=====================================================================

        let engineVersions = await getDBEngineVersions(engineName: "mysql")

        for version in engineVersions {
            if version.dbParameterGroupFamily != nil {
```

```
                parameterGroupFamilies.insert(version.dbParameterGroupFamily!)
            }
        }

        if engineVersions.count > 0 {
            selectedEngineVersion = engineVersions.last!.engineVersion
        } else {
            print("*** Unable to find a valid database engine version. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Found \(parameterGroupFamilies.count) parameter group families:")
        for family in parameterGroupFamilies {
            print("    \(family)")
        }

        //=====================================================================
        // 2. Select an engine family and create a custom DB parameter group.
        //    We select a family by sorting the set of family names, then
        //    choosing the last one.
        //=====================================================================

        let sortedFamilies = parameterGroupFamilies.sorted()

        guard let selectedFamily = sortedFamilies.last else {
            print("*** Unable to find a database engine family. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Selected database engine family \(selectedFamily)")

        dbParameterGroupName = tempName(prefix: "rds-example")
        print("Creating a database parameter group named \(dbParameterGroupName)
using \(selectedFamily)")
        dbParameterGroup = await createDBParameterGroup(groupName:
dbParameterGroupName,
                                                        familyName:
selectedFamily)

        //=====================================================================
```

```
        // 3. Get the parameter group's details.
        //======================================================================

        print("Getting the database parameter group list...")
        let dbParameterGroupList = await describeDBParameterGroups(groupName:
dbParameterGroupName)
        guard let dbParameterGroupList else {
            await cleanUp()
            return
        }

        print("Found \(dbParameterGroupList.count) parameter groups...")
        for group in dbParameterGroupList {
            print("    \(group.dbParameterGroupName ?? "<unknown>")")
        }
        print()

        //======================================================================
        // 4. Get a list of the parameter group's parameters. This list is
        //    likely to be long, so use pagination. Find the
        //    auto_increment_offset and auto_increment_increment parameters.
        //======================================================================

        let parameters = await describeDBParameters(groupName:
dbParameterGroupName)

        //======================================================================
        // 5. Parse and display each parameter's name, description, and
        //    allowed values.
        //======================================================================

        for parameter in parameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            if name == "auto_increment_offset" || name ==
"auto_increment_increment" {
                print("Parameter \(name):")
                print("          Value: \(parameter.parameterValue ??
"<undefined>")")
                print("      Data type: \(parameter.dataType ?? "<unknown>")")
```

```
                print("    Description: \(parameter.description ?? "")")
                print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
                print(String(repeating: "=", count: 78))
            }
        }

        //=====================================================================
        // 6. Modify both the auto_increment_offset and
        //    auto_increment_increment parameters in one call in the custom
        //    parameter group. Set their parameterValue fields to a new
        //    permitted value.
        //=====================================================================

        print("Setting auto_increment_offset and auto_increment_increment both to
5...")
        await modifyDBParameters(groupName: dbParameterGroupName)

        //=====================================================================
        // 7. Get and display the updated parameters, specifying a source of
        //    "user" to get only the modified parameters.
        //=====================================================================

        let updatedParameters = await describeDBParameters(groupName:
dbParameterGroupName, source: "user")

        for parameter in updatedParameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            print("Parameter \(name):")
            print("          Value: \(parameter.parameterValue ??
"<undefined>")")
            print("      Data type: \(parameter.dataType ?? "<unknown>")")
            print("    Description: \(parameter.description ?? "")")
            print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
            print(String(repeating: "=", count: 78))
        }

        //=====================================================================
```

```
        // 8. Get a list of allowed engine versions using
        //    DescribeRDSEngineVersions.
        //=====================================================================

        await listAllowedEngines(family: selectedFamily)

        //=====================================================================
        // 9. Get a list of micro instance classes available for the selected
        //    engine and engine version.
        //=====================================================================

        let dbInstanceClass = await chooseMicroInstance(engine: "mysql",
engineVersion: selectedEngineVersion)
        guard let dbInstanceClass else {
            print("Did not get a valid instance class. Canceling operations.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 10. Create an RDS database that contains a MySQL database and uses
        //     the parameter group we created.
        //=====================================================================

        print("Creating the database instance...")

        guard let instanceClass = dbInstanceClass.dbInstanceClass else {
            print("Instance class name is unknown. Canceling operations.")
            await cleanUp()
            return
        }

        dbInstanceIdentifier = tempName(prefix: "sample-identifier")
        let dbInstanceArn = await createDBInstance(
            name: "SampleDatabase\(Int.random(in: 1000000..<1000000000))",
            instanceIdentifier: dbInstanceIdentifier,
            parameterGroupName: dbParameterGroupName,
            engine: "mysql",
            engineVersion: selectedEngineVersion!,
            instanceClass: instanceClass,
            username: dbUsername,
            password: dbPassword
        )
```

```
        if dbInstanceArn == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 11. Wait for the database instance to be ready by calling
        //     DescribeDBInstances repeatedly until it reports
        //     dbInstanceStatus as "available". This can take upwards of 10
        //     minutes, let the user know that.
        //=====================================================================

        guard let endpoint = await waitUntilDBInstanceReady(instanceIdentifier:
 dbInstanceIdentifier) else {
            print("\nDid not get a valid endpoint from AWS RDS.")
            await cleanUp()
            return
        }

        guard let endpointAddress = endpoint.address else {
            print("\nNo endpoint address returned.")
            await cleanUp()
            return
        }
        guard let endpointPort = endpoint.port else {
            print("\nNo endpoint port returned.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Display connection information for the database instance.
        //=====================================================================

        print("\nTo connect to the new database instance using 'mysql' from the
 shell:")
        print("    mysql -h \(endpointAddress) -P \(endpointPort) -u
 \(self.dbUsername)")

        //=====================================================================
        // 13. Create a snapshot of the database instance.
        //=====================================================================

        dbSnapshotIdentifier = tempName(prefix: "sample-snapshot")
```

```
        await createDBSnapshot(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        //=====================================================================
        // 14. Wait for the snapshot to be ready.
        //=====================================================================

        await waitUntilDBSnapshotReady(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        // That's it! Clean up and exit!

        print("Example complete! Cleaning up...")
        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items.
    func cleanUp() async {
        print("Deleting the database instance \(dbInstanceIdentifier)...")
        await deleteDBInstance(instanceIdentifier: dbInstanceIdentifier)
        await waitUntilDBInstanceDeleted(instanceIdentifier:
dbInstanceIdentifier)

        print("Deleting the database parameter group \(dbParameterGroupName)...")
        await deleteDBParameterGroup(groupName: dbParameterGroupName)
    }

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
```

```
        } catch {
            return []
        }
    }

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
```

```
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
[RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
```

```
    }

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Output a list of the database engine versions supported by the
    /// specified family.
    ///
    /// - Parameter family: The family for which to list allowed database
    ///   engines.
    func listAllowedEngines(family: String?) async {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
```

```
                input: DescribeDBEngineVersionsInput(
                    dbParameterGroupFamily: family,
                    engine: "mysql"
                )
            )

            guard let engineVersions = output.dbEngineVersions else {
                print("No engine versions returned.")
                return
            }

            print("Found \(engineVersions.count) database engine versions:")
            for version in engineVersions {
                print("    \(version.engineVersion ?? "<unknown>"):
\(version.dbEngineDescription ?? "")")
            }
        } catch {
            print("*** Error getting database engine version list:
\(error.localizedDescription)")
            return
        }
    }

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []
```

```
            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }

            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
```

```
   ///   - instanceClass: The memory and compute capacity of the database
   ///     instance, such as `db.m5.large``.
   ///   - username: The admin user's username to establish for the new
   ///     instance.
   ///   - password: The password to use for the specified user's access.
   ///
   /// - Returns: A string indicating the ARN of the newly created database
   ///   instance, or nil if the instance couldn't be created.
   func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                         engine: String, engineVersion: String, instanceClass:
String,
                         username: String, password: String) async -> String? {
       do {
           let output = try await rdsClient.createDBInstance(
               input: CreateDBInstanceInput(
                   allocatedStorage: 100,
                   dbInstanceClass: instanceClass,
                   dbInstanceIdentifier: instanceIdentifier,
                   dbName: name,
                   dbParameterGroupName: parameterGroupName,
                   engine: engine,
                   engineVersion: engineVersion,
                   masterUserPassword: password,
                   masterUsername: username,
                   storageType: "gp2"
               )
           )

           guard let dbInstance = output.dbInstance else {
               print("*** Unable to get the database instance.")
               return nil
           }

           return dbInstance.dbInstanceArn
       } catch {
           print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
           return nil
       }
   }

   /// Wait until the specified database is available to use.
   ///
```

```
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

```
   /// Create a snapshot of the specified name.
   ///
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database instance to
   ///     snapshot.
   ///   - snapshotIdentifier: A unique identifier to give the newly-created
   ///     snapshot.
   func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
async {
       do {
           let output = try await rdsClient.createDBSnapshot(
               input: CreateDBSnapshotInput(
                   dbInstanceIdentifier: instanceIdentifier,
                   dbSnapshotIdentifier: snapshotIdentifier
               )
           )

           guard let snapshot = output.dbSnapshot else {
               print("No snapshot returned.")
               return
           }

           print("The snapshot has been created with ID
\(snapshot.dbiResourceId ?? "<unknown>")")
       } catch {
           print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
       }
   }

   /// Wait until the specified database snapshot is available to use.
   ///
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database for which the
   ///     snapshot was taken.
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
```

```
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }

                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
```

```
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }

    /// Wait until the specified database instance has been deleted.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to wait for.
    func waitUntilDBInstanceDeleted(instanceIdentifier: String) async {
        putString("Waiting for the database instance to be deleted. This may take
a few minutes...")
        do {
            var isDatabaseDeleted = false
            var foundInstance = false

            while !isDatabaseDeleted {
                let output = try await rdsClient.describeDBInstances(input:
DescribeDBInstancesInput())
                guard let instanceList = output.dbInstances else {
                    return
                }

                foundInstance = false

                for instance in instanceList {
                    guard let foundInstanceIdentifier =
instance.dbInstanceIdentifier else {
                        continue
                    }

                    if instanceIdentifier == foundInstanceIdentifier {
                        foundInstance = true
                        break
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
```

```
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
                if !foundInstance {
                    isDatabaseDeleted = true
                    print()
                }
            }
        } catch {
            print("\n*** Error waiting for the database instance to be deleted:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
```

```
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Print a string to standard output without a trailing newline, and
    /// without buffering.
    ///
    /// - Parameter str: The string to output.
    func putString(_ str: String = "") {
        if str.length >= 1 {
            let data = str.data(using: .utf8)
            guard let data else {
                return
            }
            FileHandle.standardOutput.write(data)
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance

- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Aktionen für Amazon RDS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon RDS-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-RDS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon RDS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Relational Database Service.

Beispiele

- Verwenden Sie CreateDBInstance mit einem AWS SDK oder CLI
- Verwenden Sie CreateDBParameterGroupmit einem AWS SDK oder CLI
- Verwenden Sie CreateDBSnapshot mit einem AWS SDK oder CLI
- Verwenden Sie DeleteDBInstance mit einem AWS SDK oder CLI
- Verwenden Sie DeleteDBParameterGroupmit einem AWS SDK oder CLI
- Verwenden Sie DescribeAccountAttributesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeDBEngineVersionsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeDBInstances mit einem AWS SDK oder CLI
- Verwenden Sie DescribeDBParameterGroupsmit einem AWS SDK oder CLI
- Verwenden Sie DescribeDBParameters mit einem AWS SDK oder CLI

- Verwenden Sie DescribeDBSnapshots mit einem AWS SDK oder CLI
- Verwenden Sie DescribeOrderableDBInstanceOptionsmit einem AWS SDK oder CLI
- Verwenden Sie GenerateRDSAuthTokenmit einem AWS SDK
- Verwenden Sie ModifyDBInstance mit einem AWS SDK oder CLI
- Verwenden Sie ModifyDBParameterGroupmit einem AWS SDK oder CLI
- Verwenden Sie RebootDBInstance mit einem AWS SDK oder CLI

Verwenden Sie **`CreateDBInstance mit einem`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
```

```
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB)
to allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für .NET - API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

Um eine DB-Instance zu erstellen

Im folgenden Beispiel für `create-db-instance` werden die zum Starten einer neuen DB-Instance erforderlichen Optionen verwendet.

```
aws rds create-db-instance \
    --db-instance-identifier test-mysql-instance \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username admin \
    --master-user-password secret99 \
    --allocated-storage 20
```

Ausgabe:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "12:55-13:25",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-12345abc",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql5.7",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2ff2ff2f",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
```

```
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2c"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2d"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                },
                "SubnetStatus": "Active"
            },
            {
                "SubnetIdentifier": "subnet-########",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2b"
                },
                "SubnetStatus": "Active"
            }
        ]
    },
    "PreferredMaintenanceWindow": "sun:08:07-sun:08:37",
    "PendingModifiedValues": {
        "MasterUserPassword": "****"
    },
    "MultiAZ": false,
    "EngineVersion": "5.7.22",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",
    "OptionGroupMemberships": [
        {
            "OptionGroupName": "default:mysql-5-7",
            "Status": "in-sync"
        }
    ],
```

```
        "PubliclyAccessible": true,
        "StorageType": "gp2",
        "DbInstancePort": 0,
        "StorageEncrypted": false,
        "DbiResourceId": "db-5555EXAMPLE44444444EXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "DBInstanceArn": "arn:aws:rds:us-west-2:123456789012:db:test-mysql-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": []
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer Amazon-RDS-DB-Instance im Benutzerhandbuch für Amazon RDS.

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string,
 dbInstanceClass string,
 storageType string, allocatedStorage int32, adminName string, adminPassword
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
```

```
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }
```

```
    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported
version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the database instance is available.
```

```
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für PHP - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
```

```
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB
instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
```

```
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
    func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                          engine: String, engineVersion: String, instanceClass:
String,
                          username: String, password: String) async -> String? {
        do {
            let output = try await rdsClient.createDBInstance(
                input: CreateDBInstanceInput(
                    allocatedStorage: 100,
                    dbInstanceClass: instanceClass,
                    dbInstanceIdentifier: instanceIdentifier,
                    dbName: name,
                    dbParameterGroupName: parameterGroupName,
                    engine: engine,
                    engineVersion: engineVersion,
                    masterUserPassword: password,
                    masterUsername: username,
                    storageType: "gp2"
                )
            )

            guard let dbInstance = output.dbInstance else {
                print("*** Unable to get the database instance.")
                return nil
            }

            return dbInstance.dbInstanceArn
        } catch {
            print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateDBParameterGroup`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDBParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
            client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine DB-Parametergruppe zu erstellen

Im folgenden Beispiel für `create-db-parameter-group` wird eine DB-Parametergruppe erstellt.

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL5.6 \
    --description "My new parameter group"
```

Ausgabe:

```
{
    "DBParameterGroup": {
        "DBParameterGroupName": "mydbparametergroup",
        "DBParameterGroupFamily": "mysql5.6",
        "Description": "My new parameter group",
        "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:mydbparametergroup"
    }
}
```

Weitere Informationen finden Sie unter Erstellen einer DB-Parametergruppe im Benutzerhandbuch für Amazon RDS.

- Einzelheiten zur API finden Sie CreateDBParameterGroupunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the
 specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter
 group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
```

```
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname   = 'mydbparametergroup'
" iv_dbparametergroupfamily = 'mysql8.0'
" iv_description            = 'My custom DB parameter group for MySQL 8.0'
TRY.
    oo_result = lo_rds->createdbparametergroup(
      iv_dbparametergroupname   = iv_dbparametergroupname
      iv_dbparametergroupfamily = iv_dbparametergroupfamily
```

```
          iv_description            = iv_description ).
        MESSAGE 'DB parameter group created.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbparmgralrexfault.
        MESSAGE 'DB parameter group already exists.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
        MESSAGE 'DB parameter group quota exceeded.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
 RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
```

```
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie CreateDBParameterGroupin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateDBSnapshot mit einem`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDBSnapshot` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
```

```
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für .NET - API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
```

```
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

Um einen DB-Snapshot zu erstellen

Im folgenden Beispiel für `create-db-snapshot` wird ein Upgrade für einen DB-Snapshot ausgeführt.

```
aws rds create-db-snapshot \
    --db-instance-identifier database-mysql \
    --db-snapshot-identifier mydbsnapshot
```

Ausgabe:

```
{
    "DBSnapshot": {
        "DBSnapshotIdentifier": "mydbsnapshot",
        "DBInstanceIdentifier": "database-mysql",
        "Engine": "mysql",
        "AllocatedStorage": 100,
        "Status": "creating",
        "Port": 3306,
        "AvailabilityZone": "us-east-1b",
        "VpcId": "vpc-6594f31c",
        "InstanceCreateTime": "2019-04-30T15:45:53.663Z",
        "MasterUsername": "admin",
        "EngineVersion": "5.6.40",
```

```
        "LicenseModel": "general-public-license",
        "SnapshotType": "manual",
        "Iops": 1000,
        "OptionGroupName": "default:mysql-5-6",
        "PercentProgress": 0,
        "StorageType": "io1",
        "Encrypted": true,
        "KmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/
AKIAIOSFODNN7EXAMPLE",
        "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
        "IAMDatabaseAuthenticationEnabled": false,
        "ProcessorFeatures": [],
        "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
    }
}
```

Weitere Informationen finden Sie unter Erstellen eines DB-Snapshots im Benutzerhandbuch für Amazon RDS.

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
```

```
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für Go - API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Create an Amazon RDS snapshot.
```

```
    public static void createSnapshot(RdsClient rdsClient, String
 dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
 CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
 rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
 response.dbSnapshot().dbiResourceId());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK for Java 2.x -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für PHP - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id,
DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für Ruby - API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
 async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID
 \(snapshot.dbiResourceId ?? "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
 \(snapshotIdentifier): \(error.localizedDescription)")
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der API-Referenz zum AWS -SDK für Swift.

Verwenden Sie **`DeleteDBInstance`** **mit** einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDBInstance` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier,
            SkipFinalSnapshot = true,
            DeleteAutomatedBackups = true
        });

    return response.DBInstance;
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBInstanceRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);
        request.SetSkipFinalSnapshot(true);
        request.SetDeleteAutomatedBackups(true);

        Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB instance deletion has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

Um eine DB-Instance zu löschen

Im folgenden Beispiel für `delete-db-instance` wird die angegebene DB-Instance gelöscht, nachdem der letzte DB-Snapshot `test-instance-final-snap` erstellt wurde.

```
aws rds delete-db-instance \
    --db-instance-identifier test-instance \
    --final-db-snapshot-identifier test-instance-final-snap
```

Ausgabe:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-instance",
        "DBInstanceStatus": "deleting",
        ...some output truncated...
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
```

```
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für PHP - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.
```

```
        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
```

```
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance
\(instanceIdentifier): \(error.localizedDescription)")
        }
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteDBParameterGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDBParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
```

```
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine DB-Parametergruppe

Im folgenden Beispiel für `command` wird eine DB-Parametergruppe gelöscht.

```
aws rds delete-db-parameter-group \
    --db-parameter-group-name mydbparametergroup
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Arbeiten mit DB-Parametergruppen im Benutzerhandbuch für Amazon RDS.

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while
database
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
```

```
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
 DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    TRY.
        lo_rds->deletedbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname ).
        MESSAGE 'DB parameter group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
```

```
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDBParameterGroupin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeAccountAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAccountAttributes` verwendet wird.

CLI

AWS CLI

So beschreiben Sie Kontoattribute

Im folgenden `describe-account-attributes` Beispiel werden die Attribute für das aktuelle AWS Konto abgerufen.

```
aws rds describe-account-attributes
```

Ausgabe:

```
{
    "AccountQuotas": [
        {
            "Max": 40,
            "Used": 4,
            "AccountQuotaName": "DBInstances"
        },
        {
```

```
            "Max": 40,
            "Used": 0,
            "AccountQuotaName": "ReservedDBInstances"
        },
        {
            "Max": 100000,
            "Used": 40,
            "AccountQuotaName": "AllocatedStorage"
        },
        {
            "Max": 25,
            "Used": 0,
            "AccountQuotaName": "DBSecurityGroups"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "AuthorizationsPerDBSecurityGroup"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBParameterGroups"
        },
        {
            "Max": 100,
            "Used": 3,
            "AccountQuotaName": "ManualSnapshots"
        },
        {
            "Max": 20,
            "Used": 0,
            "AccountQuotaName": "EventSubscriptions"
        },
        {
            "Max": 50,
            "Used": 1,
            "AccountQuotaName": "DBSubnetGroups"
        },
        {
            "Max": 20,
            "Used": 1,
            "AccountQuotaName": "OptionGroups"
        },
```

```
        {
            "Max": 20,
            "Used": 6,
            "AccountQuotaName": "SubnetsPerDBSubnetGroup"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "ReadReplicasPerMaster"
        },
        {
            "Max": 40,
            "Used": 1,
            "AccountQuotaName": "DBClusters"
        },
        {
            "Max": 50,
            "Used": 0,
            "AccountQuotaName": "DBClusterParameterGroups"
        },
        {
            "Max": 5,
            "Used": 0,
            "AccountQuotaName": "DBClusterRoles"
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter DescribeAccountAttributes AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```

```
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import
 software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeAccountAttributesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAccountAttributes() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API-Details finden Sie DescribeAccountAttributesin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeDBEngineVersions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDBEngineVersions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.
```

```
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie die DB-Engine-Versionen für die MySQL-DB-Engine

Im folgenden Beispiel für `describe-db-engine-versions` werden Details zu den jeweiligen DB-Engine-Versionen für die angegebene DB-Engine angezeigt.

```
aws rds describe-db-engine-versions \
```

```
    --engine mysql
```

Ausgabe:

```
{
    "DBEngineVersions": [
        {
            "Engine": "mysql",
            "EngineVersion": "5.5.46",
            "DBParameterGroupFamily": "mysql5.5",
            "DBEngineDescription": "MySQL Community Edition",
            "DBEngineVersionDescription": "MySQL 5.5.46",
            "ValidUpgradeTarget": [
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.53",
                    "Description": "MySQL 5.5.53",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.54",
                    "Description": "MySQL 5.5.54",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                {
                    "Engine": "mysql",
                    "EngineVersion": "5.5.57",
                    "Description": "MySQL 5.5.57",
                    "AutoUpgrade": false,
                    "IsMajorVersionUpgrade": false
                },
                ...some output truncated...
            ]
        }
```

Weitere Informationen finden Sie unter Was ist Amazon Relational Database Service (Amazon RDS)? im Benutzerhandbuch für Amazon RDS.

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine
 string, parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
```

```
  return output.DBEngineVersions, nil
 }
}
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
```

```
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
```

```
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter
group family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
```

```
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }
```

- Einzelheiten zur API finden Sie DescribeDBEngineVersionsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeDBInstances`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDBInstances` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

So beschreiben Sie eine DB-Instance

Im folgenden Beispiel für `describe-db-instances` werden Details zur angegebenen DB-Instance abgerufen.

```
aws rds describe-db-instances \
    --db-instance-identifier mydbinstancecf
```

Ausgabe:

```
{
    "DBInstances": [
        {
            "DBInstanceIdentifier": "mydbinstancecf",
            "DBInstanceClass": "db.t3.small",
            "Engine": "mysql",
            "DBInstanceStatus": "available",
            "MasterUsername": "masterawsuser",
            "Endpoint": {
                "Address": "mydbinstancecf.abcexample.us-
east-1.rds.amazonaws.com",
                "Port": 3306,
                "HostedZoneId": "Z2R2ITUGPM61AM"
            },
            ...some output truncated...
        }
    ]
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName
 string) (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
```

```
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response =
 rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " +
 instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest
 {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;


//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
```

```
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für PHP - API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Ruby -API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
 RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
 may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
```

```
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeDBParameterGroups`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDBParameterGroups` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string
 name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
```

```
        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie Ihre DB-Parametergruppe

Im folgenden Beispiel für `describe-db-parameter-groups` werden Details zu den DB-Parametergruppen abgerufen.

```
aws rds describe-db-parameter-groups
```

Ausgabe:

```
{
    "DBParameterGroups": [
        {
            "DBParameterGroupName": "default.aurora-mysql5.7",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "Default parameter group for aurora-mysql5.7",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-mysql5.7"
```

```
        },
        {
            "DBParameterGroupName": "default.aurora-postgresql9.6",
            "DBParameterGroupFamily": "aurora-postgresql9.6",
            "Description": "Default parameter group for aurora-postgresql9.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora-postgresql9.6"
        },
        {
            "DBParameterGroupName": "default.aurora5.6",
            "DBParameterGroupFamily": "aurora5.6",
            "Description": "Default parameter group for aurora5.6",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.aurora5.6"
        },
        {
            "DBParameterGroupName": "default.mariadb10.1",
            "DBParameterGroupFamily": "mariadb10.1",
            "Description": "Default parameter group for mariadb10.1",
            "DBParameterGroupArn": "arn:aws:rds:us-
east-1:123456789012:pg:default.mariadb10.1"
        },
        ...some output truncated...
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit DB-Parametergruppen im Benutzerhandbuch zu Amazon RDS.

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
       try {
           DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .maxRecords(20)
                   .build();

           DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
           List<DBParameterGroup> groups = response.dbParameterGroups();
           for (DBParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbParameterGroupName());
               System.out.println("The group description is " +
group.description());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
```

```
                logger.info("Parameter group %s does not exist.",
 parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
```

```
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    TRY.
        oo_result = lo_rds->describedbparametergroups(
          iv_dbparametergroupname = iv_dbparametergroupname ).
        MESSAGE 'DB parameter group retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
 [RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie DescribeDBParameterGroupsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeDBParameters`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDBParameters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter
group.</param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
```

```
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
```

```
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So beschreiben Sie die Parameter in einer DB-Parametergruppe

Im folgenden Beispiel für `describe-db-parameters` werden die Details der angegebenen DB-Parametergruppe abgerufen.

```
aws rds describe-db-parameters \
    --db-parameter-group-name mydbpg
```

Ausgabe:

```
{
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have
 only an xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        {
            "ParameterName": "auto_generate_certs",
            "Description": "Controls whether the server autogenerates SSL key and
 certificate files in the data directory, if they do not already exist.",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot"
        },
        ...some output truncated...
```

```
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit DB-Parametergruppen im Benutzerhandbuch zu Amazon RDS.

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context,
 parameterGroupName string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
```

```
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zu AWS SDK für Go .

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Retrieve parameters in the group.
   public static void describeDbParameters(RdsClient rdsClient, String
dbGroupName, int flag) {
       try {
           DescribeDbParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zu AWS SDK for Java 2.x .

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this
 prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
```

```
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zum AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'
```

```
# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für Ruby -API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " iv_source               = 'user' (optional - filters by parameter source)
    TRY.
        oo_result = lo_rds->describedbparameters(
          iv_dbparametergroupname = iv_dbparametergroupname
          iv_source               = iv_source ).
        DATA(lv_param_count) = lines( oo_result->get_parameters( ) ).
        MESSAGE |Retrieved { lv_param_count } parameters.| TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
```

```
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie unter DescribeDBParameters in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
 [RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )
```

```
            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeDBSnapshots`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDBSnapshots` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zu AWS SDK für .NET .

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen DB-Snapshot für eine DB-Instance

Im folgenden Beispiel für `describe-db-snapshots` werden die Details eines DB-Snapshots für eine DB-Instance abgerufen.

```
aws rds describe-db-snapshots \
    --db-snapshot-identifier mydbsnapshot
```

Ausgabe:

```
{
    "DBSnapshots": [
        {
```

```
            "DBSnapshotIdentifier": "mydbsnapshot",
            "DBInstanceIdentifier": "mysqldb",
            "SnapshotCreateTime": "2018-02-08T22:28:08.598Z",
            "Engine": "mysql",
            "AllocatedStorage": 20,
            "Status": "available",
            "Port": 3306,
            "AvailabilityZone": "us-east-1f",
            "VpcId": "vpc-6594f31c",
            "InstanceCreateTime": "2018-02-08T22:24:55.973Z",
            "MasterUsername": "mysqladmin",
            "EngineVersion": "5.6.37",
            "LicenseModel": "general-public-license",
            "SnapshotType": "manual",
            "OptionGroupName": "default:mysql-5-6",
            "PercentProgress": 100,
            "StorageType": "gp2",
            "Encrypted": false,
            "DBSnapshotArn": "arn:aws:rds:us-
east-1:123456789012:snapshot:mydbsnapshot",
            "IAMDatabaseAuthenticationEnabled": false,
            "ProcessorFeatures": [],
            "DbiResourceId": "db-AKIAIOSFODNN7EXAMPLE"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erstellen eines DB-Snapshots im Benutzerhandbuch für Amazon RDS.

Beispiel 2: So ermitteln Sie die Anzahl der manuell erstellten Snapshots

Im folgenden `describe-db-snapshots` Beispiel wird der `length` Operator in der `--query` Option verwendet, um die Anzahl der manuellen Schnappschüsse zurückzugeben, die in einer bestimmten AWS Region aufgenommen wurden.

```
aws rds describe-db-snapshots \
    --snapshot-type manual \
    --query "length(*[].{DBSnapshots:SnapshotType})" \
    --region eu-central-1
```

Ausgabe:

```
35
```

Weitere Informationen finden Sie unter Erstellen eines DB-Snapshots im Benutzerhandbuch für Amazon RDS.

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS CLI - Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName
 string) (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
```

```
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zu AWS SDK für Go .

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zum AWS SDK für Phyton (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'
```

```
# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if
 error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für Ruby -API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
```

```
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
           while !snapshotReady {
               let output = try await rdsClient.describeDBSnapshots(
                   input: DescribeDBSnapshotsInput(
                       dbInstanceIdentifier: instanceIdentifier,
                       dbSnapshotIdentifier: snapshotIdentifier
                   )
               )

               guard let snapshotList = output.dbSnapshots else {
                   return
               }

               for snapshot in snapshotList {
                   guard let snapshotReadyStr = snapshot.status else {
                       return
                   }

                   if snapshotReadyStr.contains("available") {
                       snapshotReady = true
                       print()
                   } else {
                       putString(".")
                       do {
                           try await Task.sleep(for: .seconds(15))
                       } catch {
                           print("\n*** Error pausing the task!")
                       }
                   }
               }
           }
       } catch {
           print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
       }
   }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DescribeOrderableDBInstanceOptions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeOrderableDBInstanceOptions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
```

```
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
```

```
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database
 engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für C ++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie bestellbare DB-Instance-Optionen

Im folgenden Beispiel für `describe-orderable-db-instance-options` werden Details zu den bestellbaren Optionen für DB-Instances abgerufen, auf denen die MySQL-DB-Engine ausgeführt wird.

```
aws rds describe-orderable-db-instance-options \
    --engine mysql
```

Ausgabe:

```
{
    "OrderableDBInstanceOptions": [
        {
            "MinStorageSize": 5,
            "ReadReplicaCapable": true,
```

```
            "MaxStorageSize": 6144,
            "AvailabilityZones": [
                {
                    "Name": "us-east-1a"
                },
                {
                    "Name": "us-east-1b"
                },
                {
                    "Name": "us-east-1c"
                },
                {
                    "Name": "us-east-1d"
                }
            ],
            "SupportsIops": false,
            "AvailableProcessorFeatures": [],
            "MultiAZCapable": true,
            "DBInstanceClass": "db.m1.large",
            "Vpc": true,
            "StorageType": "gp2",
            "LicenseModel": "general-public-license",
            "EngineVersion": "5.5.46",
            "SupportsStorageEncryption": false,
            "SupportsEnhancedMonitoring": true,
            "Engine": "mysql",
            "SupportsIAMDatabaseAuthentication": false,
            "SupportsPerformanceInsights": false
        }
    ]
    ...some output truncated...
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
```

```
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions,
 output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der AWS SDK für Go API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
```

```
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result-
>get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.|
 TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
```

```
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? =
nil) async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }

                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }
```

```
            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Einzelheiten zur API finden Sie DescribeOrderableDBInstanceOptionsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GenerateRDSAuthToken`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GenerateRDSAuthToken`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die RdsUtilitiesKlasse, um ein Authentifizierungstoken zu generieren.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie GenerateRDSAuthTokenunter AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ModifyDBInstance`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyDBInstance` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie eine DB-Instance

Im folgenden Beispiel für `modify-db-instance` wird eine Options- und Parametergruppe mit einer kompatiblen DB-Instance von Microsoft SQL Server verknüpft. Der Parameter `--apply-immediately` bewirkt, dass die Options- und Parametergruppen sofort verknüpft werden, anstatt auf das nächste Wartungsfenster zu warten.

```
aws rds modify-db-instance \
    --db-instance-identifier database-2 \
    --option-group-name test-se-2017 \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --apply-immediately
```

Ausgabe:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "database-2",
        "DBInstanceClass": "db.r4.large",
        "Engine": "sqlserver-se",
        "DBInstanceStatus": "available",

        ...output omitted...

        "DBParameterGroups": [
            {
                "DBParameterGroupName": "test-sqlserver-se-2017",
                "ParameterApplyStatus": "applying"
            }
        ],
        "AvailabilityZone": "us-west-2d",

        ...output omitted...

        "MultiAZ": true,
```

```
        "EngineVersion": "14.00.3281.6.v1",
        "AutoMinorVersionUpgrade": false,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "license-included",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "test-se-2017",
                "Status": "pending-apply"
            }
        ],
        "CharacterSetName": "SQL_Latin1_General_CP1_CI_AS",
        "SecondaryAvailabilityZone": "us-west-2c",
        "PubliclyAccessible": true,
        "StorageType": "gp2",

        ...output omitted...

        "DeletionProtection": false,
        "AssociatedRoles": [],
        "MaxAllocatedStorage": 1000
    }
}
```

Weitere Informationen finden Sie unter Ändern einer Amazon-RDS-DB-Instance im Benutzerhandbuch für Amazon RDS.

Beispiel 2: So verknüpfen Sie eine VPC-Sicherheitsgruppe mit einer DB-Instance

Im folgenden Beispiel für `modify-db-instance` werden eine bestimmte VPC-Sicherheitsgruppe zugeordnet und DB-Sicherheitsgruppen aus einer DB-Instance entfernt:

```
aws rds modify-db-instance \
    --db-instance-identifier dbName \
    --vpc-security-group-ids sg-ID
```

Ausgabe:

```
{
"DBInstance": {
    "DBInstanceIdentifier": "dbName",
    "DBInstanceClass": "db.t3.micro",
    "Engine": "mysql",
    "DBInstanceStatus": "available",
```

```
    "MasterUsername": "admin",
    "Endpoint": {
        "Address": "dbName.abcdefghijk.us-west-2.rds.amazonaws.com",
        "Port": 3306,
        "HostedZoneId": "ABCDEFGHIJK1234"
    },
    "AllocatedStorage": 20,
    "InstanceCreateTime": "2024-02-15T00:37:58.793000+00:00",
    "PreferredBackupWindow": "11:57-12:27",
    "BackupRetentionPeriod": 7,
    "DBSecurityGroups": [],
    "VpcSecurityGroups": [
        {
            "VpcSecurityGroupId": "sg-ID",
            "Status": "active"
        }
    ],
    ... output omitted ...
    "MultiAZ": false,
    "EngineVersion": "8.0.35",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",

    ... output omitted ...
    }
}
```

Weitere Informationen finden Sie unter Zugriffskontrolle mit Sicherheitsgruppen im Benutzerhandbuch für Amazon RDS.

- Weitere API-Informationen finden Sie unter ModifyDBInstance in der AWS CLI - Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }
```

```
    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master
 password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();

            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter ModifyDBInstance in der API-Referenz zu AWS SDK for Java 2.x .

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
```

```
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter ModifyDBInstance in der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ModifyDBParameterGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyDBParameterGroup` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Update a DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);
```

```
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine DB-Parametergruppe zu ändern

Im folgenden Beispiel für `modify-db-parameter-group` wird der Wert des Parameters `clr enabled` in einer DB-Parametergruppe geändert. Der Parameter `--apply-immediately` bewirkt, dass die DB-Parametergruppe sofort geändert wird, anstatt auf das nächste Wartungsfenster zu warten.

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name test-sqlserver-se-2017 \
    --parameters "ParameterName='clr
 enabled',ParameterValue=1,ApplyMethod=immediate"
```

Ausgabe:

```
{
    "DBParameterGroupName": "test-sqlserver-se-2017"
```

```
}
```

Weitere Informationen finden Sie unter Ändern von Parametern in einer DB-Parametergruppe im Benutzerhandbuch für Amazon RDS.

- Einzelheiten zur API finden Sie ModifyDBParameterGroupunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
```

```
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();
```

```
            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_dbparametergroupname = 'mydbparametergroup'
    " it_parameters - table containing parameter objects with:
    "   - parametername = 'max_connections'
    "   - parametervalue = '100'
    "   - applymethod = 'immediate' or 'pending-reboot'
    TRY.
        oo_result = lo_rds->modifydbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname
          it_parameters           = it_parameters ).
        MESSAGE 'DB parameter group modified.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
```

```
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- Einzelheiten zur API finden Sie ModifyDBParameterGroupin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`RebootDBInstance`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RebootDBInstance` verwendet wird.

CLI

AWS CLI

So starten Sie eine DB-Instance neu

Im folgenden Beispiel für `reboot-db-instance` wird die angegebene DB-Instance neu gestartet.

```
aws rds reboot-db-instance \
```

```
    --db-instance-identifier test-mysql-instance
```

Ausgabe:

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "test-mysql-instance",
        "DBInstanceClass": "db.t3.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "rebooting",
        "MasterUsername": "admin",
        "Endpoint": {
            "Address": "test-mysql-instance.############.us-
west-2.rds.amazonaws.com",
            "Port": 3306,
            "HostedZoneId": "Z1PVIF0EXAMPLE"
        },

    ... output omitted...

    }
}
```

Weitere Informationen finden Sie unter Neustarten einer DB-Instance im Benutzerhandbuch für Amazon RDS.

- Weitere API-Informationen finden Sie unter RebootDBInstance in der AWS CLI - Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
```

```
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter RebootDBInstance in der AWS SDK for Java 2.x -API-Referenz.

Szenarien für die Verwendung von Amazon RDS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon RDS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon RDS oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben
- Eine relationale Datenbankinstanz erstellen

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für .NET In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend entwickelt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS

- Amazon RDS Data Service
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript (v3) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon Aurora Aurora-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PHP-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.

- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in einer Amazon Aurora Aurora-Serverless-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldedaten enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

Eine relationale Datenbankinstanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Netzwerkkomponenten ein
- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
```

```
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"
```

```
# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
```

```
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"
```

```
echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"
```

```
CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
```

```
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress

- CreateDbInstance
- CreateDbSubnetGroup
- CreateSecret
- CreateSecurityGroup
- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteSecret
- DeleteSecurityGroup
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- Wait

Serverless-Beispiele für Amazon RDS

Die folgenden Codebeispiele zeigen, wie Amazon RDS mit AWS SDKs verwendet wird.

Beispiele

- Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion mithilfe von .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
```

```
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
$"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

$"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

$"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                $"Pwd={authToken};";


        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";

            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
"0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
"0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");
```

```
                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The
 sum is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server
 error" })
        };
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Go.

```
/*
Golang v2 code here.
```

```
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )
```

```
 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements
 RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new
 APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();
```

```
            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"),
 token);
                 PreparedStatement statement =
 connection.prepareStatement("SELECT ? + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
```

```
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response =
 rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {
```

```
    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von TypeScript.

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that
 the DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
```

```
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number;
 body: string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new
 AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
```

```
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");
```

```
        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
```

```
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
```

```
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");
```

```
async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();
```

```
    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;
```

```
    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Codebeispiele für Amazon RDS Data Service mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Relational Database Service Data Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch für Amazon RDS Data Service – Weitere Informationen zu Amazon RDS Data Service.
- API-Referenz für Amazon RDS Data Service – Details zu allen verfügbaren Aktionen von Amazon RDS Data Service.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon RDS Data Service AWS SDKs
  - Aktionen für Amazon RDS Data Service unter Verwendung AWS SDKs
    - Verwenden Sie ExecuteStatementmit einem AWS SDK oder CLI

- Szenarien für die Verwendung von Amazon RDS Data Service AWS SDKs
  - Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Grundlegende Beispiele für die Verwendung von Amazon RDS Data Service AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen des Datenservice von Amazon Relational Database Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon RDS Data Service unter Verwendung AWS SDKs
  - Verwenden Sie ExecuteStatementmit einem AWS SDK oder CLI

Aktionen für Amazon RDS Data Service unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon RDS Data Service-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-RDS-Data-Service-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon RDS Data Service AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Relational Database Service.

Beispiele

- Verwenden Sie ExecuteStatementmit einem AWS SDK oder CLI

Verwenden Sie **`ExecuteStatement`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ExecuteStatement` verwendet wird.

CLI

AWS CLI

Beispiel 1: So führen Sie eine SQL-Anweisung aus, die Teil einer Transaktion ist

Im folgenden `execute-statement`-Beispiel wird eine SQL-Anweisung ausgeführt, die Teil einer Transaktion ist.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:us-west-2:123456789012:cluster:mydbcluster" \
    --database "mydb" \
    --secret-arn "arn:aws:secretsmanager:us-west-2:123456789012:secret:mysecret"
 \
    --sql "update mytable set quantity=5 where id=201" \
    --transaction-id "ABC1234567890xyz"
```

Ausgabe:

```
{
    "numberOfRecordsUpdated": 1
}
```

Beispiel 2: So führen Sie eine SQL-Anweisung mit Parametern aus

Im folgenden Beispiel für `execute-statement` wird eine SQL-Anweisung mit Parametern ausgeführt.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster" \
    --database "mydb" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret"
 \
    --sql "insert into mytable values (:id, :val)" \
    --parameters "[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"value1\"}}]"
```

Ausgabe:

```
{
    "numberOfRecordsUpdated": 1
}
```

Weitere Informationen finden Sie unter Verwenden der Daten-API für Aurora Serverless im Benutzerhandbuch für Amazon RDS.

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);

    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der API-Referenz zum AWS SDK für Rust.

Szenarien für die Verwendung von Amazon RDS Data Service AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon RDS Data Service mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon RDS Data Service oder in Kombination

mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für .NET In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend entwickelt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

C++

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript (v3) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon Aurora Aurora-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PHP-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in einer Amazon Aurora Aurora-Serverless-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem

Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldedaten enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Codebeispiele für Amazon Redshift mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Redshift mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon Redshift Managementhandbuch – Weitere Informationen zu Amazon Redshift.
- API-Referenz für Amazon Redshift – Details zu allen verfügbaren Amazon-Redshift-Aktionen.

- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Redshift AWS SDKs
  - Hello Amazon Redshift
  - Lernen Sie die Grundlagen von Amazon Redshift mit einem AWS SDK
  - Aktionen für Amazon Redshift mit AWS SDKs
    - Verwenden Sie CreateClustermit einem AWS SDK oder CLI
    - Verwenden Sie DeleteClustermit einem AWS SDK oder CLI
    - Verwenden Sie DescribeClustersmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeStatementmit einem AWS SDK
    - Verwenden Sie ExecuteStatementmit einem AWS SDK
    - Verwenden Sie GetStatementResultmit einem AWS SDK
    - Verwenden Sie ListDatabasesmit einem AWS SDK
    - Verwenden Sie ModifyClustermit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Redshift AWS SDKs
  - Erstellen eines Amazon-Redshift-Element-Trackers
  - Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
  - Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Grundlegende Beispiele für die Verwendung von Amazon Redshift AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Redshift mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Redshift
- Lernen Sie die Grundlagen von Amazon Redshift mit einem AWS SDK
- Aktionen für Amazon Redshift mit AWS SDKs

- Verwenden Sie CreateClustermit einem AWS SDK oder CLI
- Verwenden Sie DeleteClustermit einem AWS SDK oder CLI
- Verwenden Sie DescribeClustersmit einem AWS SDK oder CLI
- Verwenden Sie DescribeStatementmit einem AWS SDK
- Verwenden Sie ExecuteStatementmit einem AWS SDK
- Verwenden Sie GetStatementResultmit einem AWS SDK
- Verwenden Sie ListDatabasesmit einem AWS SDK
- Verwenden Sie ModifyClustermit einem AWS SDK oder CLI

Hello Amazon Redshift

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon Redshift beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Main method to run the Hello Amazon Redshift example.
    /// </summary>
    /// <param name="args">Command line arguments (not used).</param>
    public static async Task Main(string[] args)
    {
        var redshiftClient = new AmazonRedshiftClient();

        Console.WriteLine("Hello, Amazon Redshift! Let's list available
clusters:");

        var clusters = new List<Cluster>();

        try
        {
```

```
            // Use pagination to retrieve all clusters.
            var clustersPaginator =
redshiftClient.Paginators.DescribeClusters(new DescribeClustersRequest());

            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");

            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }
        }
        catch (AmazonRedshiftException ex)
        {
            Console.WriteLine($"Couldn't list clusters. Here's why:
{ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK für .NET API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 redshiftClient := redshift.NewFromConfig(sdkConfig)
 count := 20
 fmt.Printf("Let's list up to %v clusters for your account.\n", count)
 result, err := redshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  MaxRecords: aws.Int32(int32(count)),
 })
 if err != nil {
  fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Clusters) == 0 {
  fmt.Println("You don't have any clusters!")
  return
 }
 for _, cluster := range result.Clusters {
  fmt.Printf("\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
```

```
}
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import
 software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloRedshift {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
```

```
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() +
 " status = " + cluster.clusterStatus()));
    }
}
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_redshift(redshift_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Redshift client and
 list
    the clusters in your account. This list might be empty if you haven't created
    any clusters.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param redshift_client: A Boto3 Redshift Client object.
    """
    print("Hello, Redshift! Let's list your clusters:")
    paginator = redshift_client.get_paginator("describe_clusters")
    clusters = []
    for page in paginator.paginate():
        clusters.extend(page["Clusters"])
```

```
    print(f"{len(clusters)} cluster(s) were found.")

    for cluster in clusters:
        print(f"  {cluster['ClusterIdentifier']}")


if __name__ == "__main__":
    hello_redshift(boto3.client("redshift"))
```

- Einzelheiten zur API finden Sie DescribeClustersin AWS SDK for Python (Boto3) API Reference.

Lernen Sie die Grundlagen von Amazon Redshift mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen eines Redshift-Clusters.
- Listen Sie die Datenbanken im Cluster auf.
- Erstellen einer Tabelle mit dem Namen „Movies“.
- Befüllen der Tabelle „Movies“.
- Abfragen der Tabelle „Movies“ nach Jahr.
- Ändern des Redshift-Clusters.
- Löschen Sie den Amazon-Redshift-Cluster.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Redshift-Wrapper-Klasse zur Verwaltung von Vorgängen.

```
/// <summary>
/// Wrapper class for Amazon Redshift operations.
/// </summary>
public class RedshiftWrapper
{
    private readonly IAmazonRedshift _redshiftClient;
    private readonly IAmazonRedshiftDataAPIService _redshiftDataClient;

    /// <summary>
    /// Constructor for RedshiftWrapper.
    /// </summary>
    /// <param name="redshiftClient">Amazon Redshift client.</param>
    /// <param name="redshiftDataClient">Amazon Redshift Data API client.</param>
    public RedshiftWrapper(IAmazonRedshift redshiftClient,
 IAmazonRedshiftDataAPIService redshiftDataClient)
    {
        _redshiftClient = redshiftClient;
        _redshiftDataClient = redshiftDataClient;
    }

    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
 string databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
 "ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
```

```
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
    {
        try
        {
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
```

```
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };
```

```
            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified
and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create a table in the Redshift database.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> CreateTableAsync(string clusterIdentifier, string
database, string dbUser)
    {
        try
        {
            var sqlStatement = @"
                CREATE TABLE Movies (
                    id INTEGER PRIMARY KEY,
                    title VARCHAR(250) NOT NULL,
                    year INTEGER NOT NULL
                )";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine("Table created: Movies");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
```

```
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create table. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Insert a record into the Movies table using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="id">The movie ID.</param>
    /// <param name="title">The movie title.</param>
    /// <param name="year">The movie year.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> InsertMovieAsync(string clusterIdentifier, string
database, string dbUser,
        int id, string title, int year)
    {
        try
        {
            var sqlStatement = "INSERT INTO Movies (id, title, year) VALUES
(:id, :title, :year)";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "id", Value = id.ToString() },
                    new SqlParameter { Name = "title", Value = title },
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };
```

```
            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine($"Inserted: {title} ({year})");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't insert movie. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Query movies by year using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="year">The year to query.</param>
    /// <returns>A list of movie titles.</returns>
    public async Task<List<string>> QueryMoviesByYearAsync(string
clusterIdentifier, string database,
        string dbUser, int year)
    {
        try
        {
            var sqlStatement = "SELECT title FROM Movies WHERE year = :year";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "year", Value = year.ToString() }
```

```
                }
            };

            var response = await
_redshiftDataClient.ExecuteStatementAsync(request);
            Console.WriteLine($"The identifier of the statement is
{response.Id}");

            await WaitForStatementToCompleteAsync(response.Id);

            var results = await GetStatementResultAsync(response.Id);
            var movieTitles = new List<string>();

            foreach (var row in results)
            {
                if (row.Count > 0)
                {
                    var title = row[0].StringValue;
                    Console.WriteLine($"The Movie title field is {title}");
                    movieTitles.Add(title);
                }
            }

            return movieTitles;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't query movies. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
```

```
   public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
   {
       try
       {
           var request = new DescribeStatementRequest
           {
               Id = statementId
           };

           var response = await
_redshiftDataClient.DescribeStatementAsync(request);
           return response;
       }
       catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
       {
           Console.WriteLine($"Statement not found: {ex.Message}");
           throw;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
           throw;
       }
   }

   /// <summary>
   /// Get the results of a statement execution.
   /// </summary>
   /// <param name="statementId">The statement ID.</param>
   /// <returns>A list of result rows.</returns>
   public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
   {
       try
       {
           var request = new GetStatementResultRequest
           {
               Id = statementId
           };

           var response = await
_redshiftDataClient.GetStatementResultAsync(request);
```

```
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait for a statement to complete execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    private async Task WaitForStatementToCompleteAsync(string statementId)
    {
        var status = StatusString.SUBMITTED;
        DescribeStatementResponse? response = null;

        while (status == StatusString.SUBMITTED || status == StatusString.PICKED
|| status == StatusString.STARTED)
        {
            await Task.Delay(1000); // Wait 1 second
            response = await DescribeStatementAsync(statementId);
            status = response.Status;
            Console.WriteLine($"...{status}");
        }

        if (status == StatusString.FINISHED)
        {
            Console.WriteLine("The statement is finished!");
        }
        else
        {
            var errorMessage = response?.Error ?? "Unknown error";
            Console.WriteLine($"The statement failed with status: {status}");
            Console.WriteLine($"Error message: {errorMessage}");
        }
```

```
    }

    /// <summary>
    /// Wait for a cluster to become available.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public async Task WaitForClusterAvailableAsync(string clusterIdentifier)
    {
        Console.WriteLine($"Wait until {clusterIdentifier} is available. This may
 take a few minutes.");

        var startTime = DateTime.Now;
        var clusters = await DescribeClustersAsync(clusterIdentifier);

        while (clusters[0].ClusterStatus != "available")
        {
            var elapsed = DateTime.Now - startTime;
            Console.WriteLine($"Elapsed Time: {elapsed:mm\\:ss} - Waiting for
 cluster...");

            await Task.Delay(5000); // Wait 5 seconds
            clusters = await DescribeClustersAsync(clusterIdentifier);
        }

        var totalElapsed = DateTime.Now - startTime;
        Console.WriteLine($"Cluster is available! Total Elapsed Time:
 {totalElapsed:mm\\:ss}");
    }
}
```

Führen Sie ein interaktives Szenario durch, in dem die Grundlagen von Redshift demonstriert werden.

```
/// <summary>
/// Amazon Redshift Getting Started Scenario.
/// </summary>
public class RedshiftBasics
{
    public static bool IsInteractive = true;
    public static RedshiftWrapper? Wrapper = null;
    public static ILogger logger = null!;
```

```
    private static readonly string _moviesFilePath = "../../../../../../
resources/sample_files/movies.json";

    /// <summary>
    /// Main method for the Amazon Redshift Getting Started scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRedshift>()
                    .AddAWSService<IAmazonRedshiftDataAPIService>()
                    .AddTransient<RedshiftWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<RedshiftBasics>();

        Wrapper = host.Services.GetRequiredService<RedshiftWrapper>();

        await RunScenarioAsync();
    }

    /// <summary>
    /// Run the complete Amazon Redshift scenario.
    /// </summary>
    public static async Task RunScenarioAsync()
    {
        // Set all variables to default values
        string userName = "awsuser";
        string userPassword = "AwsUser1000";
        string clusterIdentifier = "redshift-cluster-movies";
        var databaseName = "dev";
        int recordCount = 50;
        int year = 2013;
        try
        {
            Console.WriteLine(

 "==============================================================================");
            Console.WriteLine("Welcome to the Amazon Redshift SDK Getting Started
 scenario.");
```

```
            Console.WriteLine(
                "This .NET program demonstrates how to interact with Amazon
 Redshift by using the AWS SDK for .NET.");
            Console.WriteLine("Let's get started...");
            Console.WriteLine(

 "================================================================================");

            // Step 1: Get user credentials (if interactive)
            if (IsInteractive)
            {
                Console.WriteLine("Please enter a user name for the cluster
 (default is awsuser):");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    userName = userInput;


 Console.WriteLine("=====================================================================
                Console.WriteLine("Please enter a user password for the cluster
 (default is AwsUser1000):");
                var passwordInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(passwordInput))
                    userPassword = passwordInput;


 Console.WriteLine("=====================================================================

                // Step 2: Get cluster identifier
                Console.WriteLine("Enter a cluster id value (default is redshift-
cluster-movies):");
                var clusterInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(clusterInput))
                    clusterIdentifier = clusterInput;
            }
            else
            {
                Console.WriteLine($"Using default values: userName={userName},
 clusterIdentifier={clusterIdentifier}");
            }

            // Step 3: Create Redshift cluster
            await Wrapper!.CreateClusterAsync(clusterIdentifier, databaseName,
 userName, userPassword);
```

```
Console.WriteLine("=========================================================================

            // Step 4: Wait for cluster to become available

Console.WriteLine("=========================================================================
            await Wrapper.WaitForClusterAvailableAsync(clusterIdentifier);

Console.WriteLine("=========================================================================

            // Step 5: List databases

Console.WriteLine("=========================================================================
            Console.WriteLine($" When you created {clusterIdentifier}, the dev
database is created by default and used in this scenario.");
            Console.WriteLine(" To create a custom database, you need to have a
CREATEDB privilege.");
            Console.WriteLine(" For more information, see the documentation here:
https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

Console.WriteLine("=========================================================================


Console.WriteLine("=========================================================================
            Console.WriteLine($"List databases in {clusterIdentifier}");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ListDatabasesAsync(clusterIdentifier, userName,
databaseName);

Console.WriteLine("=========================================================================

            // Step 6: Create Movies table

Console.WriteLine("=========================================================================
            Console.WriteLine("Now you will create a table named Movies.");
```

```
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.CreateTableAsync(clusterIdentifier, databaseName,
 userName);

 Console.WriteLine("=====================================================================

            // Step 7: Populate the Movies table

 Console.WriteLine("=====================================================================
            Console.WriteLine("Populate the Movies table using the Movies.json
 file.");

            if (IsInteractive)
            {
                Console.WriteLine("Specify the number of records you would like
 to add to the Movies Table.");
                Console.WriteLine("Please enter a value between 50 and 200.");
                Console.Write("Enter a value: ");

                var recordCountInput = Console.ReadLine();
                if (int.TryParse(recordCountInput, out var inputCount) &&
 inputCount is >= 50 and <= 200)
                {
                    recordCount = inputCount;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
 {recordCount}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default record count: {recordCount}");
            }

            await PopulateMoviesTableAsync(clusterIdentifier, databaseName,
 userName, recordCount);
            Console.WriteLine($"{recordCount} records were added to the Movies
 table.");
```

```
Console.WriteLine("=========================================================================

            // Step 8 & 9: Query movies by year

Console.WriteLine("=========================================================================
            Console.WriteLine("Query the Movies table by year. Enter a value
between 2012-2014.");

            if (IsInteractive)
            {
                Console.Write("Enter a year: ");
                var yearInput = Console.ReadLine();
                if (int.TryParse(yearInput, out var inputYear) && inputYear is >=
2012 and <= 2014)
                {
                    year = inputYear;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{year}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default year: {year}");
            }

            await Wrapper.QueryMoviesByYearAsync(clusterIdentifier, databaseName,
userName, year);

Console.WriteLine("=========================================================================

            // Step 10: Modify the cluster

Console.WriteLine("=========================================================================
            Console.WriteLine("Now you will modify the Redshift cluster.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
```

```
            await Wrapper.ModifyClusterAsync(clusterIdentifier, "wed:07:30-
wed:08:00");

 Console.WriteLine("=========================================================================

            // Step 11 & 12: Delete cluster confirmation

 Console.WriteLine("=========================================================================
            if (IsInteractive)
            {
                Console.WriteLine("Would you like to delete the Amazon Redshift
 cluster? (y/n)");
                var deleteResponse = Console.ReadLine();
                if (deleteResponse?.ToLower() == "y")
                {
                    await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
                }
            }
            else
            {
                Console.WriteLine("Deleting the Amazon Redshift cluster...");
                await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            }

 Console.WriteLine("=========================================================================



 Console.WriteLine("=========================================================================
            Console.WriteLine("This concludes the Amazon Redshift SDK Getting
 Started scenario.");

 Console.WriteLine("=========================================================================
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during the scenario:
 {ex.Message}");
            Console.WriteLine("Deleting the Amazon Redshift cluster...");
            await Wrapper!.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            throw;
        }
    }
```

```
    /// <summary>
    /// Populate the Movies table with data from the JSON file.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="recordCount">Number of records to insert.</param>
    private static async Task PopulateMoviesTableAsync(string clusterIdentifier,
string database, string dbUser, int recordCount)
    {
        if (!File.Exists(_moviesFilePath))
        {
            throw new FileNotFoundException($"Required movies data file not found
at: {_moviesFilePath}");
        }

        var jsonContent = await File.ReadAllTextAsync(_moviesFilePath);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        var movies = JsonSerializer.Deserialize<List<Movie>>(jsonContent,
options);

        if (movies == null || movies.Count == 0)
        {
            throw new InvalidOperationException("Failed to parse movies JSON file
or file is empty.");
        }

        var insertCount = Math.Min(recordCount, movies.Count);

        for (int i = 0; i < insertCount; i++)
        {
            var movie = movies[i];
            await Wrapper!.InsertMovieAsync(clusterIdentifier, database, dbUser,
i, movie.Title, movie.Year);
        }
    }

    /// <summary>
    /// Movie data model.
    /// </summary>
```

```
    private class Movie
    {
        public string Title { get; set; } = string.Empty;
        public int Year { get; set; }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package scenarios

import (
 "context"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "math/rand"
```

```
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
 redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
 "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
 functions so that
// they are simpler to read and understand.
type RedshiftBasicsScenario struct {
 sdkConfig         aws.Config
 helper            IScenarioHelper
 questioner        demotools.IQuestioner
 pauser            demotools.IPausable
 filesystem        demotools.IFileSystem
 redshiftActor     *actions.RedshiftActions
 redshiftDataActor *actions.RedshiftDataActions
```

```
 secretsmanager    *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
 secure service.
type SecretsManager struct {
 SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner,
 pauser demotools.IPausable, filesystem demotools.IFileSystem, helper
 IScenarioHelper) RedshiftBasicsScenario {
 scenario := RedshiftBasicsScenario{
  sdkConfig:         sdkConfig,
  helper:            helper,
  questioner:        questioner,
  pauser:            pauser,
  filesystem:        filesystem,
  secretsmanager:    &SecretsManager{SecretsManagerClient:
 secretsmanager.NewFromConfig(sdkConfig)},
  redshiftActor:     &actions.RedshiftActions{RedshiftClient:
 redshift.NewFromConfig(sdkConfig)},
  redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
 redshiftdata.NewFromConfig(sdkConfig)},
 }
 return scenario
}


// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
 ID    int    `json:"id"`
 Title string `json:"title"`
 Year  int    `json:"year"`
}


// User makes it easier to get the User data back from SecretsManager and use it
 later.
type User struct {
 Username string `json:"userName"`
 Password string `json:"userPassword"`
}
```

```
// Run runs the RedshiftBasics interactive example that shows you how to use
 Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

 user := User{}
 secretId := "s3express/basics/secrets"
 clusterId := "demo-cluster-1"
 maintenanceWindow := "wed:07:30-wed:08:00"
 databaseName := "dev"
 tableName := "Movies"
 fileName := "Movies.json"
 nodeType := "ra3.xlplus"
 clusterType := "single-node"

 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := runner.questioner.(*demotools.MockQuestioner)
   if isMock || runner.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   runner.cleanUpResources(ctx, clusterId, databaseName, tableName,
 user.Username, runner.questioner)
```

```
  }
 }()

 // Retrieve the userName and userPassword from SecretsManager
 output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
 &secretsmanager.GetSecretValueInput{
  SecretId: aws.String(secretId),
 })
 if err != nil {
  log.Printf("There was a problem getting the secret value: %s", err)
  log.Printf("Please make sure to create a secret named 's3express/basics/
secrets' with keys of 'userName' and 'userPassword'.")
  panic(err)
 }

 err = json.Unmarshal([]byte(*output.SecretString), &user)
 if err != nil {
  log.Printf("There was a problem parsing the secret value from JSON: %s", err)
  panic(err)
 }

 // Create the Redshift cluster
 _, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username,
 user.Password, nodeType, clusterType, true)
 if err != nil {
  var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault
  if errors.As(err, &clusterAlreadyExistsFault) {
   log.Println("Cluster already exists. Continuing.")
  } else {
   log.Println("Error creating cluster.")
   panic(err)
  }
 }

 // Wait for the cluster to become available
 waiter :=
 redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Println("An error occurred waiting for the cluster.")
  panic(err)
 }
```

```
// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
 log.Println("Something went wrong trying to get information about the
cluster.")
 panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s",
*describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
 log.Printf("Failed to list databases: %v\n", err)
 panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil
result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId,
databaseName, tableName, user.Username, runner.pauser, []string{"title
VARCHAR(256)", "year INT"})
if err != nil {
 log.Printf("Failed to create table: %v\n", err)
 panic(err)
}

describeInput := redshiftdata.DescribeStatementInput{
 Id: result.Id,
}
query := actions.RedshiftQuery{
 Context: ctx,
 Input:   describeInput,
 Result:  result,
}
```

```
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 // Populate the "Movies" table
 runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName,
 user.Username, fileName)

 // Query the "Movies" table by year
 log.Println("Query the Movies table by year.")
 year := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
  demotools.InIntRange{Lower: 2012, Upper: 2014})
 runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
 year)

 // Modify the cluster's maintenance window
 runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

 // Delete the Redshift cluster if confirmed
 runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
 runner.questioner)

 log.Println("Thanks for watching!")
}

// cleanUpResources asks the user if they would like to delete each resource
 created during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further
 deletion attempts are skipped.
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 questioner demotools.IQuestioner) {
 deleted := false
 var err error = nil
 if questioner.AskBool("Do you want to delete the entire cluster? This will clean
 up all resources. (y/n)", "y") {
  deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
  if err != nil {
   log.Printf("Error deleting cluster: %v", err)
  }
```

```
 }
 if !deleted && questioner.AskBool("Do you want to delete the dev table? This
 will clean up all inserted records but keep your cluster intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId,
 databaseName, tableName, userName)
  if err != nil {
   log.Printf("Error deleting movies table: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
 table? This will clean up all inserted records but keep your cluster and table
 intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
 databaseName, tableName, userName, runner.pauser)
  if err != nil {
   log.Printf("Error deleting data rows: %v", err)
  }
 }
 if !deleted {
  log.Print("Please manually delete any unwanted resources.")
 }
}


// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
 objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string,
 filesystem demotools.IFileSystem) ([]Movie, error) {
 file, err := filesystem.OpenFile("../../resources/sample_files/" + fileName)
 if err != nil {
  return nil, err
 }
 defer filesystem.CloseFile(file)

 var movies []Movie
 err = json.NewDecoder(file).Decode(&movies)
 if err != nil {
  return nil, err
 }

 return movies, nil
}
```

```
// PopulateMoviesTable reads data from the <fileName> file and inserts records
 into the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 fileName string) {
 log.Println("Populate the " + tableName + " table using the " + fileName + "
 file.")
 numRecords := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
  demotools.InIntRange{Lower: 10, Upper: 100})

 movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
 if err != nil {
  log.Printf("Failed to load movies from JSON: %v\n", err)
  panic(err)
 }

 var sqlStatements []string

 for i, movie := range movies {
  if i >= numRecords {
   break
  }

  sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
   tableName,
   strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
 escape them
   movie.Year)

  sqlStatements = append(sqlStatements, sqlStatement)
 }

 input := &redshiftdata.BatchExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sqls:              sqlStatements,
 }

 result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to execute batch statement: %v\n", err)
```

```
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Result:  result,
  Input:   describeInput,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute batch insert query: %v\n", err)
  return
 }
 log.Printf("Successfully executed batch statement\n")

 log.Printf("%d records were added to the Movies table.\n", numRecords)
}



// QueryMoviesByYear retrieves only movies from the "Movies" table which match
 the given year.
func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, year
 int) {

 sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
 year)

 input := &redshiftdata.ExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sql:               aws.String(sqlStatement),
 }

 result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
```

```
 }

 log.Println("The identifier of the statement is ", *result.Id)

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
 *result.Id)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
 }
 for _, row := range getResultOutput.Records {
  for _, col := range row {
   title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
   if !ok {
    log.Println("Failed to parse the field")
   } else {
    log.Printf("The Movie title field is %s\n", title.Value)
   }
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.

- CreateCluster
- DescribeClusters
- DescribeStatement
- ExecuteStatement
- GetStatementResult
- ListDatabasesPaginator
- ModifyCluster

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-Redshift-Features gezeigt werden.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import
 software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
```

```
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading
 the Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);
```

```
    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate
that file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret
Manager that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift
by using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
service hosted in the cloud.

            The program's primary functionalities include cluster creation,
verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
execution of SQL statements.
            Furthermore, it demonstrates the process of querying data from the
Movie table.\s

            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
```

```
            First, we will retrieve the user name and password from Secrets
 Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information,
 such as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
 resources and storage that work together to process and analyze large volumes of
 data.");
        logger.info("Enter a cluster id value or accept the default by hitting
 Enter (default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
 userClusterId;
        try {
```

```
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error
code {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default
and used in this scenario.\s
```

```
            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}",
 rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ExecuteStatementResponse> future =
 redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
```

```
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation
to complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
```

```
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between
2012 and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(),
movieYear, clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
 redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code:
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
 rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
```

```
            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon Redshift SDK.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
```

```
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is
ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();
```

```
        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
```

```
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database
user, and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
```

```
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }

    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records
added to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
```

```
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
```

```
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
```

```
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending
program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the
results
     * @return a {@link CompletableFuture} that completes when the statement
result has been processed
```

```
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the
response is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field
is " + value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }


   /**
    * Asynchronously queries movies by a given year from a Redshift database.
    *
    * @param database    the name of the database to query
    * @param dbUser      the user to connect to the database with
    * @param year        the year to filter the movies by
    * @param clusterId   the identifier of the Redshift cluster to connect to
    * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
    */
   public CompletableFuture<String> queryMoviesByYearAsync(String database,
```

```
                                                                    String dbUser,
                                                                    int year,
                                                                    String
clusterId) {

       String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
       SqlParameter yearParam = SqlParameter.builder()
           .name("year")
           .value(String.valueOf(year))
           .build();

       ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .database(database)
           .dbUser(dbUser)
           .parameters(yearParam)
           .sql(sqlStatement)
           .build();

       return CompletableFuture.supplyAsync(() -> {
           try {
               ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
               return response.id();
           } catch (RedshiftDataException e) {
               throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
           }
       }).exceptionally(exception -> {
           logger.info("Error: {}", exception.getMessage());
           return "";
       });
   }

   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
```

```
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
       ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
           .build();

       return getAsyncClient().modifyCluster(modifyClusterRequest)
           .whenComplete((clusterResponse, exception) -> {
               if (exception != null) {
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {} ", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {} ",
exception.getMessage());
                   }
               } else {
                   logger.info("The modified cluster was successfully modified
and has "
                       + clusterResponse.cluster().preferredMaintenanceWindow()
+ " as the maintenance window");
               }
           });
   }

   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
```

```
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}",
 exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftScenario:
```

```
    """Runs an interactive scenario that shows how to get started with
Redshift."""

    def __init__(self, redshift_wrapper, redshift_data_wrapper):
        self.redshift_wrapper = redshift_wrapper
        self.redshift_data_wrapper = redshift_data_wrapper

    def redhift_scenario(self, json_file_path):
        database_name = "dev"

        print(DASHES)
        print("Welcome to the Amazon Redshift SDK Getting Started example.")
        print(
            """
      This Python program demonstrates how to interact with Amazon Redshift
      using the AWS SDK for Python (Boto3).

      Amazon Redshift is a fully managed, petabyte-scale data warehouse
      service hosted in the cloud.

      The program's primary functionalities include cluster creation,
      verification of cluster readiness, listing databases, table creation,
      populating data within the table, and executing SQL statements.

      It also demonstrates querying data from the Movies table.

      Upon completion, all AWS resources are cleaned up.
    """
        )
        if not os.path.isfile(json_file_path):
            logging.error(f"The file {json_file_path} does not exist.")
            return

        print("Let's get started...")
        user_name = q.ask("Please enter your user name (default is awsuser):")
        user_name = user_name if user_name else "awsuser"

        print(DASHES)
        user_password = q.ask(
            "Please enter your user password (default is AwsUser1000):"
        )
        user_password = user_password if user_password else "AwsUser1000"

        print(DASHES)
```

```
        print(
            """A Redshift cluster refers to the collection of computing resources
 and storage that work
            together to process and analyze large volumes of data."""
        )
        cluster_id = q.ask(
            "Enter a cluster identifier value (default is redshift-cluster-
movies): "
        )
        cluster_id = cluster_id if cluster_id else "redshift-cluster-movies"

        self.redshift_wrapper.create_cluster(
            cluster_id, "ra3.4xlarge", user_name, user_password, True, 2
        )

        print(DASHES)
        print(f"Wait until {cluster_id} is available. This may take a few
 minutes...")
        q.ask("Press Enter to continue...")

        self.wait_cluster_available(cluster_id)

        print(DASHES)

        print(
            f"""
       When you created {cluster_id}, the dev database is created by default and
 used in this scenario.

       To create a custom database, you need to have a CREATEDB privilege.
       For more information, see the documentation here:
       https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
      """
        )
        q.ask("Press Enter to continue...")
        print(DASHES)

        print(DASHES)
        print(f"List databases in {cluster_id}")
        q.ask("Press Enter to continue...")
        databases = self.redshift_data_wrapper.list_databases(
            cluster_id, database_name, user_name
        )
        print(f"The cluster contains {len(databases)} database(s).")
```

```
        for database in databases:
            print(f"    Database: {database}")
        print(DASHES)

        print(DASHES)
        print("Now you will create a table named Movies.")
        q.ask("Press Enter to continue...")

        self.create_table(cluster_id, database_name, user_name)

        print(DASHES)

        print("Populate the Movies table using the Movies.json file.")
        print(
            "Specify the number of records you would like to add to the Movies
Table."
        )
        print("Please enter a value between 50 and 200.")

        while True:
            try:
                num_records = int(q.ask("Enter a value: ", q.is_int))
                if 50 <= num_records <= 200:
                    break
                else:
                    print("Invalid input. Please enter a value between 50 and
200.")
            except ValueError:
                print("Invalid input. Please enter a value between 50 and 200.")

        self.populate_table(
            cluster_id, database_name, user_name, json_file_path, num_records
        )

        print(DASHES)
        print("Query the Movies table by year. Enter a value between 2012-2014.")

        while True:
            movie_year = int(q.ask("Enter a year: ", q.is_int))
            if 2012 <= movie_year <= 2014:
                break
            else:
                print("Invalid input. Please enter a valid year between 2012 and
2014.")
```

```
        # Function to query database
        sql_id = self.query_movies_by_year(
            database_name, user_name, movie_year, cluster_id
        )

        print(f"The identifier of the statement is {sql_id}")

        print("Checking statement status...")
        self.wait_statement_finished(sql_id)
        result = self.redshift_data_wrapper.get_statement_result(sql_id)

        self.display_movies(result)

        print(DASHES)

        print(DASHES)
        print("Now you will modify the Redshift cluster.")
        q.ask("Press Enter to continue...")

        preferred_maintenance_window = "wed:07:30-wed:08:00"
        self.redshift_wrapper.modify_cluster(cluster_id,
preferred_maintenance_window)

        print(DASHES)

        print(DASHES)
        delete = q.ask("Do you want to delete the cluster? (y/n) ", q.is_yesno)

        if delete:
            print(f"You selected to delete {cluster_id}")
            q.ask("Press Enter to continue...")
            self.redshift_wrapper.delete_cluster(cluster_id)
        else:
            print(f"Cluster {cluster_id}cluster_id was not deleted")

        print(DASHES)
        print("This concludes the Amazon Redshift SDK Getting Started scenario.")
        print(DASHES)

    def create_table(self, cluster_id, database, username):
        self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
```

```
            user_name=username,
            sql="CREATE TABLE Movies (statement_id INT PRIMARY KEY, title
VARCHAR(100), year INT)",
        )

        print("Table created: Movies")


    def populate_table(self, cluster_id, database, username, file_name, number):
        with open(file_name) as f:
            data = json.load(f)

        i = 0
        for record in data:
            if i == number:
                break

            statement_id = i
            title = record["title"]
            year = record["year"]
            i = i + 1
            parameters = [
                {"name": "statement_id", "value": str(statement_id)},
                {"name": "title", "value": title},
                {"name": "year", "value": str(year)},
            ]

            self.redshift_data_wrapper.execute_statement(
                cluster_identifier=cluster_id,
                database_name=database,
                user_name=username,
                sql="INSERT INTO Movies VALUES(:statement_id, :title, :year)",
                parameter_list=parameters,
            )

        print(f"{i} records inserted into Movies table")

    def wait_cluster_available(self, cluster_id):
        """
        Waits for a cluster to be available.

        :param cluster_id: The cluster identifier.

        Note: The cluster_available waiter can also be used.
```

```
        It is not used in this case to allow an elapsed time message.
        """
        cluster_ready = False
        start_time = time.time()

        while not cluster_ready:
            time.sleep(30)
            cluster = self.redshift_wrapper.describe_clusters(cluster_id)
            status = cluster[0]["ClusterStatus"]
            if status == "available":
                cluster_ready = True
            elif status != "creating":
                raise Exception(
                    f"Cluster {cluster_id} creation failed with status {status}."
                )

            elapsed_seconds = int(round(time.time() - start_time))
            minutes = int(elapsed_seconds // 60)
            seconds = int(elapsed_seconds % 60)

            print(f"Elapsed Time: {minutes}:{seconds:02d} - status {status}...")

            if minutes > 30:
                raise Exception(
                    f"Cluster {cluster_id} is not available after 30 minutes."
                )

    def query_movies_by_year(self, database, username, year, cluster_id):
        sql = "SELECT * FROM Movies WHERE year = :year"

        params = [{"name": "year", "value": str(year)}]

        response = self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql=sql,
            parameter_list=params,
        )

        return response["Id"]

    @staticmethod
    def display_movies(response):
```

```
        metadata = response["ColumnMetadata"]
        records = response["Records"]

        title_column_index = None
        for i in range(len(metadata)):
            if metadata[i]["name"] == "title":
                title_column_index = i
                break

        if title_column_index is None:
            print("No title column found.")
            return

        print(f"Found {len(records)} movie(s).")
        for record in records:
            print(f"   {record[title_column_index]['stringValue']}")

    def wait_statement_finished(self, sql_id):
        while True:
            time.sleep(1)
            response = self.redshift_data_wrapper.describe_statement(sql_id)
            status = response["Status"]
            print(f"Statement status is {status}.")

            if status == "FAILED":
                print(f"The query failed because {response['Error']}. Ending
 program")
                raise Exception("The Query Failed. Ending program")
            elif status == "FINISHED":
                break
```

Hauptfunktion, die die Implementierung eines Szenarios zeigt.

```
def main():
    redshift_client = boto3.client("redshift")
    redshift_data_client = boto3.client("redshift-data")
    redshift_wrapper = RedshiftWrapper(redshift_client)
    redshift_data_wrapper = RedshiftDataWrapper(redshift_data_client)
    redshift_scenario = RedshiftScenario(redshift_wrapper, redshift_data_wrapper)
    redshift_scenario.redhift_scenario(
```

```
        f"{os.path.dirname(__file__)}/../../../resources/sample_files/
movies.json"
    )
```

Die im Szenario verwendete Wrapper-Funktionen.

```
    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def execute_statement(
        self, cluster_identifier, database_name, user_name, sql,
parameter_list=None
    ):
        """
        Executes a SQL statement.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param user_name: The user's name.
```

```
        :param sql: The SQL statement.
        :param parameter_list: The optional SQL statement parameters.
        :return: The SQL statement result.
        """

        try:
            kwargs = {
                "ClusterIdentifier": cluster_identifier,
                "Database": database_name,
                "DbUser": user_name,
                "Sql": sql,
            }
            if parameter_list:
                kwargs["Parameters"] = parameter_list
            response = self.client.execute_statement(**kwargs)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't execute statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


    def list_databases(self, cluster_identifier, database_name, database_user):
        """
        Lists databases in a cluster.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param database_user: The database user.
        :return: The list of databases.
        """
        try:
            paginator = self.client.get_paginator("list_databases")
            databases = []
            for page in paginator.paginate(
                ClusterIdentifier=cluster_identifier,
                Database=database_name,
                DbUser=database_user,
            ):
                databases.extend(page["Databases"])

            return databases
        except ClientError as err:
            logging.error(
                "Couldn't list databases. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
```

```
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Aktionen für Amazon Redshift mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Redshift Redshift-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Redshift-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Redshift AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Redshift.

Beispiele

- Verwenden Sie CreateClustermit einem AWS SDK oder CLI
- Verwenden Sie DeleteClustermit einem AWS SDK oder CLI
- Verwenden Sie DescribeClustersmit einem AWS SDK oder CLI
- Verwenden Sie DescribeStatementmit einem AWS SDK

- Verwenden Sie ExecuteStatementmit einem AWS SDK
- Verwenden Sie GetStatementResultmit einem AWS SDK
- Verwenden Sie ListDatabasesmit einem AWS SDK
- Verwenden Sie ModifyClustermit einem AWS SDK oder CLI

Verwenden Sie **`CreateCluster`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCluster` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier,
string databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
```

```
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Das ParametersThis Beispiel „Cluster mit minimalem Wert erstellen“ erstellt einen Cluster mit einem minimalen Parametersatz. JSON ist das Standardausgabeformat. Befehl:

```
aws redshift create-cluster --node-type dw.hs1.xlarge --number-of-nodes 2 --
master-username adminuser --master-user-password TopSecret1 --cluster-identifier
 mycluster
```

Ergebnis:

```
{
   "Cluster": {
      "NodeType": "dw.hs1.xlarge",
      "ClusterVersion": "1.0",
      "PubliclyAccessible": "true",
      "MasterUsername": "adminuser",
      "ClusterParameterGroups": [
         {
            "ParameterApplyStatus": "in-sync",
            "ParameterGroupName": "default.redshift-1.0"
         } ],
      "ClusterSecurityGroups": [
         {
            "Status": "active",
            "ClusterSecurityGroupName": "default"
         } ],
      "AllowVersionUpgrade": true,
      "VpcSecurityGroups": \[],
      "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
      "AutomatedSnapshotRetentionPeriod": 1,
      "ClusterStatus": "creating",
      "ClusterIdentifier": "mycluster",
      "DBName": "dev",
      "NumberOfNodes": 2,
      "PendingModifiedValues": {
         "MasterUserPassword": "\****"
      }
   },
   "ResponseMetadata": {
      "RequestId": "7cf4bcfc-64dd-11e2-bea9-49e0ce183f07"
   }
}
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// CreateCluster sends a request to create a cluster with the given clusterId
 using the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
 userName string, userPassword string, nodeType string, clusterType string,
 publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
 // Create a new Redshift cluster
 input := &redshift.CreateClusterInput{
  ClusterIdentifier:  aws.String(clusterId),
  MasterUserPassword: aws.String(userPassword),
  MasterUsername:     aws.String(userName),
  NodeType:           aws.String(nodeType),
```

```
  ClusterType:        aws.String(clusterType),
  PubliclyAccessible: aws.Bool(publiclyAccessible),
 }
 var opErr *types.ClusterAlreadyExistsFault
 output, err := actor.RedshiftClient.CreateCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster already exists")
  return nil, nil
 } else if err != nil {
  log.Printf("Failed to create Redshift cluster: %v\n", err)
  return nil, err
 }

 log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
 return output, nil
}
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
 creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
```

```
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
 clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
 exception.getMessage(), exception);
                }
            });
    }
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
```

```
export { redshiftClient };
```

Erstellen Sie den -Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME", // Required
  NodeType: "NODE_TYPE", //Required
  MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
  MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
 one uppercase letter, and one number
  ClusterType: "CLUSTER_TYPE", // Required
  IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with
 permissions your cluster needs to access other AWS services on your behalf, such
 as Amazon S3.
  ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
 cluster subnet group to be associated with this cluster. Defaults to 'default'
 if not specified.
  DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
  Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new CreateClusterCommand(params));
    console.log(
      `Cluster ${data.Cluster.ClusterIdentifier} successfully created`,
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie CreateClusterin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient
 ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- API-Details finden Sie CreateClusterin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """
```

```
        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Einzelheiten zur API finden Sie CreateClusterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        " Example values: iv_node_type = 'ra3.4xlarge'
        " Example values: iv_master_username = 'awsuser'
        " Example values: iv_master_password = 'AwsUser1000'
        " Example values: iv_publicly_accessible = abap_true
        " Example values: iv_number_of_nodes = 2
        oo_result = lo_rsh->createcluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_nodetype = iv_node_type
          iv_masterusername = iv_master_username
          iv_masteruserpassword = iv_master_password
          iv_publiclyaccessible = iv_publicly_accessible
          iv_numberofnodes = iv_number_of_nodes
        ).
        MESSAGE 'Redshift cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustalrdyexfault.
        MESSAGE 'Cluster already exists.' TYPE 'I'.
      CATCH /aws1/cx_rshclstquotaexcdfault.
        MESSAGE 'Cluster quota exceeded.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateClusterin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteCluster`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit bereitgestellten Data Warehouse-Clustern

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Delete an Amazon Redshift cluster without a final snapshot.
   /// </summary>
   /// <param name="clusterIdentifier">The identifier for the cluster.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
   {
       try
       {
           var request = new DeleteClusterRequest
           {
               ClusterIdentifier = clusterIdentifier,
               SkipFinalClusterSnapshot = true
           };

           var response = await _redshiftClient.DeleteClusterAsync(request);
           Console.WriteLine($"The {clusterIdentifier} was deleted");
           return true;
       }
       catch (ClusterNotFoundException ex)
       {
           Console.WriteLine($"Cluster not found: {ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           Console.WriteLine($"Couldn't delete cluster. Here's why:
{ex.Message}");
           return false;
       }
   }
```

- Einzelheiten zur API finden Sie DeleteClusterin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Ein SnapshotThis Beispiel für das Löschen eines Clusters ohne endgültigen Cluster löscht einen Cluster und erzwingt das Löschen von Daten, sodass kein endgültiger Cluster-Snapshot erstellt wird.Befehl:

```
aws redshift delete-cluster --cluster-identifier mycluster --skip-final-cluster-
snapshot
```

Das SnapshotThis Beispiel „Cluster löschen, einen finalen Cluster zulassen“ löscht einen Cluster, gibt aber einen endgültigen Cluster-Snapshot an.Befehl:

```
aws redshift delete-cluster --cluster-identifier mycluster --final-cluster-
snapshot-identifier myfinalsnapshot
```

- Einzelheiten zur API finden Sie in der Befehlsreferenz DeleteCluster.AWS CLI

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
 (bool, error) {
 input := redshift.DeleteClusterInput{
  ClusterIdentifier:        aws.String(clusterId),
  SkipFinalClusterSnapshot: aws.Bool(true),
 }
 _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
 var opErr *types.ClusterNotFoundFault
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster was not found. Where could it be?")
  return false, err
 } else if err != nil {
  log.Printf("Failed to delete Redshift cluster: %v\n", err)
  return false, err
 }
 waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
 }
 log.Printf("The cluster %s was deleted\n", clusterId)
 return true, nil
}
```

- Einzelheiten zur API finden Sie DeleteClusterin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {}",
exception.getMessage());
                   }
               } else {
                   // Handle successful response
```

```
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
```

- Einzelheiten zur API finden Sie DeleteClusterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Erstellen Sie den -Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};
```

```
const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie DeleteClusterin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- API-Details finden Sie DeleteClusterin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier,
 SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
client = boto3.client("redshift")
redhift_wrapper = RedshiftWrapper(client)
```

- Einzelheiten zur API finden Sie DeleteClusterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster'
    lo_rsh->deletecluster(
      iv_clusteridentifier = iv_cluster_identifier
      iv_skipfinalclustersnapshot = abap_true
    ).
    MESSAGE 'Redshift cluster deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_rshclustnotfoundfault.
    MESSAGE 'Cluster not found.' TYPE 'I'.
  CATCH /aws1/cx_rshinvcluststatefault.
    MESSAGE 'Invalid cluster state for deletion.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteClusterin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeClusters`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeClusters` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Describe Amazon Redshift clusters.
   /// </summary>
   /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
   /// <returns>A list of clusters.</returns>
   public async Task<List<Cluster>> DescribeClustersAsync(string?
clusterIdentifier = null)
   {
       try
       {
           var clusters = new List<Cluster>();
           var request = new DescribeClustersRequest();
           if (!string.IsNullOrEmpty(clusterIdentifier))
           {
               request.ClusterIdentifier = clusterIdentifier;
           }

           var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
           await foreach (var response in clustersPaginator.Responses)
```

```
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Das ClustersThis Beispiel Get a Description of All gibt eine Beschreibung aller Cluster für das Konto zurück. JSON ist das Standardausgabeformat. Befehl:

```
aws redshift describe-clusters
```

Ergebnis:

```
{
   "Clusters": [
   {
      "NodeType": "dw.hs1.xlarge",
      "Endpoint": {
         "Port": 5439,
         "Address": "mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com"
      },
      "ClusterVersion": "1.0",
      "PubliclyAccessible": "true",
      "MasterUsername": "adminuser",
      "ClusterParameterGroups": [
         {
            "ParameterApplyStatus": "in-sync",
            "ParameterGroupName": "default.redshift-1.0"
         } ],
      "ClusterSecurityGroups": [
         {
            "Status": "active",
            "ClusterSecurityGroupName": "default"
         } ],
      "AllowVersionUpgrade": true,
      "VpcSecurityGroups": \[],
      "AvailabilityZone": "us-east-1a",
      "ClusterCreateTime": "2013-01-22T21:59:29.559Z",
      "PreferredMaintenanceWindow": "sat:03:30-sat:04:00",
      "AutomatedSnapshotRetentionPeriod": 1,
      "ClusterStatus": "available",
      "ClusterIdentifier": "mycluster",
      "DBName": "dev",
      "NumberOfNodes": 2,
      "PendingModifiedValues": {}
   } ],
   "ResponseMetadata": {
      "RequestId": "65b71cac-64df-11e2-8f5b-e90bd6c77476"
   }
}
```

Sie können dieselben Informationen auch mit der folgenden `--output text` Option im Textformat abrufen:

`--output text`Option.Befehl:

Option.Befehl:

```
aws redshift describe-clusters --output text
```

Ergebnis:

```
dw.hs1.xlarge       1.0     true    adminuser       True    us-east-1a
 2013-01-22T21:59:29.559Z        sat:03:30-sat:04:00     1       available
 mycluster       dev     2
ENDPOINT    5439    mycluster.coqoarplqhsn.us-east-1.redshift.amazonaws.com
in-sync     default.redshift-1.0
active      default
PENDINGMODIFIEDVALUES
RESPONSEMETADATA    934281a8-64df-11e2-b07c-f7fbdd006c67
```

- Einzelheiten zur API finden Sie DescribeClustersunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)
```

```
// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DescribeClusters returns information about the given cluster.
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId
 string) (*redshift.DescribeClustersOutput, error) {
 input, err := actor.RedshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 })
 var opErr *types.AccessToClusterDeniedFault
 if errors.As(err, &opErr) {
  println("Access to cluster denied.")
  panic(err)
 } else if err != nil {
  println("Failed to describe Redshift clusters.")
  return nil, err
 }
 return input, nil
}
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
    /**
```

```
    * Waits asynchronously for the specified cluster to become available.
    * @param clusterId the identifier of the cluster to wait for
    * @return a {@link CompletableFuture} that completes when the cluster is
ready
    */
   public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
       DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
           .clusterIdentifier(clusterId)
           .build();

       logger.info("Waiting for cluster to become available. This may take a few
minutes.");
       long startTime = System.currentTimeMillis();

       // Recursive method to poll the cluster status.
       return checkClusterStatusAsync(clustersRequest, startTime);
   }

   private CompletableFuture<Void>
checkClusterStatusAsync(DescribeClustersRequest clustersRequest, long startTime)
{
       return getAsyncClient().describeClusters(clustersRequest)
           .thenCompose(clusterResponse -> {
               List<Cluster> clusterList = clusterResponse.clusters();
               boolean clusterReady = false;
               for (Cluster cluster : clusterList) {
                   if ("available".equals(cluster.clusterStatus())) {
                       clusterReady = true;
                       break;
                   }
               }

               if (clusterReady) {
                   logger.info(String.format("Cluster is available!"));
                   return CompletableFuture.completedFuture(null);
               } else {
                   long elapsedTimeMillis = System.currentTimeMillis() -
startTime;
                   long elapsedSeconds = elapsedTimeMillis / 1000;
                   long minutes = elapsedSeconds / 60;
                   long seconds = elapsedSeconds % 60;
                   System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
```

```
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Beschreiben Sie Ihre Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie DescribeClustersin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
suspend fun describeRedshiftClusters() {
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient
 ->
```

```
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- API-Details finden Sie DescribeClustersin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.
```

```
        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Einzelheiten zur API finden Sie DescribeClustersin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-
cluster' (optional)
        oo_result = lo_rsh->describeclusters(
          iv_clusteridentifier = iv_cluster_identifier
        ).
        lt_clusters = oo_result->get_clusters( ).
        lv_cluster_count = lines( lt_clusters ).
        MESSAGE |Retrieved { lv_cluster_count } cluster(s).| TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
        MESSAGE 'Cluster not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeClustersin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeStatement`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeStatement` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeStatementin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
    *
    * @param sqlId the ID of the SQL statement to check
    * @return a {@link CompletableFuture} that completes when the SQL
statement's status is either "FINISHED" or "FAILED"
    */
   public CompletableFuture<Void> checkStatementAsync(String sqlId) {
       DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
           .id(sqlId)
           .build();

       return getAsyncDataClient().describeStatement(statementRequest)
           .thenCompose(response -> {
               String status = response.statusAsString();
               logger.info("... Status: {} ", status);

               if ("FAILED".equals(status)) {
                   throw new RuntimeException("The Query Failed. Ending
program");
               } else if ("FINISHED".equals(status)) {
                   return CompletableFuture.completedFuture(null);
               } else {
                   // Sleep for 1 second and recheck status
                   return CompletableFuture.runAsync(() -> {
                       try {
                           TimeUnit.SECONDS.sleep(1);
                       } catch (InterruptedException e) {
```

```
                                throw new RuntimeException("Error during sleep: " +
 e.getMessage(), e);
                            }
                        }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
 Recursively call until status is FINISHED or FAILED
                    }
                }).whenComplete((result, exception) -> {
                    if (exception != null) {
                        // Handle exceptions
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("The statement is finished!");
                    }
                });
    }
```

- Einzelheiten zur API finden Sie DescribeStatementin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def describe_statement(self, statement_id):
```

```
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftDataWrapper Objekt.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Einzelheiten zur API finden Sie DescribeStatementin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
```

```
        oo_result = lo_rsd->describestatement(
          iv_id = iv_statement_id
        ).
        lv_status = oo_result->get_status( ).
        MESSAGE |Statement status: { lv_status }| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeStatementin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ExecuteStatement`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ExecuteStatement` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führt eine SQL-Anweisung aus, um eine Datenbanktabelle zu erstellen.

```
    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
```

```
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the
SQL statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

Führt eine SQL-Anweisung aus, um Daten in eine Datenbanktabelle einzufügen.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records
added to the Movies table
```

```
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON
file: " + e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = iter.next();
                   int year = currentNode.get("year").asInt();
                   String title = currentNode.get("title").asText();

                   // Use SqlParameter to avoid SQL injection.
                   List<SqlParameter> parameterList = new ArrayList<>();
                   String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                   SqlParameter idParam = SqlParameter.builder()
                       .name("id")
```

```
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();


getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

Führt eine SQL-Anweisung aus, um eine Datenbanktabelle abzufragen.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
```

```
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String
clusterId) {

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK for Java 2.x API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_user_name = 'awsuser'
        " Example values: iv_sql = 'SELECT * FROM movies WHERE year = :year'
        " Example values: it_parameter_list - SQL parameters for parameterized
queries

        " Only pass parameters if the list is not empty
        IF it_parameter_list IS NOT INITIAL.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
            it_parameters = it_parameter_list
          ).
        ELSE.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
          ).
        ENDIF.
```

```
        lv_statement_id = oo_result->get_id( ).
        MESSAGE |Statement executed. ID: { lv_statement_id }| TYPE 'I'.
      CATCH /aws1/cx_rsdexecutestatementex.
        MESSAGE 'Statement execution error.' TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ExecuteStatementin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetStatementResult`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetStatementResult` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
    public async Task<List<List<Field>>> GetStatementResultAsync(string
statementId)
    {
        try
        {
```

```
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie GetStatementResultin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Überprüfen Sie das Ergebnis der Anweisung.

```
    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the
results
```

```
    * @return a {@link CompletableFuture} that completes when the statement
result has been processed
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: "
+ exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the
response is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field
is " + value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }
```

- Einzelheiten zur API finden Sie GetStatementResultin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

Der folgende Code instanziiert das RedshiftDataWrapper Objekt.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- Einzelheiten zur API finden Sie GetStatementResultin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Überprüfen Sie das Ergebnis der Anweisung.

```
    TRY.
        " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'
        " Handle pagination for large result sets

        DO.
          lo_result_page = lo_rsd->getstatementresult(
            iv_id = iv_statement_id
            iv_nexttoken = lv_next_token
          ).

          " Collect records from this page
          lt_page_records = lo_result_page->get_records( ).
          APPEND LINES OF lt_page_records TO lt_all_records.
```

```
          " Check if there are more pages
          lv_next_token = lo_result_page->get_nexttoken( ).
          IF lv_next_token IS INITIAL.
            EXIT. " No more pages
          ENDIF.
        ENDDO.

        " For the last call, set oo_result for return value
        oo_result = lo_result_page.
        lv_record_count = lines( lt_all_records ).
        MESSAGE |Retrieved { lv_record_count } record(s).| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found or results not available.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetStatementResultin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListDatabases`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListDatabases` verwendet wird.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
```

```
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie ListDatabasesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists all databases asynchronously for the specified cluster, database
user, and database.
    * @param clusterId the identifier of the cluster to list databases for
    * @param dbUser the database user to use for the list databases request
    * @param database the database to list databases for
    * @return a {@link CompletableFuture} that completes when the database
listing is complete, or throws a {@link RuntimeException} if there was an error
    */
   public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
       ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(dbUser)
           .database(database)
           .build();

       // Asynchronous paginator for listing databases.
       ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
       CompletableFuture<Void> future = databasesPaginator.subscribe(response ->
{
           response.databases().forEach(db -> {
               logger.info("The database name is {} ", db);
           });
       });

       // Return the future for asynchronous handling.
       return future.exceptionally(exception -> {
           throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
       });
```

```
    }
```

- Einzelheiten zur API finden Sie ListDatabasesin der AWS SDK for Java 2.x API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
    " Example values: iv_database_name = 'dev'
    " Example values: iv_database_user = 'awsuser'
    oo_result = lo_rsd->listdatabases(
      iv_clusteridentifier = iv_cluster_identifier
      iv_database = iv_database_name
      iv_dbuser = iv_database_user
    ).
    lt_databases = oo_result->get_databases( ).
    lv_db_count = lines( lt_databases ).
    MESSAGE |Retrieved { lv_db_count } database(s).| TYPE 'I'.
  CATCH /aws1/cx_rsddatabaseconnex.
    MESSAGE 'Database connection error.' TYPE 'I'.
  CATCH /aws1/cx_rsdresourcenotfoundex.
    MESSAGE 'Cluster not found.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListDatabasesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ModifyCluster`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyCluster` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance
window.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified
and has {response.Cluster.PreferredMaintenanceWindow} as the maintenance
window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
```

```
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why:
 {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie ModifyClusterin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

In einem ClusterThis Beispiel wird gezeigt, wie eine Cluster-Sicherheitsgruppe dem angegebenen Cluster zugeordnet wird. Befehl:

```
aws redshift modify-cluster --cluster-identifier mycluster --cluster-security-
groups mysecuritygroup
```

In Modify the Maintenance Window for a wird ClusterThis gezeigt, wie das bevorzugte wöchentliche Wartungsfenster für einen Cluster so geändert werden kann, dass es mindestens vier Stunden dauert und sonntags um 23:15 Uhr beginnt und montags um 15:15 Uhr endet: AM.Command

```
aws redshift modify-cluster --cluster-identifier mycluster --preferred-
maintenance-window Sun:23:15-Mon:03:15
```

Das Master-Passwort ändern Das ClusterThis Beispiel zeigt, wie das Master-Passwort für einen Cluster geändert wird. Befehl:

```
aws redshift modify-cluster --cluster-identifier mycluster --master-user-password
 A1b2c3d4
```

- Einzelheiten zur API finden Sie unter Befehlsreferenz ModifyCluster.AWS CLI

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
 maintenanceWindow string) *redshift.ModifyClusterOutput {
 // Modify the cluster's maintenance window
 input := &redshift.ModifyClusterInput{
  ClusterIdentifier:          aws.String(clusterId),
  PreferredMaintenanceWindow: aws.String(maintenanceWindow),
 }

 var opErr *types.InvalidClusterStateFault
 output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
```

```
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster is in an invalid state.")
  panic(err)
 } else if err != nil {
  log.Printf("Failed to modify Redshift cluster: %v\n", err)
  panic(err)
 }

 log.Printf("The cluster was successfully modified and now has %s as the
 maintenance window\n", *output.Cluster.PreferredMaintenanceWindow)
 return output
}
```

- Einzelheiten zur API finden Sie ModifyClusterin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
 modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
 clusterId) {
       ModifyClusterRequest modifyClusterRequest =
 ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
```

```
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified
and has "
                        + clusterResponse.cluster().preferredMaintenanceWindow()
+ " as the maintenance window");
                }
            });
    }
```

- Einzelheiten zur API finden Sie ModifyClusterin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Ändern Sie einen Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Einzelheiten zur API finden Sie ModifyClusterin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
suspend fun modifyCluster(clusterId: String?) {
```

```
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance
 window",
        )
    }
}
```

- API-Details finden Sie ModifyClusterin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
```

```
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- Einzelheiten zur API finden Sie ModifyClusterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        " Example values: iv_pref_maintenance_wn = 'wed:07:30-wed:08:00'
        lo_rsh->modifycluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_preferredmaintenancewin00 = iv_pref_maintenance_wn
        ).
        MESSAGE 'Redshift cluster modified successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
        MESSAGE 'Cluster not found.' TYPE 'I'.
      CATCH /aws1/cx_rshinvcluststatefault.
        MESSAGE 'Invalid cluster state for modification.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ModifyClusterin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Redshift AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Redshift mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Redshift oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen eines Amazon-Redshift-Element-Trackers
- Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
- Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Erstellen eines Amazon-Redshift-Element-Trackers

Die folgenden Code-Beispiele zeigen, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace

- Benutze Redshift CreateWorkgroup
- Benutze Redshift DeleteNamespace
- Benutze iam CreateRole

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
```

```
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
```

```
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
```

```
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5
```

```
  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "==========================================="
  echo "CLEANUP CONFIRMATION"
  echo "==========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
```

```
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
```

```
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"
```

```
# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"
```

```
# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)
```

```
# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
```

```
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
```

```
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
```

```
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
```

```
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20
```

```
  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
```

```
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Redshift-Cluster

- Erstellen Sie eine IAM-Rolle für den S3-Zugriff
- Erstellen Sie Tabellen und laden Sie Daten
- Führen Sie Beispielabfragen aus
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
```

```
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""
```

```
    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
```

```
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
```

```
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
```

```
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi
```

```
echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"
```

```
# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
```

```
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Codebeispiele für Amazon Rekognition mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Rekognition mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-Rekognition-Entwicklerhandbuch – Weitere Informationen zu Amazon Rekognition.
- API-Referenz für Amazon Rekognition – Details zu allen verfügbaren Amazon-Rekognition-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Rekognition AWS SDKs
  - Hello Amazon Rekognition
  - Aktionen für Amazon Rekognition mit AWS SDKs
    - Verwenden Sie CompareFacesmit einem AWS SDK oder CLI
    - Verwenden Sie CreateCollectionmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteCollectionmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteFacesmit einem AWS SDK oder CLI
    - Verwenden Sie DescribeCollectionmit einem AWS SDK oder CLI
    - Verwenden Sie DetectFacesmit einem AWS SDK oder CLI
    - Verwenden Sie DetectLabelsmit einem AWS SDK oder CLI
    - Verwenden Sie DetectModerationLabelsmit einem AWS SDK oder CLI
    - Verwenden Sie DetectTextmit einem AWS SDK oder CLI
    - Verwenden Sie GetCelebrityInfomit einem AWS SDK oder CLI
    - Verwenden Sie IndexFacesmit einem AWS SDK oder CLI
    - Verwenden Sie ListCollectionsmit einem AWS SDK oder CLI
    - Verwenden Sie ListFacesmit einem AWS SDK oder CLI
    - Verwenden Sie RecognizeCelebritiesmit einem AWS SDK oder CLI
    - Verwenden Sie SearchFacesmit einem AWS SDK oder CLI
    - Verwenden Sie SearchFacesByImagemit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Rekognition AWS SDKs
  - Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin nach Gesichtern mithilfe eines AWS SDK
  - Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
  - Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

- Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Gesichter in einem Bild mit einem AWS SDK
- Erkennen Sie Informationen in Videos mithilfe von Amazon Rekognition und dem AWS SDK
- Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Rekognition AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Rekognition mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon Rekognition
- Aktionen für Amazon Rekognition mit AWS SDKs
  - Verwenden Sie CompareFacesmit einem AWS SDK oder CLI
  - Verwenden Sie CreateCollectionmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteCollectionmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteFacesmit einem AWS SDK oder CLI
  - Verwenden Sie DescribeCollectionmit einem AWS SDK oder CLI
  - Verwenden Sie DetectFacesmit einem AWS SDK oder CLI
  - Verwenden Sie DetectLabelsmit einem AWS SDK oder CLI
  - Verwenden Sie DetectModerationLabelsmit einem AWS SDK oder CLI
  - Verwenden Sie DetectTextmit einem AWS SDK oder CLI
  - Verwenden Sie GetCelebrityInfomit einem AWS SDK oder CLI
  - Verwenden Sie IndexFacesmit einem AWS SDK oder CLI
  - Verwenden Sie ListCollectionsmit einem AWS SDK oder CLI
  - Verwenden Sie ListFacesmit einem AWS SDK oder CLI
  - Verwenden Sie RecognizeCelebritiesmit einem AWS SDK oder CLI

- Verwenden Sie SearchFacesmit einem AWS SDK oder CLI
- Verwenden Sie SearchFacesByImagemit einem AWS SDK oder CLI

Hello Amazon Rekognition

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon Rekognition beginnen.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
```

```
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_rekognition.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 *  A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
 *  lists the Amazon Rekognition collections in the current account and region.
 *
 *  main function
 *
 *  Usage: 'hello_rekognition'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
                rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
 outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                      << std::endl;
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Einzelheiten zur API finden Sie ListCollectionsunter AWS SDK für C++ API-Referenz.

Aktionen für Amazon Rekognition mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Rekognition Rekognition-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-Rekognition-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Rekognition AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon-Rekognition-API-Referenz.

Beispiele

- Verwenden Sie CompareFacesmit einem AWS SDK oder CLI
- Verwenden Sie CreateCollectionmit einem AWS SDK oder CLI
- Verwenden Sie DeleteCollectionmit einem AWS SDK oder CLI
- Verwenden Sie DeleteFacesmit einem AWS SDK oder CLI
- Verwenden Sie DescribeCollectionmit einem AWS SDK oder CLI
- Verwenden Sie DetectFacesmit einem AWS SDK oder CLI
- Verwenden Sie DetectLabelsmit einem AWS SDK oder CLI
- Verwenden Sie DetectModerationLabelsmit einem AWS SDK oder CLI
- Verwenden Sie DetectTextmit einem AWS SDK oder CLI
- Verwenden Sie GetCelebrityInfomit einem AWS SDK oder CLI
- Verwenden Sie IndexFacesmit einem AWS SDK oder CLI
- Verwenden Sie ListCollectionsmit einem AWS SDK oder CLI
- Verwenden Sie ListFacesmit einem AWS SDK oder CLI
- Verwenden Sie RecognizeCelebritiesmit einem AWS SDK oder CLI
- Verwenden Sie SearchFacesmit einem AWS SDK oder CLI
- Verwenden Sie SearchFacesByImagemit einem AWS SDK oder CLI

Verwenden Sie **`CompareFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CompareFaces` verwendet wird.

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// </summary>
public class CompareFaces
{
    public static async Task Main()
    {
        float similarityThreshold = 70F;
        string sourceImage = "source.jpg";
        string targetImage = "target.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

        try
        {
            using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
            byte[] data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
            imageSource.Bytes = new MemoryStream(data);
        }
```

```
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
 Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
 FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
 rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top}
 matches with {match.Similarity}% confidence.");
            });
```

```
            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
 face(s) that did not match.");
        }
    }
```

- Einzelheiten zur API finden Sie CompareFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So vergleichen Sie Gesichter in zwei Bildern

Der folgende `compare-faces`-Befehl vergleicht Gesichter in zwei Bildern, die in einem Amazon-S3-Bucket gespeichert sind.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Ausgabe:

```
{
    "UnmatchedFaces": [],
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.12368916720151901,
                    "Top": 0.16007372736930847,
                    "Left": 0.5901257991790771,
                    "Height": 0.25140416622161865
                },
                "Confidence": 100.0,
                "Pose": {
                    "Yaw": -3.7351467609405518,
                    "Roll": -0.10309021919965744,
```

```
                    "Pitch": 0.8637830018997192
                },
                "Quality": {
                    "Sharpness": 95.51618957519531,
                    "Brightness": 65.29893493652344
                },
                "Landmarks": [
                    {
                        "Y": 0.26721030473709106,
                        "X": 0.6204193830490112,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.26831310987472534,
                        "X": 0.6776827573776245,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3514654338359833,
                        "X": 0.6241428852081299,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35258132219314575,
                        "X": 0.6713621020317078,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.3140771687030792,
                        "X": 0.6428444981575012,
                        "Type": "nose"
                    }
                ]
            },
            "Similarity": 100.0
        }
    ],
    "SourceImageFace": {
        "BoundingBox": {
            "Width": 0.12368916720151901,
            "Top": 0.16007372736930847,
            "Left": 0.5901257991790771,
            "Height": 0.25140416622161865
        },
```

```
        "Confidence": 100.0
    }
}
```

Weitere Informationen finden Sie unter Vergleichen von Gesichtern in Bildern im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie CompareFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are
stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

    /**
     * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
     *
     * <p>This method takes two image keys from an S3 bucket and compares the
faces within them.
     * It prints out the confidence level of matched faces and reports the number
of unmatched faces.</p>
     *
     * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
     * @param bucketName  The name of the S3 bucket containing the images.
     * @param sourceKey   The object key (file path) for the source image in the
S3 bucket.
     * @param targetKey   The object key (file path) for the target image in the
S3 bucket.
     * @throws RuntimeException If the Rekognition service returns an error.
     */
```

```
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(targetKey)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                   .sourceImage(sourceImage)
                   .targetImage(targetImage)
                   .similarityThreshold(similarityThreshold)
                   .build();

           // Compare the two images.
           CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
           List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();

           for (CompareFacesMatch match : faceDetails) {
               ComparedFace face = match.face();
               BoundingBox position = face.boundingBox();
               System.out.println("Face at " + position.left().toString()
                       + " " + position.top()
                       + " matches with " + face.confidence().toString()
                       + "% confidence.");
           }
```

```
            List<ComparedFace> unmatchedFaces =
 compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Einzelheiten zur API finden Sie CompareFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
```

```
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
 with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- API-Details finden Sie CompareFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
 greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
 have
                 a similarity value below the specified threshold.
```

```
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- Einzelheiten zur API finden Sie CompareFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the source image
        DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_source_s3_bucket
          iv_name = iv_source_s3_key ).

        " Create source image object
        DATA(lo_source_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_source_s3obj ).

        " Create S3 object reference for the target image
        DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_target_s3_bucket
          iv_name = iv_target_s3_key ).

        " Create target image object
        DATA(lo_target_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_target_s3obj ).

        " Compare faces
        oo_result = lo_rek->comparefaces(
          io_sourceimage = lo_source_image
          io_targetimage = lo_target_image
          iv_similaritythreshold = iv_similarity ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

        " Get counts of matched and unmatched faces
        DATA(lv_matched_count) = lines( lt_face_matches ).
        DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).

        " Output detailed comparison results
        DATA(lv_message) = |Face comparison completed: | &&
                           |{ lv_matched_count } matched face(s), | &&
                           |{ lv_unmatched_count } unmatched face(s).|.
        MESSAGE lv_message TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CompareFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateCollection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCollection` verwendet wird.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };
```

```
            CreateCollectionResponse createCollectionResponse = await
 rekognitionClient.CreateCollectionAsync(createCollectionRequest);
            Console.WriteLine($"CollectionArn :
 {createCollectionResponse.CollectionArn}");
            Console.WriteLine($"Status code :
 {createCollectionResponse.StatusCode}");
        }
    }
```

- Einzelheiten zur API finden Sie CreateCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Sammlung

Der folgende `create-collection`-Befehl erstellt eine Sammlung mit dem angegebenen Namen.

```
aws rekognition create-collection \
    --collection-id "MyCollection"
```

Ausgabe:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Weitere Informationen finden Sie unter Erstellen einer Sammlung im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie CreateCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
```

```
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- API-Details finden Sie CreateCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
```

```
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Einzelheiten zur API finden Sie CreateCollectionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->createcollection(
      iv_collectionid = iv_collection_id ).
    MESSAGE 'Collection created successfully.' TYPE 'I'.
  CATCH /aws1/cx_rekresrcalrdyexistsex.
    MESSAGE 'Collection already exists.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateCollectionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteCollection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteCollection` verwendet wird.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete an existing collection.
    /// </summary>
    public class DeleteCollection
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine("Deleting collection: " + collectionId);

            var deleteCollectionRequest = new DeleteCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var deleteCollectionResponse = await
 rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
            Console.WriteLine($"{collectionId}:
 {deleteCollectionResponse.StatusCode}");
        }
    }
```

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie eine Sammlung

Der folgende `delete-collection`-Befehl löscht die angegebene Sammlung.

```
aws rekognition delete-collection \
```

```
    --collection-id MyCollection
```

Ausgabe:

```
{
    "StatusCode": 200
}
```

Weitere Informationen finden Sie unter Löschen einer Sammlung im Entwicklerhandbuch Amazon für Rekognition.

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
```

```
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
```

```
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- API-Details finden Sie DeleteCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise
```

- Einzelheiten zur API finden Sie DeleteCollectionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_rek->deletecollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
```

```
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteCollectionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteFaces` verwendet wird.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();
```

```
            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
                Console.WriteLine($"FaceID: {face}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie DeleteFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie Gesichter aus einer Sammlung

Der folgende `delete-faces`-Befehl löscht das angegebene Gesicht aus einer Sammlung.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Ausgabe:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Weitere Informationen finden Sie unter Löschen von Gesichtern in einer Sammlung im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie DeleteFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- API-Details finden Sie DeleteFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
```

```
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
```

```
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.",
self.collection_id)
            raise
        else:
            return deleted_ids
```

- Einzelheiten zur API finden Sie DeleteFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_rek->deletefaces(
          iv_collectionid = iv_collection_id
          it_faceids = it_face_ids ).

        DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
        DATA(lv_deleted_count) = lines( lt_deleted_faces ).
        DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
        MESSAGE lv_msg6 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeCollection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCollection` verwendet wird.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");

        var describeCollectionRequest = new DescribeCollectionRequest()
```

```
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
 rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
 {describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
 {describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
 {describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
 {describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie eine Sammlung

Im folgenden Beispiel für `describe-collection` werden die Details für die angegebene Sammlung angezeigt.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Ausgabe:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
}
```

Weitere Informationen finden Sie unter Beschreiben einer Sammlung im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

- Einzelheiten zur API finden Sie DescribeCollectionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- API-Details finden Sie DescribeCollectionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Einzelheiten zur API finden Sie DescribeCollectionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_rek->describecollection(
         iv_collectionid = iv_collection_id ).
       DATA(lv_face_count) = oo_result->get_facecount( ).
       DATA(lv_msg) = |Collection described: { lv_face_count } face(s)
indexed.|.
       MESSAGE lv_msg TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeCollectionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectFaces` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },

            // Attributes can be "ALL" or "DEFAULT".
```

```
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Zeigt Informationen zum Begrenzungsrahmen für alle Gesichter in einem Bild an.

```
    using System;
```

```
    using System.Collections.Generic;
    using System.Drawing;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to display the details of the
    /// bounding boxes around the faces detected in an image.
    /// </summary>
    public class ImageOrientationBoundingBox
    {
        public static async Task Main()
        {
            string photo = @"D:\Development\AWS-Examples\Rekognition
\target.jpg"; // "photo.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
 FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            int height;
            int width;

            // Used to extract original photo width/height
            using (var imageBitmap = new Bitmap(photo))
            {
                height = imageBitmap.Height;
                width = imageBitmap.Width;
```

```
            }

            Console.WriteLine("Image Information:");
            Console.WriteLine(photo);
            Console.WriteLine("Image Height: " + height);
            Console.WriteLine("Image Width: " + width);

            try
            {
                var detectFacesRequest = new DetectFacesRequest()
                {
                    Image = image,
                    Attributes = new List<string>() { "ALL" },
                };

                DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
                detectFacesResponse.FaceDetails.ForEach(face =>
                {
                    Console.WriteLine("Face:");
                    ShowBoundingBoxPositions(
                        height,
                        width,
                        face.BoundingBox,
                        detectFacesResponse.OrientationCorrection);

                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
                    Console.WriteLine($"The detected face is estimated to be
between {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Display the bounding box information for an image.
        /// </summary>
        /// <param name="imageHeight">The height of the image.</param>
        /// <param name="imageWidth">The width of the image.</param>
```

```
        /// <param name="box">The bounding box for a face found within the
 image.</param>
        /// <param name="rotation">The rotation of the face's bounding box.</
param>
        public static void ShowBoundingBoxPositions(int imageHeight, int
 imageWidth, BoundingBox box, string rotation)
        {
            float left;
            float top;

            if (rotation == null)
            {
                Console.WriteLine("No estimated orientation. Check Exif data.");
                return;
            }

            // Calculate face position based on image orientation.
            switch (rotation)
            {
                case "ROTATE_0":
                    left = imageWidth * box.Left;
                    top = imageHeight * box.Top;
                    break;
                case "ROTATE_90":
                    left = imageHeight * (1 - (box.Top + box.Height));
                    top = imageWidth * box.Left;
                    break;
                case "ROTATE_180":
                    left = imageWidth - (imageWidth * (box.Left + box.Width));
                    top = imageHeight * (1 - (box.Top + box.Height));
                    break;
                case "ROTATE_270":
                    left = imageHeight * box.Top;
                    top = imageWidth * (1 - box.Left - box.Width);
                    break;
                default:
                    Console.WriteLine("No estimated orientation information.
 Check Exif data.");
                    return;
            }

            // Display face location information.
            Console.WriteLine($"Left: {left}");
            Console.WriteLine($"Top: {top}");
```

```
            Console.WriteLine($"Face Width: {imageWidth * box.Width}");
            Console.WriteLine($"Face Height: {imageHeight * box.Height}");
        }
    }
```

- Einzelheiten zur API finden Sie DetectFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie Gesichter in einem Bild

Der folgende `detect-faces`-Befehl erkennt Gesichter im angegebenen Bild, das in einem Amazon-S3-Bucket gespeichert ist.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Ausgabe:

```
{
    "FaceDetails": [
        {
            "Confidence": 100.0,
            "Eyeglasses": {
                "Confidence": 98.91107940673828,
                "Value": false
            },
            "Sunglasses": {
                "Confidence": 99.7966537475586,
                "Value": false
            },
            "Gender": {
                "Confidence": 99.56611633300781,
                "Value": "Male"
            },
            "Landmarks": [
                {
```

```
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3514654338359833,
                    "X": 0.6241428852081299,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.35258132219314575,
                    "X": 0.6713621020317078,
                    "Type": "mouthRight"
                },
                {
                    "Y": 0.3140771687030792,
                    "X": 0.6428444981575012,
                    "Type": "nose"
                },
                {
                    "Y": 0.24662546813488007,
                    "X": 0.6001564860343933,
                    "Type": "leftEyeBrowLeft"
                },
                {
                    "Y": 0.24326619505882263,
                    "X": 0.6303644776344299,
                    "Type": "leftEyeBrowRight"
                },
                {
                    "Y": 0.23818562924861908,
                    "X": 0.6146903038024902,
                    "Type": "leftEyeBrowUp"
                },
                {
                    "Y": 0.24373626708984375,
                    "X": 0.6640064716339111,
                    "Type": "rightEyeBrowLeft"
                },
```

```
                {
                    "Y": 0.24877218902111053,
                    "X": 0.7025929093360901,
                    "Type": "rightEyeBrowRight"
                },
                {
                    "Y": 0.23938551545143127,
                    "X": 0.6823262572288513,
                    "Type": "rightEyeBrowUp"
                },
                {
                    "Y": 0.265746533870697,
                    "X": 0.6112898588180542,
                    "Type": "leftEyeLeft"
                },
                {
                    "Y": 0.2676128149032593,
                    "X": 0.6317071914672852,
                    "Type": "leftEyeRight"
                },
                {
                    "Y": 0.262735515832901,
                    "X": 0.6201658248901367,
                    "Type": "leftEyeUp"
                },
                {
                    "Y": 0.27025148272514343,
                    "X": 0.6206279993057251,
                    "Type": "leftEyeDown"
                },
                {
                    "Y": 0.268223375082016,
                    "X": 0.6658390760421753,
                    "Type": "rightEyeLeft"
                },
                {
                    "Y": 0.2672517001628876,
                    "X": 0.687832236289978,
                    "Type": "rightEyeRight"
                },
                {
                    "Y": 0.26383838057518005,
                    "X": 0.6769183874130249,
                    "Type": "rightEyeUp"
```

```
                },
                {
                    "Y": 0.27138751745224,
                    "X": 0.676596462726593,
                    "Type": "rightEyeDown"
                },
                {
                    "Y": 0.32283174991607666,
                    "X": 0.6350004076957703,
                    "Type": "noseLeft"
                },
                {
                    "Y": 0.3219289481639862,
                    "X": 0.6567046642303467,
                    "Type": "noseRight"
                },
                {
                    "Y": 0.3420318365097046,
                    "X": 0.6450609564781189,
                    "Type": "mouthUp"
                },
                {
                    "Y": 0.3664324879646301,
                    "X": 0.6455618143081665,
                    "Type": "mouthDown"
                },
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "leftPupil"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "rightPupil"
                },
                {
                    "Y": 0.26343393325805664,
                    "X": 0.5946047306060791,
                    "Type": "upperJawlineLeft"
                },
                {
                    "Y": 0.3543180525302887,
                    "X": 0.6044883728027344,
```

```
                "Type": "midJawlineLeft"
            },
            {
                "Y": 0.4084877669811249,
                "X": 0.6477024555206299,
                "Type": "chinBottom"
            },
            {
                "Y": 0.3562754988670349,
                "X": 0.707981526851654,
                "Type": "midJawlineRight"
            },
            {
                "Y": 0.26580461859703064,
                "X": 0.7234612107276917,
                "Type": "upperJawlineRight"
            }
        ],
        "Pose": {
            "Yaw": -3.7351467609405518,
            "Roll": -0.10309021919965744,
            "Pitch": 0.8637830018997192
        },
        "Emotions": [
            {
                "Confidence": 8.74203109741211,
                "Type": "SURPRISED"
            },
            {
                "Confidence": 2.501944065093994,
                "Type": "ANGRY"
            },
            {
                "Confidence": 0.7378743290901184,
                "Type": "DISGUSTED"
            },
            {
                "Confidence": 3.5296201705932617,
                "Type": "HAPPY"
            },
            {
                "Confidence": 1.7162904739379883,
                "Type": "SAD"
            },
```

```
                {
                    "Confidence": 9.518536567687988,
                    "Type": "CONFUSED"
                },
                {
                    "Confidence": 0.45474427938461304,
                    "Type": "FEAR"
                },
                {
                    "Confidence": 72.79895782470703,
                    "Type": "CALM"
                }
            ],
            "AgeRange": {
                "High": 48,
                "Low": 32
            },
            "EyesOpen": {
                "Confidence": 98.93987274169922,
                "Value": true
            },
            "BoundingBox": {
                "Width": 0.12368916720151901,
                "Top": 0.16007372736930847,
                "Left": 0.5901257991790771,
                "Height": 0.25140416622161865
            },
            "Smile": {
                "Confidence": 93.4493179321289,
                "Value": false
            },
            "MouthOpen": {
                "Confidence": 90.53053283691406,
                "Value": false
            },
            "Quality": {
                "Sharpness": 95.51618957519531,
                "Brightness": 65.29893493652344
            },
            "Mustache": {
                "Confidence": 89.85221099853516,
                "Value": false
            },
            "Beard": {
```

```
                "Confidence": 86.1991195678711,
                "Value": true
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie DetectFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source
image is stored.
                sourceImage - The name of the source image file in the Amazon S3
bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
the Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source
image is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();
```

```
            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(targetImage)
                .build();

            DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between
 "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- API-Details finden Sie DetectFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- Einzelheiten zur API finden Sie DetectFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect faces in the image with all attributes
    DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
    DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
    INSERT lo_attr_wrapper INTO TABLE lt_attributes.

    oo_result = lo_rek->detectfaces(
      io_image = lo_image
      it_attributes = lt_attributes ).

    DATA(lt_face_details) = oo_result->get_facedetails( ).
    DATA(lv_detected_count) = lines( lt_face_details ).
    DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
    MESSAGE lv_msg8 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DetectFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectLabels`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectLabels` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
```

```
                },
                MaxLabels = 10,
                MinConfidence = 75F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Erkennen Sie Labels in einer Bilddatei, die auf Ihrem Computer gespeichert ist.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect labels within an image
    /// stored locally.
    /// </summary>
    public class DetectLabelsLocalFile
    {
        public static async Task Main()
        {
            string photo = "input.jpg";
```

```
            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = image,
                MaxLabels = 10,
                MinConfidence = 77F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine($"Detected labels for {photo}");
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"{label.Name}: {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
```

```
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erkennen Sie ein Label in einem Bild

Im folgenden `detect-labels`-Beispiel werden Szenen und Objekte in einem Bild, das in einem Amazon-S3-Bucket gespeichert ist, erkannt.

```
aws rekognition detect-labels \
    --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}'
```

Ausgabe:

```
{
    "Labels": [
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
```

```
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Automobile"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Vehicle"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [],
            "Name": "Transportation"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.10616336017847061,
                        "Top": 0.5039216876029968,
                        "Left": 0.0037978808395564556,
                        "Height": 0.18528179824352264
                    },
                    "Confidence": 99.15271759033203
                },
                {
                    "BoundingBox": {
                        "Width": 0.2429988533258438,
                        "Top": 0.5251884460449219,
                        "Left": 0.7309805154800415,
                        "Height": 0.21577216684818268
                    },
```

```
                "Confidence": 99.1286392211914
            },
            {
                "BoundingBox": {
                    "Width": 0.14233611524105072,
                    "Top": 0.5333095788955688,
                    "Left": 0.6494812965393066,
                    "Height": 0.15528248250484467
                },
                "Confidence": 98.48368072509766
            },
            {
                "BoundingBox": {
                    "Width": 0.11086395382881165,
                    "Top": 0.5354844927787781,
                    "Left": 0.10355594009160995,
                    "Height": 0.10271988064050674
                },
                "Confidence": 96.45606231689453
            },
            {
                "BoundingBox": {
                    "Width": 0.06254628300666809,
                    "Top": 0.5573825240135193,
                    "Left": 0.46083059906959534,
                    "Height": 0.053911514580249786
                },
                "Confidence": 93.65448760986328
            },
            {
                "BoundingBox": {
                    "Width": 0.10105438530445099,
                    "Top": 0.534368634223938,
                    "Left": 0.5743985772132874,
                    "Height": 0.12226245552301407
                },
                "Confidence": 93.06217193603516
            },
            {
                "BoundingBox": {
                    "Width": 0.056389667093753815,
                    "Top": 0.5235804319381714,
                    "Left": 0.9427769780158997,
                    "Height": 0.17163699865341187
```

```
                    },
                    "Confidence": 92.6864013671875
                },
                {
                    "BoundingBox": {
                        "Width": 0.06003860384225845,
                        "Top": 0.5441341400146484,
                        "Left": 0.22409997880458832,
                        "Height": 0.06737709045410156
                    },
                    "Confidence": 90.4227066040039
                },
                {
                    "BoundingBox": {
                        "Width": 0.02848697081208229,
                        "Top": 0.5107086896896362,
                        "Left": 0,
                        "Height": 0.19150497019290924
                    },
                    "Confidence": 86.65286254882812
                },
                {
                    "BoundingBox": {
                        "Width": 0.04067881405353546,
                        "Top": 0.5566273927688599,
                        "Left": 0.316415935754776,
                        "Height": 0.03428703173995018
                    },
                    "Confidence": 85.36471557617188
                },
                {
                    "BoundingBox": {
                        "Width": 0.043411049991846085,
                        "Top": 0.5394920110702515,
                        "Left": 0.18293385207653046,
                        "Height": 0.0893595889210701
                    },
                    "Confidence": 82.21705627441406
                },
                {
                    "BoundingBox": {
                        "Width": 0.031183116137981415,
                        "Top": 0.5579366683959961,
                        "Left": 0.2853088080883026,
```

```
                        "Height": 0.03989990055561066
                    },
                    "Confidence": 81.0157470703125
                },
                {
                    "BoundingBox": {
                        "Width": 0.031113790348172188,
                        "Top": 0.5504819750785828,
                        "Left": 0.2580395042896271,
                        "Height": 0.056484755128622055
                    },
                    "Confidence": 56.13441467285156
                },
                {
                    "BoundingBox": {
                        "Width": 0.08586374670267105,
                        "Top": 0.5438792705535889,
                        "Left": 0.5128012895584106,
                        "Height": 0.08550430089235306
                    },
                    "Confidence": 52.37760925292969
                }
            ],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Car"
        },
        {
            "Instances": [],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Human"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
```

```
                        "Width": 0.19360728561878204,
                        "Top": 0.35072067379951477,
                        "Left": 0.43734854459762573,
                        "Height": 0.2742200493812561
                    },
                    "Confidence": 98.9914321899414
                },
                {
                    "BoundingBox": {
                        "Width": 0.03801717236638069,
                        "Top": 0.5010883808135986,
                        "Left": 0.9155802130699158,
                        "Height": 0.06597328186035156
                    },
                    "Confidence": 85.02790832519531
                }
            ],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Person"
        },
        {
            "Instances": [],
            "Confidence": 93.24951934814453,
            "Parents": [],
            "Name": "Machine"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.03561960905790329,
                        "Top": 0.6468243598937988,
                        "Left": 0.7850857377052307,
                        "Height": 0.08878646790981293
                    },
                    "Confidence": 93.24951934814453
                },
                {
                    "BoundingBox": {
                        "Width": 0.02217046171426773,
                        "Top": 0.6149078607559204,
                        "Left": 0.04757237061858177,
                        "Height": 0.07136218994855881
```

```
            },
            "Confidence": 91.5025863647461
        },
        {
            "BoundingBox": {
                "Width": 0.016197510063648224,
                "Top": 0.6274210214614868,
                "Left": 0.6472989320755005,
                "Height": 0.04955997318029404
            },
            "Confidence": 85.14686584472656
        },
        {
            "BoundingBox": {
                "Width": 0.020207518711686134,
                "Top": 0.6348286867141724,
                "Left": 0.7295016646385193,
                "Height": 0.07059963047504425
            },
            "Confidence": 83.34547424316406
        },
        {
            "BoundingBox": {
                "Width": 0.020280985161662102,
                "Top": 0.6171894669532776,
                "Left": 0.08744934946298599,
                "Height": 0.05297485366463661
            },
            "Confidence": 79.9981460571289
        },
        {
            "BoundingBox": {
                "Width": 0.018318990245461464,
                "Top": 0.623889148235321,
                "Left": 0.6836880445480347,
                "Height": 0.06730121374130249
            },
            "Confidence": 78.87144470214844
        },
        {
            "BoundingBox": {
                "Width": 0.021310249343514442,
                "Top": 0.6167286038398743,
                "Left": 0.004064912907779217,
```

```
                        "Height": 0.08317798376083374
                    },
                    "Confidence": 75.89361572265625
                },
                {
                    "BoundingBox": {
                        "Width": 0.03604431077837944,
                        "Top": 0.7030032277107239,
                        "Left": 0.9254803657531738,
                        "Height": 0.04569442570209503
                    },
                    "Confidence": 64.402587890625
                },
                {
                    "BoundingBox": {
                        "Width": 0.009834849275648594,
                        "Top": 0.5821820497512817,
                        "Left": 0.28094568848609924,
                        "Height": 0.01964157074689865
                    },
                    "Confidence": 62.79907989501953
                },
                {
                    "BoundingBox": {
                        "Width": 0.01475677452981472,
                        "Top": 0.6137543320655823,
                        "Left": 0.5950819253921509,
                        "Height": 0.039063986390829086
                    },
                    "Confidence": 59.40483474731445
                }
            ],
            "Confidence": 93.24951934814453,
            "Parents": [
                {
                    "Name": "Machine"
                }
            ],
            "Name": "Wheel"
        },
        {
            "Instances": [],
            "Confidence": 92.61514282226562,
            "Parents": [],
```

```
            "Name": "Road"
        },
        {
            "Instances": [],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Sport"
        },
        {
            "Instances": [],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Sports"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.12326609343290329,
                        "Top": 0.6332163214683533,
                        "Left": 0.44815489649772644,
                        "Height": 0.058117982000112534
                    },
                    "Confidence": 92.37877655029297
                }
            ],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                },
                {
                    "Name": "Sport"
                }
            ],
            "Name": "Skateboard"
```

```
        },
        {
            "Instances": [],
            "Confidence": 90.62931060791016,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Pedestrian"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Asphalt"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Tarmac"
        },
        {
            "Instances": [],
            "Confidence": 88.23201751708984,
            "Parents": [],
            "Name": "Path"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Urban"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
```

```
                }
            ],
            "Name": "Town"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Building"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "City"
        },
        {
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking Lot"
        },
        {
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
```

```
                "Name": "Car"
            },
            {
                "Name": "Vehicle"
            },
            {
                "Name": "Transportation"
            }
        ],
        "Name": "Parking"
    },
    {
        "Instances": [],
        "Confidence": 74.37590026855469,
        "Parents": [
            {
                "Name": "Building"
            },
            {
                "Name": "Urban"
            },
            {
                "Name": "City"
            }
        ],
        "Name": "Downtown"
    },
    {
        "Instances": [],
        "Confidence": 69.84622955322266,
        "Parents": [
            {
                "Name": "Road"
            }
        ],
        "Name": "Intersection"
    },
    {
        "Instances": [],
        "Confidence": 57.68518829345703,
        "Parents": [
            {
                "Name": "Sports Car"
            },
```

```
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Coupe"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Sports Car"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Sidewalk"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
```

```
                }
            ],
            "Name": "Pavement"
        },
        {
            "Instances": [],
            "Confidence": 55.58770751953125,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Neighborhood"
        }
    ],
    "LabelModelVersion": "2.0"
}
```

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie DetectLabelsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example, pic1.png).
\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
 Amazon Rekognition service.
```

```
     *
     * @param rekClient     the Amazon Rekognition client used to make the
 detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- API-Details finden Sie DetectLabelsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Einzelheiten zur API finden Sie DetectLabelsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect labels in the image
        oo_result = lo_rek->detectlabels(
          io_image = lo_image
          iv_maxlabels = iv_max_labels ).

        DATA(lt_labels) = oo_result->get_labels( ).
        DATA(lv_label_count) = lines( lt_labels ).
        DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
        MESSAGE lv_msg9 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetectLabelsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectModerationLabels`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectModerationLabels` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
```

```
                        Bucket = bucket,
                    },
                },
                MinConfidence = 60F,
            };

            try
            {
                var detectModerationLabelsResponse = await
 rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
 detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie unsicherer Inhalte in einem Bild

Der folgende `detect-moderation-labels`-Befehl erkennt unsichere Inhalte im angegebenen Bild, das in einem Amazon-S3-Bucket gespeichert ist.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Ausgabe:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erkennen unsicherer Bilder im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie DetectModerationLabelsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
 detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
 stored
```

```
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
detection process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- API-Details finden Sie DetectModerationLabelsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
 content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
```

```
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect moderation labels
```

```
      oo_result = lo_rek->detectmoderationlabels(
        io_image = lo_image ).

      DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
      DATA(lv_mod_count) = lines( lt_moderation_labels ).
      DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
      MESSAGE lv_msg10 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie DetectModerationLabelsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectText`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectText` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
```

```
    /// </summary>
    public class DetectText
    {
        public static async Task Main()
        {
            string photo = "Dad_photographer.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectTextRequest = new DetectTextRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

- Einzelheiten zur API finden Sie DetectText in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie Text in einem Bild

Der folgende `detect-text`-Befehl erkennt Text im angegebenen Bild.

```
aws rekognition detect-text \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Ausgabe:

```
{
    "TextDetections": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.24624845385551453,
                    "Top": 0.28288066387176514,
                    "Left": 0.391388863325119,
                    "Height": 0.022687450051307678
                },
                "Polygon": [
                    {
                        "Y": 0.28288066387176514,
                        "X": 0.391388863325119
                    },
                    {
                        "Y": 0.2826388478279114,
                        "X": 0.6376373171806335
                    },
                    {
                        "Y": 0.30532628297805786,
                        "X": 0.637677013874054
                    },
                    {
                        "Y": 0.305568128824234,
```

```
                        "X": 0.39142853021621704
                    }
                ]
            },
            "Confidence": 94.35709381103516,
            "DetectedText": "ESTD 1882",
            "Type": "LINE",
            "Id": 0
        },
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933889865875244,
                    "Top": 0.32603850960731506,
                    "Left": 0.34534579515457153,
                    "Height": 0.07126858830451965
                },
                "Polygon": [
                    {
                        "Y": 0.32603850960731506,
                        "X": 0.34534579515457153
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.3976001739501953,
                        "X": 0.684575080871582
                    },
                    {
                        "Y": 0.3973070979118347,
                        "X": 0.345236212015152
                    }
                ]
            },
            "Confidence": 99.95779418945312,
            "DetectedText": "BRAINS",
            "Type": "LINE",
            "Id": 1
        },
        {
            "Confidence": 97.22098541259766,
            "Geometry": {
```

```
                "BoundingBox": {
                    "Width": 0.061079490929841995,
                    "Top": 0.2843210697174072,
                    "Left": 0.391391396522522,
                    "Height": 0.021029088646173477
                },
                "Polygon": [
                    {
                        "Y": 0.2843210697174072,
                        "X": 0.391391396522522
                    },
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.4524524509906769
                    },
                    {
                        "Y": 0.3038259446620941,
                        "X": 0.4534534513950348
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.3923923969268799
                    }
                ]
            },
            "DetectedText": "ESTD",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 2
        },
        {
            "Confidence": 91.49320983886719,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.07007007300853729,
                    "Top": 0.2828207015991211,
                    "Left": 0.5675675868988037,
                    "Height": 0.02250562608242035
                },
                "Polygon": [
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.5675675868988037
                    },
```

```
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.5675675868988037
                    }
                ]
            },
            "DetectedText": "1882",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 3
        },
        {
            "Confidence": 99.95779418945312,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933934569358826,
                    "Top": 0.32633158564567566,
                    "Left": 0.3453453481197357,
                    "Height": 0.07127484679222107
                },
                "Polygon": [
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.3453453481197357
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.39759939908981323,
                        "X": 0.6836836934089661
                    },
                    {
                        "Y": 0.39684921503067017,
                        "X": 0.3453453481197357
```

```
                    }
                ]
            },
            "DetectedText": "BRAINS",
            "ParentId": 1,
            "Type": "WORD",
            "Id": 4
        }
    ]
}
```

- Einzelheiten zur API finden Sie DetectText in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
 Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
 Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();
```

```
            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectTextin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- API-Details finden Sie DetectText in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Einzelheiten zur API finden Sie DetectTextin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect text in the image
    oo_result = lo_rek->detecttext(
      io_image = lo_image ).

    DATA(lt_text_detections) = oo_result->get_textdetections( ).
    DATA(lv_text_count) = lines( lt_text_detections ).
    DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
    MESSAGE lv_msg11 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DetectTextin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetCelebrityInfo`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetCelebrityInfo` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();

        var celebrityInfoRequest = new GetCelebrityInfoRequest
        {
            Id = celebId,
        };

        Console.WriteLine($"Getting information for celebrity: {celebId}");

        var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

        // Display celebrity information.
        Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
        Console.WriteLine("Further information (if available):");
        celebrityInfoResponse.Urls.ForEach(url =>
        {
```

```
                Console.WriteLine(url);
            });
        }
    }
```

- Einzelheiten zur API finden Sie GetCelebrityInfoin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie Informationen über einen Prominenten ab

Der folgende `get-celebrity-info`-Befehl zeigt Informationen über den angegebenen Prominenten an. Der Parameter `id` stammt aus einem früheren Aufruf von `recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Ausgabe:

```
{
    "Name": "Celeb A",
    "Urls": [
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Weitere Informationen finden Sie unter Abrufen von Informationen über einen Prominenten im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie GetCelebrityInfoin der AWS CLI Befehlsreferenz.

Verwenden Sie **`IndexFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `IndexFaces` verwendet wird.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
public class AddFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Image
        {
            S3Object = new S3Object
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var indexFacesRequest = new IndexFacesRequest
        {
```

```
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
 rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
 {faceRecord.Face.FaceId}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie IndexFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie Gesichter zu einer Sammlung hinzu

Mit dem folgenden `index-faces`-Befehl werden die in einem Bild gefundenen Gesichter der angegebenen Sammlung hinzugefügt.

```
aws rekognition index-faces \
    --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
    --collection-id MyCollection \
    --max-faces 1 \
    --quality-filter "AUTO" \
    --detection-attributes "ALL" \
    --external-image-id "MyPicture.jpg"
```

Ausgabe:

```
{
```

```
    "FaceRecords": [
        {
            "FaceDetail": {
                "Confidence": 99.993408203125,
                "Eyeglasses": {
                    "Confidence": 99.11750030517578,
                    "Value": false
                },
                "Sunglasses": {
                    "Confidence": 99.98249053955078,
                    "Value": false
                },
                "Gender": {
                    "Confidence": 99.92769622802734,
                    "Value": "Male"
                },
                "Landmarks": [
                    {
                        "Y": 0.26750367879867554,
                        "X": 0.6202793717384338,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.26642778515815735,
                        "X": 0.6787431836128235,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.31361380219459534,
                        "X": 0.6421601176261902,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.3495299220085144,
                        "X": 0.6216195225715637,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35194727778434753,
                        "X": 0.669899046421051,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.26844894886016846,
```

```
                "X": 0.6210268139839172,
                "Type": "leftPupil"
            },
            {
                "Y": 0.26707562804222107,
                "X": 0.6817160844802856,
                "Type": "rightPupil"
            },
            {
                "Y": 0.24834522604942322,
                "X": 0.6018546223640442,
                "Type": "leftEyeBrowLeft"
            },
            {
                "Y": 0.24397172033786774,
                "X": 0.6172008514404297,
                "Type": "leftEyeBrowUp"
            },
            {
                "Y": 0.24677404761314392,
                "X": 0.6339119076728821,
                "Type": "leftEyeBrowRight"
            },
            {
                "Y": 0.24582654237747192,
                "X": 0.6619398593902588,
                "Type": "rightEyeBrowLeft"
            },
            {
                "Y": 0.23973053693771362,
                "X": 0.6804757118225098,
                "Type": "rightEyeBrowUp"
            },
            {
                "Y": 0.24441994726657867,
                "X": 0.6978968977928162,
                "Type": "rightEyeBrowRight"
            },
            {
                "Y": 0.2695908546447754,
                "X": 0.6085202693939209,
                "Type": "leftEyeLeft"
            },
            {
```

```
                "Y": 0.26716896891593933,
                "X": 0.6315826177597046,
                "Type": "leftEyeRight"
            },
            {
                "Y": 0.26289820671081543,
                "X": 0.6202316880226135,
                "Type": "leftEyeUp"
            },
            {
                "Y": 0.27123287320137024,
                "X": 0.6205548048019409,
                "Type": "leftEyeDown"
            },
            {
                "Y": 0.2668408751487732,
                "X": 0.6663622260093689,
                "Type": "rightEyeLeft"
            },
            {
                "Y": 0.26741549372673035,
                "X": 0.6910083889961243,
                "Type": "rightEyeRight"
            },
            {
                "Y": 0.2614026665687561,
                "X": 0.6785826086997986,
                "Type": "rightEyeUp"
            },
            {
                "Y": 0.27075251936912537,
                "X": 0.6789616942405701,
                "Type": "rightEyeDown"
            },
            {
                "Y": 0.3211299479007721,
                "X": 0.6324167847633362,
                "Type": "noseLeft"
            },
            {
                "Y": 0.32276326417922974,
                "X": 0.6558475494384766,
                "Type": "noseRight"
            },
```

```
                    {
                        "Y": 0.34385165572166443,
                        "X": 0.6444970965385437,
                        "Type": "mouthUp"
                    },
                    {
                        "Y": 0.3671635091304779,
                        "X": 0.6459195017814636,
                        "Type": "mouthDown"
                    }
                ],
                "Pose": {
                    "Yaw": -9.54541015625,
                    "Roll": -0.5709401965141296,
                    "Pitch": 0.6045494675636292
                },
                "Emotions": [
                    {
                        "Confidence": 39.90074157714844,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 23.38753890991211,
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 5.840933322906494,
                        "Type": "CONFUSED"
                    }
                ],
                "AgeRange": {
                    "High": 63,
                    "Low": 45
                },
                "EyesOpen": {
                    "Confidence": 99.80887603759766,
                    "Value": true
                },
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
```

```
                "Smile": {
                    "Confidence": 99.69740295410156,
                    "Value": false
                },
                "MouthOpen": {
                    "Confidence": 99.97393798828125,
                    "Value": false
                },
                "Quality": {
                    "Sharpness": 95.54405975341797,
                    "Brightness": 63.867706298828125
                },
                "Mustache": {
                    "Confidence": 97.05007934570312,
                    "Value": false
                },
                "Beard": {
                    "Confidence": 87.34505462646484,
                    "Value": false
                }
            },
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.993408203125,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            }
        }
    ],
    "UnindexedFaces": [],
    "FaceModelVersion": "3.0",
    "OrientationCorrection": "ROTATE_0"
}
```

Weitere Informationen finden Sie unter Gesichter zu einer Sammlung hinzufügen im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie IndexFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
```

```
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with
the Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
                    .maxFaces(1)
                    .qualityFilter(QualityFilter.AUTO)
                    .detectionAttributes(Attribute.DEFAULT)
                    .build();

            IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
            System.out.println("Results for the image");
            System.out.println("\n Faces indexed:");
```

```
            List<FaceRecord> faceRecords = facesResponse.faceRecords();
            for (FaceRecord faceRecord : faceRecords) {
                System.out.println("  Face ID: " + faceRecord.face().faceId());
                System.out.println("  Location:" +
 faceRecord.faceDetail().boundingBox().toString());
            }

            List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
            System.out.println("Faces not indexed:");
            for (UnindexedFace unindexedFace : unindexedFaces) {
                System.out.println("  Location:" +
 unindexedFace.faceDetail().boundingBox().toString());
                System.out.println("  Reasons:");
                for (Reason reason : unindexedFace.reasons()) {
                    System.out.println("Reason:  " + reason);
                }
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie IndexFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
```

```
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- API-Details finden Sie IndexFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Einzelheiten zur API finden Sie IndexFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie IndexFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListCollections`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListCollections` verwendet wird.

Weitere Informationen finden Sie unter Sammlungen auflisten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
```

```
            };

            var listCollectionsResponse = new ListCollectionsResponse();

            do
            {
                if (listCollectionsResponse is not null)
                {
                    listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
                }

                listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- Einzelheiten zur API finden Sie ListCollectionsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die verfügbaren Sammlungen auf

Der folgende `list-collections` Befehl listet die verfügbaren Sammlungen im AWS Konto auf.

```
aws rekognition list-collections
```

Ausgabe:

```
{
```

```
    "FaceModelVersions": [
        "2.0",
        "3.0",
        "3.0",
        "3.0",
        "4.0",
        "1.0",
        "3.0",
        "4.0",
        "4.0",
        "4.0"
    ],
    "CollectionIds": [
        "MyCollection1",
        "MyCollection2",
        "MyCollection3",
        "MyCollection4",
        "MyCollection5",
        "MyCollection6",
        "MyCollection7",
        "MyCollection8",
        "MyCollection9",
        "MyCollection10"
    ]
}
```

Weitere Informationen finden Sie unter Auflisten von Sammlungen im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie ListCollectionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie ListCollectionsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- API-Details finden Sie ListCollectionsin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
 self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
```

```
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Einzelheiten zur API finden Sie ListCollectionsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListCollectionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListFaces` verwendet wird.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
            listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
```

```
                listFacesResponse.Faces.ForEach(face =>
                {
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Einzelheiten zur API finden Sie ListFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie Gesichter in einer Sammlung auf

Der folgende `list-faces`-Befehl listet die Gesichter in der angegebenen Sammlung auf.

```
aws rekognition list-faces \
    --collection-id MyCollection
```

Ausgabe:

```
{
    "FaceModelVersion": "3.0",
    "Faces": [
        {
            "BoundingBox": {
                "Width": 0.5216310024261475,
                "Top": 0.3256250023841858,
                "Left": 0.13394300639629364,
                "Height": 0.3918749988079071
            },
            "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",
            "ExternalImageId": "image1.jpg",
            "Confidence": 100.0,
            "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"
        },
```

```
        {
            "BoundingBox": {
                "Width": 0.5074880123138428,
                "Top": 0.3774999976158142,
                "Left": 0.18302799761295319,
                "Height": 0.3812499940395355
            },
            "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
            "ExternalImageId": "image2.jpg",
            "Confidence": 99.99930572509766,
            "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
        },
        {
            "BoundingBox": {
                "Width": 0.5574039816856384,
                "Top": 0.37187498807907104,
                "Left": 0.14559100568294525,
                "Height": 0.4181250035762787
            },
            "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
            "ExternalImageId": "image3.jpg",
            "Confidence": 99.99960327148438,
            "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
            "ExternalImageId": "image4.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
        },
        {
            "BoundingBox": {
                "Width": 0.5307819843292236,
                "Top": 0.2862499952316284,
                "Left": 0.1564060002565384,
                "Height": 0.3987500071525574
            },
            "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
```

```
            "ExternalImageId": "image5.jpg",
            "Confidence": 99.99970245361328,
            "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
        },
        {
            "BoundingBox": {
                "Width": 0.5773710012435913,
                "Top": 0.34437501430511475,
                "Left": 0.12396000325679779,
                "Height": 0.4337500035762787
            },
            "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
            "ExternalImageId": "image6.jpg",
            "Confidence": 100.0,
            "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
        },
        {
            "BoundingBox": {
                "Width": 0.5349419713020325,
                "Top": 0.29124999046325684,
                "Left": 0.16389399766921997,
                "Height": 0.40187498927116394
            },
            "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
            "ExternalImageId": "image7.jpg",
            "Confidence": 99.99979400634766,
            "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
        },
        {
            "BoundingBox": {
                "Width": 0.41499999165534973,
                "Top": 0.09187500178813934,
                "Left": 0.28083300590515137,
                "Height": 0.3112500011920929
            },
            "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
            "ExternalImageId": "image8.jpg",
            "Confidence": 99.99769592285156,
            "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
        },
        {
            "BoundingBox": {
                "Width": 0.48166701197624207,
                "Top": 0.20999999344348907,
```

```
                "Left": 0.21250000596046448,
                "Height": 0.36125001311302185
            },
            "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
            "ExternalImageId": "image9.jpg",
            "Confidence": 99.99949645996094,
            "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
            "ExternalImageId": "image10.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
        }
    ]
}
```

Weitere Informationen finden Sie unter Auflisten von Gesichtern in einer Sammlung im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie ListFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
```

```
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListFacesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
```

```
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- API-Details finden Sie ListFacesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
```

```
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Einzelheiten zur API finden Sie ListFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RecognizeCelebrities`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RecognizeCelebrities` verwendet wird.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
        {
            Console.WriteLine($"Failed to load file {photo}");
            return;
```

```
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erkennen Sie Prominente in einem Bild

Der folgende `recognize-celebrities`-Befehl erkennt Prominente in dem angegebenen Bild, das in einem Amazon-S3-Bucket gespeichert ist:

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Ausgabe:

```
{
    "UnrecognizedFaces": [
        {
            "BoundingBox": {
                "Width": 0.14416666328907013,
                "Top": 0.07777778059244156,
                "Left": 0.625,
                "Height": 0.2746031880378723
            },
            "Confidence": 99.9990234375,
            "Pose": {
                "Yaw": 10.80408763885498,
                "Roll": -12.761146545410156,
                "Pitch": 10.96889877319336
            },
            "Quality": {
                "Sharpness": 94.1185531616211,
                "Brightness": 79.18367004394531
            },
            "Landmarks": [
                {
                    "Y": 0.18220913410186768,
                    "X": 0.6702951788902283,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.16337193548679352,
                    "X": 0.7188183665275574,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.20739148557186127,
                    "X": 0.7055801749229431,
                    "Type": "nose"
                },
```

```
                {
                    "Y": 0.2889308035373688,
                    "X": 0.687512218952179,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.2706988751888275,
                    "X": 0.7250053286552429,
                    "Type": "mouthRight"
                }
            ]
        }
    ],
    "CelebrityFaces": [
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.14000000059604645,
                    "Top": 0.1190476194024086,
                    "Left": 0.82833331823349,
                    "Height": 0.2666666805744171
                },
                "Confidence": 99.99359130859375,
                "Pose": {
                    "Yaw": -10.509642601013184,
                    "Roll": -14.51749324798584,
                    "Pitch": 13.799399375915527
                },
                "Quality": {
                    "Sharpness": 78.74752044677734,
                    "Brightness": 42.201324462890625
                },
                "Landmarks": [
                    {
                        "Y": 0.2290833294391632,
                        "X": 0.8709492087364197,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.20639978349208832,
                        "X": 0.9153988361358643,
                        "Type": "eyeRight"
                    },
```

```
                    {
                        "Y": 0.25417643785476685,
                        "X": 0.8907724022865295,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.32729196548461914,
                        "X": 0.8876466155052185,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.3115464746952057,
                        "X": 0.9238573312759399,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb A",
            "Urls": [
                "www.imdb.com/name/aaaaaaaaa"
            ],
            "Id": "1111111"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.13333334028720856,
                    "Top": 0.2492063492536544 8,
                    "Left": 0.4449999928474426,
                    "Height": 0.2539682686328888
                },
                "Confidence": 99.99979400634766,
                "Pose": {
                    "Yaw": 6.557040691375732,
                    "Roll": -7.316643714904785,
                    "Pitch": 9.272967338562012
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 78.83267974853516
                },
                "Landmarks": [
                    {
```

```
                        "Y": 0.3625510632991791,
                        "X": 0.48898839950561523,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.35366007685661316,
                        "X": 0.5313721299171448,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3894785940647125,
                        "X": 0.5173314809799194,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.44889405369758606,
                        "X": 0.5020005702972412,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.4408611059188843,
                        "X": 0.5351271629333496,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb B",
            "Urls": [
                "www.imdb.com/name/bbbbbbbbb"
            ],
            "Id": "2222222"
        },
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.12416666746139526,
                    "Top": 0.2968254089355469,
                    "Left": 0.2150000035762787,
                    "Height": 0.23650793731212616
                },
                "Confidence": 99.99958801269531,
                "Pose": {
                    "Yaw": 7.801797866821289,
```

```
                    "Roll": -8.326810836791992,
                    "Pitch": 7.844768047332764
                },
                "Quality": {
                    "Sharpness": 86.93206024169922,
                    "Brightness": 79.81291198730469
                },
                "Landmarks": [
                    {
                        "Y": 0.4027804136276245,
                        "X": 0.2575301229953766,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.3934555947780609,
                        "X": 0.2956969439983368,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.4309830069541931,
                        "X": 0.2837020754814148,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.48186683654785156,
                        "X": 0.26812544465065,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.47338807582855225,
                        "X": 0.29905644059181213,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb C",
            "Urls": [
                "www.imdb.com/name/ccccccccc"
            ],
            "Id": "3333333"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
```

```
                "BoundingBox": {
                    "Width": 0.11916666477918625,
                    "Top": 0.3698412775993347,
                    "Left": 0.008333333767950535,
                    "Height": 0.22698412835597992
                },
                "Confidence": 99.99999237060547,
                "Pose": {
                    "Yaw": 16.38478660583496,
                    "Roll": -1.0260354280471802,
                    "Pitch": 5.975185394287109
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 61.408443450927734
                },
                "Landmarks": [
                    {
                        "Y": 0.4632347822189331,
                        "X": 0.049406956881284714,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.46388113498687744,
                        "X": 0.08722897619009018,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.5020678639411926,
                        "X": 0.0758260041475296,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.544157862663269,
                        "X": 0.054029736667871475,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.5463630557060242,
                        "X": 0.08464983850717545,
                        "Type": "mouthRight"
                    }
                ]
            },
```

```
            "Name": "Celeb D",
            "Urls": [
                "www.imdb.com/name/ddddddddd"
            ],
            "Id": "4444444"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }

        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
```

```
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            RecognizeCelebritiesRequest request =
 RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
 rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.
\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- API-Details finden Sie RecognizeCelebritiesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
 self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
```

```
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).
```

```
      " Create image object
      DATA(lo_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_s3object ).

      " Recognize celebrities
      oo_result = lo_rek->recognizecelebrities(
        io_image = lo_image ).

      DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
      DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
      DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
      MESSAGE lv_msg12 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie RecognizeCelebritiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SearchFaces`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchFaces` verwendet wird.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
```

```
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to find faces in an image that
    /// match the face Id provided in the method request.
    /// </summary>
    public class SearchFacesMatchingId
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

            var rekognitionClient = new AmazonRekognitionClient();

            // Search collection for faces matching the face id.
            var searchFacesRequest = new SearchFacesRequest
            {
                CollectionId = collectionId,
                FaceId = faceId,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

            Console.WriteLine("Face matching faceId " + faceId);

            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie SearchFacesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So suchen Sie nach Gesichtern in einer Sammlung, die einer Gesichts-ID entsprechen.

Mit dem folgenden `search-faces`-Befehl wird in einer Sammlung nach Gesichtern gesucht, die der angegebenen Gesichts-ID entsprechen.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Ausgabe:

```
{
    "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "FaceModelVersion": "3.0",
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
                "ExternalImageId": "image1.jpg",
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 99.30997467041016
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
```

```
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 96.73158264160156
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
```

```
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 96.48291015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 96.43287658691406
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
                },
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 95.25305938720703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
```

```
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 95.22837829589844
        }
    ]
}
```

Weitere Informationen finden Sie unter Suchen nach einem Gesicht mit der Gesichts-ID im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie SearchFacesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
 File(sourceImage));
```

```
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SearchFacesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
```

```
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- Einzelheiten zur API finden Sie SearchFacesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_rek->searchfaces(
         iv_collectionid = iv_collection_id
         iv_faceid = iv_face_id
         iv_facematchthreshold = iv_threshold
         iv_maxfaces = iv_max_faces ).

       DATA(lt_face_matches) = oo_result->get_facematches( ).
       DATA(lv_match_count2) = lines( lt_face_matches ).
       DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
       MESSAGE lv_msg5 TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection or face not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie SearchFacesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SearchFacesByImage`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SearchFacesByImage` verwendet wird.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        var image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var searchFacesByImageRequest = new SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
```

```
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
 rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
 photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So suchen Sie nach Gesichtern in einer Sammlung, die dem größten Gesicht in einem Bild entsprechen.

Mit dem folgenden `search-faces-by-image`-Befehl wird in einer Sammlung nach Gesichtern gesucht, die dem größten Gesicht im angegebenen Bild entsprechen:

```
aws rekognition search-faces-by-image \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePerson.jpg"}}' \
    --collection-id MyFaceImageCollection

{
    "SearchedFaceBoundingBox": {
        "Width": 0.18562500178813934,
        "Top": 0.1618015021085739,
```

```
        "Left": 0.5575000047683716,
        "Height": 0.24770642817020416
    },
    "SearchedFaceConfidence": 99.993408203125,
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.97913360595703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.97913360595703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.41499999165534973,
                    "Top": 0.09187500178813934,
                    "Left": 0.28083300590515137,
                    "Height": 0.3112500011920929
                },
                "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
```

```
                "ExternalImageId": "image2.jpg",
                "Confidence": 99.99769592285156,
                "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
            },
            "Similarity": 99.18069458007812
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
                "ExternalImageId": "image1.jpg",
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 98.66607666015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 98.24278259277344
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
```

```
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 98.10665893554688
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 98.10526275634766
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
                },
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 97.94659423828125
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
```

```
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 97.93476867675781
        }
    ],

 "FaceModelVersion": "3.0"
}
```

Weitere Informationen finden Sie unter Suchen nach einem Gesicht mithilfe eines Bildes im Entwicklerhandbuch für Amazon Rekognition.

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
```

```
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SearchFacesByImagein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
```

```
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
```

```
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Einzelheiten zur API finden Sie SearchFacesByImagein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Search for matching faces
        oo_result = lo_rek->searchfacesbyimage(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count) = lines( lt_face_matches ).
        DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
        MESSAGE lv_msg4 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SearchFacesByImagein der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Rekognition AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Rekognition mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Rekognition oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin nach Gesichtern mithilfe eines AWS SDK
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Gesichter in einem Bild mit einem AWS SDK
- Erkennen Sie Informationen in Videos mithilfe von Amazon Rekognition und dem AWS SDK
- Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Erstellen Sie eine Amazon Rekognition Rekognition-Sammlung und suchen Sie darin nach Gesichtern mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Rekognition-Sammlung.
- Fügen Sie der Sammlung Bilder hinzu und erkennen Sie Gesichter darin.
- Durchsuchen Sie die Sammlung nach Gesichtern, die einem Referenzbild entsprechen.
- Löschen einer Sammlung.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung suchen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, die Amazon-Rekognition-Funktionen wrappen (verpacken).

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
```

```
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
 and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
 the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
```

```
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
 self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
```

```
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering
```

```
    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
```

```
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
```

```
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
```

```
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
```

```
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
```

```
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
```

```
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Verwenden Sie die Wrapper-Klassen, um eine Sammlung von Gesichtern aus einer Reihe von Bildern zu erstellen und dann nach Gesichtern in der Sammlung zu suchen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    rekognition_client = boto3.client("rekognition")
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
            rekognition_client,
            image_name="biking",
        ),
    ]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection("doc-example-collection-demo")
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())
```

```
    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the first face in the
"
        f"list (Face ID: {faces[0].face_id}."
    )
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
    print(f"Found {len(found_faces)} matching faces.")
    for face in found_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the largest face in "
        f"{images[0].image_name}."
    )
    image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
    print(f"The largest face in {images[0].image_name} is:")
    pprint(image_face.to_dict())
    print(f"Found {len(match_faces)} matching faces.")
    for face in match_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    collection.delete_collection()
    print("Thanks for watching!")
    print("-" * 88)
```

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen und Anzeigen von Elementen in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erkennen von Elementen in Bildern mithilfe von Amazon Rekognition.
- Zeigen Sie Bilder an und zeichnen Sie Begrenzungsrahmen um die erkannten Elemente.

Weitere Informationen finden Sie unter Anzeigen von Begrenzungsrahmen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, um Amazon-Rekognition-Funktionen zu umschließen.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
```

```
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)
```

```
    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.
```

```
        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
```

```
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
```

```
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Erstellen Sie Hilfsfunktionen zum Zeichnen von Begrenzungsrahmen und Polygonen.

```
import io
import logging
from PIL import Image, ImageDraw

logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.
```

```
    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Erstellen Sie Klassen, um von Amazon Rekognition zurückgegebene Objekte zu analysieren.

```
class RekognitionFace:
```

```
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
```

```
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.
```

```
        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the
 celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
```

```
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.
```

```
        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.parent_name is not None:
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering
```

```
class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Verwenden Sie die Wrapper-Klassen, um Elemente in Bildern zu erkennen und ihre Begrenzungsrahmen anzuzeigen. Die in diesem Beispiel verwendeten Bilder finden Sie GitHub zusammen mit Anweisungen und weiterem Code unter.

```
def usage_demo():
    print("-" * 88)
```

```
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/
source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image["Bytes"],
        [[face.bounding_box for face in faces]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
    box_sets = []
    colors = ["aqua", "red", "white", "blue", "yellow", "green"]
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst["BoundingBox"] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
```

```
    show_bounding_boxes(
        street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
    )
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client
    )
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image["Bytes"],
        [[celeb.face.bounding_box for celeb in celebs]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls",
rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
```

```
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
"aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Erkennen Sie Gesichter in einem Bild mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

Rust

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

Erkennen Sie Informationen in Videos mithilfe von Amazon Rekognition und dem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Starten Sie Amazon-Rekognition-Aufträge, um Elemente wie Personen, Objekte und Text in Videos zu erkennen.
- Überprüfen Sie den Auftragsstatus, bis die Aufträge abgeschlossen sind.
- Gibt die Liste der von jedem Auftrag erkannten Elemente aus.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen von Informationen aus einem Video, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();
```

```
            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient)
{
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
```

```
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
 null.
                VideoMetadata videoMetaData =
 recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
 videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
 recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Labels in einem Video mithilfe einer Labelerkennung.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
```

```
                    bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                    video - The name of the video (for example, people.mp4).\s
                    queueUrl- The URL of a SQS queue.\s
                    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
```

```
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
```

```
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
```

```
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
```

```
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
```

```
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
```

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();
```

```
            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
```

```
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
```

```
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");
```

```
                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
 detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
```

```
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
               """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startModerationDetection(rekClient, channel, bucket, video);
       getModResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startModerationDetection(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {

       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
                   .name(video)
                   .build();

           Video vidOb = Video.builder()
```

```
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();
```

```
                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
 null.
                VideoMetadata videoMetaData =
 modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
 videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
 modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie technische Signal-Segmente und Einstellungserkennungssegmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
```

```
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();

            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();
```

```
        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
```

```
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + "
milliseconds");
                    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
                    System.out.println();
                }
```

```
            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Text in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
```

```
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
```

```
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " +
detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                    System.out.println("Type: " +
detectedText.textDetection().type());
                    System.out.println("Text: " +
detectedText.textDetection().detectedText());
                    System.out.println();
```

```
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Personen in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoPersonDetection {
```

```
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
```

```
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
```

```
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }
```

```
            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }
```

```
        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
```

```
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }
```

```
        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

.NET

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition zusammen mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie mit AWS SDK für Python (Boto3) dem eine Webanwendung erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingeschriebene Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.
- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.

- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Rust

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Codebeispiele für Resource Groups mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS -Ressourcengruppen mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Benutzerhandbuch für Resource Groups – Weitere Informationen zu Resource Groups.
- API-Referenz für Resource Groups – Details zu allen verfügbaren Aktionen von Resource Groups.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Resource Groups mit AWS SDKs
  - Aktionen für Resource Groups mit AWS SDKs

- Verwendung von CreateGroup mit einer CLI
- Verwendung von DeleteGroup mit einer CLI
- Verwendung von GetGroup mit einer CLI
- Verwendung von GetGroupQuery mit einer CLI
- Verwendung von GetTags mit einer CLI
- Verwendung von ListGroupResources mit einer CLI
- Verwendung von ListGroups mit einer CLI
- Verwendung von SearchResources mit einer CLI
- Verwenden Sie Tag mit einer CLI
- Verwenden Sie Untag mit einer CLI
- Verwendung von UpdateGroup mit einer CLI
- Verwendung von UpdateGroupQuery mit einer CLI

Grundlegende Beispiele für Resource Groups mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS -Ressourcengruppen mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Resource Groups mit AWS SDKs
  - Verwendung von CreateGroup mit einer CLI
  - Verwendung von DeleteGroup mit einer CLI
  - Verwendung von GetGroup mit einer CLI
  - Verwendung von GetGroupQuery mit einer CLI
  - Verwendung von GetTags mit einer CLI
  - Verwendung von ListGroupResources mit einer CLI
  - Verwendung von ListGroups mit einer CLI
  - Verwendung von SearchResources mit einer CLI
  - Verwenden Sie Tag mit einer CLI
  - Verwenden Sie Untag mit einer CLI
  - Verwendung von UpdateGroup mit einer CLI
  - Verwendung von UpdateGroupQuery mit einer CLI

Aktionen für Resource Groups mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Ressourcengruppenaktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS -Ressourcengruppen -API-Referenz.

Beispiele

- Verwendung von CreateGroup mit einer CLI
- Verwendung von DeleteGroup mit einer CLI
- Verwendung von GetGroup mit einer CLI
- Verwendung von GetGroupQuery mit einer CLI
- Verwendung von GetTags mit einer CLI
- Verwendung von ListGroupResources mit einer CLI
- Verwendung von ListGroups mit einer CLI
- Verwendung von SearchResources mit einer CLI
- Verwenden Sie Tag mit einer CLI
- Verwenden Sie Untag mit einer CLI
- Verwendung von UpdateGroup mit einer CLI
- Verwendung von UpdateGroupQuery mit einer CLI

Verwendung von **`CreateGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGroup` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Tag-basierte Ressourcengruppe

Im folgenden Beispiel für `create-group` wird eine Tag-basierte Ressourcengruppe von Amazon-EC2-Instances in der aktuellen Region erstellt. Es basiert auf einer Abfrage nach Ressourcen, die mit dem Schlüssel `Name` und dem Wert `WebServers` gekennzeichnet sind.

Der Name der Gruppe lautet `tbq-WebServer`. Die Abfrage befindet sich in einer separaten JSON-Datei, die an den Befehl übergeben wird.

```
aws resource-groups create-group \
    --name tbq-WebServer \
    --resource-query file://query.json
```

Inhalt von `query.json`:

```
{
    "Type": "TAG_FILTERS_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters\":
[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
}
```

Ausgabe:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
    },
    "ResourceQuery": {
        "Type": "TAG_FILTERS_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters
\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
    }
}
```

Beispiel 2: So erstellen Sie eine CloudFormation stapelbasierte Ressourcengruppe

Im folgenden `create-group` Beispiel wird eine AWS CloudFormation stapelbasierte Ressourcengruppe mit dem Namen erstellt. `sampleCFNstackgroup` Die Abfrage umfasst alle Ressourcen im angegebenen CloudFormation Stack, die von AWS Resource Groups unterstützt werden.

```
aws resource-groups create-group \
    --name cbq-CFNstackgroup \
    --resource-query file://query.json
```

Inhalt von `query.json`:

```
{
    "Type": "CLOUDFORMATION_STACK_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],\"StackIdentifier
\":\"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-
z39z-11z8-97z5-500z212zz6fz\"}"
}
```

Ausgabe:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-
CFNstackgroup",
        "Name": "cbq-CFNstackgroup"
    },
    "ResourceQuery": {
        "Type": "CLOUDFORMATION_STACK_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],
\"StackIdentifier\":\"arn:aws:cloudformation:us-east-2:123456789012:stack/
MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"}'
    }
}
```

Weitere Informationen finden Sie unter Erstellen von Gruppen im Benutzerhandbuch für AWS Resource Groups.

- Einzelheiten zur API finden Sie CreateGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine neue tagbasierte AWS Ressourcengruppe für Resource Groups mit dem Namen TestPowerShellGroup erstellt. Die Gruppe umfasst Amazon EC2-Instances in der aktuellen Region, die mit dem Tag-Schlüssel „Name“ und dem Tag-Wert „test2“ versehen sind. Der Befehl gibt die Abfrage und den Typ der Gruppe sowie die Ergebnisse der Operation zurück.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
```

```
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":
["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery

New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Ausgabe:

```
Query
               Type
-----
               ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine neue tagbasierte AWS Ressourcengruppe für Resource Groups mit dem Namen TestPowerShellGroup erstellt. Die Gruppe umfasst Amazon EC2-Instances in der aktuellen Region, die mit dem Tag-Schlüssel „Name“ und dem Tag-Wert „test2“ versehen sind. Der Befehl gibt die Abfrage und den Typ der Gruppe sowie die Ergebnisse der Operation zurück.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":
["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery
```

```
New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Ausgabe:

```
Query
               Type
-----
               ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter CreateGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteGroup` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie die Beschreibung einer Ressourcengruppe

Im folgenden `delete-group`-Beispiel wird die angegebene Ressourcengruppe aktualisiert.

```
aws resource-groups delete-group \
    --group-name tbq-WebServer
```

Ausgabe:

```
{
```

```
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:1234567890:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
    }
}
```

Weitere Informationen finden Sie unter Löschen von Gruppen im Benutzerhandbuch für AWS Resource Groups.

- Einzelheiten zur API finden Sie DeleteGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die benannte Ressourcengruppe entfernt

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
  Name
----------- --------
  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-
elbv2 non-tag-cfn-elbv2
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die benannte Ressourcengruppe entfernt

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Description GroupArn
  Name
----------- --------
  ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-
elbv2 non-tag-cfn-elbv2
```

- Einzelheiten zur API finden Sie unter DeleteGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetGroup` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer Ressourcengruppe ab

Im folgenden Beispiel für `get-group` werden Details zur angegebenen Ressourcengruppe angezeigt. Um die Abfrage an die Gruppe anzuhängen, verwenden Sie `get-group-query`.

```
aws resource-groups get-group \
    --group-name tbq-WebServer
```

Ausgabe:

```
{
```

```
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer",
        "Description": "A tag-based query resource group of WebServers."
    }
}
```

- Einzelheiten zur API finden Sie GetGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Ressourcengruppe anhand des Gruppennamens abgerufen.

```
Get-RGGroup -GroupName auto-no
```

Ausgabe:

```
Description GroupArn                                                    Name
----------- --------                                                    ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Einzelheiten zur API finden Sie unter GetGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Ressourcengruppe anhand des Gruppennamens abgerufen.

```
Get-RGGroup -GroupName auto-no
```

Ausgabe:

```
Description GroupArn                                                    Name
----------- --------                                                    ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- Einzelheiten zur API finden Sie unter GetGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetGroupQuery`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetGroupQuery` verwendet wird.

CLI

AWS CLI

So hängen Sie die Abfrage an eine Ressourcengruppe an

Im folgenden `get-group-query`-Beispiel wird die Abfrage angezeigt, die der angegebenen Ressourcengruppe zugeordnet ist.

```
aws resource-groups get-group-query \
    --group-name tbq-WebServer
```

Ausgabe:

```
{
    "GroupQuery": {
        "GroupName": "tbq-WebServer",
        "ResourceQuery": {
            "Type": "TAG_FILTERS_1_0",
            "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],
\"TagFilters\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
        }
    }
}
```

- Einzelheiten zur API finden Sie GetGroupQueryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Ressourcenabfrage für die angegebene Ressourcengruppe abgerufen

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Ausgabe:

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- Einzelheiten zur API finden Sie unter GetGroupQuery AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Ressourcenabfrage für die angegebene Ressourcengruppe abgerufen

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Ausgabe:

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"auto","Values":["no"]}]} TAG_FILTERS_1_0
```

- Einzelheiten zur API finden Sie unter GetGroupQuery AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetTags` verwendet wird.

CLI

AWS CLI

So rufen Sie die Tags ab, die einer Ressourcengruppe zugeordnet sind

Im folgenden Beispiel für `get-tags` werden die Tag-Schlüssel- und Wertepaare angezeigt, die der angegebenen Ressourcengruppe (der Gruppe selbst, nicht ihren Mitgliedern) zugeordnet sind.

```
aws resource-groups get-tags \
    --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer
```

Ausgabe:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
    "Tags": {
        "QueryType": "tags",
        "QueryResources": "ec2-instances"
    }
}
```

- Einzelheiten zur API finden Sie GetTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Tags für die angegebene Ressourcengruppe arn aufgelistet

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Ausgabe:

```
Key       Value
---       -----
Instances workboxes
```

- Einzelheiten zur API finden Sie unter GetTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Tags für die angegebene Ressourcengruppe arn aufgelistet

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Ausgabe:

```
Key       Value
---       -----
Instances workboxes
```

- Einzelheiten zur API finden Sie unter GetTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListGroupResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListGroupResources` verwendet wird.

CLI

AWS CLI

So listen Sie alle Ressourcen in einer Ressourcengruppe auf

Beispiel 1: Im folgenden Beispiel für `list-resource-groups` werden alle Ressourcen aufgelistet, die Teil der angegebenen Ressourcengruppe sind.

```
aws resource-groups list-group-resources \
    --group-name tbq-WebServer
```

Ausgabe:

```
{
    "ResourceIdentifiers": [
        {
```

```
            "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/
i-09f77fa38c12345ab",
            "ResourceType": "AWS::EC2::Instance"
        }
    ]
}
```

Beispiel 2: Im folgenden Beispiel werden alle Ressourcen in der Gruppe aufgelistet, die auch den „resource-type“ „AWS::EC2::Instance“ haben:

aws resource-groups list-group-resources --group-name tbq- --filters name=Ressourcentyp, Values=: :EC2: :Instance WebServer AWS

- Einzelheiten zur API finden Sie in ListGroupResourcesder AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Gruppenressourcen auf der Grundlage einer Filterung nach Ressourcentyp aufgelistet

```
Get-RGGroupResourceList -Filter @{Name="resource-
type";Values="AWS::EC2::Instance"} -GroupName auto-yes | Select-Object -
ExpandProperty ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                    ResourceType
-----------                                                    ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012
 AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter ListGroupResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Gruppenressourcen auf der Grundlage einer Filterung nach Ressourcentyp aufgelistet

```
Get-RGGroupResourceList -Filter @{Name="resource-
type";Values="AWS::EC2::Instance"} -GroupName auto-yes | Select-Object -
ExpandProperty ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                        ResourceType
-----------                                                        ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa
 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012
 AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter ListGroupResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListGroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListGroups` verwendet wird.

CLI

AWS CLI

So listen Sie die verfügbaren Ressourcengruppen auf

Im folgenden Beispiel für `list-groups` wird eine Liste aller Ressourcengruppen angezeigt.

```
aws resource-groups list-groups
```

Ausgabe:

```
{
    "GroupIdentifiers": [
        {
            "GroupName": "tbq-WebServer",
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
tbq-WebServer3"
        },
        {
            "GroupName": "cbq-CFNStackQuery",
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
cbq-CFNStackQuery"
        }
    ],
    "Groups": [
        {
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
tbq-WebServer",
            "Name": "tbq-WebServer"
        },
        {
            "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/
cbq-CFNStackQuery",
            "Name": "cbq-CFNStackQuery"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListGroupsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden bereits erstellte Ressourcengruppen aufgelistet.

```
Get-RGGroupList
```

Ausgabe:

```
GroupArn
 GroupName
```

```
--------
 ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no              auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes             auto-
yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600
 build600
```

- Einzelheiten zur API finden Sie unter ListGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden bereits erstellte Ressourcengruppen aufgelistet.

```
Get-RGGroupList
```

Ausgabe:

```
GroupArn
 GroupName
--------
 ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no              auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes             auto-
yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600
 build600
```

- Einzelheiten zur API finden Sie unter ListGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`SearchResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `SearchResources` verwendet wird.

CLI

AWS CLI

So finden Sie Ressourcen, die einer Abfrage entsprechen

Im folgenden `search-resources` Beispiel wird eine Liste aller AWS Ressourcen abgerufen, die der angegebenen Abfrage entsprechen.

```
aws resource-groups search-resources \
    --resource-query file://query.json
```

Inhalt von `query.json`:

```
{
    "Type": "TAG_FILTERS_1_0",
    "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters\":
[{\"Key\":\"Patch Group\", \"Values\":[\"Dev\"]}]}"
}
```

Ausgabe:

```
{
    "ResourceIdentifiers": [
        {
            "ResourceArn": "arn:aws:ec2:us-west-2:123456789012:instance/
i-01a23bc45d67890ef",
            "ResourceType": "AWS::EC2::Instance"
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter SearchResources AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Ressourcentyp ResourceQuery für Instanz mit Tag-Filtern erstellt und Ressourcen gesucht.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
```

```
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                          ResourceType
-----------                                                          ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b
 AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter SearchResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Ressourcentyp ResourceQuery für Instanz mit Tag-Filtern erstellt und Ressourcen gesucht.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                          ResourceType
-----------                                                          ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b
 AWS::EC2::Instance
```

- Einzelheiten zur API finden Sie unter SearchResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie `Tag` mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `Tag` verwendet wird.

CLI

AWS CLI

So hängen Sie ein Tag an eine Ressourcengruppe an

Im folgenden Beispiel für `tag` werden die angegebenen Tag-Schlüssel- und Wertepaare an die angegebene Ressourcengruppe angehängt (die Gruppe selbst, nicht ihre Mitglieder).

```
aws resource-groups tag \
    --tags QueryType=tags,QueryResources=ec2-instances \
    --arn arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer
```

Ausgabe:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:128716708097:group/tbq-WebServer",
    "Tags": {
        "QueryType": "tags",
        "QueryResources": "ec2-instances"
    }
}
```

Weitere Informationen finden Sie unter Verwalten von Tags im Benutzerhandbuch für AWS Resource Groups.

- Weitere API-Informationen finden Sie unter Tag in der AWS CLI -Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Tag-Schlüssel „Instances“ mit dem Wert „workboxes“ zur angegebenen Ressourcengruppe arn hinzugefügt.

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Ausgabe:

```
Arn                                                          Tags
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Weitere API-Informationen finden Sie unter Tag in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Tag-Schlüssel „Instances“ mit dem Wert „workboxes“ zur angegebenen Ressourcengruppe arn hinzugefügt.

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Ausgabe:

```
Arn                                                          Tags
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Weitere API-Informationen finden Sie unter Tag in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Verwenden Sie **`Untag`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `Untag` verwendet wird.

CLI

AWS CLI

So entfernen Sie Tags aus einer Ressourcengruppe

Im folgenden Beispiel für `untags` werden alle Tags mit dem angegebenen Schlüssel aus der Ressourcengruppe selbst entfernt, nicht aus ihren Mitgliedern.

```
aws resource-groups untag \
    --arn arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer \
    --keys QueryType
```

Ausgabe:

```
{
    "Arn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-WebServer",
    "Keys": [
        "QueryType"
    ]
}
```

Weitere Informationen finden Sie unter Verwalten von Tags im Benutzerhandbuch für AWS Resource Groups.

- Weitere API-Informationen finden Sie unter Untag in der AWS CLI -Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das erwähnte Tag aus der Ressourcengruppe entfernt

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target
 "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
```

```
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Weitere API-Informationen finden Sie unter Untag in der AWS -Tools für PowerShell - Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das erwähnte Tag aus der Ressourcengruppe entfernt

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target
 "arn:aws:resource-groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Weitere API-Informationen finden Sie unter Untag in der AWS -Tools für PowerShell - Cmdlet-Referenz (V5).

Verwendung von **`UpdateGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateGroup` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie die Beschreibung einer Ressourcengruppe

Im folgenden Beispiel für `update-group` wird die Beschreibung für die angegebene Ressourcengruppe aktualisiert.

```
aws resource-groups update-group \
```

```
    --group-name tbq-WebServer \
    --description "Resource group for all web server resources."
```

Ausgabe:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
        "Description": "Resource group for all web server resources."
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren von Gruppen im Benutzerhandbuch für AWS Resource Groups.

- Einzelheiten zur API finden Sie UpdateGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Beschreibung der Gruppe aktualisiert

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Ausgabe:

```
Description             GroupArn
     Name
-----------             --------
     ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/
auto-yes auto-yes
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung der Gruppe aktualisiert

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Ausgabe:

```
Description             GroupArn
     Name
-----------             --------
     ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/
auto-yes auto-yes
```

- Einzelheiten zur API finden Sie unter UpdateGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **UpdateGroupQuery** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateGroupQuery` verwendet wird.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie die Abfrage für eine Tag-basierte Ressourcengruppe

Im folgenden Beispiel für `update-group-query` wird die Abfrage aktualisiert, die an die angegebene tagbasierte Ressourcengruppe angehängt ist.

```
aws resource-groups update-group-query \
    --group-name tbq-WebServer \
    --resource-query '{"Type":"TAG_FILTERS_1_0", "Query":"{\"ResourceTypeFilters
\":[\"AWS::EC2::Instance\"],\"TagFilters\":[{\"Key\":\"Name\", \"Values\":
[\"WebServers\"]}]}"}'
```

Ausgabe:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-east-2:123456789012:group/tbq-
WebServer",
        "Name": "tbq-WebServer"
    },
```

```
    "ResourceQuery": {
        "Type": "TAG_FILTERS_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::EC2::Instance\"],\"TagFilters
\":[{\"Key\":\"Name\", \"Values\":[\"WebServers\"]}]}"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren von Gruppen im Benutzerhandbuch für AWS Resource Groups.

Beispiel 2: Um die Abfrage für eine CloudFormation stapelbasierte Ressourcengruppe zu aktualisieren

Im folgenden `update-group-query` Beispiel wird die Abfrage aktualisiert, die an die angegebene AWS CloudFormation stapelbasierte Ressourcengruppe angehängt ist.

```
aws resource-groups update-group-query \
    --group-name cbq-CFNstackgroup \
    --resource-query '{"Type": "CLOUDFORMATION_STACK_1_0", "Query":
 "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],\"StackIdentifier\":
\"arn:aws:cloudformation:us-west-2:123456789012:stack/MyCFNStack/1415z9z0-
z39z-11z8-97z5-500z212zz6fz\"}"}'
```

Ausgabe:

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-west-2:123456789012:group/cbq-
CFNstackgroup",
        "Name": "cbq-CFNstackgroup"
    },
    "ResourceQuery": {
        "Type": "CLOUDFORMATION_STACK_1_0",
        "Query": "{\"ResourceTypeFilters\":[\"AWS::AllSupported\"],
\"StackIdentifier\":\"arn:aws:cloudformation:us-west-2:123456789012:stack/
MyCFNStack/1415z9z0-z39z-11z8-97z5-500z212zz6fz\"}"
    }
}
```

Weitere Informationen finden Sie unter Aktualisieren von Gruppen im Benutzerhandbuch für AWS Resource Groups.

- Einzelheiten zur API finden Sie unter UpdateGroupQuery AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Abfrageobjekt erstellt und die Abfrage für die Gruppe aktualisiert.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Ausgabe:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Einzelheiten zur API finden Sie unter UpdateGroupQuery AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Abfrageobjekt erstellt und die Abfrage für die Gruppe aktualisiert.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
```

```
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Ausgabe:

```
GroupName ResourceQuery
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- Einzelheiten zur API finden Sie unter UpdateGroupQuery AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Ressourcengruppen-Tagging-API mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Resource Groups Tagging API mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Benutzerhandbuch zur Resource Groups Tagging API – Weitere Informationen zur Resource Groups Tagging API.
- API-Referenz zur Resource Groups Tagging API – Details zu allen verfügbaren Aktionen zur Resource Groups Tagging API.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung der Resource Groups Tagging API AWS SDKs

- Aktionen für die Tagging-API für Resource Groups mithilfe von AWS SDKs
  - Verwendung von GetResources mit einer CLI
  - Verwendung von GetTagKeys mit einer CLI
  - Verwendung von GetTagValues mit einer CLI
  - Verwendung von TagResources mit einer CLI
  - Verwendung von UntagResources mit einer CLI

Grundlegende Beispiele für die Verwendung der Resource Groups Tagging API AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Resource Groups Tagging API mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für die Tagging-API für Resource Groups mithilfe von AWS SDKs
  - Verwendung von GetResources mit einer CLI
  - Verwendung von GetTagKeys mit einer CLI
  - Verwendung von GetTagValues mit einer CLI
  - Verwendung von TagResources mit einer CLI
  - Verwendung von UntagResources mit einer CLI

Aktionen für die Tagging-API für Resource Groups mithilfe von AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Resource Groups Groups-Tagging-API-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Resource Groups Tagging API -API-Referenz.

Beispiele

- Verwendung von GetResources mit einer CLI
- Verwendung von GetTagKeys mit einer CLI
- Verwendung von GetTagValues mit einer CLI

- Verwendung von TagResources mit einer CLI
- Verwendung von UntagResources mit einer CLI

Verwendung von **`GetResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetResources` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste mit markierten Ressourcen ab

Im folgenden Beispiel für `get-resources` wird eine Liste der Ressourcen im Konto angezeigt, die mit dem angegebenen Schlüsselnamen und Wert gekennzeichnet sind.

```
aws resourcegroupstaggingapi get-resources \
    --tag-filters Key=Environment,Values=Production \
    --tags-per-page 100
```

Ausgabe:

```
{
    "ResourceTagMappingList": [
        {
            "ResourceARN": " arn:aws:inspector:us-west-2:123456789012:target/0-
nvgVhaxX/template/0-7sbz2Kz0",
            "Tags": [
                {
                    "Key": "Environment",
                    "Value": "Production"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie GetResourcesin der Resource Groups Tagging API-Referenz.

- Einzelheiten zur API finden Sie unter GetResources AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle markierten Ressourcen in einer Region und die mit der Ressource verbundenen Tag-Schlüssel zurück. Wenn dem Cmdlet kein Parameter -Region übergeben wird, versucht es, die Region aus den Metadaten der Shell oder der EC2-Instance abzuleiten.

```
Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                         Tags
-----------                                                         ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable               {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 2: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Die Zeichenfolge für jeden Servicenamen und Ressourcentyp entspricht der Zeichenfolge, die im Amazon-Ressourcennamen (ARN) einer Ressource eingebettet ist.

```
Get-RGTResource -ResourceType "s3"
```

Ausgabe:

```
ResourceARN                                                         Tags
-----------                                                         ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 3: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Beachten Sie, dass beim Übergeben der Ressourcentypen an das Cmdlet für jeden angegebenen Ressourcentyp ein Aufruf des Services erfolgt.

```
"dynamodb","s3" | Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 4: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 5: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter und Ressourcentyp entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

Beispiel 6: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

- Einzelheiten zur API finden Sie unter GetResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle markierten Ressourcen in einer Region und die mit der Ressource verbundenen Tag-Schlüssel zurück. Wenn dem Cmdlet kein Parameter -Region übergeben wird, versucht es, die Region aus den Metadaten der Shell oder der EC2-Instance abzuleiten.

```
Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 2: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Die Zeichenfolge für jeden Servicenamen und Ressourcentyp entspricht der Zeichenfolge, die im Amazon-Ressourcennamen (ARN) einer Ressource eingebettet ist.

```
Get-RGTResource -ResourceType "s3"
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 3: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Beachten Sie, dass beim Übergeben der Ressourcentypen an das Cmdlet für jeden angegebenen Ressourcentyp ein Aufruf des Services erfolgt.

```
"dynamodb","s3" | Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}

arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 4: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket
 {stage, version, othertag}
```

Beispiel 5: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter und Ressourcentyp entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

Beispiel 6: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
```

- Einzelheiten zur API finden Sie unter GetResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetTagKeys`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetTagKeys` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste aller Tag-Schlüssel ab

Im folgenden Beispiel für `get-tag-keys` wird die Liste aller Tag-Schlüsselnamen abgerufen, die von Ressourcen im Konto verwendet werden.

```
aws resourcegroupstaggingapi get-tag-keys
```

Ausgabe:

```
{
    "TagKeys": [
        "Environment",
        "CostCenter",
        "Department"
    ]
}
```

Weitere Informationen finden Sie GetTagKeysin der Resource Groups Tagging API-Referenz.

- Einzelheiten zur API finden Sie unter GetTagKeys AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle Tag-Schlüssel in der angegebenen Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten. Beachten Sie, dass die Tag-Schlüssel nicht in einer bestimmten Reihenfolge zurückgegeben werden.

```
Get-RGTTagKey -region us-west-2
```

Ausgabe:

```
version
stage
```

- Einzelheiten zur API finden Sie unter GetTagKeys AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle Tag-Schlüssel in der angegebenen Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten. Beachten Sie, dass die Tag-Schlüssel nicht in einer bestimmten Reihenfolge zurückgegeben werden.

```
Get-RGTTagKey -region us-west-2
```

Ausgabe:

```
version
stage
```

- Einzelheiten zur API finden Sie unter GetTagKeys AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetTagValues`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetTagValues` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste aller Tag-Werte ab

Im folgenden Beispiel für `get-tag-values` werden alle Werte angezeigt, die für den angegebenen Schlüssel verwendet wurden, für alle Ressourcen in

```
aws resourcegroupstaggingapi get-tag-values \
    --key=Environment
```

Ausgabe:

```
{
    "TagValues": [
        "Alpha",
        "Gamma",
        "Production"
    ]
}
```

Weitere Informationen finden Sie GetTagValuesin der Resource Groups Tagging API-Referenz.

- Einzelheiten zur API finden Sie unter GetTagValues AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt den Wert für das angegebene Tag in einer Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Ausgabe:

```
beta
```

- Einzelheiten zur API finden Sie unter GetTagValues AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt den Wert für das angegebene Tag in einer Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Ausgabe:

```
beta
```

- Einzelheiten zur API finden Sie unter GetTagValues AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`TagResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TagResources` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Ressource einen Tag hinzu

Im folgenden Beispiel für `tag-resources` wird die angegebene Ressource mit einem Schlüsselnamen und einem Schlüsselwert gekennzeichnet.

```
aws resourcegroupstaggingapi tag-resources \
    --resource-arn-list arn:aws:s3:::MyProductionBucket \
    --tags Environment=Production,CostCenter=1234
```

Ausgabe:

```
{
    "FailedResourcesMap": {}
}
```

Weitere Informationen finden Sie TagResourcesin der Resource Groups Tagging API-Referenz.

- Einzelheiten zur API finden Sie unter TagResources AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tag-Schlüssel „stage“ und „version“ mit den Werten „beta“ und „preprod_test“ zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es wird ein einziger Aufruf an den Service gesendet, um die Tags anzuwenden.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags und Werte zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

- Einzelheiten zur API finden Sie unter TagResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tag-Schlüssel „stage“ und „version“ mit den Werten „beta“ und „preprod_test“ zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es wird ein einziger Aufruf an den Service gesendet, um die Tags anzuwenden.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags und Werte zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

- Einzelheiten zur API finden Sie unter TagResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UntagResources`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UntagResources` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein Tag von einer Ressource

Der `untag-resources`-Befehl entfernt die angegebenen Tag-Schlüssel und die zugehörigen Werte aus der angegebenen Ressource.

```
aws resourcegroupstaggingapi untag-resources \
    --resource-arn-list arn:aws:s3:::amzn-s3-demo-bucket \
    --tag-keys Environment CostCenter
```

Ausgabe:

```
{
    "FailedResourcesMap": {}
}
```

Weitere Informationen finden Sie UntagResourcesin der Resource Groups Tagging API-Referenz.

- Einzelheiten zur API finden Sie unter UntagResources AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen. Bevor die Tags entfernt werden, fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Beispiel 2: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird. Vor jedem Aufruf fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Einzelheiten zur API finden Sie unter UntagResources AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen. Bevor die Tags entfernt werden, fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"
```

```
Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Beispiel 2: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird. Vor jedem Aufruf fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- Einzelheiten zur API finden Sie unter UntagResources AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für Route 53 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Route 53 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch für Route 53 – Weitere Informationen zu Route 53.
- API-Referenz für Route 53 – Details zu allen verfügbaren Route-53-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Route 53 mit AWS SDKs
  - Aktionen für Route 53 mit AWS SDKs

- Verwendung von ChangeResourceRecordSets mit einer CLI
- Verwendung von CreateHostedZone mit einer CLI
- Verwendung von DeleteHostedZone mit einer CLI
- Verwendung von GetHostedZone mit einer CLI
- Verwenden Sie mit einem ListHostedZones AWS SDK oder CLI
- Verwendung von ListHostedZonesByName mit einer CLI
- Verwendung von ListQueryLoggingConfigs mit einer CLI

Grundlegende Beispiele für Route 53 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Route 53 mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Route 53 mit AWS SDKs
  - Verwendung von ChangeResourceRecordSets mit einer CLI
  - Verwendung von CreateHostedZone mit einer CLI
  - Verwendung von DeleteHostedZone mit einer CLI
  - Verwendung von GetHostedZone mit einer CLI
  - Verwenden Sie mit einem ListHostedZones AWS SDK oder CLI
  - Verwendung von ListHostedZonesByName mit einer CLI
  - Verwendung von ListQueryLoggingConfigs mit einer CLI

Aktionen für Route 53 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Route 53-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Route 53.

Beispiele

- Verwendung von ChangeResourceRecordSets mit einer CLI

- Verwendung von CreateHostedZone mit einer CLI
- Verwendung von DeleteHostedZone mit einer CLI
- Verwendung von GetHostedZone mit einer CLI
- Verwenden Sie mit einem ListHostedZones AWS SDK oder CLI
- Verwendung von ListHostedZonesByName mit einer CLI
- Verwendung von ListQueryLoggingConfigs mit einer CLI

Verwendung von **`ChangeResourceRecordSets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ChangeResourceRecordSets` verwendet wird.

CLI

AWS CLI

So erstellen, aktualisieren oder löschen Sie einen Ressourcendatensatz

Der folgende `change-resource-record-sets` Befehl erstellt einen Ressourcendatensatz unter Verwendung der `hosted-zone-id` `Z1R8UBAEXAMPLE` und der JSON-formatted Konfiguration in der Datei`C:\awscli\route53\change-resource-record-sets.json`:

```
aws route53 change-resource-record-sets --hosted-zone-id Z1R8UBAEXAMPLE --change-
batch file://C:\awscli\route53\change-resource-record-sets.json
```

Weitere Informationen finden Sie unter POST ChangeResourceRecordSets in der Amazon Route 53 API-Referenz.

Die Konfiguration in der JSON-Datei hängt von der Art des Ressourceneintragssatzes ab, den Sie erstellen möchten:

BasicWeightedAliasWeighted AliasLatencyLatency AliasFailoverFailover Alias

Basissyntax

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
```

```
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ]
      }
    },
    {...}
  ]
}
```

Gewichtete Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Weight": value between 0 and 255,
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Alias-Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Gewichtete Alias-Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Weight": value between 0 and 255,
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
```

```
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Latenz-Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Latenz-Alias-Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
```

```
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Region": "Amazon EC2 region name",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

Failover-Syntax:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Failover": "PRIMARY" | "SECONDARY",
        "TTL": time to live in seconds,
        "ResourceRecords": [
          {
            "Value": "applicable value for the record type"
          },
          {...}
        ],
        "HealthCheckId": "ID of an Amazon Route 53 health check"
      }
    },
    {...}
```

```
  ]
}
```

Syntax des Failover-Alias:

```
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "CREATE"|"DELETE"|"UPSERT",
      "ResourceRecordSet": {
        "Name": "DNS domain name",
        "Type": "SOA"|"A"|"TXT"|"NS"|"CNAME"|"MX"|"PTR"|"SRV"|"SPF"|"AAAA",
        "SetIdentifier": "unique description for this resource record set",
        "Failover": "PRIMARY" | "SECONDARY",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
 Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
 hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution, Amazon S3
 bucket, Elastic Load Balancing load balancer, or another resource record set in
 this hosted zone",
          "EvaluateTargetHealth": true|false
        },
        "HealthCheckId": "optional ID of an Amazon Route 53 health check"
      }
    },
    {...}
  ]
}
```

- Einzelheiten zur API finden Sie ChangeResourceRecordSetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein A-Eintrag für www.example.com erstellt und der A-Eintrag für test.example.com von 192.0.2.3 in 192.0.2.1 geändert. Beachten Sie, dass Werte für TXT-type Änderungsdatensätze in doppelten Anführungszeichen stehen müssen. Weitere

Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Beispiel 2: Dieses Beispiel zeigt, wie Alias-Ressourceneinträge erstellt werden. „Z222222222“ ist die ID der von Amazon Route 53 gehosteten Zone, in der Sie den Alias-Ressourceneintragssatz erstellen. „example.com“ ist der Zonenapex, für den Sie einen Alias erstellen möchten, und „www.example.com“ ist eine Subdomain, für die Sie ebenfalls einen Alias erstellen möchten. „Z1111111111111“ ist ein Beispiel für eine gehostete Zonen-ID für den Load Balancer und „example-load-balancer-1111111111.us-east-1.elb.amazonaws.com“

ist ein Beispiel für einen Load-Balancer-Domainnamen, mit dem Amazon Route 53 auf Anfragen für example.com und www.example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets,
 one for the zone apex, example.com, and one for www.example.com, that both point
 to example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 3: In diesem Beispiel werden zwei A-Datensätze für www.example.com erstellt. One-fourth zu dem Zeitpunkt (1/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden Werten für den ersten Ressourcendatensatz (192.0.2.9

und 192.0.2.10). Three-fourths der Zeit (3/ (1+3)) Amazon Route 53 beantwortet Anfragen für www.example.com mit den beiden Werten für den zweiten Ressourcendatensatz (192.0.2.11 und 192.0.2.12). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets,
 each of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 4: Dieses Beispiel zeigt, wie gewichtete Alias-Ressourcendatensätze erstellt werden, vorausgesetzt, dass example.com die Domain ist, für die Sie gewichtete Alias-Ressourcendatensätze erstellen möchten. SetIdentifier unterscheidet die beiden gewichteten Alias-Ressourcendatensätze voneinander. Dieses Element ist erforderlich, da die Elemente „Name“ und „Type“ für beide Ressourceneintragsätze dieselben Werte

haben. Z1111111111111 und Z3333333333333 sind Beispiele für IDs gehosteter Zonen für den ELB-Load-Balancer, die durch den Wert von DNSName angegeben werden. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com und example-load-balancer-4444444444.us-east-1.elb.amazonaws.com sind Beispiele für Domains von Elastic Load Balancing, von denen aus Amazon Route 53 auf Abfragen für example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first
 ELB domain 3/4ths of the times and the second one 1/4th of the time."
```

```
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 5: In diesem Beispiel werden zwei Latenz-Alias-Ressourceneinträge erstellt, einer für einen ELB-Load Balancer in der Region US-West (Oregon) (us-west-2) und einer für einen Load Balancer in der Region Asien-Pazifik (Singapur) (ap-southeast-1). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
```

```
    ChangeBatch_Comment="This change batch creates two latency resource
 record sets, one for the US West (Oregon) region and one for the Asia Pacific
 (Singapore) region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Einzelheiten zur API finden Sie unter ChangeResourceRecordSets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein A-Eintrag für www.example.com erstellt und der A-Eintrag für test.example.com von 192.0.2.3 in 192.0.2.1 geändert. Beachten Sie, dass Werte für TXT-type Änderungsdatensätze in doppelten Anführungszeichen stehen müssen. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
```

```
$change3.ResourceRecordSet.ResourceRecords = @()
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Beispiel 2: Dieses Beispiel zeigt, wie Alias-Ressourceneinträge erstellt werden. „Z222222222“ ist die ID der von Amazon Route 53 gehosteten Zone, in der Sie den Alias-Ressourceneintragssatz erstellen. „example.com“ ist der Zonenapex, für den Sie einen Alias erstellen möchten, und „www.example.com“ ist eine Subdomain, für die Sie ebenfalls einen Alias erstellen möchten. „Z1111111111111“ ist ein Beispiel für eine gehostete Zonen-ID für den Load Balancer und „example-load-balancer-1111111111.us-east-1.elb.amazonaws.com“ ist ein Beispiel für einen Load-Balancer-Domainnamen, mit dem Amazon Route 53 auf Anfragen für example.com und www.example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
```

```
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets,
 one for the zone apex, example.com, and one for www.example.com, that both point
 to example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 3: In diesem Beispiel werden zwei A-Datensätze für www.example.com erstellt. One-fourth zu dem Zeitpunkt (1/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden Werten für den ersten Ressourcendatensatz (192.0.2.9 und 192.0.2.10). Three-fourths der Zeit (3/ (1+3)) Amazon Route 53 beantwortet Anfragen für www.example.com mit den beiden Werten für den zweiten Ressourcendatensatz (192.0.2.11 und 192.0.2.12). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
```

```
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets,
 each of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 4: Dieses Beispiel zeigt, wie gewichtete Alias-Ressourcendatensätze erstellt werden, vorausgesetzt, dass example.com die Domain ist, für die Sie gewichtete Alias-Ressourcendatensätze erstellen möchten. SetIdentifier unterscheidet die beiden gewichteten Alias-Ressourcendatensätze voneinander. Dieses Element ist erforderlich, da die Elemente „Name“ und „Type“ für beide Ressourceneintragsätze dieselben Werte haben. Z1111111111111 und Z3333333333333 sind Beispiele für IDs gehosteter Zonen für den ELB-Load-Balancer, die durch den Wert von DNSName angegeben werden. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com und example-load-balancer-4444444444.us-east-1.elb.amazonaws.com sind Beispiele für Domains von Elastic Load Balancing, von denen aus Amazon Route 53 auf Abfragen für example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
```

```
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first
 ELB domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 5: In diesem Beispiel werden zwei Latenz-Alias-Ressourceneinträge erstellt, einer für einen ELB-Load Balancer in der Region US-West (Oregon) (us-west-2) und einer für einen Load Balancer in der Region Asien-Pazifik (Singapur) (ap-southeast-1). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Mit dem Get-R53Change Cmdlet können Sie abfragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
```

```
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object
 Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource
 record sets, one for the US West (Oregon) region and one for the Asia Pacific
 (Singapore) region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- Einzelheiten zur API finden Sie unter ChangeResourceRecordSets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateHostedZone`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateHostedZone` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine gehostete Zone

Mit dem folgenden `create-hosted-zone`-Befehl wird mithilfe der Aufruferreferenz `2014-04-01-18:47` eine gehostete Zone namens `example.com` hinzugefügt. Der optionale Kommentar enthält ein Leerzeichen und muss daher in Anführungszeichen gesetzt werden:

```
aws route53 create-hosted-zone --name example.com --caller-
reference 2014-04-01-18:47 --hosted-zone-config Comment="command-line version"
```

Weitere Informationen finden Sie unter Arbeiten mit gehosteten Zonen im Entwicklerhandbuch für Amazon Route 53.

- Einzelheiten zur API finden Sie CreateHostedZonein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellt eine neue gehostete Zone namens „example.com“, die mit einem wiederverwendbaren Delegationssatz verknüpft ist. Beachten Sie, dass Sie einen Wert für den CallerReference Parameter angeben müssen, damit Anfragen, die bei Bedarf wiederholt werden müssen, ohne das Risiko eingehen zu müssen, dass der Vorgang zweimal ausgeführt wird. Da die gehostete Zone in einer VPC erstellt wird, ist sie automatisch privat und Sie sollten den PrivateZone Parameter - HostedZoneConfig _ nicht festlegen.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Einzelheiten zur API finden Sie unter CreateHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt eine neue gehostete Zone namens „example.com“, die mit einem wiederverwendbaren Delegationssatz verknüpft ist. Beachten Sie, dass Sie einen Wert für

den CallerReference Parameter angeben müssen, damit Anfragen, die bei Bedarf wiederholt werden müssen, ohne das Risiko eingehen zu müssen, dass der Vorgang zweimal ausgeführt wird. Da die gehostete Zone in einer VPC erstellt wird, ist sie automatisch privat und Sie sollten den PrivateZone Parameter - HostedZoneConfig _ nicht festlegen.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- Einzelheiten zur API finden Sie unter CreateHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteHostedZone`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteHostedZone` verwendet wird.

CLI

AWS CLI

So löschen Sie eine gehostete Zone

Mit dem folgenden `delete-hosted-zone`-Befehl wird die gehostete Zone mit einer `id` von `Z36KTIQEXAMPLE` gelöscht:

```
aws route53 delete-hosted-zone --id Z36KTIQEXAMPLE
```

- Einzelheiten zur API finden Sie DeleteHostedZonein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht die gehostete Zone mit der angegebenen ID. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird, es sei denn, Sie fügen den Schalterparameter -Force hinzu.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Einzelheiten zur API finden Sie unter DeleteHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht die gehostete Zone mit der angegebenen ID. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird, es sei denn, Sie fügen den Schalterparameter -Force hinzu.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- Einzelheiten zur API finden Sie unter DeleteHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetHostedZone`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetHostedZone` verwendet wird.

CLI

AWS CLI

So rufen Sie Informationen zu einer gehosteten Zone ab

Mit dem folgenden `get-hosted-zone`-Befehl werden Informationen über die gehostete Zone mit einem `id` von `Z1R8UBAEXAMPLE` abgerufen:

```
aws route53 get-hosted-zone --id Z1R8UBAEXAMPLE
```

- Einzelheiten zur API finden Sie GetHostedZonein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt Details der gehosteten Zone mit der ID Z1D633PJN98FT9 zurück.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Einzelheiten zur API finden Sie unter GetHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt Details der gehosteten Zone mit der ID Z1D633PJN98FT9 zurück.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- Einzelheiten zur API finden Sie unter GetHostedZone AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListHostedZones`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListHostedZones` verwendet wird.

CLI

AWS CLI

Um die Hosting-Zonen aufzulisten, die dem aktuellen AWS Konto zugeordnet sind

Der folgende `list-hosted-zones` Befehl listet zusammenfassende Informationen zu den ersten 100 Hostzonen auf, die dem aktuellen AWS Konto zugeordnet sind. :

```
aws route53 list-hosted-zones
```

Wenn Sie mehr als 100 gehostete Zonen haben oder wenn Sie sie in Gruppen von weniger als 100 auflisten möchten, fügen Sie den Parameter `--max-items` ein. Um zum Beispiel eine gehostete Zone nach der anderen aufzulisten, verwenden Sie den folgenden Befehl:

```
aws route53 list-hosted-zones --max-items 1
```

Um Informationen über die nächste gehostete Zone anzuzeigen, übernehmen Sie den Wert von `NextToken` aus der Antwort auf den vorherigen Befehl und fügen ihn in den Parameter `--starting-token` ein, zum Beispiel:

```
aws route53 list-hosted-zones --max-items 1 --starting-token Z3M3LMPEXAMPLE
```

- Einzelheiten zur API finden Sie ListHostedZonesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen aus.

```
Get-R53HostedZoneList
```

Beispiel 2: Gibt alle gehosteten Zonen aus, die mit dem wiederverwendbaren Delegationssatz mit der ID NZ8X2CISAMPLE verknüpft sind.

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Einzelheiten zur API finden Sie unter ListHostedZones AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen aus.

```
Get-R53HostedZoneList
```

Beispiel 2: Gibt alle gehosteten Zonen aus, die mit dem wiederverwendbaren Delegationssatz mit der ID NZ8X2CISAMPLE verknüpft sind.

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- Einzelheiten zur API finden Sie unter ListHostedZones AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),
 aws_sdk_route53::Error> {
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;

    println!(
        "Number of hosted zones in region : {}",
        hosted_zone_count.hosted_zone_count(),
    );

    let hosted_zones = client.list_hosted_zones().send().await?;

    println!("Zones:");

    for hz in hosted_zones.hosted_zones() {
        let zone_name = hz.name();
        let zone_id = hz.id();

        println!("  ID :   {}", zone_id);
        println!("  Name : {}", zone_name);
        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListHostedZonesin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`ListHostedZonesByName`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListHostedZonesByName` verwendet wird.

CLI

AWS CLI

Der folgende Befehl listet bis zu 100 gehostete Zonen auf, sortiert nach Domainnamen:

```
aws route53 list-hosted-zones-by-name
```

Ausgabe:

```
{
  "HostedZones": [
      {
          "ResourceRecordSetCount": 2,
          "CallerReference": "test20150527-2",
          "Config": {
              "Comment": "test2",
              "PrivateZone": false
          },
          "Id": "/hostedzone/Z119WBBTVP5WFX",
          "Name": "2.example.com."
      },
      {
          "ResourceRecordSetCount": 2,
          "CallerReference": "test20150527-1",
          "Config": {
              "Comment": "test",
              "PrivateZone": false
          },
          "Id": "/hostedzone/Z3P5QSUBK4POTI",
          "Name": "www.example.com."
      }
  ],
  "IsTruncated": false,
  "MaxItems": "100"
}
```

Der folgende Befehl listet die gehosteten Zonen nach Namen geordnet auf, beginnend mit `www.example.com`:

```
aws route53 list-hosted-zones-by-name --dns-name www.example.com
```

Ausgabe:

```
{
  "HostedZones": [
      {
          "ResourceRecordSetCount": 2,
          "CallerReference": "mwunderl20150527-1",
          "Config": {
              "Comment": "test",
              "PrivateZone": false
          },
          "Id": "/hostedzone/Z3P5QSUBK4POTI",
          "Name": "www.example.com."
      }
  ],
  "DNSName": "www.example.com",
  "IsTruncated": false,
  "MaxItems": "100"
}
```

- Einzelheiten zur API finden Sie ListHostedZonesByNamein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück.

```
Get-R53HostedZonesByName
```

Beispiel 2: Gibt Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück, beginnend mit dem angegebenen DNS-Namen.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- Einzelheiten zur API finden Sie unter ListHostedZonesByName AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück.

```
Get-R53HostedZonesByName
```

Beispiel 2: Gibt Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück, beginnend mit dem angegebenen DNS-Namen.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- Einzelheiten zur API finden Sie unter ListHostedZonesByName AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListQueryLoggingConfigs`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListQueryLoggingConfigs` verwendet wird.

CLI

AWS CLI

So listen Sie die Konfigurationen zur Abfrageprotokollierung auf

Im folgenden `list-query-logging-configs` Beispiel werden Informationen zu den ersten 100 Konfigurationen für die Abfrageprotokollierung in Ihrem AWS Konto für die gehostete Zone aufgeführt`Z1OX3WQEXAMPLE`.

```
aws route53 list-query-logging-configs \
    --hosted-zone-id Z1OX3WQEXAMPLE
```

Ausgabe:

```
{
    "QueryLoggingConfigs": [
        {
            "Id": "964ff34e-ae03-4f06-80a2-9683cexample",
            "HostedZoneId": "Z1OX3WQEXAMPLE",
```

```
            "CloudWatchLogsLogGroupArn": "arn:aws:logs:us-
east-1:111122223333:log-group:/aws/route53/example.com:*"
        }
    ]
}
```

Weitere Informationen finden Sie unter Protokollieren von DNS-Abfragen im Entwicklerhandbuch für Amazon Route 53.

- Einzelheiten zur API finden Sie ListQueryLoggingConfigsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Konfigurationen für die DNS-Abfrageprotokollierung zurückgegeben, die mit dem aktuellen AWS-Konto verknüpft sind.

```
Get-R53QueryLoggingConfigList
```

Ausgabe:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Einzelheiten zur API finden Sie unter ListQueryLoggingConfigs AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Konfigurationen für die DNS-Abfrageprotokollierung zurückgegeben, die mit dem aktuellen AWS-Konto verknüpft sind.

```
Get-R53QueryLoggingConfigList
```

Ausgabe:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- Einzelheiten zur API finden Sie unter ListQueryLoggingConfigs AWS -Tools für PowerShellCmdlet-Referenz (V5).

Codebeispiele für die Route 53-Domainregistrierung mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es Amazon Route 53 domain registration mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Entwicklerhandbuch zur Route-53-Domainregistrierung – Weitere Informationen zur Route 53-Domainregistrierung.
- API-Referenz für die Route-53-Domainregistrierung – Details zu allen verfügbaren Aktionen für die Route-53-Domainregistrierung.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Route 53-Domainregistrierung mit AWS SDKs
  - Hallo Route-53-Domainregistrierung
  - Lernen Sie die Grundlagen der Route 53-Domainregistrierung mit einem AWS SDK
  - Aktionen für die Route 53-Domainregistrierung mit AWS SDKs
    - Verwenden Sie CheckDomainAvailabilitymit einem AWS SDK oder CLI
    - Verwenden Sie CheckDomainTransferabilitymit einem AWS SDK oder CLI
    - Verwenden Sie GetDomainDetailmit einem AWS SDK oder CLI
    - Verwenden Sie GetDomainSuggestionsmit einem AWS SDK oder CLI
    - Verwenden Sie GetOperationDetailmit einem AWS SDK oder CLI
    - Verwenden Sie ListDomainsmit einem AWS SDK oder CLI
    - Verwenden Sie ListOperationsmit einem AWS SDK oder CLI
    - Verwenden Sie ListPricesmit einem AWS SDK
    - Verwenden Sie RegisterDomainmit einem AWS SDK oder CLI
    - Verwenden Sie ViewBillingmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Route 53-Domainregistrierung mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Route 53 domain registration mit AWS SDKs verwendet werden.

Beispiele

- Hallo Route-53-Domainregistrierung
- Lernen Sie die Grundlagen der Route 53-Domainregistrierung mit einem AWS SDK
- Aktionen für die Route 53-Domainregistrierung mit AWS SDKs
  - Verwenden Sie CheckDomainAvailabilitymit einem AWS SDK oder CLI
  - Verwenden Sie CheckDomainTransferabilitymit einem AWS SDK oder CLI
  - Verwenden Sie GetDomainDetailmit einem AWS SDK oder CLI
  - Verwenden Sie GetDomainSuggestionsmit einem AWS SDK oder CLI
  - Verwenden Sie GetOperationDetailmit einem AWS SDK oder CLI
  - Verwenden Sie ListDomainsmit einem AWS SDK oder CLI
  - Verwenden Sie ListOperationsmit einem AWS SDK oder CLI

- Verwenden Sie ListPricesmit einem AWS SDK
- Verwenden Sie RegisterDomainmit einem AWS SDK oder CLI
- Verwenden Sie ViewBillingmit einem AWS SDK oder CLI

Hallo Route-53-Domainregistrierung

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung der Route-53-Domainregistrierung beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static class HelloRoute53Domains
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon Route 53 domain registration service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRoute53Domains>()
            ).Build();

        // Now the client is available for injection.
        var route53Client =
 host.Services.GetRequiredService<IAmazonRoute53Domains>();

        // You can use await and any of the async methods to get a response.
        var response = await route53Client.ListPricesAsync(new ListPricesRequest
 { Tld = "com" });
```

```
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices
 for .com domain operations:");
        var comPrices = response.Prices.FirstOrDefault();
        if (comPrices != null)
        {
            Console.WriteLine($"\tRegistration:
 {comPrices.RegistrationPrice?.Price} {comPrices.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {comPrices.RenewalPrice?.Price}
 {comPrices.RenewalPrice?.Currency}");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPricesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type
 * and displays the prices for Registration and Renewal.
 *
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <hostedZoneId> \n\n" +
                "Where:\n" +
                "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient =
 Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient,
 String domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .maxItems(10)
                    .tld(domainType)
                    .build();
```

```
            ListPricesResponse response =
 route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
                System.out.println(
                        "Registration: " + pr.registrationPrice().price() + " " +
 pr.registrationPrice().currency());
                System.out.println("Renewal: " + pr.renewalPrice().price() + " "
 + pr.renewalPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + "
 " + pr.transferPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + "
 " + pr.transferPrice().currency());
                System.out.println("Change Ownership: " +
 pr.changeOwnershipPrice().price() + " "
                        + pr.changeOwnershipPrice().currency());
                System.out.println(
                        "Restoration: " + pr.restorationPrice().price() + " " +
 pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPricesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
```

```
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListPricesin der API-Referenz zum AWS SDK für Kotlin.

Lernen Sie die Grundlagen der Route 53-Domainregistrierung mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Auflisten der aktuellen Domains und der Vorgänge des letzten Jahres
- Anzeigen der Abrechnung für das vergangene Jahr und der Preise für Domaintypen
- Abrufen von Domainvorschlägen
- Überprüfen der Verfügbarkeit und Übertragbarkeit von Domains
- Optional: Anfordern einer Domainregistrierung
- Abrufen eines Vorgangsdetails
- Optional: Abrufen eines Domaindetails

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public static class Route53DomainScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. List current domains.
        2. List operations in the past year.
        3. View billing for the account in the past year.
        4. View prices for domain types.
        5. Get domain suggestions.
        6. Check domain availability.
        7. Check domain transferability.
        8. Optionally, request a domain registration.
        9. Get an operation detail.
       10. Optionally, get a domain detail.
   */

    private static Route53Wrapper _route53Wrapper = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
                    .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonRoute53Domains>()
                .AddTransient<Route53Wrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();
```

```
        var logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger(typeof(Route53DomainScenario));

        _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Route 53 domains example
scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListDomains();
            await ListOperations();
            await ListBillingRecords();
            await ListPrices();
            await ListDomainSuggestions();
            await CheckDomainAvailability();
            await CheckDomainTransferability();
            var operationId = await RequestDomainRegistration();
            await GetOperationalDetail(operationId);
            await GetDomainDetails();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon Route 53 domains example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List account registered domains.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomains()
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"1. List account domains.");
        var domains = await _route53Wrapper.ListDomains();
        for (int i = 0; i < domains.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {domains[i].DomainName}");
        }

        if (!domains.Any())
        {
            Console.WriteLine("\tNo domains found in this account.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListOperations()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List account domain operations in the past
year.");
        var operations = await _route53Wrapper.ListOperations(
            DateTime.Today.AddYears(-1));
        for (int i = 0; i < operations.Count; i++)
        {
            Console.WriteLine($"\tOperation Id: {operations[i].OperationId}");
            Console.WriteLine($"\tStatus: {operations[i].Status}");
            Console.WriteLine($"\tDate: {operations[i].SubmittedDate}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListBillingRecords()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. View billing for the account in the past year.");
        var billingRecords = await _route53Wrapper.ViewBilling(
```

```
            DateTime.Today.AddYears(-1),
            DateTime.Today);
        for (int i = 0; i < billingRecords.Count; i++)
        {
            Console.WriteLine($"\tBill Date:
{billingRecords[i].BillDate.ToShortDateString()}");
            Console.WriteLine($"\tOperation: {billingRecords[i].Operation}");
            Console.WriteLine($"\tPrice: {billingRecords[i].Price}");
        }
        if (!billingRecords.Any())
        {
            Console.WriteLine("\tNo billing records found in this account for the
past year.");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List prices for a few domain types.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListPrices()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. View prices for domain types.");
        var domainTypes = new List<string> { "net", "com", "org", "co" };

        var prices = await _route53Wrapper.ListPrices(domainTypes);
        foreach (var pr in prices)
        {
            Console.WriteLine($"\tName: {pr.Name}");
            Console.WriteLine($"\tRegistration: {pr.RegistrationPrice?.Price}
{pr.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {pr.RenewalPrice?.Price}
{pr.RenewalPrice?.Currency}");
            Console.WriteLine($"\tTransfer: {pr.TransferPrice?.Price}
{pr.TransferPrice?.Currency}");
            Console.WriteLine($"\tChange Ownership:
{pr.ChangeOwnershipPrice?.Price} {pr.ChangeOwnershipPrice?.Currency}");
            Console.WriteLine($"\tRestoration: {pr.RestorationPrice?.Price}
{pr.RestorationPrice?.Currency}");
            Console.WriteLine();
        }
        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// List domain suggestions for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomainSuggestions()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Get domain suggestions.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to get available domain
suggestions.");
            domainName = Console.ReadLine();
        }

        var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName,
true, 5);
        foreach (var suggestion in suggestions)
        {
            Console.WriteLine($"\tSuggestion Name: {suggestion.DomainName}");
            Console.WriteLine($"\tAvailability: {suggestion.Availability}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check availability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainAvailability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Check domain availability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain
availability.");
            domainName = Console.ReadLine();
        }
```

```
        var availability = await
_route53Wrapper.CheckDomainAvailability(domainName);
        Console.WriteLine($"\tAvailability: {availability}");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainTransferability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Check domain transferability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain
transferability.");
            domainName = Console.ReadLine();
        }

        var transferability = await
_route53Wrapper.CheckDomainTransferability(domainName);
        Console.WriteLine($"\tTransferability: {transferability}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> RequestDomainRegistration()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Optionally, request a domain registration.");

        Console.WriteLine($"\tNote: This example uses domain request settings in
settings.json.");
        Console.WriteLine($"\tTo change the domain registration settings, set the
values in that file.");
        Console.WriteLine($"\tRemember, registering an actual domain will incur
an account billing cost.");
```

```
        Console.WriteLine($"\tWould you like to begin a domain registration? (y/
n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string domainName = _configuration["DomainName"];
            ContactDetail contact = new ContactDetail();
            contact.CountryCode =
 CountryCode.FindValue(_configuration["Contact:CountryCode"]);
            contact.ContactType =
 ContactType.FindValue(_configuration["Contact:ContactType"]);

            _configuration.GetSection("Contact").Bind(contact);

            var operationId = await _route53Wrapper.RegisterDomain(
                domainName,
                Convert.ToBoolean(_configuration["AutoRenew"]),
                Convert.ToInt32(_configuration["DurationInYears"]),
                contact);
            if (operationId != null)
            {
                Console.WriteLine(
                    $"\tRegistration requested. Operation Id: {operationId}");
            }

            return operationId;
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }

    /// <summary>
    /// Get details for an operation.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetOperationalDetail(string? operationId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Get an operation detail.");

        var operationDetails =
            await _route53Wrapper.GetOperationDetail(operationId);
```

```
        Console.WriteLine(operationDetails);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Optionally, get details for a registered domain.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> GetDomainDetails()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Get details on a domain.");

        Console.WriteLine($"\tNote: you must have a registered domain to get
 details.");
        Console.WriteLine($"\tWould you like to get domain details? (y/n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string? domainName = null;
            while (domainName == null)
            {
                Console.WriteLine($"\tEnter a domain name to get details.");
                domainName = Console.ReadLine();
            }

            var domainDetails = await
 _route53Wrapper.GetDomainDetail(domainName);
            Console.WriteLine(domainDetails);
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }
}
```

Wrapper-Methoden, die vom Szenario für Route-53-Domainregistrierungsaktionen verwendet werden.

```
public class Route53Wrapper
{
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;
    private readonly ILogger<Route53Wrapper> _logger;
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,
 ILogger<Route53Wrapper> logger)
    {
        _amazonRoute53Domains = amazonRoute53Domains;
        _logger = logger;
    }


    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
 ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }


    /// <summary>
    /// Check the availability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for availability.</param>
    /// <returns>An availability result string.</returns>
    public async Task<string> CheckDomainAvailability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
            new CheckDomainAvailabilityRequest
            {
                DomainName = domain
```

```
            }
        );
        return result.Availability.Value;
    }


    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Transferability.Transferable.Value;
    }


    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</
param>
    /// <param name="suggestionCount">The number of suggestions to return.
 Defaults to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
```

```
    }


    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
 {operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }


    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
```

```
    /// <param name="duration">The duration in years for the domain
registration.</param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
int duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant,
and tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
{domainName}");
            return null;
        }
    }


    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate,
DateTime endDate)
    {
```

```
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }


    /// <summary>
    /// List the domains for the account.
    /// </summary>
    /// <returns>A collection of domain summary records.</returns>
    public async Task<List<DomainSummary>> ListDomains()
    {
        var results = new List<DomainSummary>();
        var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
            new ListDomainsRequest());

        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }


    /// <summary>
    /// List operations for the account that are submitted after a specified
date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
    {
```

```
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }


    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on
 {result.CreationDate.ToShortDateString()}.\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
```

```
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/
pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainType> <phoneNumber> <email> <domainSuggestion>
 <firstName> <lastName> <city>

                Where:
                    domainType - The domain type (for example, com).\s
                    phoneNumber - The phone number to use (for example,
 +91.9966564xxx)      email - The email address to use.      domainSuggestion -
 The domain suggestion (for example, findmy.accountants).\s
                    firstName - The first name to use to register a domain.\s
                    lastName -  The last name to use to register a domain.\s
                    city - the city to use to register a domain.\s
                    """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String domainType = args[0];
        String phoneNumber = args[1];
        String email = args[2];
        String domainSuggestion = args[3];
        String firstName = args[4];
        String lastName = args[5];
        String city = args[6];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient =
Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest =
GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
```

```
            System.out.println("The contact last name is " +
 response.registrantContact().lastName());
            System.out.println("The contact org name is " +
 response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
 route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
 GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
 route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " +
 response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
 route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
```

```
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " +
suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
```

```
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient,
String domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " +
content.registrationPrice().price() + " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient
route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);
```

```
            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient)
{
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is "
 + content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin code example performs the following operations:

1. List current domains.
2. List operations in the past year.
3. View billing for the account in the past year.
4. View prices for domain types.
5. Get domain suggestions.
6. Check domain availability.
7. Check domain transferability.
8. Request a domain registration.
9. Get operation details.
10. Optionally, get domain details.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
 <lastName> <city>
        Where:
           domainType - The domain type (for example, com).
```

```
          phoneNumber - The phone number to use (for example, +1.2065550100)

          email - The email address to use.
          domainSuggestion - The domain suggestion (for example,
findmy.example).
          firstName - The first name to use to register a domain.
          lastName -  The last name to use to register a domain.
          city - The city to use to register a domain.
   """

   if (args.size != 7) {
       println(usage)
       exitProcess(1)
   }

   val domainType = args[0]
   val phoneNumber = args[1]
   val email = args[2]
   val domainSuggestion = args[3]
   val firstName = args[4]
   val lastName = args[5]
   val city = args[6]

   println(DASHES)
   println("Welcome to the Amazon Route 53 domains example scenario.")
   println(DASHES)

   println(DASHES)
   println("1. List current domains.")
   listDomains()
   println(DASHES)

   println(DASHES)
   println("2. List operations in the past year.")
   listOperations()
   println(DASHES)

   println(DASHES)
   println("3. View billing for the account in the past year.")
   listBillingRecords()
   println(DASHES)

   println(DASHES)
   println("4. View prices for domain types.")
```

```
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
 firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is
 ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
```

```
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
```

```
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
```

```
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Aktionen für die Route 53-Domainregistrierung mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Route 53-Domänenregistrierungsaktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Route 53 domain registration -API-Referenz.

Beispiele

- Verwenden Sie CheckDomainAvailabilitymit einem AWS SDK oder CLI
- Verwenden Sie CheckDomainTransferabilitymit einem AWS SDK oder CLI
- Verwenden Sie GetDomainDetailmit einem AWS SDK oder CLI
- Verwenden Sie GetDomainSuggestionsmit einem AWS SDK oder CLI
- Verwenden Sie GetOperationDetailmit einem AWS SDK oder CLI
- Verwenden Sie ListDomainsmit einem AWS SDK oder CLI
- Verwenden Sie ListOperationsmit einem AWS SDK oder CLI
- Verwenden Sie ListPricesmit einem AWS SDK
- Verwenden Sie RegisterDomainmit einem AWS SDK oder CLI
- Verwenden Sie ViewBillingmit einem AWS SDK oder CLI

Verwenden Sie **`CheckDomainAvailability`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CheckDomainAvailability` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Availability.Value;
}
```

- Einzelheiten zur API finden Sie CheckDomainAvailabilityin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So stellen Sie fest, ob Sie einen Domainnamen mit Route 53 registrieren können

Der folgende `check-domain-availability`-Befehl gibt Informationen darüber zurück, ob der Domainname `example.com` für die Registrierung über Route 53 verfügbar ist.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains check-domain-availability \
    --region us-east-1 \
    --domain-name example.com
```

Ausgabe:

```
{
    "Availability": "UNAVAILABLE"
}
```

Route 53 unterstützt eine große Anzahl von Top-Level-Domains (TLDs), z. B. `.com` und `.jp`, wir unterstützen jedoch nicht alle verfügbaren TLDs. Wenn Sie die Verfügbarkeit einer Domain überprüfen und Route 53 die TLD nicht unterstützt, gibt `check-domain-availability` die folgende Meldung zurück.

```
An error occurred (UnsupportedTLD) when calling the CheckDomainAvailability
 operation: <top-level domain> tld is not supported.
```

Eine Liste der TLDs, die Sie bei einer Domainregistrierung mit Route 53 verwenden können, finden Sie unter Domains, die Sie mit Amazon Route 53 registrieren können im Entwicklerhandbuch für Amazon Route 53. Weitere Informationen zum Registrieren einer Domain mit Route 53 finden Sie unter Registrieren einer neuen Domain im Entwicklerhandbuch für Amazon Route 53.

- Einzelheiten zur API finden Sie CheckDomainAvailabilityin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void checkDomainAvailability(Route53DomainsClient
 route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
 CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
 response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie CheckDomainAvailabilityin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
```

```
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- Einzelheiten zur API finden Sie CheckDomainAvailabilityin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CheckDomainTransferability`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CheckDomainTransferability` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
```

```
                DomainName = domain
            }
        );
        return result.Transferability.Transferable.Value;
    }
```

- Einzelheiten zur API finden Sie CheckDomainTransferabilityin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So stellen Sie fest, ob eine Domain auf Route 53 übertragen werden kann 53

Der folgende `check-domain-transferability`-Befehl gibt Informationen darüber zurück, ob Sie den Domainnamen `example.com` auf Route 53 übertragen können.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains check-domain-transferability \
    --region us-east-1 \
    --domain-name example.com
```

Ausgabe:

```
{
    "Transferability": {
        "Transferable": "UNTRANSFERABLE"
    }
}
```

Weitere Informationen finden Sie unter Übertragen der Registrierung für eine Domain an Amazon Route 53 im Entwicklerhandbuch für Amazon Route 53.

- Einzelheiten zur API finden Sie CheckDomainTransferabilityin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           CheckDomainTransferabilityResponse response = route53DomainsClient
                   .checkDomainTransferability(transferabilityRequest);
           System.out.println("Transferability: " +
response.transferability().transferable().toString());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie CheckDomainTransferabilityin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- Einzelheiten zur API finden Sie CheckDomainTransferabilityin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetDomainDetail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDomainDetail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on
{result.CreationDate.ToShortDateString()}.\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
```

- Einzelheiten zur API finden Sie GetDomainDetailin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie detaillierte Informationen zu einer bestimmten Domain ab

Der folgende `get-domain-detail`-Befehl zeigt detaillierte Informationen über die angegebene Domain an.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains get-domain-detail \
    --region us-east-1 \
    --domain-name example.com
```

Ausgabe:

```
{
    "DomainName": "example.com",
    "Nameservers": [
        {
            "Name": "ns-2048.awsdns-64.com",
            "GlueIps": []
        },
        {
            "Name": "ns-2049.awsdns-65.net",
            "GlueIps": []
        },
        {
            "Name": "ns-2050.awsdns-66.org",
            "GlueIps": []
        },
        {
            "Name": "ns-2051.awsdns-67.co.uk",
            "GlueIps": []
        }
    ],
    "AutoRenew": true,
    "AdminContact": {
        "FirstName": "Saanvi",
        "LastName": "Sarkar",
        "ContactType": "COMPANY",
```

```
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "ssarkar@example.com",
        "ExtraParams": []
    },
    "RegistrantContact": {
        "FirstName": "Alejandro",
        "LastName": "Rosalez",
        "ContactType": "COMPANY",
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "arosalez@example.com",
        "ExtraParams": []
    },
    "TechContact": {
        "FirstName": "Wang",
        "LastName": "Xiulan",
        "ContactType": "COMPANY",
        "OrganizationName": "Example",
        "AddressLine1": "123 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "wxiulan@example.com",
        "ExtraParams": []
    },
    "AdminPrivacy": true,
    "RegistrantPrivacy": true,
    "TechPrivacy": true,
    "RegistrarName": "Amazon Registrar, Inc.",
    "WhoIsServer": "whois.registrar.amazon",
    "RegistrarUrl": "http://registrar.amazon.com",
```

```
    "AbuseContactEmail": "abuse@registrar.amazon.com",
    "AbuseContactPhone": "+1.2062661000",
    "CreationDate": 1444934889.601,
    "ExpirationDate": 1602787689.0,
    "StatusList": [
        "clientTransferProhibited"
    ]
}
```

- Einzelheiten zur API finden Sie GetDomainDetailin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void getDomainDetails(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainDetailRequest detailRequest =
GetDomainDetailRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
           System.out.println("The contact first name is " +
response.registrantContact().firstName());
           System.out.println("The contact last name is " +
response.registrantContact().lastName());
           System.out.println("The contact org name is " +
response.registrantContact().organizationName());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- Einzelheiten zur API finden Sie GetDomainDetailin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is
 ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}
```

- Einzelheiten zur API finden Sie GetDomainDetailin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetDomainSuggestions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDomainSuggestions` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</
param>
    /// <param name="suggestionCount">The number of suggestions to return.
 Defaults to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }
```

- Einzelheiten zur API finden Sie GetDomainSuggestionsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste der vorgeschlagenen Domainnamen ab

Der folgende `get-domain-suggestions`-Befehl zeigt eine Liste mit vorgeschlagenen Domainnamen an, die auf dem Domainnamen `example.com` basieren. Die Antwort enthält nur verfügbare Domainnamen. Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains get-domain-suggestions \
    --region us-east-1 \
    --domain-name example.com \
    --suggestion-count 10 \
    --only-available
```

Ausgabe:

```
{
    "SuggestionsList": [
        {
            "DomainName": "egzaampal.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplelaw.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplehouse.net",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "homeexample.net",
            "Availability": "AVAILABLE"
        },
        {
```

```
            "DomainName": "examplelist.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "examplenews.net",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "officeexample.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "exampleworld.com",
            "Availability": "AVAILABLE"
        },
        {
            "DomainName": "exampleart.com",
            "Availability": "AVAILABLE"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetDomainSuggestionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
```

```
                    .build();

            GetDomainSuggestionsResponse response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " +
 suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie GetDomainSuggestionsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
```

```
        val response =
 route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetDomainSuggestionsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetOperationDetail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetOperationDetail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
```

```
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
{operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }
```

- Einzelheiten zur API finden Sie GetOperationDetailin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den aktuellen Status einer Operation ab

Einige Domainregistrierungsvorgänge werden asynchron ausgeführt und geben eine Antwort zurück, bevor sie abgeschlossen sind. Diese Operationen geben eine Vorgangs-ID zurück, mit der Sie den aktuellen Status abrufen können. Der folgende `get-operation-detail`-Befehl gibt den Status des angegebenen Vorgangs zurück.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains get-operation-detail \
    --region us-east-1 \
    --operation-id edbd8d63-7fe7-4343-9bc5-54033example
```

Ausgabe:

```
{
    "OperationId": "edbd8d63-7fe7-4343-9bc5-54033example",
    "Status": "SUCCESSFUL",
    "DomainName": "example.com",
    "Type": "DOMAIN_LOCK",
    "SubmittedDate": 1573749367.864
}
```

- Einzelheiten zur API finden Sie GetOperationDetailin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
       try {
           GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                   .operationId(operationId)
                   .build();

           GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
           System.out.println("Operation detail message is " +
response.message());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie GetOperationDetailin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- Einzelheiten zur API finden Sie GetOperationDetailin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListDomains`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListDomains` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
    var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
        new ListDomainsRequest());

    // Get the entire list using the paginator.
    await foreach (var domain in paginateDomains.Domains)
    {
        results.Add(domain);
    }
    return results;
}
```

- Einzelheiten zur API finden Sie ListDomainsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um die Domains aufzulisten, die mit dem aktuellen AWS Konto registriert sind

Der folgende `list-domains` Befehl listet zusammenfassende Informationen zu den Domänen auf, die mit dem aktuellen AWS Konto registriert sind.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains list-domains
    --region us-east-1
```

Ausgabe:

```
{
    "Domains": [
        {
            "DomainName": "example.com",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602712345.0
        },
        {
            "DomainName": "example.net",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602723456.0
        },
        {
            "DomainName": "example.org",
            "AutoRenew": true,
            "TransferLock": true,
            "Expiry": 1602734567.0
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListDomainsunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is "
+ content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListDomainsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomains() {
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListDomainsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListOperations`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListOperations` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List operations for the account that are submitted after a specified
 date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
 submittedSince)
```

```
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie ListOperationsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie den Status von Vorgängen auf, die eine Vorgangs-ID zurückgeben

Einige Domainregistrierungsvorgänge werden asynchron ausgeführt und geben eine Antwort zurück, bevor sie abgeschlossen sind. Diese Operationen geben eine Vorgangs-ID zurück, mit der Sie den aktuellen Status abrufen können. Der folgende `list-operations`-Befehl listet zusammenfassende Informationen, einschließlich des Status, zu den aktuellen Domainregistrierungsvorgängen auf.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains list-operations
    --region us-east-1
```

Ausgabe:

```
{
    "Operations": [
```

```
        {
            "OperationId": "aab9822f-1da0-4bf3-8a15-fd4e0example",
            "Status": "SUCCESSFUL",
            "Type": "DOMAIN_LOCK",
            "SubmittedDate": 1455321739.986
        },
        {
            "OperationId": "c24379ed-76be-42f8-bdad-9379bexample",
            "Status": "SUCCESSFUL",
            "Type": "UPDATE_NAMESERVER",
            "SubmittedDate": 1468960475.109
        },
        {
            "OperationId": "f47e1297-ef9e-4c2b-ae1e-a5fcbexample",
            "Status": "SUCCESSFUL",
            "Type": "RENEW_DOMAIN",
            "SubmittedDate": 1473561835.943
        },
        {
            "OperationId": "75584f23-b15f-459e-aed7-dc6f5example",
            "Status": "SUCCESSFUL",
            "Type": "UPDATE_DOMAIN_CONTACT",
            "SubmittedDate": 1547501003.41
        }
    ]
}
```

Die Ausgabe umfasst alle Operationen, die eine Vorgangs-ID zurückgeben und die Sie für alle Domains ausgeführt haben, die Sie jemals mit dem AWS Girokonto registriert haben. Wenn Sie nur die Operationen abrufen möchten, die Sie nach einem bestimmten Datum eingereicht haben, können Sie den `submitted-since`-Parameter einbeziehen und ein Datum im Unix-Format und in koordinierter Weltzeit (UTC) angeben. Der folgende Befehl ruft den Status aller Operationen ab, die am 1. Januar 2020 nach 12:00 Uhr UTC gesendet wurden.

```
aws route53domains list-operations \
    --submitted-since 1577836800
```

- Einzelheiten zur API finden Sie ListOperationsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listOperations(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           localDateTime = localDateTime.minusYears(1);
           Instant myTime = localDateTime.toInstant(zoneOffset);

           ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                   .submittedSince(myTime)
                   .build();

           ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
           listRes.stream()
                   .flatMap(r -> r.operations().stream())
                   .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                           " Status: " + content.statusAsString() +
                           " Date: " + content.submittedDate()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListOperationsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListOperationsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListPrices`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListPrices` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }
```

- Einzelheiten zur API finden Sie ListPricesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listPrices(Route53DomainsClient route53DomainsClient,
String domainType) {
       try {
           ListPricesRequest pricesRequest = ListPricesRequest.builder()
                   .tld(domainType)
                   .build();

           ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
           listRes.stream()
                   .flatMap(r -> r.prices().stream())
                   .forEach(content -> System.out.println(" Name: " +
content.name() +
                           " Registration: " +
content.registrationPrice().price() + " "
                           + content.registrationPrice().currency() +
                           " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie ListPricesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListPricesin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`RegisterDomain`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterDomain` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain
 registration.</param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
 int duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant,
 and tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
```

```
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
 {domainName}");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie RegisterDomainin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So registrieren Sie eine Domain

Der folgende `register-domain` Befehl registriert eine Domain und ruft alle Parameterwerte aus einer JSON-formatted Datei ab.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains register-domain \
    --region us-east-1 \
    --cli-input-json file://register-domain.json
```

Inhalt von `register-domain.json`:

```
{
    "DomainName": "example.com",
    "DurationInYears": 1,
    "AutoRenew": true,
    "AdminContact": {
```

```
        "FirstName": "Martha",
        "LastName": "Rivera",
        "ContactType": "PERSON",
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "mrivera@example.com"
    },
    "RegistrantContact": {
        "FirstName": "Li",
        "LastName": "Juan",
        "ContactType": "PERSON",
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "ljuan@example.com"
    },
    "TechContact": {
        "FirstName": "Mateo",
        "LastName": "Jackson",
        "ContactType": "PERSON",
        "OrganizationName": "Example",
        "AddressLine1": "1 Main Street",
        "City": "Anytown",
        "State": "WA",
        "CountryCode": "US",
        "ZipCode": "98101",
        "PhoneNumber": "+1.8005551212",
        "Email": "mjackson@example.com"
    },
    "PrivacyProtectAdminContact": true,
    "PrivacyProtectRegistrantContact": true,
    "PrivacyProtectTechContact": true
}
```

Ausgabe:

```
{
    "OperationId": "b114c44a-9330-47d1-a6e8-a0b11example"
}
```

Sie können `get-operation-detail` ausführen, um zu bestätigen, dass der Vorgang erfolgreich war. Weitere Informationen finden Sie unter get-operation-detail.

Weitere Informationen finden Sie unter Registrieren einer neuen Domain im Entwicklerhandbuch zu Amazon Route 53.

Informationen darüber, für welche Top-Level-Domains (TLDs) Werte erforderlich sind `ExtraParams` und welche Werte gültig sind, finden Sie ExtraParamin der Amazon Route 53 API-Referenz.

- Einzelheiten zur API finden Sie RegisterDomainin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
           String domainSuggestion,
           String phoneNumber,
           String email,
           String firstName,
           String lastName,
           String city) {

       try {
           ContactDetail contactDetail = ContactDetail.builder()
                   .contactType(ContactType.COMPANY)
```

```
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie RegisterDomainin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- Einzelheiten zur API finden Sie RegisterDomainin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ViewBilling`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ViewBilling` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate,
 DateTime endDate)
```

```
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie ViewBillingin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um Abrechnungsinformationen für die Gebühren für die Domainregistrierung für das AWS Girokonto zu erhalten

Mit dem folgenden `view-billing`-Befehl werden alle domainbezogenen Abrechnungsdatensätze für das aktuelle Konto für den Zeitraum vom 1. Januar 2018 (1514764800 in Unix-Zeit) bis Mitternacht am 31. Dezember 2019 (1577836800 in Unix-Zeit) zurückgegeben.

Dieser Befehl wird nur in der `us-east-1`-Region ausgeführt. Wenn Ihre Standardregion auf `us-east-1` gesetzt ist, können Sie den `region`-Parameter weglassen.

```
aws route53domains view-billing \
    --region us-east-1 \
    --start-time 1514764800 \
    --end-time 1577836800
```

Ausgabe:

```
{
    "BillingRecords": [
        {
            "DomainName": "example.com",
            "Operation": "RENEW_DOMAIN",
            "InvoiceId": "149962827",
            "BillDate": 1536618063.181,
            "Price": 12.0
        },
        {
            "DomainName": "example.com",
            "Operation": "RENEW_DOMAIN",
            "InvoiceId": "290913289",
            "BillDate": 1568162630.884,
            "Price": 12.0
        }
    ]
}
```

Weitere Informationen finden Sie ViewBillingin der Amazon Route 53 API-Referenz.

- Einzelheiten zur API finden Sie ViewBillingunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listBillingRecords(Route53DomainsClient
 route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
```

```
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
 route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
 content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ViewBillingin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
```

```
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ViewBillingin der API-Referenz zum AWS SDK für Kotlin.

Codebeispiele für Amazon S3 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Simple Storage Service mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-S3-Benutzerhandbuch – Weitere Informationen zu Amazon S3.
- API-Referenz für Amazon S3 – Details zu allen verfügbaren Amazon-S3-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon S3 AWS SDKs
  - Hello Amazon S3
  - Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK
  - Aktionen für Amazon S3 mit AWS SDKs
    - Verwenden Sie AbortMultipartUploadmit einem AWS SDK oder CLI
    - Verwenden Sie mit einem CompleteMultipartUpload AWS SDK oder CLI
    - Verwenden Sie CopyObjectmit einem AWS SDK oder CLI
    - Verwenden Sie CreateBucketmit einem AWS SDK oder CLI
    - Verwenden Sie CreateMultiRegionAccessPointmit einem AWS SDK
    - Verwenden Sie CreateMultipartUploadmit einem AWS SDK oder CLI
    - Verwenden Sie CreatePresignedPostmit einem AWS SDK
    - Verwenden Sie DeleteBucketmit einem AWS SDK oder CLI
    - Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI
    - Verwenden Sie mit einem DeleteBucketCors AWS SDK oder CLI
    - Verwendung von DeleteBucketEncryption mit einer CLI
    - Verwendung von DeleteBucketInventoryConfiguration mit einer CLI
    - Verwenden Sie mit einem DeleteBucketLifecycle AWS SDK oder CLI
    - Verwendung von DeleteBucketMetricsConfiguration mit einer CLI
    - Verwenden Sie mit einem DeleteBucketPolicy AWS SDK oder CLI
    - Verwendung von DeleteBucketReplication mit einer CLI
    - Verwendung von DeleteBucketTagging mit einer CLI
    - Verwenden Sie mit einem DeleteBucketWebsite AWS SDK oder CLI

- Verwenden Sie mit einem DeleteObject AWS SDK oder CLI
- Verwendung von DeleteObjectTagging mit einer CLI
- Verwenden Sie mit einem DeleteObjects AWS SDK oder CLI
- Verwendung von DeletePublicAccessBlock mit einer CLI
- Verwendung von GetBucketAccelerateConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketAcl AWS SDK oder CLI
- Verwendung von GetBucketAnalyticsConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketCors AWS SDK oder CLI
- Verwenden Sie GetBucketEncryptionmit einem AWS SDK oder CLI
- Verwendung von GetBucketInventoryConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie GetBucketLocationmit einem AWS SDK oder CLI
- Verwendung von GetBucketLogging mit einer CLI
- Verwendung von GetBucketMetricsConfiguration mit einer CLI
- Verwendung von GetBucketNotification mit einer CLI
- Verwenden Sie mit einem GetBucketPolicy AWS SDK oder CLI
- Verwendung von GetBucketPolicyStatus mit einer CLI
- Verwenden Sie mit einem GetBucketReplication AWS SDK oder CLI
- Verwendung von GetBucketRequestPayment mit einer CLI
- Verwendung von GetBucketTagging mit einer CLI
- Verwendung von GetBucketVersioning mit einer CLI
- Verwenden Sie mit einem GetBucketWebsite AWS SDK oder CLI
- Verwenden Sie mit einem GetObject AWS SDK oder CLI
- Verwenden Sie GetObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectRetentionmit einem AWS SDK oder CLI
- Verwendung von GetObjectTagging mit einer CLI

- Verwendung von GetPublicAccessBlock mit einer CLI

- Verwenden Sie mit einem HeadBucket AWS SDK oder CLI
- Verwenden Sie HeadObjectmit einem AWS SDK oder CLI
- Verwendung von ListBucketAnalyticsConfigurations mit einer CLI
- Verwendung von ListBucketInventoryConfigurations mit einer CLI
- Verwenden Sie mit einem ListBuckets AWS SDK oder CLI
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK oder CLI
- Verwenden Sie ListObjectVersionsmit einem AWS SDK oder CLI
- Verwendung von ListObjects mit einer CLI
- Verwenden Sie V2 mit ListObjects einem AWS SDK oder CLI
- Verwenden Sie PutBucketAccelerateConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketAcl AWS SDK oder CLI
- Verwenden Sie PutBucketCorsmit einem AWS SDK oder CLI
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketLoggingmit einem AWS SDK oder CLI
- Verwendung von PutBucketNotification mit einer CLI
- Verwenden Sie mit einem PutBucketNotificationConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI
- Verwenden Sie PutBucketReplicationmit einem AWS SDK oder CLI
- Verwendung von PutBucketRequestPayment mit einer CLI
- Verwendung von PutBucketTagging mit einer CLI
- Verwenden Sie mit einem PutBucketVersioning AWS SDK oder CLI
- Verwenden Sie PutBucketWebsitemit einem AWS SDK oder CLI
- Verwenden Sie PutObjectmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectRetentionmit einem AWS SDK oder CLI
- Verwenden Sie RestoreObjectmit einem AWS SDK oder CLI

- Verwenden Sie SelectObjectContentmit einem AWS SDK oder CLI

    - Verwenden Sie UploadPartmit einem AWS SDK oder CLI
    - Verwenden Sie UploadPartCopymit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon S3 AWS SDKs
  - Prüfen, ob ein Bucket vorhanden ist
  - Prüfen Sie, ob ein Objekt existiert
  - Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK
  - Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK
  - Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
  - Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK
  - Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
  - Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK
  - Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK
  - Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
  - Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK
  - Erkennen Sie Gesichter in einem Bild mit einem AWS SDK
  - Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
  - Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
  - Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service (Amazon S3)
  - Herunterladen aller Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis
  - Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK
  - Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK
  - Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers
  - Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK

  - Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK

- Erste Schritte mit Big-Data-Verarbeitungsclustern
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit der Textextraktion von Dokumenten
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit Objektspeicher
- Erste Schritte mit Query Analytics
- Arbeiten Sie mit den Objektsperrfunktionen von Amazon S3 mithilfe eines AWS SDK
- Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK
- Verwalten Sie Zugriffskontrolllisten (ACLs) für Amazon S3 S3-Buckets mithilfe eines AWS SDK
- Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK
- Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK
- Analysieren Sie Amazon S3 S3-URIs mit einem AWS SDK
- Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK
- Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK
- Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK
- Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK
- Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda
- Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK
- Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service (Amazon S3)-Bucket
- Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK
- Laden Sie einen Stream unbekannter Größe mit einem in ein Amazon S3 S3-Objekt hoch AWS SDK
- Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie ein AWS SDK
- Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK

- Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK
- Serverless-Beispiele für Amazon S3
  - Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Grundlegende Beispiele für die Verwendung von Amazon S3 AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Storage Service mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon S3
- Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK
- Aktionen für Amazon S3 mit AWS SDKs
  - Verwenden Sie AbortMultipartUploadmit einem AWS SDK oder CLI
  - Verwenden Sie mit einem CompleteMultipartUpload AWS SDK oder CLI
  - Verwenden Sie CopyObjectmit einem AWS SDK oder CLI
  - Verwenden Sie CreateBucketmit einem AWS SDK oder CLI
  - Verwenden Sie CreateMultiRegionAccessPointmit einem AWS SDK
  - Verwenden Sie CreateMultipartUploadmit einem AWS SDK oder CLI
  - Verwenden Sie CreatePresignedPostmit einem AWS SDK
  - Verwenden Sie DeleteBucketmit einem AWS SDK oder CLI
  - Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI
  - Verwenden Sie mit einem DeleteBucketCors AWS SDK oder CLI
  - Verwendung von DeleteBucketEncryption mit einer CLI
  - Verwendung von DeleteBucketInventoryConfiguration mit einer CLI
  - Verwenden Sie mit einem DeleteBucketLifecycle AWS SDK oder CLI
  - Verwendung von DeleteBucketMetricsConfiguration mit einer CLI
  - Verwenden Sie mit einem DeleteBucketPolicy AWS SDK oder CLI
  - Verwendung von DeleteBucketReplication mit einer CLI
  - Verwendung von DeleteBucketTagging mit einer CLI
  - Verwenden Sie mit einem DeleteBucketWebsite AWS SDK oder CLI
  - Verwenden Sie mit einem DeleteObject AWS SDK oder CLI

- Verwendung von DeleteObjectTagging mit einer CLI
- Verwenden Sie mit einem DeleteObjects AWS SDK oder CLI
- Verwendung von DeletePublicAccessBlock mit einer CLI
- Verwendung von GetBucketAccelerateConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketAcl AWS SDK oder CLI
- Verwendung von GetBucketAnalyticsConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketCors AWS SDK oder CLI
- Verwenden Sie GetBucketEncryptionmit einem AWS SDK oder CLI
- Verwendung von GetBucketInventoryConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie GetBucketLocationmit einem AWS SDK oder CLI
- Verwendung von GetBucketLogging mit einer CLI
- Verwendung von GetBucketMetricsConfiguration mit einer CLI
- Verwendung von GetBucketNotification mit einer CLI
- Verwenden Sie mit einem GetBucketPolicy AWS SDK oder CLI
- Verwendung von GetBucketPolicyStatus mit einer CLI
- Verwenden Sie mit einem GetBucketReplication AWS SDK oder CLI
- Verwendung von GetBucketRequestPayment mit einer CLI
- Verwendung von GetBucketTagging mit einer CLI
- Verwendung von GetBucketVersioning mit einer CLI
- Verwenden Sie mit einem GetBucketWebsite AWS SDK oder CLI
- Verwenden Sie mit einem GetObject AWS SDK oder CLI
- Verwenden Sie GetObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectRetentionmit einem AWS SDK oder CLI
- Verwendung von GetObjectTagging mit einer CLI
- Verwendung von GetPublicAccessBlock mit einer CLI

- Verwenden Sie mit einem HeadBucket AWS SDK oder CLI

- Verwenden Sie HeadObjectmit einem AWS SDK oder CLI
- Verwendung von ListBucketAnalyticsConfigurations mit einer CLI
- Verwendung von ListBucketInventoryConfigurations mit einer CLI
- Verwenden Sie mit einem ListBuckets AWS SDK oder CLI
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK oder CLI
- Verwenden Sie ListObjectVersionsmit einem AWS SDK oder CLI
- Verwendung von ListObjects mit einer CLI
- Verwenden Sie V2 mit ListObjects einem AWS SDK oder CLI
- Verwenden Sie PutBucketAccelerateConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketAcl AWS SDK oder CLI
- Verwenden Sie PutBucketCorsmit einem AWS SDK oder CLI
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketLoggingmit einem AWS SDK oder CLI
- Verwendung von PutBucketNotification mit einer CLI
- Verwenden Sie mit einem PutBucketNotificationConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI
- Verwenden Sie PutBucketReplicationmit einem AWS SDK oder CLI
- Verwendung von PutBucketRequestPayment mit einer CLI
- Verwendung von PutBucketTagging mit einer CLI
- Verwenden Sie mit einem PutBucketVersioning AWS SDK oder CLI
- Verwenden Sie PutBucketWebsitemit einem AWS SDK oder CLI
- Verwenden Sie PutObjectmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectRetentionmit einem AWS SDK oder CLI
- Verwenden Sie RestoreObjectmit einem AWS SDK oder CLI
- Verwenden Sie SelectObjectContentmit einem AWS SDK oder CLI

- Verwenden Sie UploadPartmit einem AWS SDK oder CLI

- Verwenden Sie UploadPartCopymit einem AWS SDK oder CLI

Hello Amazon S3

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon S3 beginnen.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Hello Amazon Simple Storage Service
// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();
```

```
            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
            {
                Console.WriteLine($"Found {buckets.Count} S3 buckets:");
                Console.WriteLine();

                foreach (var bucket in buckets)
                {
                    Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                    Console.WriteLine($"  Creation Date:
 {bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No S3 buckets found in your account.");
            }

            Console.WriteLine("Hello S3 completed successfully.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"S3 service error occurred: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list S3 buckets. Here's why:
 {ex.Message}");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_s3.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Einzelheiten zur API finden Sie ListBucketsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "errors"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  var ae smithy.APIError
```

```
  if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
   fmt.Println("You don't have permission to list buckets for this account.")
  } else {
   fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  }
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
```

```
      // Something besides S3 failed.
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
```

```
def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50,  # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")


if __name__ == "__main__":
    hello_s3()
```

- Einzelheiten zur API finden Sie ListBucketsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
```

```
  manager.list_buckets
end
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
 object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
 present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
 bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
```

```
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }

    // Prepare a ByteStream around the file, and upload the object using that
ByteStream.
    let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
        .await
        .map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to create bytestream for {filepath:?} ({err:?})"
            ))
        })?;
    let resp = client
        .put_object()
        .bucket(bucket)
        .key(key)
        .body(body)
        .send()
        .await?;

    println!(
        "Upload success. Version: {:?}",
        resp.version_id()
            .expect("S3 Object upload missing version ID")
    );
```

```
    // Retrieve the just-uploaded object.
    let resp = client.get_object().bucket(bucket).key(key).send().await?;
    println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
    println!("version: {}", resp.version_id().unwrap_or("(missing)"));

    Ok(())
}
```

S3ExampleError Versorgungsunternehmen.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}
```

```
impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- Einzelheiten zur API finden Sie ListBucketsin der API-Referenz zum AWS SDK für Rust.

Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
```

```
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole()))
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
```

```
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
 scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;

        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }

            var success = await s3Wrapper.CreateBucketAsync(bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                // Get the local path and filename for the file to upload.
```

```
                while (string.IsNullOrEmpty(filePath))
                {
                    Console.Write("Please enter the path and filename of the file
 to upload: ");
                    filePath = Console.ReadLine();

                    // Confirm that the file exists on the local computer.
                    if (!File.Exists(filePath))
                    {
                        Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                        filePath = string.Empty;
                    }
                }
            }
            else
            {
                // Use the public variable if set, otherwise create a temp file
                if (!string.IsNullOrEmpty(TempFilePath))
                {
                    filePath = TempFilePath;
                    Console.WriteLine($"Using provided test file: {filePath}");
                }
                else
                {
                    // Create a temporary test file for non-interactive mode
                    filePath = Path.GetTempFileName();
                    var testContent = "This is a test file for S3 basics
 scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
 UTC");
                    await File.WriteAllTextAsync(filePath, testContent);
                    Console.WriteLine($"Created temporary test file:
 {filePath}");
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
 filePath);

            if (success)
            {
```

```
                Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set up download path
            string downloadPath = string.Empty;

            if (IsInteractive)
            {
                // Now get a new location where we can save the file.
                while (string.IsNullOrEmpty(downloadPath))
                {
                    // First get the path to which the file will be downloaded.
                    Console.Write("Please enter the path where the file will be
downloaded: ");
                    downloadPath = Console.ReadLine();

                    // Confirm that the file doesn't already exist on the local
computer.
                    if (File.Exists($"{downloadPath}\\{keyName}"))
                    {
                        Console.WriteLine($"Sorry, the file already exists in
that location.\n");
                        downloadPath = string.Empty;
                    }
                }
            }
            else
            {
                downloadPath = Path.GetTempPath();
                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                }

                Console.WriteLine($"Using download path: {downloadPath}");
            }

            // Download an object from a bucket.
```

```
            success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            if (IsInteractive)
            {
                while (string.IsNullOrEmpty(folderName))
                {
                    Console.Write("Please enter the name of the folder to copy
your object to: ");
                    folderName = Console.ReadLine();
                }
            }
            else
            {
                folderName = "test-folder";
                Console.WriteLine($"Using folder name: {folderName}");
            }

            await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await s3Wrapper.ListBucketContentsAsync(bucketName);

            // Delete the contents of the bucket.
            if (IsInteractive)
            {
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
                _ = Console.ReadLine();
            }
```

```
            var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
            if (deleteContentsSuccess)
            {
                Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
            }

            if (IsInteractive)
            {
                // Deleting the bucket too quickly after separately deleting its
contents can
                // cause an error that the bucket isn't empty. To delete contents
and bucket in one
                // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
                _ = Console.ReadLine();
            }
            else
            {
                // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
                Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
                await Task.Delay(2000);
            }

            // Delete the bucket.
            var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
            if (deleteSuccess)
            {
                Console.WriteLine($"Successfully deleted {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
            }
```

```
            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (File.Exists(filePath))
                    {
                        File.Delete(filePath);
                        Console.WriteLine("Cleaned up temporary test file.");
                    }

                    var downloadFile = Path.Combine(downloadPath, keyName);
                    if (File.Exists(downloadFile))
                    {
                        File.Delete(downloadFile);
                        Console.WriteLine("Cleaned up downloaded test file.");
                    }
                }
                catch (Exception ex)
                {
                    scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
                }
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

            // Clean up on error - delete bucket if it exists
            try
            {
                if (!string.IsNullOrEmpty(bucketName))
                {
                    await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                    await s3Wrapper.DeleteBucketAsync(bucketName);
                }
            }
            catch (Exception cleanupEx)
            {
                scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
            }
```

```
            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                    {
                        File.Delete(filePath);
                    }
                }
                catch (Exception fileCleanupEx)
                {
                    scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
 temporary files during error handling.");
                }
            }

            throw;
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
```

```
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }


    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
```

```
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
```

```
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
 _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
```

```
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
```

```
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' getting list of objects.");
            return null;
        }
    }



    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
```

```
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
 o.Key }).ToList(),
                });
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
```

```
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
```

```
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon S3 getting started demo."
  echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
 Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
  local bucket_name
  bucket_name=$(generate_random_name "$bucket_name_prefix")

  local region_code
  region_code=$(aws configure get region)

  if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
  else
    errecho "The bucket failed to create. This demo will exit."
    return 1
  fi

  local file_name
  while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
      echo "Could not find file $file_name. Are you sure it exists?"
      file_name=""
    fi
  done

  local key
  key="$(basename "$file_name")"

  local result=0
```

```
 if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
   echo "Uploaded file $file_name into bucket $bucket_name with key $key."
 else
   result=1
 fi

 local destination_file
 destination_file="$file_name.download"
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi

 if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
   local to_key
   to_key="demo/$key"
   if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
     echo "Copied $key in the bucket $bucket_name to the  $to_key."
   else
     result=1
   fi
 fi

 local bucket_items
 bucket_items=$(list_items_in_bucket "$bucket_name")

 # shellcheck disable=SC2181
 if [[ $? -ne 0 ]]; then
   result=1
 fi

 echo "Your bucket contains the following items."
 echo -e "Name\t\tSize"
 echo "$bucket_items"

 if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
```

```
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

Die in diesem Szenario verwendeten Amazon-S3-Funktionen.

```
###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
```

```
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
```

```
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"
```

```
  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

##############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
```

```
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param bucketNamePrefix: A prefix for a bucket name.
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
        const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());
```

```
   // 1. Create a bucket.
   {
       Aws::S3::Model::CreateBucketRequest request;
       request.SetBucket(bucketName);

       if (clientConfig.region != Aws::Region::US_EAST_1) {
           Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
           createBucketConfiguration.WithLocationConstraint(

Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                           clientConfig.region));
           request.WithCreateBucketConfiguration(createBucketConfiguration);
       }

       Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

       if (!outcome.IsSuccess()) {
           const Aws::S3::S3Error &err = outcome.GetError();
           std::cerr << "Error: createBucket: " <<
                     err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
           return false;
       } else {
           std::cout << "Created the bucket, '" << bucketName <<
                     "', in the region, '" << clientConfig.region << "'." <<
std::endl;
       }
   }

   // 2. Upload a local file to the bucket.
   Aws::String key = "key-for-test";
   {
       Aws::S3::Model::PutObjectRequest request;
       request.SetBucket(bucketName);
       request.SetKey(key);

       std::shared_ptr<Aws::FStream> input_data =
               Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                             uploadFilePath,
                                             std::ios_base::in |
                                             std::ios_base::binary);
```

```
        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;
```

```
            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;
```

```
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
 objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
 bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
```

```
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Struktur, die die vom Szenario verwendete Bucket- und Objektaktionen umschließt.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
 error) {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
```

```
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
```

```
    ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
   }
  }
  return err
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
 string, objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
```

```
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
   err = noKey
  } else {
   log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  }
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
```

```
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
 objectKey string, folderName string) error {
 objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(objectDest),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, bucketName)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectDest)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
  }
 }
 return err
```

```
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
```

```
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
 return objects, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
 objectKeys []string) error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
 })
 if err != nil || len(output.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucketName)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
  } else if len(output.Errors) > 0 {
   for _, outErr := range output.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *output.Errors[0].Message)
  }
 } else {
  for _, delObjs := range output.Deleted {
```

```
    err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
 time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
  _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucketName)})
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   } else {
    log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
   }
  } else {
   err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucketName)
   } else {
    log.Printf("Deleted %s.\n", bucketName)
   }
  }
  return err
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie mit S3-Buckets und Objekten arbeiten.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
```

```
    log.Println("Something went wrong with the demo.")
    _, isMock := questioner.(*demotools.MockQuestioner)
    if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
     log.Println(r)
    }
   }
  }()

  log.Println(strings.Repeat("-", 88))
  log.Println("Welcome to the Amazon S3 getting started demo.")
  log.Println(strings.Repeat("-", 88))

  s3Client := s3.NewFromConfig(sdkConfig)
  bucketBasics := actions.BucketBasics{S3Client: s3Client}

  count := 10
  log.Printf("Let's list up to %v buckets for your account:", count)
  buckets, err := bucketBasics.ListBuckets(ctx)
  if err != nil {
   panic(err)
  }
  if len(buckets) == 0 {
   log.Println("You don't have any buckets!")
  } else {
   if count > len(buckets) {
    count = len(buckets)
   }
   for _, bucket := range buckets[:count] {
    log.Printf("\t%v\n", *bucket.Name)
   }
  }

  bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
  bucket:",
   demotools.NotEmpty{})
  bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
  if err != nil {
   panic(err)
  }
  if !bucketExists {
   err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
   if err != nil {
    panic(err)
```

```
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
```

```
 for _, object := range objects {
  objKeys = append(objKeys, *object.Key)
  log.Printf("\t%v\n", *object.Key)
 }
 log.Println(strings.Repeat("-", 88))

 if questioner.AskBool("Do you want to delete your bucket and all of its "+
  "contents? (y/n)", "y") {
  log.Println("Deleting objects.")
  err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(ctx, bucketName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting downloaded file %v.\n", downloadFileName)
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ein Beispielszenario.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
```

```
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final Logger logger =
 LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3
 bucket.
                objectPath - The full file path of the object within the S3
 bucket (e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be
 downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
 copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
```

```
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to
store and retrieve
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets
and objects as
            part of their application deployment pipelines. With S3, teams can
focus on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for
this service.
            Let's get started...
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
```

```
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}'
and key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
```

```
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse, die die Operationen enthält.

```
public class S3Actions {

    private static final Logger logger =
 LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API
call timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
```

```
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }


    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
     * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
```

```
        });
    }


    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the key (or name) of the S3 object to retrieve
     * @param path       the local file path where the object's bytes will be
written
     * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
     */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }


    /**
     * Asynchronously lists all objects in the specified S3 bucket.
```

```
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have
been listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }


    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
```

```
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key)
{
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
```

```
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
```

```
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has
been deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
```

```
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during
delete", ex);
            }
        });

        return response.thenApply(r -> null);
    }


    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();
```

```
        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
 UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
 exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie zunächst alle erforderlichen Module.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
```

```
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

Die vorherigen Importe verweisen auf einige hilfreiche Dienstprogramme. Diese Dienstprogramme befinden sich lokal in dem GitHub Repository, auf das am Anfang dieses Abschnitts verwiesen wird. Zu Ihrer Information finden Sie im Folgenden Implementierungen dieser Dienstprogramme.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
```

```
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };

  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

Objekte werden in „Buckets“ gespeichert. Definieren wir eine Funktion zum Erstellen eines neuen Buckets.

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

Buckets enthalten „Objekte“. Diese Funktion lädt den Inhalt eines Verzeichnisses als Objekte in Ihren Bucket hoch.

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (const file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

Überprüfen Sie nach dem Hochladen von Objekten, ob sie korrekt hochgeladen wurden. Sie können dafür verwenden ListObjects . Sie werden die Eigenschaft „Key“ verwenden, die Antwort enthält jedoch noch weitere nützliche Eigenschaften.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

Gelegentlich möchten Sie vielleicht ein Objekt von einem Bucket in einen anderen kopieren. Verwenden Sie dafür den CopyObject Befehl.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });

      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};
```

Es gibt keine SDK-Methode zum Abrufen mehrerer Objekte aus einem Bucket. Stattdessen erstellen Sie eine Liste von Objekten, die Sie herunterladen und durchlaufen können.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

Es ist Zeit, Ihre Ressourcen zu bereinigen. Ein Bucket muss leer sein, bevor er gelöscht werden kann. Mit diesen beiden Funktionen leeren und löschen Sie den Bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
```

```
  console.log(`${bucketName} deleted successfully.\n`);
};
```

Die Funktion „main“ fasst alles zusammen. Wenn Sie diese Datei direkt ausführen, wird die Funktion „main“ aufgerufen.

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
```

```
    console.error(err);
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Kotlin

SDK für Kotlin

> **Note**
>
> Da ist noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
```

```
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
```

```
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

```
suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
```

```
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
```

```
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
```

```
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import io
import os
import uuid
```

```
import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)

    bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")
```

```
    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
```

```
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
 destination
```

```
 # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
 def upload_file(bucket)
   File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
   s3_object = bucket.object(File.basename('demo.txt'))
   s3_object.upload_file('demo.txt')
   puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
 rescue Aws::Errors::ServiceError => e
   puts("Couldn't upload file demo.txt to #{bucket.name}.")
   puts("\t#{e.code}: #{e.message}")
   raise
 else
   s3_object
 end

 # Downloads an Amazon S3 object to a file.
 #
 # @param s3_object [Aws::S3::Object] The object to download.
 def download_file(s3_object)
   puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
   answer = gets.chomp.downcase
   if answer == 'y'
     puts('Enter a name for the downloaded file: ')
     file_name = gets.chomp
     s3_object.download_file(file_name)
     puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
   end
 rescue Aws::Errors::ServiceError => e
   puts("Couldn't download #{s3_object.key}.")
   puts("\t#{e.code}: #{e.message}")
   raise
 end

 # Copies an Amazon S3 object to a subfolder within the same bucket.
 #
 # @param source_object [Aws::S3::Object] The source object to copy.
 # @return [Aws::S3::Object, nil] The destination object.
 def copy_object(source_object)
   dest_object = nil
   puts("\nDo you want to copy #{source_object.key} to a subfolder in your
bucket (y/n)? ")
   answer = gets.chomp.downcase
   if answer == 'y'
```

```
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

```
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die binäre Crate-Datei, die das Szenario ausführt.

```
#![allow(clippy::result_large_err)]

//!  Purpose
//!  Shows how to use the AWS SDK for Rust to get started using
//!  Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
 into and out of it,
//!  and delete all resources at the end of the demo.
//!
//!  This example follows the steps in "Getting started with Amazon S3" in the
 Amazon S3
//!  user guide.
//!  - https://docs.aws.amazon.com/AmazonS3/latest/userguide/
GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
```

```
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
 &key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
 &key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
 &target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

Allgemeine Aktionen, die im Szenario verwendet werden.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
```

```
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
}

/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
 with etag {}",
        response
```

```
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}

pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
 Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
```

```
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

    " Create an Amazon Simple Storage Service (Amazon S3) bucket. "
    TRY.
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
```

```
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.


    "Upload an object to an S3 bucket."
    TRY.
        "Get contents of file from application server."
        DATA lv_file_content TYPE xstring.
        OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
        READ DATASET iv_key INTO lv_file_content.
        CLOSE DATASET iv_key.

        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
            iv_body = lv_file_content ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                    iv_bucket = iv_bucket_name
                    iv_key = iv_key ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

```
    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
```

```
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
```

```
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
         //   configuration.region = "us-east-2" // Uncomment this to set the
 region programmatically.
            client = S3Client(config: configuration)
        }
        catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            throw error
        }
    }


    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the
 locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
```

```
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
```

```
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
 async throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

```
    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
```

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
 read data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }


    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
```

```
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
"ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
```

```
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String
```

```
    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = try await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
```

```
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
 destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket:
 bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
 \(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
```

```
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen für Amazon S3 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon S3 S3-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-S3-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon S3 AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Simple Storage Service.

Beispiele

- Verwenden Sie AbortMultipartUploadmit einem AWS SDK oder CLI
- Verwenden Sie mit einem CompleteMultipartUpload AWS SDK oder CLI
- Verwenden Sie CopyObjectmit einem AWS SDK oder CLI
- Verwenden Sie CreateBucketmit einem AWS SDK oder CLI

- Verwenden Sie CreateMultiRegionAccessPointmit einem AWS SDK
- Verwenden Sie CreateMultipartUploadmit einem AWS SDK oder CLI
- Verwenden Sie CreatePresignedPostmit einem AWS SDK
- Verwenden Sie DeleteBucketmit einem AWS SDK oder CLI
- Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI
- Verwenden Sie mit einem DeleteBucketCors AWS SDK oder CLI
- Verwendung von DeleteBucketEncryption mit einer CLI
- Verwendung von DeleteBucketInventoryConfiguration mit einer CLI
- Verwenden Sie mit einem DeleteBucketLifecycle AWS SDK oder CLI
- Verwendung von DeleteBucketMetricsConfiguration mit einer CLI
- Verwenden Sie mit einem DeleteBucketPolicy AWS SDK oder CLI
- Verwendung von DeleteBucketReplication mit einer CLI
- Verwendung von DeleteBucketTagging mit einer CLI
- Verwenden Sie mit einem DeleteBucketWebsite AWS SDK oder CLI
- Verwenden Sie mit einem DeleteObject AWS SDK oder CLI
- Verwendung von DeleteObjectTagging mit einer CLI
- Verwenden Sie mit einem DeleteObjects AWS SDK oder CLI
- Verwendung von DeletePublicAccessBlock mit einer CLI
- Verwendung von GetBucketAccelerateConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketAcl AWS SDK oder CLI
- Verwendung von GetBucketAnalyticsConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketCors AWS SDK oder CLI
- Verwenden Sie GetBucketEncryptionmit einem AWS SDK oder CLI
- Verwendung von GetBucketInventoryConfiguration mit einer CLI
- Verwenden Sie mit einem GetBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie GetBucketLocationmit einem AWS SDK oder CLI
- Verwendung von GetBucketLogging mit einer CLI
- Verwendung von GetBucketMetricsConfiguration mit einer CLI
- Verwendung von GetBucketNotification mit einer CLI
- Verwenden Sie mit einem GetBucketPolicy AWS SDK oder CLI

- Verwendung von GetBucketPolicyStatus mit einer CLI
- Verwenden Sie mit einem GetBucketReplication AWS SDK oder CLI
- Verwendung von GetBucketRequestPayment mit einer CLI
- Verwendung von GetBucketTagging mit einer CLI
- Verwendung von GetBucketVersioning mit einer CLI
- Verwenden Sie mit einem GetBucketWebsite AWS SDK oder CLI
- Verwenden Sie mit einem GetObject AWS SDK oder CLI
- Verwenden Sie GetObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie GetObjectRetentionmit einem AWS SDK oder CLI
- Verwendung von GetObjectTagging mit einer CLI
- Verwendung von GetPublicAccessBlock mit einer CLI
- Verwenden Sie mit einem HeadBucket AWS SDK oder CLI
- Verwenden Sie HeadObjectmit einem AWS SDK oder CLI
- Verwendung von ListBucketAnalyticsConfigurations mit einer CLI
- Verwendung von ListBucketInventoryConfigurations mit einer CLI
- Verwenden Sie mit einem ListBuckets AWS SDK oder CLI
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK oder CLI
- Verwenden Sie ListObjectVersionsmit einem AWS SDK oder CLI
- Verwendung von ListObjects mit einer CLI
- Verwenden Sie V2 mit ListObjects einem AWS SDK oder CLI
- Verwenden Sie PutBucketAccelerateConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketAcl AWS SDK oder CLI
- Verwenden Sie PutBucketCorsmit einem AWS SDK oder CLI
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK oder CLI
- Verwenden Sie mit einem PutBucketLifecycleConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketLoggingmit einem AWS SDK oder CLI
- Verwendung von PutBucketNotification mit einer CLI

- Verwenden Sie mit einem PutBucketNotificationConfiguration AWS SDK oder CLI
- Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI
- Verwenden Sie PutBucketReplicationmit einem AWS SDK oder CLI
- Verwendung von PutBucketRequestPayment mit einer CLI
- Verwendung von PutBucketTagging mit einer CLI
- Verwenden Sie mit einem PutBucketVersioning AWS SDK oder CLI
- Verwenden Sie PutBucketWebsitemit einem AWS SDK oder CLI
- Verwenden Sie PutObjectmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectAclmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLegalHoldmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectLockConfigurationmit einem AWS SDK oder CLI
- Verwenden Sie PutObjectRetentionmit einem AWS SDK oder CLI
- Verwenden Sie RestoreObjectmit einem AWS SDK oder CLI
- Verwenden Sie SelectObjectContentmit einem AWS SDK oder CLI
- Verwenden Sie UploadPartmit einem AWS SDK oder CLI
- Verwenden Sie UploadPartCopymit einem AWS SDK oder CLI

Verwenden Sie **`AbortMultipartUpload`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AbortMultipartUpload` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Löschen von unvollständigen mehrteiligen Uploads
- Arbeiten mit Amazon-S3-Objektintegrität

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AbortMultipartUploadin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So brechen Sie den angegebenen mehrteiligen Upload ab

Der folgende `abort-multipart-upload`-Befehl bricht einen mehrteiligen Upload für den Schlüssel `multipart/01` im Bucket `amzn-s3-demo-bucket` ab.

```
aws s3api abort-multipart-upload \
    --bucket amzn-s3-demo-bucket \
    --key multipart/01 \
    --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

Die für diesen Befehl benötigte Upload-ID wird von `create-multipart-upload` ausgegeben und kann auch mit `list-multipart-uploads` abgerufen werden.

- Einzelheiten zur API finden Sie AbortMultipartUploadin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
```

```
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
```

```
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
```

```
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
            }
        }
    }

    /**
     * Aborts incomplete multipart uploads older than the specified point in
time.
     *
     * @param bucketName  the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are
aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();
```

```
                AbortMultipartUploadResponse abortMultipartUploadResponse =
 s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
 (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
 aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
 {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }

    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
 days.
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules =
 List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
```

```
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }

    /************************
     Multipart upload methods
     ************************/

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
```

```
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
```

```
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
        new Thread(() -> {
            try {
```

```
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /*************************
     Resource handling methods
     ************************/

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }
```

```
    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
 PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
 Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}
```

- Einzelheiten zur API finden Sie AbortMultipartUploadin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl bricht mehrteilige Uploads ab, die vor mehr als 5 Tagen erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014 erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "Thursday, January 02, 2014"
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014, 10:45:37, erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "2014/01/02 10:45:37"
```

- Einzelheiten zur API finden Sie unter AbortMultipartUpload AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl bricht mehrteilige Uploads ab, die vor mehr als 5 Tagen erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014 erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "Thursday, January 02, 2014"
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014, 10:45:37, erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
 "2014/01/02 10:45:37"
```

- Einzelheiten zur API finden Sie unter AbortMultipartUpload AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CompleteMultipartUpload`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CompleteMultipartUpload` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erstellen einer mehrteiligen Kopie
- Verwenden der Prüfsummen
- Arbeiten mit Amazon-S3-Objektintegrität

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- Einzelheiten zur API finden Sie CompleteMultipartUploadin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl schließt einen mehrteiligen Upload für den Schlüssel `multipart/01` im Bucket `amzn-s3-demo-bucket` ab:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

Die für diesen Befehl benötigte Upload-ID wird von `create-multipart-upload` ausgegeben und kann auch mit `list-multipart-uploads` abgerufen werden.

Die Option für mehrteilige Uploads im obigen Befehl erwartet eine JSON-Struktur, die die Teile des mehrteiligen Uploads beschreibt, die zu einer vollständigen Datei zusammengesetzt werden sollen. In diesem Beispiel wird das Präfix `file://` verwendet, um die JSON-Struktur aus einer Datei im lokalen Ordner mit dem Namen `mpustruct` zu laden.

mpustruct:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
```

```
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

Der ETag-Wert für jeden Teil, der hochgeladen wird, wird jedes Mal ausgegeben, wenn Sie einen Teil mit dem `upload-part`-Befehl hochladen. Er kann auch durch Aufrufen von `list-parts` abgerufen werden, indem die MD5-Prüfsumme für jeden Teils kalkuliert wird.

Ausgabe:

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "amzn-s3-demo-bucket",
    "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
}
```

- Einzelheiten zur API finden Sie CompleteMultipartUploadin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
```

```
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
```

```
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

- Einzelheiten zur API finden Sie CompleteMultipartUploadin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`CopyObject`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CopyObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Verschlüsselung
- Erste Schritte mit Objektspeicher
- Senden von bedingten Anforderungen

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```
    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
 destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
```

```
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response
```

```
  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                            const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
```

```
    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl kopiert ein Objekt von `bucket-1` nach `bucket-2`:

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

Ausgabe:

```
{
    "CopyObjectResult": {
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
```

```
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren eines Objekts mit einem S3Client.

```
    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
 a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
 exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
 String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
 getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
 copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

Verwenden Sie an S3TransferManager, um ein Objekt von einem Bucket in einen anderen zu kopieren. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
```

```
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- Einzelheiten zur API finden Sie CopyObjectunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie das Objekt.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/
${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
      console.error(
```

```
        `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the
 active tier.`,
      );
    } else {
      throw caught;
    }
  }
};
```

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
```

```
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
 "${sourceBucketName}":  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
```

```
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string, eTag: string }}
 */
```

```
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
 "${sourceBucketName}":  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
```

```
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Kopieren Sie das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Kopiert das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
 destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
```

```
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
```

```
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
```

```
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Einzelheiten zur API finden Sie CopyObjectin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einfache Kopie eines Objekts.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in denselben Bucket, jedoch mit einem neuen Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Beispiel 2: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in den Bucket „backup-files” mit dem Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Beispiel 3: Dieser Befehl lädt das Objekt „sample.txt” aus dem Bucket „test-files” in eine lokale Datei mit dem Namen „local-sample.txt” herunter.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder
 c:\downloads
```

Beispiel 5: Lädt alle Objekte, die mit dem angegebenen Schlüsselpräfix übereinstimmen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

- Einzelheiten zur API finden Sie unter CopyObject AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in denselben Bucket, jedoch mit einem neuen Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Beispiel 2: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in den Bucket „backup-files” mit dem Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Beispiel 3: Dieser Befehl lädt das Objekt „sample.txt” aus dem Bucket „test-files” in eine lokale Datei mit dem Namen „local-sample.txt” herunter.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder
 c:\downloads
```

Beispiel 5: Lädt alle Objekte, die mit dem angegebenen Schlüsselpräfix übereinstimmen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

Beispiel 6: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallel Downloads in eine lokale Datei herunter, um die Leistung bei großen Dateien zu verbessern.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt -UseMultipartDownload
```

Beispiel 7: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Teilgröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanforderungen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile
 local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -
PartSize 64MB
```

Beispiel 8: Lädt alle Objekte, die dem key prefix entsprechen, mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Beispiel 9: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um den Durchsatz in Netzwerken mit hoher Bandbreite zu maximieren.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile
 local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Beispiel 10: Lädt alle Objekte, die dem key prefix entsprechen, mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter, wobei einzelne Dateifehler überwunden werden, anstatt den Vorgang abzubrechen.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Einzelheiten zur API finden Sie unter CopyObject AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
```

```
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
 request.
```

```
        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Einzelheiten zur API finden Sie CopyObjectin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Kopieren Sie ein Objekt und fügen Sie dem Zielobjekt eine serverseitige Verschlüsselung hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
```

```
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
 with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- Einzelheiten zur API finden Sie CopyObjectin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_dest_bucket
          iv_key = iv_dest_object
          iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
        MESSAGE 'Object copied to another bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CopyObjectin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
```

```
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CopyObjectin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateBucket`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateBucket` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit Objektspeicher
- Verwalten großer Nachrichten mit S3
- Mit versionierten Objekten arbeiten

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
```

```
#                           create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
```

```
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Bucket

Im folgenden Beispiel für `create-bucket` wird ein Bucket mit dem Namen `amzn-s3-demo-bucket` erstellt:

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region us-east-1
```

Ausgabe:

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

Weitere Informationen finden Sie unter Erstellen eines Buckets im Amazon-S3-Benutzerhandbuch.

Beispiel 2: So erstellen Sie einen Bucket, bei dem der Besitzer erzwungen wird

Im folgenden Beispiel für `create-bucket` wird ein Bucket namens `amzn-s3-demo-bucket` erstellt, der die vom Bucket-Eigentümer erzwungene Einstellung für S3 Object Ownership verwendet.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

Ausgabe:

```
{
    "Location": "/amzn-s3-demo-bucket"
}
```

Weitere Informationen finden Sie unter Steuern des Eigentums an Objekten und Deaktivieren von ACLs im Benutzerhandbuch für Amazon S3.

Beispiel 3: So erstellen Sie einen Bucket außerhalb der Region „us-east“

Im folgenden Beispiel für `create-bucket` wird ein Bucket namens `amzn-s3-demo-bucket` in der Region `eu-west-1` erstellt. Für Regionen außerhalb von `us-east-1` müssen die entsprechenden `LocationConstraint` angegeben werden, um den Bucket in der gewünschten Region zu erstellen.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

Ausgabe:

```
{
    "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"
}
```

Weitere Informationen finden Sie unter Erstellen eines Buckets im Amazon-S3-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateBucketin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Buckets mit Standardkonfiguration.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
```

```
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
```

```
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

Erstellen eines Buckets mit Object Lock und warten, bis er existiert.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
```

```
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket.

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
               s3Waiter.waitUntilBucketExists(bucketRequestWait);
           return waiterResponseFuture.thenAccept(waiterResponse -> {
               waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                   logger.info(bucketName + " is ready");
               });
           });
```

```
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
 bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Bucket.

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
 constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
 names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
```

```
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- Einzelheiten zur API finden Sie CreateBucketin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket mit Standardeinstellungen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

Erstellen Sie einen versionierten Bucket mit einer Lebenszyklus-Konfiguration.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
```

```
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- Einzelheiten zur API finden Sie CreateBucketin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
```

```
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
```

```
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}
```

- Einzelheiten zur API finden Sie CreateBucketin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " determine our region from our session
```

```
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateBucketin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )
```

```
        // For regions other than "us-east-1", you must set the
 locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You
 may wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie CreateBucketin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`CreateMultiRegionAccessPoint`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateMultiRegionAccessPoint`.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Steuerungs-Client so, dass er eine Anfrage an die Region us-west-2 sendet.

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

Erstellen Sie den Multi-Region Access Point.

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
```

```
                        },
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Warten Sie, bis der Multi-Region Access Point verfügbar ist.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
```

```
            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- Einzelheiten zur API finden Sie CreateMultiRegionAccessPointin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateMultipartUpload`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateMultipartUpload` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Erstellen einer mehrteiligen Kopie
- Verwenden der Prüfsummen
- Arbeiten mit Amazon-S3-Objektintegrität

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return uploadID;
}
```

- Einzelheiten zur API finden Sie CreateMultipartUploadin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt einen mehrteiligen Upload im Bucket `amzn-s3-demo-bucket` mit dem Schlüssel `multipart/01`:

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
 'multipart/01'
```

Ausgabe:

```
{
    "Bucket": "amzn-s3-demo-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

Die fertige Datei in einem Ordner namens `multipart` im Bucket `amzn-s3-demo-bucket` wird `01` benannt. Speichern Sie die Upload-ID, den Schlüssel und den Bucket-Namen zur Verwendung mit dem `upload-part`-Befehl.

- Einzelheiten zur API finden Sie CreateMultipartUploadin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
```

```
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Einzelheiten zur API finden Sie CreateMultipartUploadin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`CreatePresignedPost`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreatePresignedPost`.

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte POST URL.

```
    /// <summary>
    /// Create a presigned POST URL with conditions.
    /// </summary>
    /// <param name="s3Client">The Amazon S3 client.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <param name="objectKey">The object key (path) where the uploaded file
 will be stored.</param>
    /// <param name="expires">When the presigned URL expires.</param>
    /// <param name="fields">Dictionary of fields to add to the form.</param>
    /// <param name="conditions">List of conditions to apply.</param>
    /// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
    public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
        IAmazonS3 s3Client,
        string bucketName,
        string objectKey,
        DateTime expires,
        Dictionary<string, string>? fields = null,
        List<S3PostCondition>? conditions = null)
    {
        var request = new CreatePresignedPostRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = expires
        };

        // Add custom fields if provided
        if (fields != null)
        {
            foreach (var field in fields)
            {
                request.Fields.Add(field.Key, field.Value);
            }
        }

        // Add conditions if provided
        if (conditions != null)
        {
            foreach (var condition in conditions)
            {
                request.Conditions.Add(condition);
```

```
            }
        }

        return await s3Client.CreatePresignedPostAsync(request);
    }
```

- Einzelheiten zur API finden Sie CreatePresignedPostin der AWS SDK für .NET API-Referenz.

Verwenden Sie **`DeleteBucket`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucket` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit Objektspeicher

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
    /// <returns>A boolean value that represents the success or failure of
```

```
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
```

```
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);
```

```
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht einen Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
```

```
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket
 deletion is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();
```

```
        CompletableFuture<DeleteBucketResponse> response =
 getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during
 bucket deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
```

```
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie einen leeren Bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket „test-files“ und löscht anschließend den Bucket. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass nicht leere Buckets nicht gelöscht werden können.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Einzelheiten zur API finden Sie unter DeleteBucket AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket „test-files“ und löscht anschließend den Bucket. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass nicht leere Buckets nicht gelöscht werden können.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Einzelheiten zur API finden Sie unter DeleteBucket AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- Einzelheiten zur API finden Sie DeleteBucketin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteBucketin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteBucketin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteBucketin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteBucketAnalyticsConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketAnalyticsConfiguration` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Analytikkonfiguration für einen Bucket

Im folgenden Beispiel für `delete-bucket-analytics-configuration` wird die Analytics-Konfiguration für den angegebenen Bucket und die angegebene ID entfernt.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteBucketAnalyticsConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen „testfilter” im angegebenen S3-Bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Einzelheiten zur API finden Sie unter DeleteBucketAnalyticsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen „testfilter” im angegebenen S3-Bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Einzelheiten zur API finden Sie unter DeleteBucketAnalyticsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteBucketCors`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketCors` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Cross-Origin Resource Sharing-Konfiguration aus einem Bucket mit dem Namen`amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3->deletebucketcors(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteBucketCorsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteBucketEncryption`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketEncryption` verwendet wird.

CLI

AWS CLI

So löschen Sie die serverseitige Verschlüsselungskonfiguration eines Buckets

Im folgenden Beispiel für `delete-bucket-encryption` wird die serverseitige Verschlüsselungskonfiguration des angegebenen Buckets gelöscht.

```
aws s3api delete-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteBucketEncryptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dadurch wird die für den angegebenen S3-Bucket aktivierte Verschlüsselung deaktiviert.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketEncryption AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dadurch wird die für den angegebenen S3-Bucket aktivierte Verschlüsselung deaktiviert.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketEncryption AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteBucketInventoryConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketInventoryConfiguration` verwendet wird.

CLI

AWS CLI

So löschen Sie die Inventarkonfiguration eines Buckets

Im folgenden Beispiel für `delete-bucket-inventory-configuration` wird die Inventarkonfiguration mit der ID 1 für den angegebenen Bucket gelöscht.

```
aws s3api delete-bucket-inventory-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteBucketInventoryConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen InventoryName 'test', das dem angegebenen S3-Bucket entspricht.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -
InventoryId 'testInventoryName'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketInventoryConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen InventoryName 'test', das dem angegebenen S3-Bucket entspricht.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -
InventoryId 'testInventoryName'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketInventoryConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteBucketLifecycle`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketLifecycle` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
 string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- Einzelheiten zur API finden Sie DeleteBucketLifecyclein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Lebenszykluskonfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketLifecyclein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
 self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteBucketLifecyclein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3->deletebucketlifecycle(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteBucketLifecyclein der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteBucketMetricsConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketMetricsConfiguration` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Metrik-Konfiguration für einen Bucket

Im folgenden Beispiel für `delete-bucket-metrics-configuration` wird die Metrik-Konfiguration für den angegebenen Bucket und die angegebene ID entfernt.

```
aws s3api delete-bucket-metrics-configuration \
```

```
    --bucket amzn-s3-demo-bucket \
    --id 123
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteBucketMetricsConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen „testmetrics” im angegebenen S3-Bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Einzelheiten zur API finden Sie unter DeleteBucketMetricsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen „testmetrics” im angegebenen S3-Bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- Einzelheiten zur API finden Sie unter DeleteBucketMetricsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteBucketPolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketPolicy` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Bucket-Richtlinie aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
deleted
     *
     * @throws S3Exception if there is an error deleting the bucket policy
     */
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Bucket-Richtlinie.

```
import {
  DeleteBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
```

```
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
```

```
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Der Befehl entfernt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter DeleteBucketPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter DeleteBucketPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3->deletebucketpolicy(
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket policy deleted.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteBucketReplication`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketReplication` verwendet wird.

CLI

AWS CLI

Der folgende Befehl löscht eine Replikationskonfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketReplicationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht die Replikationskonfiguration, die mit dem Bucket namens „amzn-s3-demo-bucket“ verknüpft ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die DeleteReplicationConfiguration Aktion s3: erforderlich ist. Sie werden vor der Ausführung des Vorgangs zur Bestätigung aufgefordert. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter DeleteBucketReplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht die Replikationskonfiguration, die mit dem Bucket namens „amzn-s3-demo-bucket“ verknüpft ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die DeleteReplicationConfiguration Aktion s3: erforderlich ist. Sie werden vor der Ausführung des

Vorgangs zur Bestätigung aufgefordert. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter DeleteBucketReplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteBucketTagging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketTagging` verwendet wird.

CLI

AWS CLI

Der folgende Befehl löscht eine Tagging-Konfiguration aus einem Bucket namens `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketTaggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem angegebenen Bucket verknüpft sind.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem angegebenen Bucket verknüpft sind.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteBucketWebsite`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteBucketWebsite` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);
```

```
    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteBucketWebsitein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Website-Konfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie DeleteBucketWebsitein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
 configuration from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
```

```
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteBucketWebsitein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Website-Konfiguration aus dem Bucket.

```
import {
  DeleteBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    // The response code will be successful for both removed configurations and
    // configurations that did not exist in the first place.
    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it
 never was.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteBucketWebsitein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteObject`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Arbeiten mit Amazon-S3-Objektintegrität
- Mit versionierten Objekten arbeiten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt in einem nicht versionierten S3-Bucket.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete an object from a non-versioned Amazon
    /// Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteObject
```

```
    {
        /// <summary>
        /// The Main method initializes the necessary variables and then calls
        /// the DeleteObjectNonVersionedBucketAsync method to delete the object
        /// named by the keyName parameter.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
```

```
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

Löschen Sie ein Objekt in einem versionierten S3-Bucket.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
 client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
 keyName);

                // Delete the object by specifying an object key and a version
 ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
```

```
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
```

```
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
```

```
                                const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht ein Objekt namens `test.txt` aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Wenn die Bucket-Versionierung aktiviert ist, enthält die Ausgabe die Versions-ID der Löschmarkierung:

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
```

```
}
```

Weitere Informationen über das Löschen von Objekten finden Sie unter Löschen von Objekten im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie DeleteObjectin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
```

```
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
  } else {
   deleted = true
  }
 }
```

```
 return deleted, err
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an object from an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has
been deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during
delete", ex);
           }
```

```
        });

        return response.thenApply(r -> null);
    }
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
```

```
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
 exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteObject(string $bucketName, string $fileName, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
```

```
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

Setzen Sie ein Objekt auf eine vorherige Version zurück, indem Sie spätere Versionen des Objekts löschen.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
```

```
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

Reaktivieren Sie ein gelöschtes Objekt, indem Sie die aktive Löschmarke des Objekts entfernen.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
```

```
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Erstellen Sie einen Lambda-Handler, der eine Löschmarke aus einem S3-Objekt entfernt. Dieser Handler kann verwendet werden, um irrelevante Löschmarkierungen in einem versionierten Bucket effizient zu bereinigen.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
 the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None
```

```
    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
 marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
 obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
```

```
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- Einzelheiten zur API finden Sie DeleteObjectin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    // There are no modeled errors to handle when deleting an object.

    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteObjectin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
```

```
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeleteObjectTagging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteObjectTagging` verwendet wird.

CLI

AWS CLI

So löschen Sie die Tag-Sets eines Objekts

Im folgenden Beispiel für `delete-object-tagging` wird das Tag mit dem angegebenen Schlüssel aus dem Objekt `doc1.rtf` gelöscht.

```
aws s3api delete-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteObjectTaggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem Objekt mit dem Schlüssel „testfile.txt" im angegebenen S3-Bucket verknüpft sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -
Select '^Key'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Einzelheiten zur API finden Sie unter DeleteObjectTagging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem Objekt mit dem Schlüssel „testfile.txt" im angegebenen S3-Bucket verknüpft sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -
Select '^Key'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- Einzelheiten zur API finden Sie unter DeleteObjectTagging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteObjects`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteObjects` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Löschen Sie alle Objekte im Bucket.
- Erste Schritte mit Objektspeicher

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
            }
```

```
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte in einem nicht versionierten S3-Bucket.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
```

```
            // If the Amazon S3 bucket from which you wish to delete objects is
not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
```

```
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
```

```
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

Löschen Sie mehrere Objekte in einem versionierten S3-Bucket.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
```

```
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
```

```
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }
```

```
        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
```

```
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
```

```
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
```

```
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
 bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
 in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
```

```
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht ein Objekt aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` ist ein JSON-Dokument im aktuellen Verzeichnis, in dem das zu löschende Objekt angegeben wird:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Ausgabe:

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
```

```
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
```

```
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 } else {
  for _, delObjs := range delOut.Deleted {
   err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.Key)
   }
  }
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName>
```

```
            Where:
               bucketName - the Amazon S3 bucket name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
```

```
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte.

```
import {
```

```
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while deleting objects from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Reihe von Objekten aus einer Schlüsselliste.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
```

```
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt das Objekt „sample.txt" aus dem Bucket „test-files". Sie werden vor der Ausführung des Befehls um eine Bestätigung gebeten. Um den Prompt zu unterdrücken, verwenden Sie den -Force-Schalter.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts „sample.txt" aus dem Bucket „test-files", vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert sind.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte „sample1.txt", „sample2.txt" und „sample3.txt" als einzelne Stapelverarbeitung aus dem Bucket „test-files". Die Antwort des Services listet alle verarbeiteten Schlüssel auf, unabhängig davon, ob die Löschung erfolgreich war oder fehlgeschlagen ist. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Object gibt eine

Sammlung von Amazon.S3.Model.S3Object Instanzen zurück, von denen jede ein Key-Element vom Typ string hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein key prefix "prefix/subprefix" im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Beispiel 6: In diesem Beispiel wird eine Sammlung von Amazon.S3.Model.S3ObjectVersion Instanzen, die Löschmarkierungen darstellen, zur Löschung an das Cmdlet übergeben. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten (in diesem Fall Löschmarken) im Batch-Modus gelöscht werden kann, indem ein Array von Objekten erstellt wird, die mit dem Parameter - verwendet werden sollen. KeyAndVersionCollection

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- Einzelheiten zur API finden Sie unter DeleteObjects AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt das Objekt „sample.txt” aus dem Bucket „test-files”. Sie werden vor der Ausführung des Befehls um eine Bestätigung gebeten. Um den Prompt zu unterdrücken, verwenden Sie den -Force-Schalter.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts „sample.txt“ aus dem Bucket „test-files“, vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert sind.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte „sample1.txt“, „sample2.txt“ und „sample3.txt“ als einzelne Stapelverarbeitung aus dem Bucket „test-files“. Die Antwort des Services listet alle verarbeiteten Schlüssel auf, unabhängig davon, ob die Löschung erfolgreich war oder fehlgeschlagen ist. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Object gibt eine Sammlung von Amazon.S3.Model.S3Object Instanzen zurück, von denen jede ein Key-Element vom Typ string hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein key prefix "prefix/subprefix" im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden

Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Beispiel 6: In diesem Beispiel wird eine Sammlung von Amazon.S3.Model.S3ObjectVersion Instanzen, die Löschmarkierungen darstellen, zur Löschung an das Cmdlet übergeben. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten (in diesem Fall Löschmarken) im Batch-Modus gelöscht werden kann, indem ein Array von Objekten erstellt wird, die mit dem Parameter - verwendet werden sollen. KeyAndVersionCollection

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- Einzelheiten zur API finden Sie unter DeleteObjects AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Reihe von Objekten mithilfe einer Liste von Objektschlüsseln.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
```

```
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

Löschen Sie alle Objekte im Bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.
```

```
        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

Löschen Sie ein versioniertes Objekt dauerhaft, indem Sie alle seine Versionen löschen.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- Einzelheiten zur API finden Sie DeleteObjectsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building
 object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
 delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object
 input {err:?}"))
                })?,
        )
        .send()
        .await?;
    Ok(())
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->deleteobjects(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteObjectsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
```

```
                objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
                quiet: true
            )
        )

        do {
            _ = try await client.deleteObjects(input: input)
        } catch {
            print("ERROR: deleteObjects:", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectsin der API-Referenz zum AWS SDK für Swift.

Verwendung von **`DeletePublicAccessBlock`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePublicAccessBlock` verwendet wird.

CLI

AWS CLI

So löschen Sie die Konfiguration zum Blockieren des öffentlichen Zugriffs für einen Bucket

Im folgenden Beispiel für `delete-public-access-block` wird die Konfiguration zum Blockieren des öffentlichen Zugriffs für den angegebenen Bucket entfernt.

```
aws s3api delete-public-access-block \
    --bucket amzn-s3-demo-bucket
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeletePublicAccessBlockin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlicher Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Ausgabe:

```
amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter DeletePublicAccessBlock AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlicher Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Ausgabe:

```
amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter DeletePublicAccessBlock AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketAccelerateConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketAccelerateConfiguration` verwendet wird.

CLI

AWS CLI

So rufen Sie die Accelerate-Konfiguration eines Buckets ab

Im folgenden Beispiel für `get-bucket-accelerate-configuration` wird die Accelerate-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-accelerate-configuration \
```

```
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Status": "Enabled"
}
```

- Einzelheiten zur API finden Sie GetBucketAccelerateConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Übertragungsbeschleunigungseinstellungen für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter GetBucketAccelerateConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Übertragungsbeschleunigungseinstellungen für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
```

```
-----
Enabled
```

- Einzelheiten zur API finden Sie unter GetBucketAccelerateConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketAcl`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketAcl` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Prüfen, ob ein Bucket vorhanden ist
- Verwalten von Zugriffssteuerungslisten (ACL)

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
```

```
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- Einzelheiten zur API finden Sie GetBucketAclin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();
```

```
        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

```
//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
```

```
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- Einzelheiten zur API finden Sie GetBucketAclin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Zugriffskontrollliste für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetBucketAclin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <objectKey>

            Where:
              bucketName - The Amazon S3 bucket to get the access control list
 (ACL) for.
              objectKey - The object to get the ACL for.\s
            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
```

```
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- Einzelheiten zur API finden Sie GetBucketAclin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die ACL-Berechtigungen ab.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
```

```
  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetBucketAclin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V5

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Ausgabe:

```
DisplayName Id
----------- --
testusername
 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Einzelheiten zur API finden Sie unter GetBucketAcl AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
```

```
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- Einzelheiten zur API finden Sie GetBucketAclin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketacl(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetBucketAclin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GetBucketAnalyticsConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketAnalyticsConfiguration` verwendet wird.

CLI

AWS CLI

So rufen Sie die Analytikkonfiguration für einen Bucket mit einer bestimmten ID ab

Im folgenden Beispiel für `get-bucket-analytics-configuration` wird die Analytics-Konfiguration für den angegebenen Bucket und die angegebene ID angezeigt.

```
aws s3api get-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Ausgabe:

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- Einzelheiten zur API finden Sie GetBucketAnalyticsConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen „testfilter” im angegebenen S3-Bucket zurück.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketAnalyticsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen „testfilter” im angegebenen S3-Bucket zurück.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketAnalyticsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketCors`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketCors` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- Einzelheiten zur API finden Sie GetBucketCorsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Cross-Origin Resource Sharing-Konfiguration für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
        {
            "AllowedHeaders": [
                "Authorization"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
```

```
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetBucketCorsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die CORS-Richtlinie für den Bucket ab.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
```

```
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The
 bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetBucketCorsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
 self.bucket.name))
            raise
        else:
            return cors
```

- Einzelheiten zur API finden Sie GetBucketCorsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- Einzelheiten zur API finden Sie GetBucketCorsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketcors(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetBucketCorsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetBucketEncryption`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketEncryption` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get and print the encryption settings of a bucket.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task GetEncryptionSettings(string bucketName)
    {
        // Check and print the bucket encryption settings.
        Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

        try
        {
            var settings =
                await _s3Client.GetBucketEncryptionAsync(
                    new GetBucketEncryptionRequest() { BucketName =
bucketName });

            foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
            {
                Console.WriteLine(
                    $"\tAlgorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
                Console.WriteLine(
                    $"\tKey:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServi
            }
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
```

- Einzelheiten zur API finden Sie GetBucketEncryptionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die serverseitige Verschlüsselungskonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-bucket-encryption` wird die serverseitige Verschlüsselungskonfiguration für den Bucket `amzn-s3-demo-bucket` abgerufen.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- Einzelheiten zur API finden Sie GetBucketEncryptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketEncryption AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketEncryption AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketInventoryConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketInventoryConfiguration` verwendet wird.

CLI

AWS CLI

So rufen Sie die Bestandskonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-bucket-inventory-configuration` wird die Bestandskonfiguration für den angegebenen Bucket mit der ID 1 abgerufen.

```
aws s3api get-bucket-inventory-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Ausgabe:

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
```

```
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    }
}
```

- Einzelheiten zur API finden Sie GetBucketInventoryConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen „testinventory” für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Einzelheiten zur API finden Sie unter GetBucketInventoryConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen „testinventory” für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- Einzelheiten zur API finden Sie unter GetBucketInventoryConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketLifecycleConfiguration`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketLifecycleConfiguration` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- Einzelheiten zur API finden Sie GetBucketLifecycleConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Lebenszyklus-Konfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetBucketLifecycleConfigurationin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
```

```
            return config.rules
```

- Einzelheiten zur API finden Sie GetBucketLifecycleConfigurationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketlifecycleconf(         " oo_result is
returned for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetBucketLifecycleConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetBucketLocation`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketLocation` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Standortbeschränkung für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen, falls eine Einschränkung besteht:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "LocationConstraint": "us-west-2"
}
```

- Einzelheiten zur API finden Sie GetBucketLocationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket „amzn-s3-demo-bucket“ zurück, sofern eine Beschränkung vorhanden ist.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
ap-south-1
```

- Einzelheiten zur API finden Sie unter GetBucketLocation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket „amzn-s3-demo-bucket“ zurück, sofern eine Beschränkung vorhanden ist.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
```

```
-----
ap-south-1
```

- Einzelheiten zur API finden Sie unter GetBucketLocation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
```

```
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie GetBucketLocationin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`GetBucketLogging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketLogging` verwendet wird.

CLI

AWS CLI

So rufen Sie den Protokollierungsstatus für einen Bucket ab

Im folgenden Beispiel für `get-bucket-logging` wird der Protokollierungsstatus für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-logging \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
```

```
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "amzn-s3-demo-bucket-logs"
          }
}
```

- Einzelheiten zur API finden Sie GetBucketLoggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Protokollierungsstatus für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Einzelheiten zur API finden Sie unter GetBucketLogging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Protokollierungsstatus für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- Einzelheiten zur API finden Sie unter GetBucketLogging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketMetricsConfiguration`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketMetricsConfiguration` verwendet wird.

CLI

AWS CLI

So rufen Sie die Metrik-Konfiguration für einen Bucket mit einer bestimmten ID ab

Im folgenden Beispiel für `get-bucket-metrics-configuration` wird die Metrik-Konfiguration für den angegebenen Bucket und die angegebene ID angezeigt.

```
aws s3api get-bucket-metrics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 123
```

Ausgabe:

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- Einzelheiten zur API finden Sie GetBucketMetricsConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen „testfilter” für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketMetricsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen „testfilter” für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- Einzelheiten zur API finden Sie unter GetBucketMetricsConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketNotification`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketNotification` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Benachrichtigungskonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
}
```

- Einzelheiten zur API finden Sie GetBucketNotificationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Ausgabe:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Einzelheiten zur API finden Sie unter GetBucketNotification AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Ausgabe:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Einzelheiten zur API finden Sie unter GetBucketNotification AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketPolicy`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketPolicy` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Bucket-Richtlinie für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::amzn-s3-demo-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",
\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-
s3-demo-bucket/secret/*\"}]}"
}
```

So rufen Sie eine Bucket-Richtlinie ab und fügen sie ein Das folgende Beispiel zeigt, wie Sie eine Amazon-S3-Bucket-Richtlinie herunterladen, Änderungen an der Datei vornehmen, und die geänderte Bucket-Richtlinie mithilfe von `put-bucket-policy` anwenden können. Führen Sie folgenden Befehl aus, um die Bucket-Richtlinie in eine Datei herunterzuladen:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output
 text > policy.json
```

Anschließend können Sie die `policy.json`-Datei nach Bedarf ändern. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

`policy.json`-Datei nach Bedarf. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

-Datei nach Bedarf. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
```

```
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the
policy
     * @return the policy text for the specified bucket, or an empty string if an
error occurs
     */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Bucket-Richtlinie ab.

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
```

```
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }
```

```
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Dieser Befehl gibt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie aus.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicy AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie aus.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicy AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- Einzelheiten zur API finden Sie GetBucketPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketpolicy(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       DATA(lv_policy) = oo_result->get_policy( ).
       MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`GetBucketPolicyStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketPolicyStatus` verwendet wird.

CLI

AWS CLI

So rufen Sie den Richtlinienstatus eines Buckets ab, der angibt, ob es sich um einen öffentlichen Bucket handelt

Im folgenden Beispiel für `get-bucket-policy-status` wird der Richtlinienstatus für den Bucket `amzn-s3-demo-bucket` abgerufen.

```
aws s3api get-bucket-policy-status \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicyStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketPolicyStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketReplication`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketReplication` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Replikationskonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- Einzelheiten zur API finden Sie GetBucketReplicationin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client           the S3 client used to interact with the S3
service
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication
details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
           .bucket(sourceBucketName)
           .build();

       try {
           ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
           ReplicationRule rule = replicationConfig.rules().get(0);
           System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
           System.out.println("Retrieved priority: " + rule.priority());
           System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

       } catch (S3Exception e) {
           System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
       }
   }
```

- Einzelheiten zur API finden Sie GetBucketReplicationin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die informationen der Replikationskonfiguration zurück, die für den Bucket mit dem Namen „amzn-s3-demo-bucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter GetBucketReplication AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die informationen der Replikationskonfiguration zurück, die für den Bucket mit dem Namen „amzn-s3-demo-bucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter GetBucketReplication AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketRequestPayment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketRequestPayment` verwendet wird.

CLI

AWS CLI

So rufen Sie die Zahlungsanforderungskonfiguration eines Buckets auf

Im folgenden Beispiel für `get-bucket-request-payment` wird die Zahlung-durch-den-Anforderer-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-request-payment \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Payer": "BucketOwner"
}
```

- Einzelheiten zur API finden Sie GetBucketRequestPaymentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Zahlungsanforderungskonfiguration für den Bucket mit dem Namen „amzn-s3-demo-bucket“ zurück. Standardmäßig zahlt der Bucket-Eigentümer für die Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter GetBucketRequestPayment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Zahlungsanforderungskonfiguration für den Bucket mit dem Namen „amzn-s3-demo-bucket“ zurück. Standardmäßig zahlt der Bucket-Eigentümer für die Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter GetBucketRequestPayment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketTagging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketTagging` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Tag-Konfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "TagSet": [
```

```
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetBucketTaggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die mit dem angegebenen Bucket verknüpft sind.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die mit dem angegebenen Bucket verknüpft sind.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetBucketVersioning`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketVersioning` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Versionierungskonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Status": "Enabled"
}
```

- Einzelheiten zur API finden Sie GetBucketVersioningin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketVersioning AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketVersioning AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetBucketWebsite`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketWebsite` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
 GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
 client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
 {getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
 {getResponse.WebsiteConfiguration.ErrorDocument}");
```

- Einzelheiten zur API finden Sie GetBucketWebsitein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
```

```
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetBucketWebsitein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die statische Webseitenkonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- Einzelheiten zur API finden Sie GetBucketWebsitein der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Website-Konfiguration ab.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
```

```
  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
 bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie GetBucketWebsitein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetObject`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Abrufen eines Objekts aus einem Bucket abrufen, wenn es geändert wurde
- Holen Sie sich ein Objekt von einem Multi-Region Access Point
- Erste Schritte mit der Verschlüsselung
- Erste Schritte mit Objektspeicher
- Senden von bedingten Anforderungen
- Verfolgen von Uploads und Downloads

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await
 _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```
    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
 'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
```

```
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read
 successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
```

```
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie GetObjectin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Das folgende Beispiel veranschaulicht, wie Sie den `get-object`-Befehl verwenden können, um ein Objekt aus Amazon S3 herunterzuladen:

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Beachten Sie, dass der Outfile-Parameter ohne einen Optionsnamen wie „--outfile“ angegeben wird. Der Name der Ausgabedatei muss der letzte Parameter im Befehl sein.

Das folgende Beispiel veranschaulicht die Verwendung von `--range`, um einen bestimmten Bytebereich aus einem Objekt herunterzuladen. Beachten Sie, dass den Bytebereichen das Präfix „bytes=“ vorangestellt werden muss:

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

Weitere Informationen über das Abrufen von Objekten finden Sie unter „Objekte abrufen“ im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie GetObjectin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
```

```
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
   err = noKey
  } else {
   log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  }
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lesen Sie Daten als Byte-Array mit einem S3Client.

```
   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes
have been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
               Files.write(filePath, data);
               logger.info("Successfully obtained bytes from an S3 object");
           } catch (IOException ex) {
               throw new RuntimeException("Failed to write data to file", ex);
           }
       }).whenComplete((resp, ex) -> {
```

```
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
 ex);
            }
        });
    }
```

Verwenden Sie an S3TransferManager, um ein Objekt in einem S3-Bucket in eine lokale Datei herunterzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);
```

```
        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Lesen Sie Tags, die zu einem Objekt gehören, mit einem S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags =
 s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie eine URL für ein Objekt mit einem S3Client ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
 retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
 specified object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie ein Objekt mithilfe des S3Presigner-Client-Objekts mit einem S3Client ab.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();
```

```
        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating
the pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
service.
     * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
```

```
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

Rufen Sie ein Objekt mithilfe eines ResponseTransformer Objekts und S3Client ab.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
 them to a local file.
```

```
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3
 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
 local file.
     * @throws S3Exception If an error occurs while retrieving the object from
 the S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie unter GetObject AWS SDK for Java 2.xAPI-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie das Objekt herunter.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
```

```
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
```

```
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
```

```
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
```

```
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt ab.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft das Objekt "sample.txt" aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei „local-sample.txt” muss vor dem Aufruf dieses Befehls nicht vorhanden sein.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Beispiel 2: Mit diesem Befehl wird das virtuelle Verzeichnis „DIR“ aus dem Bucket „amzn-s3-demo-bucket“ abgerufen und in einem Ordner mit dem Namen "" am aktuellen Speicherort gespeichert. Local-DIR Der Ordner "Local-DIR" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf „.json“ enden, aus Buckets, deren Name „config“ enthält, in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Einzelheiten zur API finden Sie unter GetObject AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft das Objekt "sample.txt" aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei „local-sample.txt” muss vor dem Aufruf dieses Befehls nicht vorhanden sein.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Beispiel 2: Mit diesem Befehl wird das virtuelle Verzeichnis „DIR“ aus dem Bucket „amzn-s3-demo-bucket“ abgerufen und in einem Ordner mit dem Namen "" am aktuellen Speicherort gespeichert. Local-DIR Der Ordner "Local-DIR" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf „.json“ enden, aus Buckets, deren Name „config“ enthält, in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Beispiel 4: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallel Downloads herunter, um die Leistung bei großen Dateien zu verbessern.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt -UseMultipartDownload
```

Beispiel 5: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Teilgröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanforderungen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-
large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Beispiel 6: Lädt alle Objekte mit dem key prefix „data/“ mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-
data -UseMultipartDownload -DownloadFilesConcurrently
```

Beispiel 7: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um den Durchsatz in Netzwerken mit hoher Bandbreite zu maximieren.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Beispiel 8: Lädt alle Objekte mit dem key prefix „data/“ mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter, wobei einzelne Dateifehler überwunden werden, anstatt den Vorgang abzubrechen.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-
data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Einzelheiten zur API finden Sie unter GetObject AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
            raise
        else:
            return body
```

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
```

```
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
 the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Einzelheiten zur API finden Sie GetObjectin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt ab.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rufen Sie ein Objekt ab und melden Sie seinen serverseitigen Verschlüsselungsstatus.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
 {err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
```

```
    Ok(byte_count)
}
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                 iv_bucket = iv_bucket_name
                 iv_key = iv_object_key ).
       DATA(lv_object_data) = oo_result->get_body( ).
       MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func downloadFile(bucket: String, key: String, to: String) async
 throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
 read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }
```

```
import AWSS3
```

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
  }
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetObjectAcl`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectAcl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffssteuerungslisten (ACL)

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```

```
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
```

```
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- Einzelheiten zur API finden Sie GetObjectAclin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Zugriffskontrollliste für ein Objekt in einem Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Ausgabe:

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetObjectAclin der AWS CLI Befehlsreferenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetObjectAclin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V5 PowerShell

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 *).Owner
```

Ausgabe:

```
DisplayName Id
----------- --
testusername
 9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Einzelheiten zur API finden Sie unter GetObjectAcl AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
```

```
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- Einzelheiten zur API finden Sie GetObjectAclin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getobjectacl(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name
         iv_key = iv_object_key ).
       MESSAGE 'Retrieved object ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetObjectAclin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetObjectAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectAttributes` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Arbeiten mit Amazon-S3-Objektintegrität

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
```

```
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
```

```
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
```

```
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

- Einzelheiten zur API finden Sie GetObjectAttributesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Metadaten von einem Objekt ab, ohne das Objekt selbst zurückzugeben

Im folgenden Beispiel für `get-object-attributes` werden Metadaten aus dem Objekt `doc1.rtf` abgerufen.

```
aws s3api get-object-attributes \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"
```

Ausgabe:

```
{
    "LastModified": "2022-03-15T19:37:31+00:00",
    "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
    "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
    "StorageClass": "STANDARD",
    "ObjectSize": 405
}
```

Weitere Informationen finden Sie GetObjectAttributesin der Amazon S3 S3-API-Referenz.

- Einzelheiten zur API finden Sie GetObjectAttributesin der AWS CLI Befehlsreferenz.

Verwenden Sie **`GetObjectLegalHold`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectLegalHold` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
```

```
                                $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den Status „Rechtliche Aufbewahrungsfrist“ eines Objekts ab

Im folgenden Beispiel für `get-object-legal-hold` wird der Status Rechtliche Aufbewahrungsfrist für das angegebene Objekt abgerufen.

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Ausgabe:

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
```

```
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Get the legal hold details for an S3 object.
```

```
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
 objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
 GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
 getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
 bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
 ex.getMessage() + "'");
        }

        return null;
    }
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectLegalHoldCommand,
```

```
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
 bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
 ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
 ${bucketName} from ${bucketName}.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
```

```
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
 bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getobjectlegalhold(         " oo_result is returned
for testing purposes. "
         iv_bucket = iv_bucket_name
         iv_key = iv_object_key ).
       MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetObjectLegalHoldin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetObjectLockConfiguration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectLockConfiguration` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Get the object lock configuration details for an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to get details.</param>
   /// <returns>The bucket's object lock configuration details.</returns>
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
           return new ObjectLockConfiguration();
       }
```

```
    }
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Objektsperrkonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-object-lock-configuration` wird die Object Lock-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

Ausgabe:

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }
```

```
 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
```

```
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
 bucket>",
      }),
    );
    console.log(
      `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
```

```
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter GetObjectLockConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter GetObjectLockConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Object-Lock-Konfiguration ab.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
 "Enabled"
```

```
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getobjectlockconfiguration(         " oo_result is
returned for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetObjectLockConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetObjectRetention`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectRetention` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectRetentionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die Konfiguration für die Aufbewahrung eines Objekts ab

Im folgenden Beispiel für `get-object-retention` wird die Konfiguration für die Aufbewahrung des angegebenen Objekts abgerufen.

```
aws s3api get-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Ausgabe:

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- Einzelheiten zur API finden Sie GetObjectRetentionin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
```

```
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- Einzelheiten zur API finden Sie GetObjectRetentionin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
```

```
            GetObjectRetentionRequest retentionRequest =
 GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
 getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
 in "+ bucketName +": " + response.retention().mode() +" until "+
 response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectRetentionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
```

```
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until
 ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
 ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
 "${bucketName}".  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```

```
const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie GetObjectRetentionin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt aufbewahrt wird.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Einzelheiten zur API finden Sie unter GetObjectRetention AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt aufbewahrt wird.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Einzelheiten zur API finden Sie unter GetObjectRetention AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetObjectTagging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Tags

CLI

AWS CLI

So rufen Sie die an ein Objekt angehängten Tags ab

Im folgenden Beispiel für `get-object-tagging` werden die Werte für den angegebenen Schlüssel aus dem angegebenen Objekt abgerufen.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

Ausgabe:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
}
```

Im folgenden Beispiel für `get-object-tagging` wird versucht, die Tag-Sätze des Objekts `doc2.rtf` abzurufen, das jedoch keine Tags hat.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc2.rtf
```

Ausgabe:

```
{
    "TagSet": []
}
```

Im folgenden Beispiel für `get-object-tagging` werden die Tag-Sätze des Objekts `doc3.rtf` abgerufen, das über mehrere Tags verfügt.

```
aws s3api get-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc3.rtf
```

Ausgabe:

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
            "Key": "team"
        }
    ]
}
```

- Einzelheiten zur API finden Sie GetObjectTaggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags zurückgegeben, die mit dem Objekt im angegebenen S3-Bucket verknüpft sind.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Key  Value
---  -----
test value
```

- Einzelheiten zur API finden Sie unter GetObjectTagging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags zurückgegeben, die mit dem Objekt im angegebenen S3-Bucket verknüpft sind.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Key  Value
---  -----
test value
```

- Einzelheiten zur API finden Sie unter GetObjectTagging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPublicAccessBlock`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPublicAccessBlock` verwendet wird.

CLI

AWS CLI

So richten Sie die Konfiguration zum Blockieren des öffentlichen Zugriffs für einen Bucket ein oder ändern sie

Im folgenden Beispiel für `get-public-access-block` wird die Konfiguration zum Blockieren des öffentlichen Zugriff für den angegebenen Bucket angezeigt.

```
aws s3api get-public-access-block \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- Einzelheiten zur API finden Sie GetPublicAccessBlockin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl gibt die Konfiguration der öffentlichen Zugriffssperre des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetPublicAccessBlock AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl gibt die Konfiguration der öffentlichen Zugriffssperre des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter GetPublicAccessBlock AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`HeadBucket`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `HeadBucket` verwendet wird.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
###############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
```

```
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- Einzelheiten zur API finden Sie HeadBucketin der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Der folgende Befehl überprüft den Zugriff auf einen Bucket namens `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Wenn der Bucket vorhanden ist und Sie Zugriff darauf haben, wird keine Ausgabe zurückgegeben. Andernfalls wird eine Fehlermeldung ausgegeben. Beispiel:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Einzelheiten zur API finden Sie HeadBucketunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
```

```
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
```

```
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- Einzelheiten zur API finden Sie HeadBucketin der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Ausgabe mit dem HTTP-Statuscode 200 OK für einen vorhandenen Bucket zurück, wenn der Benutzer die Erlaubnis hat, darauf zuzugreifen. BucketArn Der Parameter wird nur für S3-Verzeichnis-Buckets unterstützt.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Ausgabe:

```
AccessPointAlias   : False
BucketArn          :
BucketLocationName :
BucketLocationType :
BucketRegion       : us-east-2
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 0
HttpStatusCode     : OK
```

Beispiel 2: Dieser Befehl gibt einen Fehler mit HTTP-Statuscode NotFound für einen nicht existierenden Bucket aus.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
 Code NotFound. No further error information was returned by the service.
```

Beispiel 3: Dieser Befehl löst einen Fehler mit dem HTTP-Statuscode „Forbidden“ für einen vorhandenen Bucket aus, auf den der Benutzer keine Zugriffsberechtigung hat.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
 Code Forbidden. No further error information was returned by the service.
```

- Einzelheiten zur API finden Sie unter HeadBucket AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- Einzelheiten zur API finden Sie HeadBucketin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->headbucket(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie HeadBucketin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`HeadObject`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `HeadObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Prüfen Sie, ob ein Objekt existiert
- Erste Schritte mit Objektspeicher

CLI

AWS CLI

Der folgende Befehl ruft Metadaten für ein Objekt in einem Bucket mit dem Namen `amzn-s3-demo-bucket` ab:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Ausgabe:

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- Einzelheiten zur API finden Sie HeadObjectin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Bestimmen Sie den Inhaltstyp eines Objekts.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie den Wiederherstellungsstatus eines Objekts ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to
 interact with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is
 stored
     * @param keyName    the name of the Amazon S3 object to be checked
```

```
     * @throws S3Exception if an error occurs while interacting with the Amazon
 S3 service
     */
    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie HeadObjectin der AWS SDK for Java 2.x API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie HeadObjectin der AWS SDK für Ruby API-Referenz.

Verwendung von **`ListBucketAnalyticsConfigurations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListBucketAnalyticsConfigurations` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste der Analytikkonfigurationen für einen Bucket ab

Im folgenden Beispiel für `list-bucket-analytics-configurations` wird eine Liste von Analytics-Konfigurationen für den angegebenen Bucket abgerufen.

```
aws s3api list-bucket-analytics-configurations \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "AnalyticsConfigurationList": [
        {
            "StorageClassAnalysis": {},
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- Einzelheiten zur API finden Sie ListBucketAnalyticsConfigurationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die ersten 100 Analysekonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter ListBucketAnalyticsConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Analysekonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter ListBucketAnalyticsConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListBucketInventoryConfigurations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListBucketInventoryConfigurations` verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste der Bestandskonfigurationen für einen Bucket ab

Im folgenden Beispiel für `list-bucket-inventory-configurations` werden die Listen der Bestandskonfigurationen für den angegebenen Bucket aufgelistet.

```
aws s3api list-bucket-inventory-configurations \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
```

```
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- Einzelheiten zur API finden Sie ListBucketInventoryConfigurationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter ListBucketInventoryConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter ListBucketInventoryConfigurations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListBuckets`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListBuckets` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
```

```
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
```

```
    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl verwendet den `list-buckets`-Befehl, um die Namen aller Ihrer Amazon-S3-Buckets (in allen Regionen) anzuzeigen:

```
aws s3api list-buckets --query "Buckets[].Name"
```

Die Abfrageoption filtert die Ausgabe von `list-buckets` nach unten, sodass nur die Bucket-Namen angezeigt werden.

Weitere Informationen über Buckets finden Sie unter „Arbeiten mit Amazon-S3-Buckets“ im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie ListBucketsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
 error) {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Buckets auf.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
```

```
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing buckets.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter ListBuckets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter ListBuckets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
 Boto3
                            that contains collections and factory methods to
 create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
```

```
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- Einzelheiten zur API finden Sie ListBucketsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
```

```
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
```

```
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListBucketsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->listbuckets(         " oo_result is returned for
 testing purposes. "
        ).
        DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
        MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list buckets.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListBucketsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    /// Return an array containing information about every available bucket.
    ///
    /// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
    ///   each bucket.
    public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
        return try await client.listBuckets(input: ListBucketsInput())
    }
```

- Einzelheiten zur API finden Sie ListBucketsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`ListMultipartUploads`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListMultipartUploads` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Löschen von unvollständigen mehrteiligen Uploads

CLI

AWS CLI

Der folgende Befehl listet alle aktiven mehrteiligen Uploads für einen Bucket mit dem Namen `amzn-s3-demo-bucket` auf:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
    "CommonPrefixes": []
}
```

Bei laufenden mehrteiligen Uploads fallen Speicherkosten in Amazon S3 an. Schließen Sie einen aktiven mehrteiligen Upload ab oder brechen Sie ihn ab, um seine Teile aus Ihrem Konto zu entfernen.

- Einzelheiten zur API finden Sie ListMultipartUploadsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon
 S3 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListMultipartUploadsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListObjectVersions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListObjectVersions` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erste Schritte mit Objektspeicher
- Mit versionierten Objekten arbeiten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
```

```
            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
 constant
            // for the AWS Region to the client constructor like this:
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
```

```
                        Console.WriteLine($"key: {entry.Key} size:
 {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListObjectVersionsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl werden Versionsinformationen für ein Objekt in einem Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Ausgabe:

```
{
    "DeleteMarkers": [
```

```
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": true,
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
```

```
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListObjectVersionsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
```

```
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- Einzelheiten zur API finden Sie ListObjectVersionsin der AWS SDK für Go API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListObjectVersionsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->listobjectversions(         " oo_result is returned
 for testing purposes. "
          iv_bucket = iv_bucket_name
          iv_prefix = iv_prefix ).
        MESSAGE 'Retrieved object versions.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListObjectVersionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`ListObjects`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListObjects` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Eine Webseite erstellen, die Amazon-S3-Objekte auflistet

CLI

AWS CLI

Im folgenden Beispiel wird der Befehl `list-objects` verwendet, um die Namen aller Objekte im angegebenen Bucket anzuzeigen:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

Im Beispiel wird das Argument `--query` verwendet, um die Ausgabe von `list-objects` nach Schlüsselwert und Größe für jedes Objekt zu filtern

Weitere Informationen über Objekte finden Sie unter „Arbeiten mit Amazon-S3-Objekten“ im Entwicklerhandbuch zu Amazon S3.

- Einzelheiten zur API finden Sie ListObjectsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt „sample.txt“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Einzelheiten zur API finden Sie unter ListObjects AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt „sample.txt“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Einzelheiten zur API finden Sie unter ListObjects AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie V2 mit ListObjects einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListObjectsV2` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Löschen Sie alle Objekte im Bucket.
- Erste Schritte mit Objektspeicher

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
 bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
```

```
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' getting list of objects.");
            return null;
        }
    }
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte mit einem Paginator auf.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "amzn-s3-demo-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
 bucketName)
        {
```

```
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
 ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());
```

```
    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste von Objekten in einem Bucket ab

Im folgenden Beispiel für `list-objects-v2` werden die Objekte im angegebenen Bucket aufgelistet.

```
aws s3api list-objects-v2 \
    --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
    "Contents": [
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
```

```
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
    ]
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
 return objects, err
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
been listed
    */
   public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
       ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .maxKeys(1)
           .build();

       ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
       return paginator.subscribe(response -> {
           response.contents().forEach(s3Object -> {
               logger.info("Object key: " + s3Object.key());
           });
       }).thenRun(() -> {
           logger.info("Successfully listed all objects in the bucket: " +
bucketName);
       }).exceptionally(ex -> {
           throw new RuntimeException("Failed to list objects", ex);
       });
   }
```

Listet Objekte mithilfe der Paginierung auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
```

```
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
 " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Objekte in Ihrem Bucket auf. Wenn es mehr als ein Objekt gibt, IsTruncated NextContinuationToken wird es verwendet, um über die gesamte Liste zu iterieren.

```
import {
  S3Client,
  S3ServiceException,
```

```
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
        `Page ${pageNum + 1}\n------\n${objectList.map((o) => `•
 ${o}`).join("\n")}\n`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}". The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}".
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
```

```
    }
  }
};
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API-Details finden Sie unter ListObjectsV2 im AWS SDK für die Kotlin-API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte in einem Bucket auf.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
 are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
 bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
 bucket.name)
            raise
        else:
            return objects
```

- API-Details finden Sie unter ListObjectsV2 in der AWS API-Referenz zum SDK for Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
 Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
```

```
        }
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
 "ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
```

```
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK-Referenz zur Swift-API.

Verwenden Sie **`PutBucketAccelerateConfiguration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketAccelerateConfiguration` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
```

```
        public static async Task Main()
        {
            var s3Client = new AmazonS3Client();
            const string bucketName = "amzn-s3-demo-bucket";

            await EnableAccelerationAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method sets the configuration to enable transfer acceleration
        /// for the bucket referred to in the bucketName parameter.
        /// </summary>
        /// <param name="client">An Amazon S3 client used to enable the
        /// acceleration on an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
```

```
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
 setting transfer acceleration");
            }
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketAccelerateConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Beschleunigungskonfiguration eines vorhandenen Buckets fest

Im folgenden Beispiel für `put-bucket-accelerate-configuration` wird die Accelerate-Konfiguration für den angegebenen Bucket aktiviert.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket \
    --accelerate-configuration Status=Enabled
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutBucketAccelerateConfigurationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Einzelheiten zur API finden Sie unter PutBucketAccelerateConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- Einzelheiten zur API finden Sie unter PutBucketAccelerateConfiguration AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutBucketAcl`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketAcl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffssteuerungslisten (ACL)

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
```

```
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- Einzelheiten zur API finden Sie PutBucketAclin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));
```

```
    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
```

```
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Einzelheiten zur API finden Sie PutBucketAclin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

In diesem Beispiel werden zwei `full control` AWS Benutzern (user1@example.com und user2@example.com) und allen Benutzern die `read` folgenden Berechtigungen erteilt:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.htmlEinzelheiten zu benutzerdefinierten ACLs finden Sie unter (die s3api-ACL-Befehle verwenden z. `put-bucket-acl` B. dieselbe Kurzschreibweise für Argumente).

- Einzelheiten zur API finden Sie unter Befehlsreferenz. PutBucketAclAWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <id>\s

            Where:
              bucketName - The Amazon S3 bucket to grant permissions on.\s
```

```
              id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
           """;

       if (args.length != 2) {
           System.out.println(usage);
           return;
       }

       String bucketName = args[0];
       String id = args[1];
       System.out.format("Setting access \n");
       System.out.println(" in bucket: " + bucketName);
       Region region = Region.US_EAST_1;
       S3Client s3 = S3Client.builder()
           .region(region)
           .build();

       setBucketAcl(s3, bucketName, id);
       System.out.println("Done!");
       s3.close();
   }

   /**
    * Sets the Access Control List (ACL) for an Amazon S3 bucket.
    *
    * @param s3 the S3Client instance to be used for the operation
    * @param bucketName the name of the S3 bucket to set the ACL for
    * @param id the ID of the AWS user or account that will be granted full
control of the bucket
    * @throws S3Exception if an error occurs while setting the bucket ACL
    */
   public static void setBucketAcl(S3Client s3, String bucketName, String id) {
       try {
           Grant ownerGrant = Grant.builder()
               .grantee(builder -> builder.id(id)
                   .type(Type.CANONICAL_USER))
               .permission(Permission.FULL_CONTROL)
               .build();

           List<Grant> grantList2 = new ArrayList<>();
           grantList2.add(ownerGrant);

           AccessControlPolicy acl = AccessControlPolicy.builder()
```

```
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutBucketAclin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Bucket-ACL fest.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
```

```
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
```

```
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutBucketAclin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
```

```
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Einzelheiten zur API finden Sie PutBucketAclin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
 recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
```

```
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- Einzelheiten zur API finden Sie PutBucketAclin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Grant log delivery access to a bucket
        " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
        lo_s3->putbucketacl(
          iv_bucket = iv_bucket_name
          iv_grantwrite = iv_grantwrite ).
        MESSAGE 'Bucket ACL updated.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutBucketAclin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutBucketCors`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketCors` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- Einzelheiten zur API finden Sie PutBucketCors in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Im folgenden Beispiel werden PUT-, POST- und DELETE-Anfragen von www.example.com aktiviert und es werden GET-Anfragen von jeder beliebigen Domain aktiviert:

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-
configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- Einzelheiten zur API finden Sie PutBucketCorsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
```

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
Amazon S3 bucket.
     *
     * @param s3            the {@link S3Client} instance used to interact with
the Amazon S3 service
     * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
     * @param accountId     the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS
configuration for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
```

```
    *
    * @param s3 the S3Client instance to use for the operation
    * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
    * @param accountId the expected bucket owner's account ID
    *
    * @throws S3Exception if there is an error retrieving the CORS configuration
    */
   public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
       try {
           GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
               .bucket(bucketName)
               .expectedBucketOwner(accountId)
               .build();

           GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
           List<CORSRule> corsRules = corsResponse.corsRules();
           for (CORSRule rule : corsRules) {
               System.out.println("allowOrigins: " + rule.allowedOrigins());
               System.out.println("AllowedMethod: " + rule.allowedMethods());
           }

       } catch (S3Exception e) {

           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   /**
    * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
    *
    * @param s3 The S3Client object used to interact with the Amazon S3 service.
    * @param bucketName The name of the S3 bucket to set the CORS rules for.
    * @param accountId The AWS account ID of the bucket owner.
    */
   public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
       List<String> allowMethods = new ArrayList<>();
       allowMethods.add("PUT");
```

```
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                .allowedMethods(allowMethods)
                .allowedOrigins(allowOrigins)
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutBucketCorsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie eine CORS-Regel hinzu.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
              AllowedMethods: ["GET", "PUT"],
              // Allow only requests from the specified origin.
              AllowedOrigins: ["https://www.example.com"],
              // Allow the entity tag (ETag) header to be returned in the
 response. The ETag header
              // The entity tag represents a specific version of the object. The
 ETag reflects
```

```
              // changes only to the contents of an object, not its metadata.
              ExposeHeaders: ["ETag"],
              // How long the requesting browser should cache the preflight
 response. After
              // this time, the preflight request will have to be made again.
              MaxAgeSeconds: 3600,
            },
          ],
        },
      }),
    );
    console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutBucketCorsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
 are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- Einzelheiten zur API finden Sie PutBucketCorsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
```

```
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- Einzelheiten zur API finden Sie PutBucketCorsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Allow PUT, POST, DELETE methods from http://www.example.com
        lo_s3->putbucketcors(
          iv_bucket = iv_bucket_name
          io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
            it_corsrules = it_cors_rules ) ).
        MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutBucketCorsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **PutBucketEncryption**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketEncryption` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Objektspeicher

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <param name="kmsKeyId">The Id of the KMS Key.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> SetBucketServerSideEncryption(string
 bucketName, string kmsKeyId)
    {
        var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
        {
            ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
            {
                new ServerSideEncryptionRule
                {
                    ServerSideEncryptionByDefault = new
 ServerSideEncryptionByDefault
                    {
                        ServerSideEncryptionAlgorithm =
 ServerSideEncryptionMethod.AWSKMS,
                        ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
```

```
                    }
                }
            }
        };
        try
        {
            var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
        }
        return false;
    }
```

- Einzelheiten zur API finden Sie PutBucketEncryptionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So konfigurieren Sie die serverseitige Verschlüsselung für einen Bucket

Im folgenden Beispiel für `put-bucket-encryption` wird die AES256-Verschlüsselung als Standard für den angegebenen Bucket festgelegt.

```
aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
```

```
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutBucketEncryptionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl aktiviert die serverseitige AES256-Standardverschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Einzelheiten zur API finden Sie unter PutBucketEncryption AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl aktiviert die serverseitige AES256-Standardverschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Einzelheiten zur API finden Sie PutBucketEncryptionin der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutBucketLifecycleConfiguration`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketLifecycleConfiguration` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Löschen von unvollständigen mehrteiligen Uploads
- Verwalten großer Nachrichten mit S3
- Mit versionierten Objekten arbeiten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- Einzelheiten zur API finden Sie PutBucketLifecycleConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl wendet eine Lebenszykluskonfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket` an:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
lifecycle-configuration  file://lifecycle.json
```

Die Datei `lifecycle.json` ist ein JSON-Dokument im aktuellen Ordner, das zwei Regeln angibt:

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

Die erste Regel verschiebt Dateien mit dem Präfix `rotated` am angegebenen Datum nach Glacier. Die zweite Regel verschiebt alte Objektversionen nach Glacier, wenn sie nicht mehr

aktuell sind. Informationen zu akzeptablen Zeitstempelformaten finden Sie unter „Angeben von Parameterwerten“ im Benutzerhandbuch für AWS CLI.

- Einzelheiten zur API finden Sie PutBucketLifecycleConfigurationin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <accountId>\s

            Where:
              bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 to upload an object into.
              accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
 Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
```

```
    * @throws S3Exception if there is an error setting the lifecycle
configuration.
    */
   public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
       try {
           // Create a rule to archive objects with the "glacierobjects/" prefix
to the
           // S3 Glacier Flexible Retrieval storage class immediately.
           LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
               .prefix("glacierobjects/")
               .build();

           Transition transition = Transition.builder()
               .storageClass(TransitionStorageClass.GLACIER)
               .days(0)
               .build();

           LifecycleRule rule1 = LifecycleRule.builder()
               .id("Archive immediately rule")
               .filter(ruleFilter)
               .transitions(transition)
               .status(ExpirationStatus.ENABLED)
               .build();

           // Create a second rule.
           Transition transition2 = Transition.builder()
               .storageClass(TransitionStorageClass.GLACIER)
               .days(0)
               .build();

           List<Transition> transitionList = new ArrayList<>();
           transitionList.add(transition2);

           LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
               .prefix("glacierobjects/")
               .build();

           LifecycleRule rule2 = LifecycleRule.builder()
               .id("Archive and then delete rule")
               .filter(ruleFilter2)
               .transitions(transitionList)
               .status(ExpirationStatus.ENABLED)
               .build();
```

```
            // Add the LifecycleRule objects to an ArrayList.
            ArrayList<LifecycleRule> ruleList = new ArrayList<>();
            ruleList.add(rule1);
            ruleList.add(rule2);

            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
new lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
```

```
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();
```

```
 s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
 configuration
     */
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
 String accountId) {
        try {
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
 DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutBucketLifecycleConfigurationin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
 configuration can
        be used to archive or delete the objects in the bucket according to
 specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- Einzelheiten zur API finden Sie PutBucketLifecycleConfigurationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example: Expire objects with prefix 'logs/' after 30 days
    lo_s3->putbucketlifecycleconf(
      iv_bucket = iv_bucket_name
      io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
        it_rules = it_lifecycle_rule ) ).
    MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutBucketLifecycleConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutBucketLogging`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketLogging` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();
```

```
            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
```

```
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
```

```
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
```

```
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketLoggingin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So legen Sie die Protokollierung für eine Bucket-Richtlinie fest

Im folgenden Beispiel für `put-bucket-logging` wird die Protokollierungsrichtlinie für amzn-s3-demo-bucket festgelegt. Erteilen Sie zunächst dem Prinzipal des Protokollservices mit dem `put-bucket-policy`-Befehl die Berechtigung in Ihrer Bucket-Richtlinie.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://policy.json
```

Inhalt von `policy.json`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

Wenden Sie dann die Protokollierungsrichtlinie mit `put-bucket-logging` an.

```
aws s3api put-bucket-logging \
    --bucket amzn-s3-demo-bucket \
    --bucket-logging-status file://logging.json
```

Inhalt von `logging.json`:

```
{
     "LoggingEnabled": {
         "TargetBucket": "amzn-s3-demo-bucket",
         "TargetPrefix": "Logs/"
     }
 }
```

Der `put-bucket-policy`-Befehl ist erforderlich, um dem Protokollierungsserviceprinzipal `s3:PutObject`-Berechtigungen zu erteilen.

Weitere Informationen finden Sie unter Amazon-S3-Server-Zugriffsprotokoll im Benutzerhandbuch für Amazon S3.

Beispiel 2: So legen Sie eine Bucket-Richtlinie für die Zugriffsprotokollierung auf einen einzelnen Benutzer fest

Im folgenden Beispiel für `put-bucket-logging` wird die Protokollierungsrichtlinie für amzn-s3-demo-bucket festgelegt. Der AWS Benutzer bob@example.com hat die volle Kontrolle über die Protokolldateien, und niemand sonst hat Zugriff. Erteilen Sie zunächst S3 die Berechtigung mit `put-bucket-acl`.

```
aws s3api put-bucket-acl \
    --bucket amzn-s3-demo-bucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

Wenden Sie dann die Protokollierungsrichtlinie mit `put-bucket-logging` an.

```
aws s3api put-bucket-logging \
    --bucket amzn-s3-demo-bucket \
    --bucket-logging-status file://logging.json
```

Inhalt von `logging.json`:

```
{
```

```
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-bucket",
        "TargetPrefix": "amzn-s3-demo-bucket-logs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

Der Befehl `put-bucket-acl` wird benötigt, um dem System zur Bereitstellung der Amazon-S3-Protokolle die erforderlichen Berechtigungen („write-acp“ und „read-acp“) zu gewähren.

Weitere Informationen finden Sie unter Amazon-S3-Server-Zugriffsprotokoll im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie PutBucketLoggingin der AWS CLI Befehlsreferenz.

Verwendung von **`PutBucketNotification`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketNotification` verwendet wird.

CLI

AWS CLI

Im Beispiel wird eine Benachrichtigungskonfiguration auf einen Bucket namens `amzn-s3-demo-bucket` angewendet:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-
configuration file://notification.json
```

Die Datei `notification.json` ist eine JSON-Datei im aktuellen Ordner, die ein SNS-Thema und einen zu überwachenden Ereignistyp angibt:

```
{
  "TopicConfiguration": {
```

```
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

Dem SNS-Thema muss eine IAM-Richtlinie angehängt sein, die es Amazon S3 ermöglicht, darin zu veröffentlichen.

```
{
 "Version":"2012-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
    }
   }
  }
 ]
}
```

- Einzelheiten zur API finden Sie PutBucketNotificationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis konfiguriert ObjectRemovedDelete und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
```

```
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen ObjectCreatedAll für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
```

```
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Einzelheiten zur API finden Sie unter PutBucketNotification AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis konfiguriert ObjectRemovedDelete und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen ObjectCreatedAll für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
```

```
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- Einzelheiten zur API finden Sie unter PutBucketNotification AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutBucketNotificationConfiguration`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketNotificationConfiguration` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Process-S3-Ereignisbenachrichtigungen
- Senden Sie Ereignisbenachrichtigungen an EventBridge

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic,
 sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
```

```
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketNotificationConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So aktivieren Sie die angegebenen Benachrichtigungen für einen Bucket

Im folgenden Beispiel für `put-bucket-notification-configuration` wird veranschaulicht, wie eine Benachrichtigungskonfiguration auf einen Bucket namens `amzn-s3-demo-bucket` angewendet wird. Die Datei `notification.json` ist eine JSON-Datei im aktuellen Ordner, die ein SNS-Thema und einen zu überwachenden Ereignistyp angibt:

```
aws s3api put-bucket-notification-configuration \
    --bucket amzn-s3-demo-bucket \
    --notification-configuration file://notification.json
```

Inhalt von `notification.json`:

```
{
    "TopicConfigurations": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
}
```

Dem SNS-Thema muss eine IAM-Richtlinie angehängt sein, die es Amazon S3 ermöglicht, darin zu veröffentlichen.

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
```

```
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
                }
            }
        }
    ]
}
```

- Einzelheiten zur API finden Sie PutBucketNotificationConfigurationin der AWS CLI Befehlsreferenz.

Verwenden Sie **`PutBucketPolicy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketPolicy` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;
```

```
    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
```

```
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

In diesem Beispiel können alle Benutzer jedes Objekt im amzn-s3-demo-bucket abrufen, mit Ausnahme der Objekte im. MySecretFolder Es gewährt dem Root-Benutzer des `put` Kontos auch `delete` die folgenden Berechtigungen: AWS `1234-5678-9012`

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file:// policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      },
      {
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
      },
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
         },
         "Action": [
```

```
            "s3:DeleteObject",
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }

  ]
}
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
 Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
 Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String
 policyText) {
```

```
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
```

```
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie die Richtlinie hinzu.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
```

```
    // This is a resource-based policy. For more information on resource-based
 policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            AWS: iamRoleArn,
          },
          Action: "s3:GetObject",
          Resource: `arn:aws:s3:::${bucketName}/*`,
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log(
      `GetObject access to the bucket "${bucketName}" was granted to the provided
 IAM role.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "MalformedPolicy"
    ) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". The policy was malformed.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
```

```
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- Einzelheiten zur API finden Sie PutBucketPolicyin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
```

```
  end

end
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example policy JSON string
        " iv_policy = '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
        lo_s3->putbucketpolicy(
          iv_bucket = iv_bucket_name
          iv_policy = iv_policy ).
        MESSAGE 'Bucket policy set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutBucketReplication`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketReplication` verwendet wird.

CLI

AWS CLI

So konfigurieren Sie die Replikation für einen S3-Bucket

Im folgenden Beispiel für `put-bucket-replication` wird eine Replikationskonfiguration auf den angegebenen S3-Bucket angewendet.

```
aws s3api put-bucket-replication \
    --bucket amzn-s3-demo-bucket1 \
    --replication-configuration file://replication.json
```

Inhalt von `replication.json`:

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"
            }
        }
    ]
}
```

Für den Ziel-Bucket muss die Versionsverwaltung aktiviert sein. Die angegebene Rolle muss über Schreibberechtigungen für den Ziel-Bucket verfügen und eine Vertrauensbeziehung besitzen, die es Amazon S3 ermöglicht, die Rolle zu übernehmen.

Beispiel einer Rollenberechtigungsrichtlinie:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
        }
    ]
}
```

Beispiel einer Vertrauensstellungsrichtlinie:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Titel des Themas im Benutzerhandbuch für Amazon Simple Storage Service Console.

- Einzelheiten zur API finden Sie PutBucketReplicationin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Sets the replication configuration for an Amazon S3 bucket.
    *
    * @param s3Client             the S3Client instance to use for the operation
    * @param sourceBucketName     the name of the source bucket
    * @param destBucketName       the name of the destination bucket
    * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
    * @param roleARN              the ARN of the IAM role to use for the
replication configuration
    */
   public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
       try {
           Destination destination = Destination.builder()
               .bucket(destinationBucketARN)
               .storageClass(StorageClass.STANDARD)
               .build();

           // Define a prefix filter for replication.
           ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
               .prefix("documents/")
               .build();
```

```
            // Define delete marker replication setting.
            DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
                .status(DeleteMarkerReplicationStatus.DISABLED)
                .build();

            // Create the replication rule.
            ReplicationRule replicationRule = ReplicationRule.builder()
                .priority(1)
                .filter(ruleFilter)
                .status(ReplicationRuleStatus.ENABLED)
                .deleteMarkerReplication(deleteMarkerReplication)
                .destination(destination)
                .build();

            List<ReplicationRule> replicationRuleList = new ArrayList<>();
            replicationRuleList.add(replicationRule);

            // Define the replication configuration with IAM role.
            ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
                .role(roleARN)
                .rules(replicationRuleList)
                .build();

            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());
```

```
        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketReplicationin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel legt eine Replikationskonfiguration mit einer einzigen Regel fest, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix "" im Bucket 'amzn-s3-demo-bucket' erstellt wurden, im Bucket 'amzn-s3-demo-bucket' ermöglicht. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: Dieses Beispiel legt eine Replikationskonfiguration mit mehreren Regeln fest, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "TaxDocs" erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglichen. OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
```

```
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Beispiel 3: In diesem Beispiel wird die Replikationskonfiguration für den angegebenen Bucket aktualisiert, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "" in den Bucket 'amzn-s3-demo-bucket' steuert. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). PutBucketReplicationAWS - Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel legt eine Replikationskonfiguration mit einer einzigen Regel fest, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix "" im Bucket 'amzn-s3-demo-bucket' erstellt wurden, im Bucket 'amzn-s3-demo-bucket' ermöglicht. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: Dieses Beispiel legt eine Replikationskonfiguration mit mehreren Regeln fest, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "TaxDocs" erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglichen. OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
```

```
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Beispiel 3: In diesem Beispiel wird die Replikationskonfiguration für den angegebenen Bucket aktualisiert, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "" in den Bucket 'amzn-s3-demo-bucket' steuert. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). PutBucketReplicationAWS - Tools für PowerShell

Verwendung von **`PutBucketRequestPayment`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketRequestPayment` verwendet wird.

CLI

AWS CLI

Beispiel 1: So aktivieren Sie die „requester pays“-Konfiguration für einen Bucket

Im folgenden Beispiel für `put-bucket-request-payment` wird `requester pays` für den angegebenen Bucket aktiviert.

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 1: So deaktivieren Sie die „requester pays“-Konfiguration für einen Bucket

Im folgenden Beispiel für `put-bucket-request-payment` wird `requester pays` für den angegebenen Bucket deaktiviert.

```
aws s3api put-bucket-request-payment \
    --bucket amzn-s3-demo-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutBucketRequestPaymentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert die Konfiguration für Zahlungsanforderungen für den Bucket mit dem Namen „amzn-s3-demo-bucket“, sodass der Person, die Downloads aus dem Bucket anfordert, die Kosten für den Download in Rechnung gestellt werden. Standardmäßig zahlt der Bucket-Eigentümer für Downloads. Verwenden Sie 'BucketOwner' für den Parameter RequestPaymentConfiguration _Payer, um die Zahlung für die Anforderung wieder auf die Standardwerte zurückzusetzen.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Einzelheiten zur API finden Sie unter PutBucketRequestPayment AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert die Konfiguration für Zahlungsanforderungen für den Bucket mit dem Namen „amzn-s3-demo-bucket“, sodass der Person, die Downloads aus dem Bucket anfordert, die Kosten für den Download in Rechnung gestellt werden. Standardmäßig zahlt der Bucket-Eigentümer für Downloads. Verwenden Sie 'BucketOwner' für den Parameter RequestPaymentConfiguration _Payer, um die Zahlung für die Anforderung wieder auf die Standardwerte zurückzusetzen.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Einzelheiten zur API finden Sie unter PutBucketRequestPayment AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutBucketTagging`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Objektspeicher

CLI

AWS CLI

Der folgende Befehl wendet eine Tagging-Konfiguration auf einen Bucket mit dem Namen `amzn-s3-demo-bucket` an:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://
tagging.json
```

Die Datei `tagging.json` ist ein JSON-Dokument im aktuellen Ordner, das Tags angibt:

```
{
   "TagSet": [
     {
       "Key": "organization",
```

```
        "Value": "marketing"
      }
    ]
}
```

Alternativ können Sie eine Tagging-Konfiguration für `amzn-s3-demo-bucket` direkt von der Befehlszeile aus anwenden:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- Einzelheiten zur API finden Sie PutBucketTaggingin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket namens „**`cloudtrail-test-2018`**“ an: ein Tag mit dem Schlüssel „Stage“ und dem Wert „Test“ sowie ein Tag mit dem Schlüssel „Environment“ und dem Wert „Alpha“. Um zu überprüfen, ob die Tags zum Bucket hinzugefügt wurden, führen Sie **`Get-S3BucketTagging -BucketName bucket_name`** aus. Die Ergebnisse sollten die Tags anzeigen, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **`Write-S3BucketTagging`** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die Cmdlets Resource Groups und Tagging API aus, **`Add-RGTResourceTag`** und **`Remove-RGTResourceTag`**. Verwenden Sie alternativ den Tag-Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Dieser Befehl übergibt einen Bucket namens **`cloudtrail-test-2018`** in das **`Write-S3BucketTagging`**-Cmdlet. Es wendet Tags Stage:Production und auf Department:Finance den Bucket an. Beachten Sie, dass **`Write-S3BucketTagging`** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- Einzelheiten zur API finden Sie unter PutBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket namens „**cloudtrail-test-2018**“ an: ein Tag mit dem Schlüssel „Stage“ und dem Wert „Test“ sowie ein Tag mit dem Schlüssel „Environment“ und dem Wert „Alpha“. Um zu überprüfen, ob die Tags zum Bucket hinzugefügt wurden, führen Sie **Get-S3BucketTagging -BucketName bucket_name** aus. Die Ergebnisse sollten die Tags anzeigen, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die Cmdlets Resource Groups und Tagging API aus, **Add-RGTResourceTag** und **Remove-RGTResourceTag**. Verwenden Sie alternativ den Tag-Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Dieser Befehl übergibt einen Bucket namens **cloudtrail-test-2018** in das **Write-S3BucketTagging**-Cmdlet. Es wendet Tags Stage:Production und auf Department:Finance den Bucket an. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- Einzelheiten zur API finden Sie unter PutBucketTagging AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **PutBucketVersioning** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketVersioning` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Objektspeicher

CLI

AWS CLI

Mit dem folgenden Befehl wird die Versionsverwaltung auf dem `amzn-s3-demo-bucket`-Bucket aktiviert:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

Der folgende Befehl aktiviert die Versionierung und verwendet einen MFA-Code

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled --mfa "SERIAL 123456"
```

- Einzelheiten zur API finden Sie PutBucketVersioningin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String
 bucketName){
        VersioningConfiguration versioningConfiguration =
 VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```
        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- Einzelheiten zur API finden Sie PutBucketVersioningin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Einzelheiten zur API finden Sie unter PutBucketVersioning AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Einzelheiten zur API finden Sie unter PutBucketVersioning AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example: Enable versioning on a bucket
    " iv_status = 'Enabled'
    lo_s3->putbucketversioning(
      iv_bucket = iv_bucket_name
      io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(
        iv_status = iv_status ) ).
    MESSAGE 'Bucket versioning enabled.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutBucketVersioningin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutBucketWebsite`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketWebsite` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
 PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
 client.PutBucketWebsiteAsync(putRequest);
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);
```

```
    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Es wird eine statische Webseitenkonfiguration auf einen Bucket mit dem Namen `amzn-s3-demo-bucket` angewendet:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-
configuration file://website.json
```

Die Datei `website.json` ist ein JSON-Dokument im aktuellen Ordner, das Index- und Fehlerseiten für die Webseite angibt:

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
```

```
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName> [indexdoc]\s

            Where:
```

```
              bucketName   - The Amazon S3 bucket to set the website
configuration on.\s
              indexdoc - The index document, ex. 'index.html'
                         If not specified, 'index.html' will be set.
          """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucketName = args[0];
       String indexDoc = "index.html";
       Region region = Region.US_EAST_1;
       S3Client s3 = S3Client.builder()
           .region(region)
           .build();

       setWebsiteConfig(s3, bucketName, indexDoc);
       s3.close();
   }

   /**
    * Sets the website configuration for an Amazon S3 bucket.
    *
    * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
    * @param bucketName The name of the S3 bucket to configure.
    * @param indexDoc The name of the index document to use for the website
configuration.
    */
   public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
       try {
           WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
               .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
               .build();

           PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
               .bucket(bucketName)
               .websiteConfiguration(websiteConfig)
               .build();

           s3.putBucketWebsite(pubWebsiteReq);
```

```
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Website-Konfiguration fest.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
```

```
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indexdokument „index.html“ und dem Fehlerdokument „error.html“.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Einzelheiten zur API finden Sie unter PutBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indexdokument „index.html“ und dem Fehlerdokument „error.html“.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Einzelheiten zur API finden Sie unter PutBucketWebsite AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'
```

```
# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für Ruby API-Referenz.

Verwenden Sie **`PutObject`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit Objektspeicher
- Senden von bedingten Anforderungen
- Verfolgen von Uploads und Downloads
- Arbeiten mit Amazon-S3-Objektintegrität

.NET

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
```

```
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für .NET API-Referenz.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie ein Objekt mit serverseitiger Verschlüsselung hoch.

```
    using System;
    using System.Threading.Tasks;
```

```
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
 bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
 ServerSideEncryptionMethod.AES256,
```

```
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

Fügen Sie ein Objekt mithilfe einer bedingten Anforderung ein.

```
    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
    public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
    {
```

```
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- Einzelheiten zur API finden Sie PutObjectin der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So laden Sie ein Objekt auf Amazon S3 hoch

Der folgende `put-object`-Befehl lädt ein Objekt in Amazon S3 hoch.

```
aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/MySampleImage.png \
    --body MySampleImage.png
```

Weitere Informationen zum Hochladen von Objekten finden Sie unter Hochladen von Objekten < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html > im Amazon S3 Developer Guide.

Beispiel 2: So laden Sie eine Videodatei auf Amazon S3 hoch3

Der folgende `put-object`-Befehl lädt eine Videodatei hoch.

```
aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/big-video-file.mp4 \
    --body /media/videos/f-sharp-3-data-services.mp4
```

Weitere Informationen zum Hochladen von Objekten finden Sie unter Hochladen von Objekten < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html > im Amazon S3 Developer Guide.

- Einzelheiten zur API finden Sie PutObjectin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Platzieren Sie ein Objekt in einem Bucket, indem Sie die Low-Level-API verwenden.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
```

```
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}
```

Laden Sie ein Objekt mithilfe eines Transfer-Managers in einen Bucket hoch.

```
import (
 "bytes"
 "context"
 "errors"
```

```
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
```

```
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine Datei mit einem S3Client in einen Bucket hoch.

```
   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();
```

```
        CompletableFuture<PutObjectResponse> response =
 getAsyncClient().putObject(objectRequest,
 AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }
```

Verwenden Sie an S3TransferManager, um eine Datei in einen Bucket hochzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie Tags fest.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   /**
    * Updates the tags associated with an object in an Amazon S3 bucket.
    *
    * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
    * @param bucketName the name of the S3 bucket containing the object
    * @param objectKey the key (or name) of the object in the S3 bucket
    * @throws S3Exception if there is an error updating the object's tags
    */
   public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
       try {
           GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
           List<Tag> obTags = getTaggingRes.tagSet();
           for (Tag sinTag : obTags) {
               System.out.println("The tag key is: " + sinTag.key());
               System.out.println("The tag value is: " + sinTag.value());
           }

           // Replace the object's tags with two new tags.
           Tag tag3 = Tag.builder()
               .key("Tag 3")
               .value("This is tag 3")
               .build();

           Tag tag4 = Tag.builder()
               .key("Tag 4")
               .value("This is tag 4")
               .build();

           List<Tag> tags = new ArrayList<>();
           tags.add(tag3);
           tags.add(tag4);

           Tagging updatedTags = Tagging.builder()
               .tagSet(tags)
```

```
                .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an
error occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
```

```
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie Metadaten fest.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
              <bucketName> <objectKey> <objectPath>\s
```

```
            Where:
              bucketName - The Amazon S3 bucket to upload an object into.
              objectKey - The object to upload (for example, book.pdf).
              objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
```

```
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie einen Wert für die Objektaufbewahrung fest.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3
service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);
```

```
            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutObjectunter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie das Objekt hoch.

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console
 (160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
```

```
};
```

Laden Sie das Objekt unter der Bedingung hoch, dass sein ETag mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
 such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Einzelheiten zur API finden Sie PutObjectin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie ein Objekt in einen Bucket hoch.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl lädt die einzelne Datei „local-sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei „sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“. Wenn der Parameter -Key nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Beispiel 3: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle*.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel „sample.txt“ enthält.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
 -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Einzelheiten zur API finden Sie unter PutObject AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl lädt die einzelne Datei „local-sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei „sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“. Wenn der Parameter -Key nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Beispiel 3: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle*.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel „sample.txt“ enthält.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
 -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Einzelheiten zur API finden Sie unter PutObject AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
When this
                     argument is a string, it is interpreted as a file name,
which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
```

```
                put_data.close()
```

Laden Sie ein Objekt mithilfe einer bedingten Anforderung hoch.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
 bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
 {source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
```

```
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Einzelheiten zur API finden Sie PutObjectin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine Datei mit einem verwalteten Uploader (Object.upload_file) hoch.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei hoch mit. Object.put

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
```

```
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei mit serverseitiger Verschlüsselung hoch Object.put und fügen Sie sie hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie PutObjectunter AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
```

```
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- Einzelheiten zur API finden Sie PutObjectin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
            iv_body = lv_body ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutObjectin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3
import Smithy

    public func uploadFile(bucket: String, key: String, file: String) async
 throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

```
import AWSS3
import Smithy

    public func createFile(bucket: String, key: String, withData data: Data)
 async throws {
        let dataStream = ByteStream.data(data)
```

```
        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`PutObjectAcl`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObjectAcl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verwalten von Zugriffssteuerungslisten (ACL)

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                              const Aws::String &granteePermission, const
 Aws::String &granteeType,
```

```
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
   Aws::S3::S3Client s3Client(clientConfig);

   Aws::S3::Model::Owner owner;
   owner.SetID(ownerID);

   Aws::S3::Model::Grantee grantee;
   grantee.SetType(setGranteeType(granteeType));

   if (!granteeEmailAddress.empty()) {
       grantee.SetEmailAddress(granteeEmailAddress);
   }

   if (!granteeID.empty()) {
       grantee.SetID(granteeID);
   }

   if (!granteeURI.empty()) {
       grantee.SetURI(granteeURI);
   }

   Aws::S3::Model::Grant grant;
   grant.SetGrantee(grantee);
   grant.SetPermission(setGranteePermission(granteePermission));

   Aws::Vector<Aws::S3::Model::Grant> grants;
   grants.push_back(grant);

   Aws::S3::Model::AccessControlPolicy acp;
   acp.SetOwner(owner);
   acp.SetGrants(grants);

   Aws::S3::Model::PutObjectAclRequest request;
   request.SetAccessControlPolicy(acp);
   request.SetBucket(bucketName);
   request.SetKey(objectKey);

   Aws::S3::Model::PutObjectAclOutcome outcome =
           s3Client.PutObjectAcl(request);

   if (!outcome.IsSuccess()) {
```

```
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
```

```
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Einzelheiten zur API finden Sie PutObjectAclin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl erhalten `full control` zwei AWS Benutzer (user1@example.com und user2@example.com) Zugriff `read` auf alle Benutzer:

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-full-control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.htmlEinzelheiten zu benutzerdefinierten ACLs finden Sie unter (die s3api-ACL-Befehle verwenden z. `put-object-acl` B. dieselbe Kurzschreibweise für Argumente).

- Einzelheiten zur API finden Sie unter Befehlsreferenz. PutObjectAclAWS CLI

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- Einzelheiten zur API finden Sie PutObjectAclin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example: Grant read access to an AWS user
    " iv_grantread = 'emailAddress=user@example.com'
    lo_s3->putobjectacl(
      iv_bucket = iv_bucket_name
      iv_key = iv_object_key
      iv_grantread = iv_grantread ).
    MESSAGE 'Object ACL updated.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie PutObjectAclin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutObjectLegalHold`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObjectLegalHold` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
```

```
    }
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So wenden Sie eine gesetzliche Aufbewahrungsfrist auf ein Objekt an

Im folgenden Beispiel für `put-object-legal-hold` wird für das Objekt `doc1.rtf` eine gesetzliche Aufbewahrungsfrist eingerichtet.

```
aws s3api put-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
       System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
   }
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" |
 "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'.",
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });
```

```
  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
 "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
      required: true,
    },
```

```
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
```

```
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
        )
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Set legal hold status to ON
        " iv_status = 'ON'
        lo_s3->putobjectlegalhold(
```

```
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
            iv_status = iv_status ) ).
        MESSAGE 'Object legal hold status set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutObjectLegalHoldin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutObjectLockConfiguration`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObjectLockConfiguration` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
    /// <summary>
```

```
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };
```

```
            var response = await
 _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
 {bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Object-Lock-Konfiguration eines Buckets fest

Im folgenden Beispiel für `put-object-lock-configuration` wird ein 50-tägiges Object Lock für den angegebenen Bucket eingerichtet.

```
aws s3api put-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
 { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
```

```
    MFADelete: types.MFADeleteDisabled,
    Status:    types.BucketVersioningStatusEnabled,
   },
  }
  _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   return err
  }

  input := &s3.PutObjectLockConfigurationInput{
   Bucket: aws.String(bucket),
   ObjectLockConfiguration: &types.ObjectLockConfiguration{
    ObjectLockEnabled: types.ObjectLockEnabledEnabled,
   },
  }
  _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  }

  return err
}
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }
```

```
 return err
}
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
```

```
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
```

```
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
 PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


 getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
 +".");
    }
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
```

```
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
 bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
 bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
 bucket.
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
```

```
          Rule: {
            // The default Object Lock retention mode and period that you want to
apply
            // to new objects placed in the specified bucket. Bucket settings
require
            // both a mode and a period. The period can be either Days or Years
but
            // you must select one.
            DefaultRetention: {
              // In governance mode, users can't overwrite or delete an object
version
              // or alter its lock settings unless they have special permissions.
With
              // governance mode, you protect objects against being deleted by
most users,
              // but you can still grant some users permission to alter the
retention settings
              // or delete the objects if necessary.
              Mode: "GOVERNANCE",
              Days: Number.parseInt(retentionDays),
            },
          },
        },
      },
    );
    console.log(
      `Set default retention mode to "GOVERNANCE" with a retention period of
${retentionDays} day(s).`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
${caught.name}: ${caught.message}`,
      );
    } else {
```

```
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

PUT-Object Lock-Konfiguration.

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}},
        )
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Enable object lock with default retention
        " iv_enabled = 'Enabled'
        lo_s3->putobjectlockconfiguration(
          iv_bucket = iv_bucket_name
          io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
            iv_objectlockenabled = iv_enabled ) ).
        MESSAGE 'Object lock configuration set.' TYPE 'I'.
```

```
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutObjectLockConfigurationin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutObjectRetention`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutObjectRetention` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Sperren von Amazon-S3-Objekten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
```

```
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
 until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
 '{ex.Message}'");
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectRetentionin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Konfiguration für die Aufbewahrung eines Objekts fest

Im folgenden Beispiel für `put-object-retention` wird eine Konfiguration für die Aufbewahrung des angegebenen Objekts bis zum 01.01.2025 festgelegt.

```
aws s3api put-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
 "2025-01-01T00:00:00" }'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie PutObjectRetentionin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
```

```
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- Einzelheiten zur API finden Sie PutObjectRetentionin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);
```

```
        // Convert the Instant to a ZonedDateTime object with a specific time
zone.
        ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }
```

- Einzelheiten zur API finden Sie PutObjectRetentionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
 users,
      // but you can still grant some users permission to alter the retention
 settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
```

```
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
 on an object. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period
 on an object. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Einzelheiten zur API finden Sie PutObjectRetentionin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31st Dec 2019 00:00:00“ für das Objekt „testfile.txt“ im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Einzelheiten zur API finden Sie unter PutObjectRetention AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31st Dec 2019 00:00:00“ für das Objekt „testfile.txt“ im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Einzelheiten zur API finden Sie unter PutObjectRetention AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine Objektaufbewahrung fest.

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
                BypassGovernanceRetention=True,
            )
```

- Einzelheiten zur API finden Sie PutObjectRetentionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Set retention mode to GOVERNANCE for 30 days
        " iv_mode = 'GOVERNANCE'
        " iv_retain_date should be a timestamp in the future
        lo_s3->putobjectretention(
```

```
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
          iv_bypassgovernanceretention = abap_true ).
        MESSAGE 'Object retention set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutObjectRetentionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`RestoreObject`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `RestoreObject` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
```

```
    public class RestoreArchivedObject
    {
        public static void Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string objectKey = "archived-object.txt";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            IAmazonS3 client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        /// <summary>
        /// This method restores an archived object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
 client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
```

```
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- Einzelheiten zur API finden Sie RestoreObjectin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Wiederherstellungsanforderung für ein Objekt

Im folgenden Beispiel für `restore-object` wird das angegebene Objekt von Amazon S3 Glacier für den Bucket `my-glacier-bucket` für 10 Tage wiederhergestellt.

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie RestoreObjectin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
```

```
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
 Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3                   an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName           the name of the S3 bucket where the object is
 stored
     * @param keyName              the key (object name) of the S3 object to be
 restored
```

```
     * @param expectedBucketOwner  the AWS account ID of the expected bucket
owner
     */
    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie RestoreObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`SelectObjectContent`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SelectObjectContent` verwendet wird.

CLI

AWS CLI

So filtern Sie den Inhalt eines Amazon-S3-Objekts auf der Grundlage einer SQL-Anweisung

Im folgenden Beispiel für `select-object-content` wird das Objekt `my-data-file.csv` mit der angegebenen SQL-Anweisung gefiltert und die Ausgabe wird an eine Datei gesendet.

```
aws s3api select-object-content \
    --bucket amzn-s3-demo-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SelectObjectContentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Beispiel zeigt eine Abfrage mit einem JSON-Objekt. Das vollständige Beispiel zeigt auch die Verwendung eines CSV-Objekts.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
```

```
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
 bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
```

```
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
```

```
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
 buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
 information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
 SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
```

```
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- Einzelheiten zur API finden Sie SelectObjectContentin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UploadPart`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UploadPart` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Verwenden der Prüfsummen
- Arbeiten mit Amazon-S3-Objektintegrität

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
 &calculatedHash,
```

```
                                                                const
std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
&client) {
   Aws::S3::Model::UploadPartRequest request;
   request.SetBucket(bucket);
   request.SetKey(key);
   request.SetUploadId(uploadID);
   request.SetPartNumber(partNumber);
   if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
       request.SetChecksumAlgorithm(checksumAlgorithm);
   }
   request.SetBody(body);

   if (!calculatedHash.empty()) {
       switch (checksumAlgorithm) {
           case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
               request.SetContentMD5(calculatedHash);
               break;
           case Aws::S3::Model::ChecksumAlgorithm::CRC32:
               request.SetChecksumCRC32(calculatedHash);
               break;
           case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
               request.SetChecksumCRC32C(calculatedHash);
               break;
           case Aws::S3::Model::ChecksumAlgorithm::SHA1:
               request.SetChecksumSHA1(calculatedHash);
               break;
           case Aws::S3::Model::ChecksumAlgorithm::SHA256:
               request.SetChecksumSHA256(calculatedHash);
               break;
       }
   }

   return client.UploadPart(request);
}
```

- Einzelheiten zur API finden Sie UploadPartin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird der erste Teil eines mehrteiligen Uploads hochgeladen, der mit dem Befehl `create-multipart-upload` initiiert wurde:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
 'multipart/01' --part-number 1 --body part01 --upload-id
  "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

Die `body`-Option verwendet den Namen oder Pfad einer lokalen Datei für den Upload (verwenden Sie nicht das Präfix file://). Die Mindestgröße von Teilen beträgt 5 MB. Die Upload-ID wird von `create-multipart-upload` zurückgegeben und kann auch mit `list-multipart-uploads` abgerufen werden. Bucket und Schlüssel werden angegeben, wenn Sie den mehrteiligen Upload erstellen.

Ausgabe:

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Speichern Sie den ETag-Wert der einzelnen Teile für später. Diese Werte werden benötigt, um den mehrteiligen Upload abzuschließen.

- Einzelheiten zur API finden Sie UploadPartin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();
```

```
    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;
```

```
    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- Einzelheiten zur API finden Sie UploadPartin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`UploadPartCopy`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UploadPartCopy` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen einer mehrteiligen Kopie

CLI

AWS CLI

So laden Sie einen Teil eines Objekts durch das Kopieren von Daten aus einem vorhandenen Objekt als Datenquelle hoch

Im folgenden Beispiel für `upload-part-copy` wird ein Teil durch Kopieren von Daten aus einem vorhandenen Objekt hochgeladen, das als Datenquelle dient.

```
aws s3api upload-part-copy \
    --bucket amzn-s3-demo-bucket \
    --key "Map_Data_June.mp4" \
    --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \
    --part-number 1 \
    --upload-
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0ggOH6b13pVRY7QjvLlf75iFdJqp_2wztk5hvpUM2SesXgrz
```

Ausgabe:

```
{
    "CopyPartResult": {
        "LastModified": "2019-12-13T23:16:03.000Z",
        "ETag": "\"711470fc377698c393d94aed6305e245\""
    }
}
```

- Einzelheiten zur API finden Sie UploadPartCopyin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public CompletableFuture<String> performMultiCopy(String toBucket, String
 bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
```

```
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- Einzelheiten zur API finden Sie UploadPartCopyin der AWS SDK for Java 2.x API-Referenz.

Szenarien für die Verwendung von Amazon S3 AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon S3 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon S3 oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Prüfen, ob ein Bucket vorhanden ist
- Prüfen Sie, ob ein Objekt existiert
- Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK
- Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK
- Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK
- Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK
- Erkennen Sie Gesichter in einem Bild mit einem AWS SDK
- Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service (Amazon S3)
- Herunterladen aller Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis
- Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK
- Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK
- Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers
- Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK
- Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK

- Erste Schritte mit Big-Data-Verarbeitungsclustern
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit der Textextraktion von Dokumenten
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit Objektspeicher
- Erste Schritte mit Query Analytics
- Arbeiten Sie mit den Objektsperrfunktionen von Amazon S3 mithilfe eines AWS SDK
- Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK
- Verwalten Sie Zugriffskontrolllisten (ACLs) für Amazon S3 S3-Buckets mithilfe eines AWS SDK
- Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK
- Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK
- Analysieren Sie Amazon S3 S3-URIs mit einem AWS SDK
- Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK
- Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK
- Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK
- Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK
- Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK
- Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda
- Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK
- Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service (Amazon S3)-Bucket
- Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK
- Laden Sie einen Stream unbekannter Größe mit einem in ein Amazon S3 S3-Objekt hoch AWS SDK
- Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie ein AWS SDK
- Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK
- Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK

Prüfen, ob ein Bucket vorhanden ist

Das folgende Codebeispiel zeigt, wie Sie überprüfen, ob ein Bucket vorhanden ist.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sie können die folgende `doesBucketExists` Methode als Ersatz für die SDK for Java V1 AmazonS3Client #does BucketExist V2 (String) -Methode verwenden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
 bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
 s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
```

```
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
 be used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks
 for the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the
 value false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket
 exists but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
```

```
    }
}
```

- Einzelheiten zur API finden Sie GetBucketAclin AWS SDK for Java 2.x der API-Referenz.

Prüfen Sie, ob ein Objekt existiert

Das folgende Codebeispiel zeigt, wie überprüft wird, ob ein Objekt existiert.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sie können die folgende `doesObjectExist` Methode als Ersatz für die SDK for Java V1 AmazonS3Client #does ObjectExist (String, String) -Methode verwenden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
 bucket name.
        final String key = "my-key"; // Replace with your object key.
```

```
        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
 s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with
 404 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may
 return 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName   The name of the bucket containing the object.
     * @param key          The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
 exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
 s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
 string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build());
            return true;
```

```
        } catch (NoSuchKeyException e) {
            return false;
        }
    }
}
```

- Einzelheiten zur API finden Sie HeadObjectin AWS SDK for Java 2.x der API-Referenz.

Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Klartext-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

Rust

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, die Audiodatei in einen Amazon S3 S3-Bucket hochzuladen, Amazon Transcribe zu verwenden, um diese Audiodatei in Text zu konvertieren und den Text anzuzeigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine Amazon-S3-Aktion ausführen kann.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
set to Version 4 as shown below.
```

```
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName,
 objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
 bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }
```

```
            return urlString;
        }
    }
```

Generieren Sie eine vorsignierte URL und führen Sie ein Upload mit dieser URL durch.

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
KMS, you must specify Signature Version 4.
```

```
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
 passed in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
```

```
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

SDK für .NET (v4)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und verwenden Sie vorsignierte POST URLs für direkte Browser-Uploads.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
 host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Perform the actions defined for the Amazon S3 Presigned POST scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
```

```
        _isInteractive = !args.Contains("--non-interactive");

        // Set up dependency injection for Amazon S3
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

        SetUpServices(host);

        try
        {
            // Display overview
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(_isInteractive);

            // Step 1: Create bucket
            await CreateBucketAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 2: Create presigned URL
            _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
            var response = await CreatePresignedPostAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 3: Display URL and fields
            _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
            DisplayPresignedPostFields(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 4: Upload file
            _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
            await UploadFileAsync(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 5: Verify file exists
            await VerifyFileExistsAsync();
            _uiMethods.PressEnter(_isInteractive);
```

```
            // Step 6: Cleanup
            _uiMethods.DisplayTitle("Step 6: Clean up resources");
            await CleanupAsync();

            _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
 successfully!");
            _uiMethods.PressEnter(_isInteractive);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error in scenario");
            Console.WriteLine($"Error: {ex.Message}");

            // Attempt cleanup if there was an error
            if (!string.IsNullOrEmpty(_bucketName))
            {
                _uiMethods.DisplayTitle("Cleaning up resources after error");
                await _s3Wrapper.DeleteBucketAsync(_bucketName);
                Console.WriteLine($"Cleaned up bucket: {_bucketName}");
            }
        }
    }

    /// <summary>
    /// Create an S3 bucket for the scenario.
    /// </summary>
    private static async Task CreateBucketAsync()
    {
        _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

        // Generate a default bucket name for the scenario
        var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

        // Prompt user for bucket name or use default in non-interactive mode
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'): ",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrWhiteSpace(_bucketName))
        {
            _bucketName = defaultBucketName;
```

```
        }

        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }


    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
 CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
 the demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse
 response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
 scenario.";
```

```
        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
    {
        try
        {
            _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

            using var httpClient = _httpClientFactory.CreateClient();
            using var formContent = new MultipartFormDataContent();
```

```
            // Add all the fields from the presigned POST response
            foreach (var field in response.Fields)
            {
                formContent.Add(new StringContent(field.Value), field.Key);
            }

            // Add the file content
            var fileStream = File.OpenRead(filePath);
            var fileName = Path.GetFileName(filePath);
            var fileContent = new StreamContent(fileStream);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
            formContent.Add(fileContent, "file", fileName);

            // Send the POST request
            var httpResponse = await httpClient.PostAsync(response.Url,
 formContent);
            var responseContent = await httpResponse.Content.ReadAsStringAsync();

            // Log and return the result
            _logger.LogInformation("Upload completed with status code
 {statusCode}", httpResponse.StatusCode);

            return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
 responseContent);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error uploading file");
            return (false, HttpStatusCode.InternalServerError, ex.Message);
        }
    }

    /// <summary>
    /// Verify that the uploaded file exists in the S3 bucket.
    /// </summary>
    private static async Task VerifyFileExistsAsync()
    {
        _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

        Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

        try
```

```
        {
            var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
_objectKey!);

            Console.WriteLine($"File verification successful! File exists in the
bucket.");
            Console.WriteLine($"File size: {metadata.ContentLength} bytes");
            Console.WriteLine($"File type: {metadata.Headers.ContentType}");
            Console.WriteLine($"Last modified: {metadata.LastModified}");
        }
        catch (AmazonS3Exception ex) when (ex.StatusCode ==
System.Net.HttpStatusCode.NotFound)
        {
            Console.WriteLine($"Error: File was not found in the bucket.");
            throw;
        }
    }

    private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
    {
        Console.WriteLine($"Presigned POST URL: {response.Url}");
        Console.WriteLine("Form fields to include:");

        foreach (var field in response.Fields)
        {
            Console.WriteLine($"  {field.Key}: {field.Value}");
        }
    }

    /// <summary>
    /// Clean up resources created by the scenario.
    /// </summary>
    private static async Task CleanupAsync()
    {
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
```

```
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
 already deleted");
            }
        }
    }
}
```

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, um ein Objekt herunterzuladen.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
```

```
                                            expirationSeconds);
}
```

Führen Sie einen Download unter Verwendung von libcurl durch.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }
```

```
    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

Generieren Sie eine vorsignierte URL, um ein Objekt hochzuladen.

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
```

```
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                        expirationSeconds);
}
```

Führen Sie einen Upload unter Verwendung von libcurl durch.

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Vorsignierungsaktionen umschließen.

```
import (
 "context"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(ctx,
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
```

```
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(ctx,
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
 objectKey string) (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(ctx,
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}



func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
 string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
 {
 request, err := presigner.PresignClient.PresignPostObject(ctx,
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(options *s3.PresignPostOptions) {
```

```
  options.Expires = time.Duration(lifetimeSecs) * time.Second
 })
 if err != nil {
  log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
\n", bucketName, objectKey, err)
 }
 return request, nil
}
```

Führen Sie ein interaktives Beispiel aus, das vorsignierte URLs generiert und verwendet, um ein S3-Objekt hochzuladen, herunterzuladen und zu löschen.

```
import (
 "bytes"
 "context"
 "io"
 "log"
 "mime/multipart"
 "net/http"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)



// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
```

```
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 presigning demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
 presignClient := s3.NewPresignClient(s3Client)
 presigner := actions.Presigner{PresignClient: presignClient}

 bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
  "you own or one you want to create:", demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
```

```
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
 panic(err)
}
```

```
 defer uploadFile.Close()
 multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
 presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
 if err != nil {
  panic(err)
 }
 log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
 multiPartResponse.StatusCode)

 log.Println("Let's presign a request to delete the object.")
 questioner.Ask("Press Enter when you're ready.")
 presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Got a presigned %v request to URL:\n\t%v\n",
 presignedDelRequest.Method,
  presignedDelRequest.URL)
 log.Println("Using net/http to send the request...")
 delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie einen HTTP-Request-Wrapper, der im Beispiel verwendet wird, um HTTP-Anfragen zu stellen.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
```

```
 Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
 *http.Response, err error) {
 postRequest, err := http.NewRequest("POST", url, body)
 if err != nil {
  return nil, err
 }
 postRequest.Header.Set("Content-Type", contentType)
 return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im Folgenden werden drei Beispiele gezeigt, wie vorsignierte URLs erstellt und die URLs mit HTTP-Clientbibliotheken verwendet werden:

- Eine HTTP-GET-Anfrage, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-GET-Anfrage mit Metadaten in den Headern, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-PUT-Anfrage mit Abfrageparametern, die die URL mit einer HTTP-Clientbibliothek verwendet

Generieren Sie eine vorsignierte URL für ein Objekt und laden Sie sie dann herunter (GET-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;
```

```
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generieren Sie die URL.

```
    /* Create a pre-signed URL to download an object in a subsequent GET request.
 */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will
 expire in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
 presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]",
 presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
```

```
    }
```

Laden Sie das Objekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `HttpURLConnection` (seit v1.1), um den Download durchzuführen.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Download durchzuführen.

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
```

```
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Verwenden Sie die `SdkHttpClient` Klasse AWS SDK for Java, um den Download durchzuführen.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
```

```
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Generieren Sie eine vorsignierte URL mit Metadaten in den Headern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
```

```
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generieren Sie die URL.

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
 Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
 PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
 expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
 presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());
```

```
            return presignedRequest.url().toExternalForm();
        }
    }
```

Laden Sie ein Dateiobjekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `HttpURLConnection` (seit v1.1), um den Upload durchzuführen.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File
 fileToPut, Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());
```

```
        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Upload durchzuführen.

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

 .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

```
    /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);
```

```
            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());
            // Add headers
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
 k, v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Generieren Sie eine vorsignierte URL mit Abfrageparametern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generieren Sie die URL.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 adds query parameters
    *  to the request instead of using headers. By using query parameters, you
 do not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned
 URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName,
 Map<String, String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
 = AwsRequestOverrideConfiguration.builder();


 queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);
```

```
            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)

.overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
configuration.
                    .build();

            PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


            PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

```
    /**
     * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
request. Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
        logger.info("Begin [{}] upload", fileToPut.toString());
```

```
        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte URL, um ein Objekt in einen Bucket hochzuladen.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
```

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
```

```
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
```

```
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Erstellen Sie eine vorsignierte URL, um ein Objekt aus einem Bucket herunterzuladen.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
```

```
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte `GetObject`-Anfrage und verwenden Sie die URL, um ein Objekt herunterzuladen.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Erstellen Sie eine `GetObject`-vorsignierte Anfrage mit erweiterten Optionen.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Erstellen Sie eine vorsignierte `PutObject`-Anfrage und verwenden Sie sie, um ein Objekt hochzuladen.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }
```

```
    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';
```

```
class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
```

```
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.
\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }
```

```
   public function putObject(string $bucketName, string $key, array $args = [])
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
       try {
           $this->client->putObject($parameters);
           if ($this->verbose) {
               echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object uploading before continuing.";
           }
           throw $exception;
       }
   }



   public function getObject(string $bucketName, string $key, array $args = []):
Result
   {
       $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
       try {
           $object = $this->client->getObject($parameters);
           if ($this->verbose) {
               echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
           }
       } catch (AwsException $exception) {
           if ($this->verbose) {
               echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
               echo "Please fix error with object downloading before
continuing.";
           }
           throw $exception;
```

```
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
```

```
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args
 = [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
 ['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
```

```
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
 error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
```

```
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine S3-Aktion ausführen kann. Verwenden Sie das Anforderungspaket, um eine Anforderung mit der URL zu stellen.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
```

```
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

Generieren Sie eine vorsignierte POST-Anforderung zum Hochladen einer Datei.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url
```

```
  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
 'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Vorsignieranforderungen für GET-S3-Objekte.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;
```

```
    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Erstellen Sie Vorsignieranforderungen für GET- und PUT-S3-Objekte.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
 std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}
```

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie vorsignierte Anforderungen für GET-S3-Objekte.

```
" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service.  It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
  iv_bucket = iv_bucket_name
  iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file.  The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK

Im folgenden Codebeispiel wird veranschaulicht, wie Sie Amazon-S3-Objekte auf einer Webseite auflisten.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Der folgende Code ist die relevante React-Komponente, die das AWS SDK aufruft. Eine lauffähige Version der Anwendung, die diese Komponente enthält, finden Sie unter dem vorherigen GitHub Link.

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
```

```
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- Einzelheiten zur API finden Sie unter ListObjects AWS SDK für JavaScriptAPI-Referenz.

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die

Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie alle Objekte aus einem Amazon-S3-Bucket löschen.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie alle Objekte für einen gegebenen Amazon-S3-Bucket.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
```

```
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - DeleteObjects
  - ListObjectsV2

Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie unvollständige mehrteilige Amazon-S3-Uploads löschen oder beenden.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um mehrteilige Uploads zu beenden, die gerade bearbeitet werden oder aus irgendeinem Grund unvollständig sind, können Sie eine Liste mit Uploads abrufen und sie dann löschen, wie im folgenden Beispiel gezeigt.

```
   /**
    * Aborts all incomplete multipart uploads from the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket
    */
   public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(upload.key())
               .expectedBucketOwner(accountId)
               .uploadId(upload.uploadId())
               .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
           }
       }
   }
```

Um unvollständige mehrteilige Uploads zu löschen, die vor oder nach einem Datum initiiert wurden, können Sie mehrteilige Uploads anhand eines bestimmten Zeitpunkts selektiv löschen, wie im folgenden Beispiel gezeigt.

```
   /**
    * Aborts incomplete multipart uploads older than the specified point in
time.
    *
    * @param bucketName  the name of the S3 bucket
    * @param pointInTime the cutoff time; uploads initiated before this are
aborted
    */
   static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
           if (upload.initiated().isBefore(pointInTime)) {
               abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(upload.key())
                   .expectedBucketOwner(accountId)
                   .uploadId(upload.uploadId())
                   .build();

               AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
               if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                   logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
               }
           }
       }
   }
```

Wenn Sie nach dem Start eines mehrteiligen Uploads Zugriff auf die Upload-ID haben, können Sie den laufenden Upload mithilfe der ID löschen.

```
    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
 {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

Um unvollständige mehrteilige Uploads, die älter als eine bestimmte Anzahl von Tagen sind, dauerhaft zu löschen, richten Sie eine Bucket-Lebenszykluskonfiguration für den Bucket ein. Das folgende Beispiel zeigt, wie Sie eine Regel erstellen, um unvollständige Uploads zu löschen, die älter als 7 Tage sind.

```
    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
 days.
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules =
 List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
```

```
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

Python

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter TextractAndComprehendNotebook.ipynb.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Erkennen Sie Gesichter in einem Bild mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

Rust

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

.NET

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition zusammen mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.

- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie mit AWS SDK für Python (Boto3) dem eine Webanwendung erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.

- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingeschriebene Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.
- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service (Amazon S3)

Das folgende Codebeispiel zeigt, wie Sie den Inhalt der Verzeichnisse von Amazon-S3-Buckets herunterladen und filtern.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie die S3TransferManagerin verwenden AWS SDK for Java 2.x , um „Verzeichnisse“ aus einem Amazon S3 S3-Bucket herunterzuladen. Es zeigt auch, wie man es DownloadFiltersin der Anfrage verwendet.

```
    /**
     * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
     * an object to an S3 bucket, you specify a key, which is essentially the
"path" to the object. The key can contain
     * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
```

```
     * logical representation, not an actual directory.
     * <p><pre>
     * In this example, our S3 bucket contains the following objects:
     *
     * folder1/file1.txt
     * folder1/file2.txt
     * folder1/file3.txt
     * folder2/file1.txt
     * folder2/file2.txt
     * folder2/file3.txt
     * folder3/file1.txt
     * folder3/file2.txt
     * folder3/file3.txt
     *
     * When method `downloadS3Directories` is invoked with
     * `destinationPathURI` set to `/test`, the downloaded
     * directory looks like:
     *
     * |- test
     *    |- folder1
     *       |- file1.txt
     *       |- file2.txt
     *       |- file3.txt
     *    |- folder3
     *       |- file1.txt
     *       |- file2.txt
     *       |- file3.txt
     * </pre>
     *
     * @param transferManager    An S3TransferManager instance.
     * @param destinationPathURI local directory to hold the downloaded S3
 'directories' and files.
     * @param bucketName         The S3 bucket that contains the 'directories' to
 download.
     * @return The number of objects (files, in this case) that were downloaded.
     */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String
 bucketName) {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder1/");
```

```
        DownloadFilter folder3Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
 folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
 was not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }
```

Herunterladen aller Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis

Das folgende Codebeispiel zeigt, wie Sie alle Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis herunterladen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie an S3TransferManager, um alle S3-Objekte im selben S3-Bucket herunterzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();
```

```
        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- Einzelheiten zur API finden Sie DownloadDirectoryunter AWS SDK for Java 2.x API-Referenz.

Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe aus einem Amazon S3-Objekt herunterladen.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Download a file from the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to get the file from.
    ///   - key: The name (or path) of the file to download from the bucket.
    ///   - destPath: The pathname on the local filesystem at which to store
```

```
   ///     the downloaded file.
   func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
       let fileURL: URL

       // If no destination path was provided, use the key as the name to use
       // for the file in the downloads folder.

       if destPath == nil {
           do {
               try fileURL = FileManager.default.url(
                   for: .downloadsDirectory,
                   in: .userDomainMask,
                   appropriateFor: URL(string: key),
                   create: true
               ).appendingPathComponent(key)
           } catch {
               throw TransferError.directoryError
           }
       } else {
           fileURL = URL(fileURLWithPath: destPath!)
       }

       let config = try await S3Client.S3ClientConfiguration(region: region)
       let s3Client = S3Client(config: config)

       // Create a `FileHandle` referencing the local destination. Then
       // create a `ByteStream` from that.

       FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
       let fileHandle = try FileHandle(forWritingTo: fileURL)

       // Download the file using `GetObject`.

       let getInput = GetObjectInput(
           bucket: bucket,
           key: key
       )

       do {
           let getOutput = try await s3Client.getObject(input: getInput)

           guard let body = getOutput.body else {
```

```
                throw TransferError.downloadError("Error: No data returned for
download")
            }

            // If the body is returned as a `Data` object, write that to the
            // file. If it's a stream, read the stream chunk by chunk,
            // appending each chunk to the destination file.

            switch body {
            case .data:
                guard let data = try await body.readData() else {
                    throw TransferError.downloadError("Download error")
                }

                // Write the `Data` to the file.

                do {
                    try data.write(to: fileURL)
                } catch {
                    throw TransferError.writeError
                }
                break

            case .stream(let stream as ReadableStream):
                while (true) {
                    let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)
                    guard let chunk = chunk else {
                        break
                    }

                    // Write the chunk to the destination file.

                    do {
                        try fileHandle.write(contentsOf: chunk)
                    } catch {
                        throw TransferError.writeError
                    }
                }

                break
            default:
                throw TransferError.downloadError("Received data is unknown
object type")
```

```
            }
        } catch {
            throw TransferError.downloadError("Error downloading the file:
\(error)")
        }

        print("File downloaded to \(fileURL.path).")
    }
```

Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK

Das folgende Codebeispiel zeigt, wie ein Objekt von einem Multi-Region Access Point abgerufen wird.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Client so, dass er den Signaturalgorithmus Asymmetric Sigv4 (SigV4a) verwendet.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)
signing algorithm.
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Verwenden Sie den Multi-Region Access Point-ARN anstelle eines Bucket-Namens, um das Objekt abzurufen.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Kotlin.

Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers

Das folgende Codebeispiel zeigt, wie Sie Daten aus einem Objekt in einem S3 Bucket lesen, jedoch nur, wenn dieser Bucket seit dem letzten Abruf nicht geändert wurde.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
```

```
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
```

```
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
```

```
        Err(err) => match err {
            SdkError::ServiceError(err) => {
                // Get the raw HTTP response. If its status is 304, the
                // object has not changed. This is the expected code path.
                let http = err.raw();
                match http.status().as_u16() {
                    // If the HTTP status is 304: Not Modified, return a
                    // tuple containing the values of the HTTP
                    // `last-modified` and `etag` headers.
                    304 => (
                        Ok(DateTime::from_str(
                            http.headers().get("last-modified").unwrap(),
                            DateTimeFormat::HttpDate,
                        )
                        .unwrap()),
                        http.headers().get("etag").map(|t| t.into()).unwrap(),
                    ),
                    // Any other HTTP status code is returned as an
                    // `SdkError::ServiceError`.
                    _ => (Err(SdkError::ServiceError(err)), String::new()),
                }
            }
            // Any other kind of error is returned in a tuple containing the
            // error and an empty string.
            _ => (Err(err), String::new()),
        },
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;
```

```
    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- Einzelheiten zur API finden Sie GetObjectin der API-Referenz zum AWS SDK für Rust.

Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht die ersten Schritte mit der Verschlüsselung von Amazon-S3-Objekten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to apply client encryption to an object in an
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class SSEClientEncryption
```

```
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "exampleobject.txt";
            string copyTargetKeyName = "examplecopy.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
```

```
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectAsync.</param>
```

```
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
 S3.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
 client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
 StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }
```

```
                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
```

```
            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
```

```
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CopyObject
  - GetObject
  - GetObjectMetadata

Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt die ersten Schritte mit Tags für Amazon-S3-Objekte.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
```

```
    /// Service (Amazon S3) objects.
    /// </summary>
    public class ObjectTag
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "newobject.txt";
            string filePath = @"*** file path ***";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
 bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
```

```
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
```

```
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectTaggingin der AWS SDK für .NET API-Referenz.

Erste Schritte mit Big-Data-Verarbeitungsclustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}
```

```
# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
 true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
 then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi
```

```
# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 || true
```

```
# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
 find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
 's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
 violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
```

```
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi
```

```
unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
 data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
 text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
```

```
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
```

```
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
 period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
```

```
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
 "$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
 files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
 | head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
 | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv
```

```
echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb

- Rb
- Rm
- SSH
- TerminateClusters
- Wait

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
```

```
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
```

```
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
```

```
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
```

```
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
```

```
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
```

```
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
```

```
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
```

```
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
```

```
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole

- DeleteRolePolicy
- DeleteTopic
- DescribeConfigurationRecorderStatus
- DescribeConfigurationRecorders
- DescribeDeliveryChannels
- DetachRolePolicy
- GetCallerIdentity
- PutConfigurationRecorder
- PutDeliveryChannel
- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Erste Schritte mit der Textextraktion von Dokumenten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines S3-Buckets
- Laden Sie ein Dokument auf S3 hoch
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "====================================================="
echo "This script will guide you through using Amazon Textract to analyze
 document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi
```

```
    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
 delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
 then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi
```

```
# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
```

```
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
 50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
```

```
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "==================================================="
echo "Analysis Summary"
echo "==================================================="
```

```
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
 '{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
 echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
 echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
 echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
 echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
 | wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
 wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "====================================================="
echo "RESOURCES CREATED"
echo "====================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "====================================================="
echo "CLEANUP CONFIRMATION"
echo "====================================================="
```

```
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "==================================================="
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AnalyzeDocument
  - Cp
  - Hilfe
  - Mb

- Rb
- Rm

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
```

```
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
```

```
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)
```

```
    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
```

```
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
```

```
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
```

```
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
```

```
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
```

```
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi
```

```
echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
```

```
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
```

```
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Erste Schritte mit Objektspeicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Ihren ersten S3-Bucket
- Hochladen eines Objekts
- Aktivieren von Versioning
- Konfigurieren Sie die Standardverschlüsselung
- Fügen Sie Ihrem Bucket Tags hinzu
- Listet Objekte und Versionen auf
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# ===============================================================================
# Prerequisites check
# ===============================================================================

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
 AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# ===============================================================================
# Setup: logging, temp directory, resource tracking
# ===============================================================================

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
```

```
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

echo "============================================"
echo "S3 Getting Started"
echo "============================================"
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# ============================================================================
# Helper functions
# ============================================================================

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
```

```
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
 then
        return 0
    fi

    echo "$versions" | jq -r '.[] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
 IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
 "$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# ==============================================================================
# Error handling and cleanup functions
# ==============================================================================

cleanup() {
    echo ""
    echo "============================================="
    echo "CLEANUP"
    echo "============================================="

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[].
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[].
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
```

```
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "============================================="
    echo "ERROR on line ${line_number}"
    echo "============================================="
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# ==============================================================================
# Step 1: Create a bucket
# ==============================================================================

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
```

```
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
 then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# ==============================================================================
# Step 2: Upload a sample text file
# ==============================================================================
```

```
echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# ==============================================================================
# Step 3: Download the object
# ==============================================================================

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# ==============================================================================
# Step 4: Copy the object to a folder prefix
# ==============================================================================

echo "Step 4: Copying object to a folder prefix..."
```

```
if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# ==============================================================================
# Step 5: Enable versioning and upload a second version
# ==============================================================================

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# ==============================================================================
```

```
# Step 6: Configure SSE-S3 encryption
# ==============================================================================

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# ==============================================================================
# Step 7: Block all public access
# ==============================================================================

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# ==============================================================================
```

```
# Step 8: Configure bucket logging
# ==============================================================================

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
 true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
```

```
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# ==============================================================================
# Step 9: Tag the bucket
# ==============================================================================

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
```

```
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# ==============================================================================
# Step 10: List objects and versions
# ==============================================================================

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# ==============================================================================
# Step 11: Cleanup
# ==============================================================================

echo ""
echo "============================================="
echo "TUTORIAL COMPLETE"
echo "============================================="
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${RESOURCE}"
    done
else
```

```
    echo "  (none)"
fi
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - HeadObject
  - ListObjectVersions
  - ListObjectsV2
  - PutBucketEncryption
  - PutBucketTagging
  - PutBucketVersioning
  - PutObject
  - PutPublicAccessBlock

Erste Schritte mit Query Analytics

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen S3-Bucket für Abfrageergebnisse
- Erstellen einer Datenbank.
- Erstellen einer Tabelle

- Ausführen einer -Abfrage
- Benannte Abfragen erstellen und verwenden
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
 [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
```

```
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
```

```
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
 ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
```

```
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
```

```
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
```

```
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^
\\\\(]+[\\\\(][^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
```

```
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
```

```
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi
```

```
DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID"
 \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
```

```
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
```

```
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
 then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
```

```
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
 | cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
```

```
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi
```

```
        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults
  - ListDatabases
  - ListNamedQueries
  - ListTableMetadata

- Mb
- Rb
- Rm
- StartQueryExecution

Arbeiten Sie mit den Objektsperrfunktionen von Amazon S3 mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie mit S3-Object-Lock-Features arbeiten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
```

```
        1. Create test Amazon Simple Storage Service (S3) buckets with different
lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
or overwrite objects.
        5. Clean up objects and buckets.
  */

   public static S3ActionsWrapper _s3ActionsWrapper = null!;
   public static IConfiguration _configuration = null!;
   private static string _resourcePrefix = null!;
   private static string noLockBucketName = null!;
   private static string lockEnabledBucketName = null!;
   private static string retentionAfterCreationBucketName = null!;
   private static List<string> bucketNames = new List<string>();
   private static List<string> fileNames = new List<string>();

   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonS3>()
                   .AddTransient<S3ActionsWrapper>()
           )
           .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       ConfigurationSetup();

       ServicesSetup(host);
```

```
        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
 Object Locking Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";
```

```
        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
 true);
        await
 _s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
 false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();
```

```
        Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
```

```
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
```

```
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                        }
                    }
                }
            }
            Console.WriteLine(new string('-', 80));
            return true;
        }

        // <summary>
        /// List all of the current buckets and objects.
        /// </summary>
        /// <param name="interactive">True to run as interactive.</param>
        /// <returns>The list of buckets and objects.</returns>
        public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
        {
            var allObjects = new List<S3ObjectVersion>();
            foreach (var bucketName in bucketNames)
            {
                var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
                foreach (var objectKey in objectsInBucket.Versions)
                {
                    allObjects.Add(objectKey);
                }
            }

            if (interactive)
            {
                Console.WriteLine("\nCurrent buckets and objects:\n");
                int i = 0;
                foreach (var bucketObject in allObjects)
                {
                    i++;
                    Console.WriteLine(
```

```
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the scenario."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
```

```
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
```

```
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
```

```
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
                await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
```

```
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
```

```
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };
```

```
            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
```

```
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
```

```
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };
```

```
            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                                $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                                $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
```

```
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
 {bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
```

```
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
ex.Message);
            return false;
        }
```

```
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "s3_object_lock/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")
```

```
  for _, info := range createInfo {
   log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
```

```
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
 bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
 context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")
```

```
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
 to %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
```

```
      default:
       panic(err)
      }
     } else {
      log.Printf("Legal hold set on %s.\n", objKey)
     }
    }
   case 1:
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
 %s in %s?\n"+
     "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
 able to delete this object\n"+
     "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
    if scenario.questioner.AskBool(q, "y") {
     err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
      default:
       panic(err)
      }
     } else {
      log.Printf("Retention period set to 1 for %s.", objKey)
      bucket.retentionEnabled = true
     }
    }
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
```

```
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
 for choice != Finish {
  objList := scenario.GetAllObjects(ctx)
  objChoices := scenario.makeObjectChoiceList(objList)
  choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
 interactiveChoices)
  switch choice {
  case ListAll:
   log.Println("The current objects in the example buckets are:")
   for _, objChoice := range objChoices {
    log.Println("\t", objChoice)
   }
  case DeleteObject, DeleteRetentionObject:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 delete:\n", objChoices)
   obj := objList[objChoice]
   deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
 obj.versionId, choice == DeleteRetentionObject)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Println("Nothing to delete.")
    default:
     panic(err)
    }
   } else if deleted {
    log.Printf("Object %s deleted.\n", obj.key)
   }
```

```
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Println("Couldn't upload to nonexistent bucket.")
   default:
    panic(err)
   }
  } else {
   log.Printf("Uploaded new content to object %s.\n", obj.key)
  }
 case ViewRetention:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Can't get retention configuration for %s.\n", obj.key)
   default:
    panic(err)
   }
  } else if retention != nil {
   log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
  } else {
   log.Printf("Object %s does not have object retention configured.\n", obj.key)
  }
 case ViewLegalHold:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
```

```
      log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
     default:
      panic(err)
     }
    } else if legalHold != nil {
     log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
     log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
   case Finish:
    log.Println("Let's clean up.")
   }
   log.Println(strings.Repeat("-", 88))
  }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
```

```
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete S3-Aktionen umschließt.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
```

```
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
```

```
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
```

```
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
```

```
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
```

```
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
```

```
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
```

```
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
```

```
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
  } else {
   deleted = true
  }
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
```

```
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  } else {
   for _, delObjs := range delOut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
  time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
  *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "s3_object_lock/actions"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
```

```
 name             string
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
```

```
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
 bucket.name)
  } else {
   log.Printf("Deleted %s.\n", bucket.name)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
```

```
 switch err.(type) {
 case *types.NoSuchBucket:
  log.Printf("No objects to get from %s.\n", bucket.name)
 default:
  panic(err)
 }
}
delObjects := make([]types.ObjectIdentifier, len(versions))
for i, version := range versions {
 if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
  status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
   default:
    panic(err)
   }
  } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
   err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
    default:
     panic(err)
    }
   }
  }
 }
 delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
bucket.retentionEnabled)
if err != nil {
 switch err.(type) {
 case *types.NoSuchBucket:
  log.Println("Nothing to delete.")
```

```
  default:
   panic(err)
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
```

```
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
 Object Locking Feature Scenario.");
```

```
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
```

```
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
```

```
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
```

```
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                    }
                }
```

```
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal
lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);

        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();
```

```
        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
```

```
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
```

```
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
 zone.
        ZonedDateTime zonedDateTime =
 futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();
```

```
        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
```

```
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
```

```
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
```

```
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
```

```
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
```

```
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

```
    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einstiegspunkt für das Szenario (index.js). Dadurch werden alle Schritte orchestriert. Die Implementierungsdetails für Szenario, ScenarioInput ScenarioOutput, und finden GitHub Sie unter ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
```

```
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
```

```
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Geben Sie Willkommensnachrichten an die Konsole aus (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Stellen Sie Buckets, Objekte und Dateieinstellungen bereit (setup.steps.js).

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
```

```
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-no-lock with object lock False.
         ${state.bucketPrefix}-lock-enabled with object lock True.
         ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
```

```
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: lockEnabledBucketName,
          ObjectLockEnabledForBucket: true,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
      );
      await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
      );
      await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

      state.noLockBucketName = noLockBucketName;
      state.lockEnabledBucketName = lockEnabledBucketName;
      state.retentionBucketName = retentionBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
```

```
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file0.txt in ${state.bucketPrefix}-no-lock.
         file1.txt in ${state.bucketPrefix}-no-lock.
         file0.txt in ${state.bucketPrefix}-lock-enabled.
         file1.txt in ${state.bucketPrefix}-lock-enabled.
         file0.txt in ${state.bucketPrefix}-retention-after-creation.
         file1.txt in ${state.bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
```

```
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
```

```
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
 retention period.
A default retention period will be configured for ${state.bucketPrefix}-
retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
```

```
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
      if (Status !== "Enabled") {
        throw new Error("Bucket versioning is not enabled.");
      }
    });

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
```

```
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
```

```
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
```

```
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
```

```
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
```

```
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

Zeigen Sie Dateien in den Buckets an und löschen Sie sie (repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
```

```
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
```

```
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
```

```
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
```

```
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
 ${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
 bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
```

```
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

Vernichten Sie alle erstellten Ressourcen (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
```

```
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
```

```
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        }
        throw e;
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (legalHold.LegalHold?.Status === "ON") {
            await client.send(
              new PutObjectLegalHoldCommand({
                Bucket: bucket,
                Key,
                VersionId,
                LegalHold: {
                  Status: "OFF",
                },
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        try {
          const retention = await client.send(
            new GetObjectRetentionCommand({
              Bucket: bucket,
              Key,
```

```
              VersionId,
            }),
          );

          if (retention.Retention?.Mode === "GOVERNANCE") {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration

- GetObjectRetention
- PutObjectLegalHold
- PutObjectLockConfiguration
- PutObjectRetention

Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Vorbedingungen zu Amazon-S3-Anforderungen hinzufügen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der bedingten Amazon-S3-Anforderung zeigt.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
```

```
   This example demonstrates the use of conditional requests for S3 operations.
   You can use conditional requests to add preconditions to S3 read requests to
return or copy
   an object based on its Entity tag (ETag), or last modified date.
   You can use a conditional write requests to prevent overwrites by ensuring
   there is no existing object with the same key.
  */

   public static S3ActionsWrapper _s3ActionsWrapper = null!;
   public static IConfiguration _configuration = null!;
   public static string _resourcePrefix = null!;
   public static string _sourceBucketName = null!;
   public static string _destinationBucketName = null!;
   public static string _sampleObjectKey = null!;
   public static string _sampleObjectEtag = null!;
   public static bool _interactive = true;


   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonS3>()
                   .AddTransient<S3ActionsWrapper>()
           )
           .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       ServicesSetup(host);

       try
```

```
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            ConfigurationSetup();
            _sampleObjectEtag = await Setup(_sourceBucketName,
_destinationBucketName, _sampleObjectKey);

            await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
_sampleObjectKey, _sampleObjectEtag, 0);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await CleanupScenario(_sourceBucketName, _destinationBucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
```

```
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        _sourceBucketName = _resourcePrefix + "-source";
        _destinationBucketName = _resourcePrefix + "-dest";
        _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
 uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
 bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
 destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
 requests.\n" +
            "This example demonstrates the use of conditional requests for S3
 operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
 requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
\n" +
            "You can use a conditional write requests to prevent overwrites by
 ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
\r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
 example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();
```

```
        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
 sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task CleanupScenario(string sourceBucket, string
 destBucket)
    {
        await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
        await _s3ActionsWrapper.CleanupBucketByName(destBucket);
    }

    /// <summary>
    /// Displays a list of the objects in the test buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task DisplayBuckets(string sourceBucket, string
 destBucket)
    {
        await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
        await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
    }

    /// <summary>
    /// Displays the menu of conditional request options for the user.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The key of the test object in the source
 bucket.</param>
    /// <param name="etag">The ETag of the test object in the source bucket.</
param>
```

```
    public static async Task DisplayDemoChoices(string sourceBucket, string
 destBucket, string objectKey, string etag, int defaultChoice)
    {
        var actions = new[]
        {
            "Print a list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit."
        };

        var conditions = new[]
        {
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
 fail.",
            "If-Modified-Since: using yesterday's date. This condition should
 succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
 fail."
        };

        var conditionTypes = new[]
        {
            S3ConditionType.IfMatch,
            S3ConditionType.IfNoneMatch,
            S3ConditionType.IfModifiedSince,
            S3ConditionType.IfUnmodifiedSince,
        };

        var yesterdayDate = DateTime.UtcNow.AddDays(-1);

        int choice;
        while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
 features by selecting one of the following choices:", actions, defaultChoice)) !
= 4)
        {
            switch (choice)
            {
                case 0:
                    Console.WriteLine("Listing the objects and buckets.");
                    await DisplayBuckets(sourceBucket, destBucket);
                    break;
```

```
                case 1:
                    int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                    if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                    }
                    else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                    }
                    break;
                case 2:
                    int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
                    string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
                    if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
                    }
                    else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
                    }
                    break;
                case 3:
                    Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
                    Console.WriteLine("If the key is a duplicate, the write will
fail.");
                    string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
                    await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
                    break;
```

```
            }

            if (!_interactive)
            {
                break;
            }
        }

        Console.WriteLine("Proceeding to cleanup.");
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
 and buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
```

```
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
 defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
```

```
    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not
 interactive.</param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string
 defaultAnswer)
    {
        string? answer = "";
        if (_interactive)
        {
            do
            {
                Console.WriteLine(question);
                answer = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(answer));
        }
        else
        {
            answer = defaultAnswer;
        }

        return answer;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
```

```
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
 'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
```

```
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read
 successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
```

```
    }

    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
    public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
```

```
    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
 destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
 string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
```

```
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithName(string bucketName)
    {
        Console.WriteLine($"\tCreating bucket {bucketName}.");
        try
```

```
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            };

            await _amazonS3.PutBucketAsync(request);
            var bucketReady = false;
            var retries = 5;
            while (!bucketReady && retries > 0)
            {
                Thread.Sleep(5000);
                bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
                retries--;
            }

            return bucketReady;
        }
        catch (BucketAlreadyExistsException ex)
        {
            Console.WriteLine($"Bucket already exists: '{ex.Message}'");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Cleans up objects and deletes the bucket by name.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task CleanupBucketByName(string bucketName)
    {
        try
        {
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            foreach (var obj in listObjectsResponse.S3Objects)
```

```
            {
                await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
            }
            await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
            Console.WriteLine($"Cleaned up bucket: {bucketName}.");
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }

    /// <summary>
    /// List the contents of the bucket with their ETag.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
    {
        var results = new List<S3Object>();
        try
        {
            Console.WriteLine($"\t Items in bucket {bucketName}");
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            if (listObjectsResponse.S3Objects.Count == 0)
            {
                Console.WriteLine("\t\tNo objects found.");
            }
            else
            {
                foreach (var obj in listObjectsResponse.S3Objects)
                {
```

```
                    Console.WriteLine($"\t\t object: {obj.Key} ETag {obj.ETag}");
                }
            }
            results = listObjectsResponse.S3Objects;

        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                _logger.LogError($"Bucket {bucketName} does not exist.");
            }
            else
            {
                _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
                throw;
            }
        }

        return results;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
 {bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket by deleting the objects and then the bucket
 itself.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CleanUpBucketByName(string bucketName)
    {
        try
        {
            var allFiles = await ListBucketContentsByName(bucketName);

            foreach (var fileInfo in allFiles)
            {
                await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
            }

            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CopyObject
  - GetObject
  - PutObject

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einstiegspunkt für den Workflow (index.js). Dadurch werden alle Schritte orchestriert. Die Implementierungsdetails für Szenario, ScenarioInput ScenarioOutput, und finden GitHub Sie unter ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
```

```
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(scenarios), replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Conditional Requests - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
```

```
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Geben Sie Willkommensnachrichten an die Konsole aus (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
 operations." +
      " You can use conditional requests to add preconditions to S3 read requests
 to return " +
      "or copy an object based on its Entity tag (ETag), or last modified
 date.You can use " +
      "a conditional write requests to prevent overwrites by ensuring there is no
 existing " +
      "object with the same key.\n" +
      "This example will enable you to perform conditional reads and writes that
 will succeed " +
      "or fail based on your selected options.\n" +
```

```
      "Sample buckets and a sample object will be created as part of the example.
\n" +
      "Some steps require a key name prefix to be defined by the user. Before you
 begin, you can " +
      "optionally edit this prefix in ./object_name.json. If you do so, please
 reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Stellen Sie Buckets und Objekte bereit (setup.steps.js).

```
import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
```

```
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );
/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-source-bucket.
         ${state.bucketPrefix}-destination-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
```

```
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: destinationBucketName },
      );

      state.sourceBucketName = sourceBucketName;
      state.destinationBucketName = destinationBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
```

```
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Rufen Sie Objekte mithilfe von bedingten S3-Anfragen (repl.steps.js) ab, kopieren Sie sie und fügen Sie sie ein.

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
```

```
    "Explore the S3 conditional request features by selecting one of the
 following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
        {
          name: "Perform a conditional copy. These examples use the key name
 prefix defined in ./object_name.json.",
          value: choices.CONDITIONAL_COPY,
        },
        {
          name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
          value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key } = version;
      files.push({ bucket, key: Key });
    }
  }
  return files;
};
```

```
/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
  const objectsResponse = await client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    }),
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
```

```
        },
      );
      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condCopyOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional copy action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condWriteOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional write action would you like to take?",
        {
          type: "select",
          choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
          ],
        },
```

```
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
            )
            .join("\n");
          break;
        }
        case choices.CONDITIONAL_READ:
          {
            const selectedCondRead = await condReadOptions.handle(state);
            if (
              selectedCondRead ===
              "If-Match: using the object's ETag. This condition should succeed."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
```

```
            if (
              selectedCondRead ===
              "If-None-Match: using the object's ETag. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfNoneMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Modified-Since: using yesterday's date. This condition should
succeed."
            ) {
              const date = new Date();
              date.setDate(date.getDate() - 1);

              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfModifiedSince: date,
                  }),
                );
```

```
                state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Unmodified-Since: using yesterday's date. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";

              const date = new Date();
              date.setDate(date.getDate() - 1);
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfUnmodifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
          }
          break;
        case choices.CONDITIONAL_COPY: {
          const selectedCondCopy = await condCopyOptions.handle(state);
          if (
            selectedCondCopy ===
            "If-Match: using the object's ETag. This condition should succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
```

```
            const ETag = await getEtag(client, bucket, key);

            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;
            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfMatch: ETag,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-None-Match: using the object's ETag. This condition should fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
```

```
                  CopySourceIfNoneMatch: ETag,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
            } catch (err) {
              state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Modified-Since: using yesterday's date. This condition should
succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfModifiedSince: date,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
            }
            break;
```

```
          }
          if (
            selectedCondCopy ===
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfUnmodifiedSince: date,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
            }
          }
          break;
        }
        case choices.CONDITIONAL_WRITE:
          {
            const selectedCondWrite = await condWriteOptions.handle(state);
            if (
              selectedCondWrite ===
              "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
            ) {
```

```
              // Optionally edit the default key name prefix of the copied object
 in ./object_name.json.
              const key = "text02.txt";
              const __filename = fileURLToPath(import.meta.url);
              const __dirname = dirname(__filename);
              const filePath = path.join(__dirname, "text02.txt");
              try {
                await client.send(
                  new PutObjectCommand({
                    Bucket: `${state.destinationBucketName}`,
                    Key: `${key}`,
                    Body: await readFile(filePath),
                    IfNoneMatch: "*",
                  }),
                );
                state.replOutput = `${key} uploaded to bucket
 ${state.destinationBucketName} because the key is not a duplicate.`;
              } catch (err) {
                state.replOutput = `Unable to upload object to bucket
 ${state.destinationBucketName}:${err.message}`;
              }
              break;
            }
          }
          break;

        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
          preformatted: true,
        }),
      },
    },
  );

export { replInput, choices };
```

Vernichten Sie alle erstellten Ressourcen (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
```

```
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Objects and buckets have already been deleted.");
          continue;
        }
        throw e;
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CopyObject
  - GetObject
  - PutObject

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die bedingten Amazon-S3-Anforderungen zeigt.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
 requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
 setup.
sys.path.append("../../../..")
import demo_tools.question as q  # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)
```

```
class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
 actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
 str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
 {dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
            response = self.s3_client.put_object(
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
            return object_etag
```

```
        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
            )


    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """
        Cleans up the bucket by deleting all objects and then the bucket itself.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            for obj in objs:
                key = obj["Key"]
                self.s3_client.delete_object(Bucket=bucket_name, Key=key)
            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Cleaned up bucket: {bucket_name}.")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
            else:
                logger.error(f"Error deleting bucket: {error_code}")
                raise


    def display_buckets(self, source_bucket: str, dest_bucket: str):
```

```
        """
        Display a list of the objects in the test buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.list_bucket_contents(source_bucket)
        self.list_bucket_contents(dest_bucket)

    def list_bucket_contents(self, bucket_name):
        """
        Display a list of the objects in the bucket.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            print(f"\t Items in bucket {bucket_name}")
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            if not objs:
                print("\t\tNo objects found.")
            for obj in objs:
                key = obj["Key"]
                print(f"\t\t object: {key} ETag {obj['ETag']}")
            return objs
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist.")
            else:
                logger.error(f"Error listing bucket and objects: {error_code}")
                raise


    def display_menu(
        self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
    ):
        """
        Displays the menu of conditional request options for the user.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The key of the test object in the source bucket.
```

```
        :param etag: The etag of the test object in the source bucket.
        """

        actions = [
            "Print list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit.",
        ]

        conditions = [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ]

        condition_types = [
            "IfMatch",
            "IfNoneMatch",
            "IfModifiedSince",
            "IfUnmodifiedSince",
        ]
        copy_condition_types = [
            "CopySourceIfMatch",
            "CopySourceIfNoneMatch",
            "CopySourceIfModifiedSince",
            "CopySourceIfUnmodifiedSince",
        ]

        yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

        choice = 0
        while choice != 4:
            print("-" * 88)
            print("Choose an action to explore some example conditional
requests.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                print("Listing the objects and buckets.")
```

```
                self.display_buckets(source_bucket, dest_bucket)
            elif choice == 1:
                print("Perform a conditional read.")
                condition_type = q.choose("Enter the condition type : ",
conditions)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.get_object_conditional(
                        object_key, source_bucket,
condition_types[condition_type], etag
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.get_object_conditional(
                        object_key,
                        source_bucket,
                        condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 2:
                print("Perform a conditional copy.")
                condition_type = q.choose("Enter the condition type : ",
conditions)
                dest_key = q.ask("Enter an object key: ", q.non_empty)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        source_bucket,
                        dest_bucket,
                        copy_condition_types[condition_type],
                        etag,
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        copy_condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 3:
                print(
                    "Perform a conditional write using IfNoneMatch condition on
the object key."
                )
                print("If the key is a duplicate, the write will fail.")
```

```
                object_key = q.ask("Enter an object key: ", q.non_empty)
                self.conditional_requests.put_object_conditional(
                    object_key, source_bucket, b"Conditional write example data."
                )
            elif choice == 4:
                print("Proceeding to cleanup.")


    def run_scenario(self):
        """
        Runs the interactive scenario.
        """
        print("-" * 88)
        print("Welcome to the Amazon S3 conditional requests example.")
        print("-" * 88)

        print(
            f"""\
        This example demonstrates the use of conditional requests for S3
operations.
        You can use conditional requests to add preconditions to S3 read requests
to return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by
ensuring
        there is no existing object with the same key.

        This example will allow you to perform conditional reads
        and writes that will succeed or fail based on your selected options.

        Sample buckets and a sample object will be created as part of the
example.
        """
        )

        bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
        source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
        dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
        object_key = "test-upload-file.txt"

        try:
            etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
```

```
            self.display_menu(source_bucket_name, dest_bucket_name, object_key,
 etag)
        finally:
            self.cleanup_scenario(source_bucket_name, dest_bucket_name)

        print("-" * 88)
        print("Thanks for watching.")
        print("-" * 88)


if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```

Eine Wrapper-Klasse, die die Operationen der bedingten Anfrage enthält.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)


class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)
```

```
    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
```

```
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
```

```
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CopyObject
  - GetObject
  - PutObject

Verwalten Sie Zugriffskontrolllisten (ACLs) für Amazon S3 S3-Buckets mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine neue Zugriffssteuerungsliste (ACL) für Amazon-S3-Buckets verwalten.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();

        await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
    }

    /// <summary>
```

```
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
```

```
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
 get the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
```

```
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
```

```
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - GetBucketAcl

- GetObjectAcl
- PutBucketAcl
- PutObjectAcl

Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie die erweiterte Clientbibliothek von Amazon SQS verwenden, um mit großen Amazon-SQS-Nachrichten zu arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
```

```
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
```

```
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(),
e);
            throw e;
        }
    }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie versionierte S3-Objekte in Batches mit einer Lambda-Funktion verwaltet werden.

Python

SDK für Python (Boto3)

Zeigt, wie versionierte Objekte von Amazon Simple Storage Service (Amazon S3) stapelweise bearbeitet werden, indem Jobs erstellt werden, die AWS Lambda Funktionen aufrufen, um die Verarbeitung durchzuführen. In diesem Beispiel wird ein versionsfähiger Bucket erstellt, der die Strophen des Gedichts You Are Old, Father William von Lewis Carroll hochlädt und Amazon-S3-Batch-Aufträge verwendet, um das Gedicht auf verschiedene Arten zu verdrehen.

Lernen Sie Folgendes:

- Erstellen Sie Lambda-Funktionen, die mit versionierten Objekten arbeiten.
- Erstellen Sie ein Manifest von zu aktualisierenden Objekten.
- Erstellen Sie Batch-Aufträge, die Lambda-Funktionen zum Aktualisieren von Objekten aufrufen.
- Löschen Sie Lambda-Funktionen.
- Leeren und löschen Sie einen versionierten Bucket.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon S3

Analysieren Sie Amazon S3 S3-URIs mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Amazon S3-URIs analysieren, um wichtige Komponenten wie den Bucket-Namen und Objektschlüssel zu extrahieren.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Analysieren Sie eine Amazon S3-URI mithilfe der S3Uri-Klasse.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                    capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
 method.
        URI uri = URI.create(s3ObjectUrl);
```

```
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```

```
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine mehrteilige Kopie eines Amazon-S3-Objekts erstellen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
```

```
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit S3-Ereignisbenachrichtigungen auf objektorientierte Weise arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie ein S3-Benachrichtigungsereignis mithilfe von Amazon SQS verarbeiten.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
```

```
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String
queueArn) {
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the
queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)
```

```
.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
```

```
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Rust

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie ein Bucket aktiviert wird, um S3-Ereignisbenachrichtigungen an ein Amazon SNS-Thema EventBridge und eine Amazon SQS SQS-Warteschlange zu senden und Benachrichtigungen an diese weiterzuleiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
     * to route the S3 object created events to a topic and a queue.
     *
     * @param bucketName Name of existing bucket
```

```
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
                           logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                           throw new RuntimeException(t.getCause().getMessage(),
t);
                       }
```

```
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie einen Upload oder Download in ein bzw. von einem Amazon-S3-Objekt verfolgen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verfolgen Sie den Fortschritt eines Datei-Uploads.

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
```

```
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

Verfolgen Sie den Fortschritt eines Datei-Downloads.

```
    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
```

```
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetObject
  - PutObject

Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda

Das folgende Codebeispiel zeigt, wie Sie Daten für Ihre Anwendung mit S3 Object Lambda transformieren können.

.NET

SDK für .NET

Zeigt, wie Sie benutzerdefinierten Code zu standardmäßigen S3-GET-Anforderungen hinzufügen, um das angeforderte Objekt, das von S3 abgerufen wurde, so zu ändern, dass das Objekt den Anforderungen des anfordernden Clients oder der anfordernden Anwendung entspricht.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Amazon S3

Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK

Das folgende Codebeispiel zeigt Beispiele für bewährte Techniken beim Schreiben von Einheiten- und Integrationstests mit einem AWS SDK.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Cargo.toml zum Testen von Beispielen.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
```

```
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

Beispiel für Modultests mit automock und einem Service-Wrapper

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}
```

```
#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }
```

```
        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
```

```
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}
```

Beispiel für Integrationstests mit StaticReplayClient.

```
use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
```

```
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
```

```
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }

    #[tokio::test]
```

```
    async fn test_multiple_pages() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
```

```
        replay_client.assert_requests_match(&[]);
    }
}
```

Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service (Amazon S3)-Bucket

Das folgende Codebeispiel zeigt, wie Sie ein lokales Verzeichnis rekursiv in einen Amazon Simple Storage Service (Amazon S3)-Bucket hochladen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie an S3TransferManager, um ein lokales Verzeichnis hochzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
```

```
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- Einzelheiten zur API finden Sie UploadDirectoryunter AWS SDK for Java 2.x API-Referenz.

Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien mithilfe von Amazon S3 zu und von einem S3-Bucket übertragen TransferUtility.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
```

```
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
{
```

```
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
```

```
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);
```

```
    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
```

```
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

Laden Sie eine einzelne Datei hoch.

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</
param>
```

```
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
```

```
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
```

```
            }
        }
```

Laden Sie eine einzelne Datei herunter.

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

Laden Sie den Inhalt eines S3 Buckets herunter.

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
```

```
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
```

```
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

Verfolgen Sie den Fortschritt eines Uploads mit dem TransferUtility.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;

    /// <summary>
    /// This example shows how to track the progress of a multipart upload
    /// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
    /// upload to an Amazon S3 bucket.
    /// </summary>
    public class TrackMPUUsingHighLevelAPI
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "sample_pic.png";
            string path = "filepath/directory/";
            string filePath = $"{path}{keyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            await TrackMPUAsync(client, bucketName, filePath, keyName);
        }

        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
```

```
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
```

```
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

Laden Sie ein Objekt mit Verschlüsselung hoch.

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
    {
        public static async Task Main()
        {
            string existingBucketName = "amzn-s3-demo-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();
```

```
            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
```

```
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
 client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }
```

```
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");

                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die Upload- und Download-Manager verwenden, um die Daten in Teile zu zerlegen und sie gleichzeitig zu übertragen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
 string, objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
```

```
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
```

```
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie die Upload- und Download-Manager im Kontext verwenden.

```
import (
 "context"
 "crypto/rand"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
```

```
// This package can be found in the ..\..\demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 large object demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
 bucket:",
  demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
  if err != nil {
   panic(err)
  } else {
   log.Println("Bucket created.")
  }
 }
 log.Println(strings.Repeat("-", 88))

 mibs := 30
 log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
 bucket. ", mibs)
 questioner.Ask("Press Enter when you're ready.")
 largeBytes := make([]byte, 1024*1024*mibs)
 _, _ = rand.Read(largeBytes)
```

```
 largeKey := "doc-example-large"
 log.Println("Uploading...")
 err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
 if err != nil {
  panic(err)
 }
 log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
 log.Println(strings.Repeat("-", 88))

 log.Printf("Let's download the %v MiB object.", mibs)
 questioner.Ask("Press Enter when you're ready.")
 log.Println("Downloading...")
 largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
 largeKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Downloaded %v bytes.", len(largeDownload))
 log.Println(strings.Repeat("-", 88))

 if questioner.AskBool("Do you want to delete your bucket and all of its "+
  "contents? (y/n)", "y") {
  log.Println("Deleting object.")
  err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(ctx, bucketName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien mit dem zu und von einem S3-Bucket übertragen S3TransferManager.

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
```

```
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Laden Sie eine einzelne Datei hoch.

```
    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

In den Codebeispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Verwenden Sie den S3 Transfer Manager auf dem AWS CRT-based S3-Client, um einen mehrteiligen Upload transparent durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the file path of the file to be uploaded
     */
    public void multipartUploadWithTransferManager(String bucketName, String key,
 String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Verwenden Sie die S3Client-API, um einen mehrteiligen Upload durchzuführen.

```
    /**
```

```
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the path to the file to be uploaded
     */
    public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
```

```
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Verwenden Sie die S3AsyncClient API mit aktivierter mehrteiliger Unterstützung, um einen mehrteiligen Upload durchzuführen.

```
    /**
     * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
multipart support.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the local file path of the file to be uploaded
     */
    public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
            .multipartEnabled(true)
            .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                .bucket(bucketName)
```

```
                .key(key),
            Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
 S3AsyncClient.");
    }
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine große Datei hoch.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
```

```
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Laden Sie eine große Datei herunter.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
```

```
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
```

```
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die Dateien mit mehreren der verfügbaren Transfer-Manager-Einstellungen übertragen. Verwenden Sie eine Callback-Klasse, um während der Dateiübertragung Callback-Fortschritte zu schreiben.

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
```

```
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
```

```
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstrieren Sie die Funktionen des Transfer-Managers und melden Sie Ergebnisse.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
```

```
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
```

```
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
```

```
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.
```

```
        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')
```

```
    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
```

```
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        return Err(S3ExampleError::new("Bad file size."));
    }
    if chunk_count > MAX_CHUNKS {
        return Err(S3ExampleError::new(
            "Too many chunks! Try increasing your chunk size.",
        ));
    }

    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
```

```
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;

    let data: GetObjectOutput =
        s3_code_examples::download_object(&client, &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");
```

```
    Ok(())
}
```

Laden Sie einen Stream unbekannter Größe mit einem in ein Amazon S3 S3-Objekt hoch AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Stream unbekannter Größe in ein Amazon S3-Objekt hochladen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie den AWS CRT-based S3-Client.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

   public static void main(String[] args) {
       String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
       String key = UUID.randomUUID().toString();

       AsyncExampleUtils.createBucket(bucketName);
       try {
           PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
           S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
           PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
           logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
           logger.info("Object {} uploaded to bucket {}.", key, bucketName);
       } catch (SdkException e) {
           logger.error(e.getMessage(), e);
       } finally {
           AsyncExampleUtils.deleteObject(bucketName, key);
           AsyncExampleUtils.deleteBucket(bucketName);
       }
   }

   /**
    * @param s33CrtAsyncClient - To upload content from a stream of unknown
size, use can the AWS CRT-based S3 client.
    * @param bucketName - The name of the bucket.
    * @param key - The name of the object.
    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
    */
   public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s33CrtAsyncClient, String bucketName, String key) {

       // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
       String randomString = AsyncExampleUtils.randomString();
       logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
       InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());
```

```
        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Verwenden Sie den standardmäßigen asynchronen S3-Client mit aktiviertem mehrteiligem Upload.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);
```

```
   public static void main(String[] args) {
       String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
       String key = UUID.randomUUID().toString();

       AsyncExampleUtils.createBucket(bucketName);
       try {
           PutObjectFromStreamAsyncMp example = new
PutObjectFromStreamAsyncMp();
           S3AsyncClient s3AsyncClientMp =
S3AsyncClient.builder().multipartEnabled(true).build();
           PutObjectResponse putObjectResponse =
example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
           logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
           logger.info("Object {} uploaded to bucket {}.", key, bucketName);
       } catch (SdkException e) {
           logger.error(e.getMessage(), e);
       } finally {
           AsyncExampleUtils.deleteObject(bucketName, key);
           AsyncExampleUtils.deleteBucket(bucketName);
       }
   }

   /**
    * @param s3AsyncClientMp - To upload content from a stream of unknown size,
use can the S3 asynchronous client with multipart enabled.
    * @param bucketName - The name of the bucket.
    * @param key - The name of the object.
    * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
    */
   public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
String bucketName, String key) {

       // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
       String randomString = AsyncExampleUtils.randomString();
       logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
       InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());
```

```
        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Verwenden Sie den Amazon-S3-Transfer-Manager.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
 LoggerFactory.getLogger(UploadStream.class);
```

```
    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
```

```
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
 null, executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Upload a file to the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to store the file into.
    ///   - key: The name (or path) of the file to upload to in the `bucket`.
```

```
    ///   - sourcePath: The pathname on the local filesystem of the file to
    ///     upload.
    func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
        let fileURL: URL = URL(fileURLWithPath: sourcePath)
        let fileName: String

        // If no key was provided, use the last component of the filename.

        if key == nil {
            fileName = fileURL.lastPathComponent
        } else {
            fileName = key!
        }

        let s3Client = try await S3Client()

        // Create a FileHandle for the source file.

        let fileHandle = FileHandle(forReadingAtPath: sourcePath)
        guard let fileHandle = fileHandle else {
            throw TransferError.readError
        }

        // Create a byte stream to retrieve the file's contents. This uses the
        // Smithy FileStream and ByteStream types.

        let stream = FileStream(fileHandle: fileHandle)
        let body = ByteStream.stream(stream)

        // Create a `PutObjectInput` with the ByteStream as the body of the
        // request's data. The AWS SDK for Swift will handle sending the
        // entire file in chunks, regardless of its size.

        let putInput = PutObjectInput(
            body: body,
            bucket: bucket,
            key: fileName
        )

        do {
            _ = try await s3Client.putObject(input: putInput)
        } catch {
            throw TransferError.uploadError("Error uploading the file: \(error)")
```

```
        }

        print("File uploaded to \(fileURL.path).")
    }
```

Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie ein AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Prüfsummen verwenden, um mit einem Amazon-S3-Objekt zu arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In den Codebeispielen wird eine Teilmenge der folgenden Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
```

```
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Geben Sie einen Prüfsummenalgorithmus für die `putObject`-Methode an, wenn Sie `PutObjectRequest` erstellen.

```
    public void putObjectWithChecksum(String bucketName, String key) {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
            RequestBody.fromString("This is a test"));
    }
```

Überprüfen Sie die Prüfsumme für die `getObject` Methode, wenn Sie die GetObjectRequest erstellen.

```
    public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
 {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
            .response();
    }
```

Pre-calculate eine Prüfsumme für die `putObject` Methode, wenn Sie die erstellen. `PutObjectRequest`

```
   public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
       String checksum = calculateChecksum(filePath, "SHA-256");

       s3Client.putObject((b -> b
               .bucket(bucketName)
               .key(key)
               .checksumSHA256(checksum)),
           RequestBody.fromFile(Paths.get(filePath)));
   }
```

Verwenden Sie den S3 Transfer Manager zusätzlich zum AWS CRT-based S3-Client, um einen mehrteiligen Upload transparent durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

Sie können einen Prüfsummenalgorithmus angeben, den das SDK verwenden soll. Standardmäßig verwendet das SDK den CRC32-Algorithmus.

```
   public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
       S3TransferManager transferManager = S3TransferManager.create();
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b
               .bucket(bucketName)
               .key(key)
               .checksumAlgorithm(ChecksumAlgorithm.SHA1))
           .source(Paths.get(filePath))
           .build();
       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
       fileUpload.completionFuture().join();
       transferManager.close();
   }
```

Verwenden Sie die S3Client-API oder (S3AsyncClient API), um einen mehrteiligen Upload durchzuführen. Wenn Sie eine zusätzliche Prüfsumme angeben, müssen Sie den Algorithmus

angeben, der bei der Initiierung des Uploads verwendet werden soll. Sie müssen auch den Algorithmus für jede Teilanforderung angeben und die für jedes Teil nach dem Hochladen berechnete Prüfsumme bereitstellen.

```
   public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
       ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

       // Initiate the multipart upload.
       CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
           .bucket(bucketName)
           .key(key)
           .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
       String uploadId = createMultipartUploadResponse.uploadId();

       // Upload the parts of the file.
       int partNumber = 1;
       List<CompletedPart> completedParts = new ArrayList<>();
       ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

       try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
           long fileSize = file.length();
           long position = 0;
           while (position < fileSize) {
               file.seek(position);
               long read = file.getChannel().read(bb);

               bb.flip(); // Swap position and limit before reading from the
buffer.
               UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                   .bucket(bucketName)
                   .key(key)
                   .uploadId(uploadId)
                   .checksumAlgorithm(algorithm) // Checksum specified on each
part.
                   .partNumber(partNumber)
                   .build();

               UploadPartResponse partResponse = s3Client.uploadPart(
                   uploadPartRequest,
                   RequestBody.fromByteBuffer(bb));
```

```
                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit Features der S3-Objektintegrität arbeiten.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Objektintegrität zeigt.

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
 std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
 std::endl;
        return false;
```

```
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
              << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
                  << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                          SHA256);


    gUseCalculatedChecksum = !askYesNoQuestion(
            "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
              << std::endl;
    std::cout << "Object integrity will be verified using the "
              << stringForHashMethod(chosenHashMethod) << " algorithm."
              << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
                << "A checksum computed by this workflow will be used for object
integrity verification,"
                << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
                << "A checksum computed by the SDK will be used for object
integrity verification."
                << std::endl;
    }
```

```
    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
```

```
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                 client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
```

```
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                            transferManagerHashMethod,
                            retrievedTransferManagerFinalHash,
                            &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                    UPLOAD_BUFFER_SIZE,
                                    locallyCalculatedFinalHash,
                                    locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                         locallyCalculatedFinalHash, "TransferManager upload",
                         transferManagerHashMethod,
                         retrievedTransferManagerPartHashes,
                         locallyCalculatedPartHashes);

    printAsterisksLine();

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;
```

```
    multiPartUploadExplanations(key, chosenHashMethod);

    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          MULTI_PART_TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5;  // The default hash method for
multipart uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
              stringForHashMethod(multipartUploadHashMethod) << std::endl;
```

```
    std::cout << "Now we will retrieve the checksums from the server." <<
 std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
 multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
 workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
 integrity hashing methods.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded
 object.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
```

```
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

 request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
```

```
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
```

```
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
```

```
        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
```

```
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual
 parts retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                      const Hasher &localHash,
                                      const Aws::String &uploadtype,
                                      HASH_METHOD hashMethod,
                                      const std::vector<Aws::String>
 &retrievedPartHashes,
                                      const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
 << std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
 retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
```

```
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
 std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
 not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
 "'"
                      << std::endl;
            std::cerr << "Remote number of hashes - '"
                      << retrievedPartHashes.size()
                      << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
 match for part " << i + 1
                          << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                          << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                          << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
```

```
                                            const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
 std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                           const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
 Transfer Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    bool useDefaultHashMethod,
                                    const std::shared_ptr<Aws::S3::S3Client>
 &client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
```

```
            transfer_config.checksumAlgorithm =
 getChecksumAlgorithmForHashMethod(
                    hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
 transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
            transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
 transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

              bucket, key,

              "text/plain",

              Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
            uploadHandle->GetStatus() ==
 Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
 uploaded to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash
 values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
```

```
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
   \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes)
 {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
 uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();
```

```
        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
```

```
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                          const Aws::String &key,
                                                          const Aws::String
 &uploadID,
                                                          int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                          const Aws::String
 &calculatedHash,
                                                          const
 std::shared_ptr<Aws::IOStream> &body,
                                                          const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
```

```
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
```

```
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

```
//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
 hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
 concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
 &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {
    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
        }
    }
    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
 client);
    if (uploadID.empty()) {
        return false;
    }
```

```
   std::vector<unsigned char> totalHashBuffer;
   bool uploadSucceeded = true;
   std::streamsize uploadedBytes = 0;
   int partNumber = 1;
   Aws::Vector<Aws::S3::Model::CompletedPart> parts;
   while (uploadedBytes < objectSize) {
       std::cout << "Uploading part " << partNumber << "." << std::endl;

       std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
       std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
       ioStream->read((char *) buffer.data(), bytesToRead);
       Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
       std::shared_ptr<Aws::IOStream> body =
               Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                              &preallocatedStreamBuf);

       Hasher hasher;
       if (!hasher.calculateObjectHash(*body, hashMethod)) {
           std::cerr << "Error calculating hash." << std::endl;
           uploadSucceeded = false;
           break;
       }

       Aws::String base64HashString = hasher.getBase64HashString();
       partHashes.push_back(base64HashString);

       Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

       totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                              hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

       Aws::String calculatedHash;
       if (gUseCalculatedChecksum) {
           calculatedHash = base64HashString;
       }
```

```
        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,
                                                                        client);
        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }

            parts.push_back(completedPart);
        } else {
            std::cerr << "Error uploading part. " <<
                      uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }
```

```
        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
 completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
 {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
```

```
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
 {
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
 valid." << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
```

```
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration
 &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
```

```
            << std::endl;
    printAsterisksLine();
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an
 object.\n"
            << "The default checksum algorithm for PutObject and MultiPart upload
 is an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
 the SDK calculate the value."
              << std::endl;
    std::cout
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;

    std::cout
            << "This workflow will locally compute checksums for files uploaded
 to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums
 are calculated."
            << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
```

```
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have
 failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using
 the GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
            << std::endl;
    std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }
```

```
    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
 uploading large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally
 and then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part.
 The final checksum is a concatenation of"
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
            << " in the section \"Using part-level checksums for multipart
 uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
```

```
    // Generate a large file by writing this source file multiple times to a new
 file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                  | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- GetObjectAttributes
- PutObject
- UploadPart

Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen versionierten S3 Bucket.
- Rufen Sie alle Versionen eines Objekts ab.
- Setzen Sie eine Richtlinie auf eine frühere Version zurück.
- Löschen Sie ein versioniertes Objekt und stellen Sie es wieder her.
- Löschen Sie alle Versionen eines Objekts dauerhaft.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Aktionen umschließen.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.
```

```
    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
```

```
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )
```

```
    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )



def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
```

```
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
```

```
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Laden Sie die Strophe eines Gedichts in ein versioniertes Objekt hoch und führen Sie eine Reihe von Aktionen dafür aus.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
```

```
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
```

```
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateBucket
  - DeleteObject
  - ListObjectVersions
  - PutBucketLifecycleConfiguration

Serverless-Beispiele für Amazon S3

Die folgenden Codebeispiele zeigen, wie Amazon S3 mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von .NET

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }
```

```
        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
```

```
    return err
   }
   log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
  *headOutput.ContentType)
  }

  return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung JavaScript.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
```

```
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```

```
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

Codebeispiele für Amazon S3 Control mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon S3 Control mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- Benutzerhandbuch zu Amazon S3 Control – Weitere Informationen zu Amazon S3 Control.
- API-Referenz für Amazon S3 Control – Details zu allen verfügbaren Aktionen von Amazon S3 Control.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon S3 Control AWS SDKs
  - Hello Amazon S3 Control
  - Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK
  - Aktionen für Amazon S3 Control mit AWS SDKs
    - Verwenden Sie CreateJobmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteJobTaggingmit einem AWS SDK
    - Verwenden Sie DescribeJobmit einem AWS SDK oder CLI
    - Verwenden Sie GetJobTaggingmit einem AWS SDK
    - Verwenden Sie PutJobTaggingmit einem AWS SDK
    - Verwenden Sie UpdateJobPrioritymit einem AWS SDK oder CLI
    - Verwenden Sie UpdateJobStatusmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung von Amazon S3 Control AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon S3 Control mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon S3 Control
- Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK
- Aktionen für Amazon S3 Control mit AWS SDKs
  - Verwenden Sie CreateJobmit einem AWS SDK oder CLI

- Verwenden Sie DeleteJobTaggingmit einem AWS SDK
- Verwenden Sie DescribeJobmit einem AWS SDK oder CLI
- Verwenden Sie GetJobTaggingmit einem AWS SDK
- Verwenden Sie PutJobTaggingmit einem AWS SDK
- Verwenden Sie UpdateJobPrioritymit einem AWS SDK oder CLI
- Verwenden Sie UpdateJobStatusmit einem AWS SDK oder CLI

Hello Amazon S3 Control

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon S3 Control beginnen können.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
```

```
/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 not configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/
latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 in the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
```

```
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation
is finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
```

```
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
```

```
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
 'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise
```

- Einzelheiten zur API finden Sie ListJobsin AWS SDK for Python (Boto3) API Reference.

Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Kernoperationen von Amazon S3 Control erlernen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erlernen Sie die Kernoperationen.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;
```

```
public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
 of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
 varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
 operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
 tags on thousands or millions of
            objects in a single operation, streamlining the management and
 organization of their data.

            This can be particularly useful for tasks such as cost allocation,
 lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
 essential.
            S3 Batch's scalability and serverless nature make it an ideal
 solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
 scenario.
```

```
        System.out.println("Use CloudFormation to stand up the resource required
 for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
 CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
 with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
 manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the
number, the higher the priority.
             So, a job with a priority of `30` would have a higher priority than
a job with
             a priority of `20`. This is a common way to represent the priority
of a task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
```

```
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
```

```
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
 completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }


}
```

Erstellen Sie eine Klasse, die Operationen umschließt.

```
public class S3BatchActions {
```

```
   private static S3ControlAsyncClient asyncClient;

   private static S3AsyncClient s3AsyncClient ;
   /**
    * Retrieves the asynchronous S3 Control client instance.
    * <p>
    * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
    * has not been created yet, it will be initialized with the following
configuration:
    * <ul>
    *   <li>Maximum concurrency: 100</li>
    *   <li>Connection timeout: 60 seconds</li>
    *   <li>Read timeout: 60 seconds</li>
    *   <li>Write timeout: 60 seconds</li>
    *   <li>API call timeout: 2 minutes</li>
    *   <li>API call attempt timeout: 90 seconds</li>
    *   <li>Retry policy: 3 retries</li>
    *   <li>Region: US_EAST_1</li>
    *   <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
   private static S3ControlAsyncClient getAsyncClient() {
       if (asyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryPolicy(RetryPolicy.builder()
                   .numRetries(3)
                   .build())
               .build();
```

```
            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
```

```
    */
   public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
       UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
           .build();

       return asyncClient.updateJobStatus(updateJobStatusRequest)
           .thenAccept(updateJobStatusResponse -> {
               System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
           })
           .exceptionally(ex -> {
               System.err.println("Failed to cancel job: " + ex.getMessage());
               throw new RuntimeException(ex); // Propagate the exception
           });
   }

   /**
    * Updates the priority of a job asynchronously.
    *
    * @param jobId     the ID of the job to update
    * @param accountId the ID of the account associated with the job
    * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
    */
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
```

```
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }

    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
```

```
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job
description is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
```

```
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " +
response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " +
manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " +
report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
```

```
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of
 the S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);
```

```
                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
```

```
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is
stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }
```

```
    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
```

```
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
```

```
            .build();

        CompletableFuture<PutObjectResponse> future =
 getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
 bucket " + bucketName);
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
```

```
    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has
been deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
deletion.
    */
   public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
       ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
       toDelete.add(ObjectIdentifier.builder()
           .key(objectName)
           .build());

       DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
           .bucket(bucketName)
           .delete(Delete.builder()
               .objects(toDelete).build())
           .build();

       return getS3AsyncClient().deleteObjects(dor)
           .thenAccept(result -> {
               System.out.println("The object was deleted!");
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
           });
   }

   /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
   public void deleteBucketFolderAsync(String bucketName) {
       String folderName = "reports/";
       ListObjectsV2Request request = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .prefix(folderName)
```

```
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
```

```
     * @return a {@link CompletableFuture} that completes when the bucket has
been deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or
uploading the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
```

```
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file
 deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
 S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                   actions.deleteBucketObjects(bucketName, fileName)
                  .thenRun(() -> System.out.println("Object deletion completed"))
                  .exceptionally(ex -> {
                      System.err.println("Error occurred: " + ex.getMessage());
                      return null;
                  });
        }
        System.out.println("All files have been deleted from the bucket " +
 bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\
\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " +
 path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
 filePath.toString());

                    // Create and write content to the new file.
```

```
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority

- UpdateJobStatus

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lernen Sie das Szenario S3 Batch Basics kennen.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
 None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                          access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
 provides
                                 low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                           access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
```

```
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]

    def create_bucket(self, bucket_name: str) -> None:
        """
        Create an S3 bucket.

        Args:
            bucket_name (str): Name of the bucket to create

        Raises:
            ClientError: If bucket creation fails
        """
        try:
            if self.region_name and self.region_name != 'us-east-1':
                self.s3_client.create_bucket(
                    Bucket=bucket_name,
                    CreateBucketConfiguration={
                        'LocationConstraint': self.region_name
                    }
                )
            else:
                self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
        except ClientError as e:
            print(f"Error creating bucket: {e}")
            raise

    def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
        """
        Upload files to S3 bucket including manifest file.

        Args:
            bucket_name (str): Target bucket name
            file_names (list): List of file names to upload

        Returns:
            str: ETag of the manifest file

        Raises:
            ClientError: If file upload fails
        """
```

```
        try:
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    content = f"Content for {file_name}"
                    self.s3_client.put_object(
                        Bucket=bucket_name,
                        Key=file_name,
                        Body=content.encode('utf-8')
                    )
                    print(f"Uploaded {file_name} to {bucket_name}")

            manifest_content = ""
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    manifest_content += f"{bucket_name},{file_name}\n"

            manifest_response = self.s3_client.put_object(
                Bucket=bucket_name,
                Key="job-manifest.csv",
                Body=manifest_content.encode('utf-8')
            )
            print(f"Uploaded manifest file to {bucket_name}")
            print(f"Manifest content:\n{manifest_content}")
            return manifest_response['ETag'].strip('"')

        except ClientError as e:
            print(f"Error uploading files: {e}")
            raise

    def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
            str: Job ID
```

```
        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
```

```
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
str = 'Ready') -> bool:
        """
        Wait for a job to reach the desired status.
```

```
        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
            desired_status (str): Target status to wait for

        Returns:
            bool: True if desired status is reached, False otherwise

        Raises:
            ClientError: If checking job status fails
        """
        print(f"Waiting for job to become {desired_status}...")
        max_attempts = 60
        attempt = 0
        while attempt < max_attempts:
            try:
                response = self.s3control_client.describe_job(
                    AccountId=account_id,
                    JobId=job_id
                )
                current_status = response['Job']['Status']
                print(f"Current job status: {current_status}")
                if current_status == desired_status:
                    return True
                if current_status == 'Suspended':
                    print("Job is in Suspended state, can proceed with
activation")
                    return True
                if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
                    print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                    if 'FailureReasons' in response['Job']:
                        print("Failure reasons:")
                        for reason in response['Job']['FailureReasons']:
                            print(f"- {reason}")
                    return False

                time.sleep(20)
                attempt += 1
            except ClientError as e:
                print(f"Error checking job status: {e}")
                raise
        print(f"Timeout waiting for job to become {desired_status}")
```

```
        return False

    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
```

```
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return

    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be
cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
```

```
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise

    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
```

```
                JobId=job_id
            )
            tags = response.get('Tags', [])
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
                    print(f"  {tag['Key']}: {tag['Value']}")
            else:
                print(f"No tags found for job ID: {job_id}")
        except ClientError as e:
            print(f"Error getting job tags: {e}")
            raise

    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
                ]
            )
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise

    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
```

```
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise

    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise

    def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
        """
        Clean up all resources created during the scenario.

        Args:
            bucket_name (str): Name of the bucket to clean up
            file_names (list): List of files to delete

        Raises:
            ClientError: If cleanup fails
        """
        try:
            for file_name in file_names:
                self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
```

```
                print(f"Deleted {file_name}")

            response = self.s3_client.list_objects_v2(
                Bucket=bucket_name,
                Prefix='batch-op-reports/'
            )
            if 'Contents' in response:
                for obj in response['Contents']:
                    self.s3_client.delete_object(
                        Bucket=bucket_name,
                        Key=obj['Key']
                    )
                    print(f"Deleted {obj['Key']}")

            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Deleted bucket {bucket_name}")
        except ClientError as e:
            print(f"Error in cleanup: {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

Aktionen für Amazon S3 Control mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon S3 Control-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon S3.

Beispiele

- Verwenden Sie CreateJobmit einem AWS SDK oder CLI
- Verwenden Sie DeleteJobTaggingmit einem AWS SDK
- Verwenden Sie DescribeJobmit einem AWS SDK oder CLI
- Verwenden Sie GetJobTaggingmit einem AWS SDK
- Verwenden Sie PutJobTaggingmit einem AWS SDK
- Verwenden Sie UpdateJobPrioritymit einem AWS SDK oder CLI
- Verwenden Sie UpdateJobStatusmit einem AWS SDK oder CLI

Verwenden Sie **`CreateJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie einen neuen Auftrag für Amazon S3 Batch Operations

Im folgenden Beispiel für `create-job` wird ein Auftrag von Amazon S3 Batch Operations erstellt, um Objekte als `confidential` ` in the bucket ``employee-records` zu kennzeichnen.

```
aws s3control create-job \
    --account-id 123456789012 \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
 "Value":"true"}] }}' \
    --report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
create-job",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
```

```
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv","ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
    --priority 42 \
    --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

Ausgabe:

```
{
    "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Einzelheiten zur API finden Sie CreateJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen asynchronen S3-Auftrag.

```
    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of
 the S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
```

```
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                        String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();
```

```
                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
```

```
                return jobId;
            });
    }
```

Erstellen Sie einen Auftrag zur Compliance-Aufbewahrung.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows
 you to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API
 call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
```

```
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Erstellen Sie einen Auftrag für eine gesetzliche Aufbewahrungsfrist.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows
 you to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API
 call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();
```

```
        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Erstellen Sie einen neuen Auftrag zur Governance-Aufbewahrung.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
 <manifestObjectVersionId>

            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3
 object that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will
 be stored.
                roleArn - The ARN of the IAM role that will be used to perform
 the governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the
 `eTag` property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
 jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
 s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
```

```
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
        final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
```

```
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports
```

```
        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
```

```
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise
```

- Einzelheiten zur API finden Sie CreateJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_manifest_arn  = 'arn:aws:s3:::my-bucket/job-manifest.csv'
        " iv_manifest_etag = 'abc123def456'
        " iv_report_bucket = 'arn:aws:s3:::my-report-bucket'
        DATA(lo_result) = lo_s3c->createjob(
          iv_accountid            = iv_account_id
          iv_rolearn              = iv_role_arn
          iv_confirmationrequired = abap_true
          iv_priority             = 10
          iv_description          = 'Batch job for tagging objects'
          io_operation            = NEW /aws1/cl_s3cjoboperation(
            io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
              it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                ( NEW /aws1/cl_s3cs3tag(
```

```
                    iv_key   = 'BatchTag'
                    iv_value = 'BatchValue' ) )
              )
            )
          )
          io_manifest             = NEW /aws1/cl_s3cjobmanifest(
            io_spec     = NEW /aws1/cl_s3cjobmanifestspec(
              iv_format = 'S3BatchOperations_CSV_20180820'
              it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
              )
            )
            io_location = NEW /aws1/cl_s3cjobmanifestloc(
              iv_objectarn = iv_manifest_arn
              iv_etag      = iv_manifest_etag
            )
          )
          io_report               = NEW /aws1/cl_s3cjobreport(
            iv_bucket      = iv_report_bucket
            iv_format      = 'Report_CSV_20180820'
            iv_enabled     = abap_true
            iv_prefix      = 'batch-op-reports'
            iv_reportscope = 'AllTasks'
          )
        ).
        ov_job_id = lo_result->get_jobid( ).
        MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteJobTagging`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteJobTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously deletes the tags associated with a specific batch job.
    *
    * @param jobId     The ID of the batch job whose tags should be deleted.
    * @param accountId The ID of the account associated with the batch job.
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
```

```
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Einzelheiten zur API finden Sie DeleteJobTaggingin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise
```

- Einzelheiten zur API finden Sie DeleteJobTaggingin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_s3c->deletejobtagging(
      iv_accountid = iv_account_id
      iv_jobid     = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
  CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_nf.
  CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_cli.
  CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteJobTaggingin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So beschreiben Sie einen Auftrag von Amazon S3 Batch Operations

Im folgenden `describe-job` werden die Konfigurationsparameter und der Status für den angegebenen Batchoperationsauftrag bereitgestellt.

```
aws s3control describe-job \
    --account-id 123456789012 \
    --job-id 93735294-df46-44d5-8638-6356f335324e
```

Ausgabe:

```
{
    "Job": {
        "TerminationDate": "2019-10-03T21:49:53.944Z",
        "JobId": "93735294-df46-44d5-8638-6356f335324e",
        "FailureReasons": [],
        "Manifest": {
            "Spec": {
                "Fields": [
                    "Bucket",
                    "Key"
                ],
                "Format": "S3BatchOperations_CSV_20180820"
            },
            "Location": {
                "ETag": "69f52a4e9f797e987155d9c8f5880897",
                "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-
report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
            }
        },
```

```
        "Operation": {
            "S3PutObjectTagging": {
                "TagSet": [
                    {
                        "Value": "true",
                        "Key": "confidential"
                    }
                ]
            }
        },
        "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
        "ProgressSummary": {
            "TotalNumberOfTasks": 8,
            "NumberOfTasksFailed": 0,
            "NumberOfTasksSucceeded": 8
        },
        "Priority": 42,
        "Report": {
            "ReportScope": "AllTasks",
            "Format": "Report_CSV_20180820",
            "Enabled": true,
            "Prefix": "batch-op-create-job",
            "Bucket": "arn:aws:s3:::employee-records-logs"
        },
        "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-
df46-44d5-8638-6356f335324e",
        "CreationTime": "2019-10-03T21:48:48.048Z",
        "Status": "Complete"
    }
}
```

- Einzelheiten zur API finden Sie DescribeJobin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the specified job.
    *
    * @param jobId     the ID of the job to describe
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job
description is available
    * @throws RuntimeException if an error occurs while describing the job
    */
   public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
       DescribeJobRequest jobRequest = DescribeJobRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return getAsyncClient().describeJob(jobRequest)
           .thenAccept(response -> {
               System.out.println("Job ID: " + response.job().jobId());
               System.out.println("Description: " +
response.job().description());
               System.out.println("Status: " + response.job().statusAsString());
               System.out.println("Role ARN: " + response.job().roleArn());
               System.out.println("Priority: " + response.job().priority());
               System.out.println("Progress Summary: " +
response.job().progressSummary());

               // Print out details about the job manifest.
               JobManifest manifest = response.job().manifest();
               System.out.println("Manifest Location: " +
manifest.location().objectArn());
               System.out.println("Manifest ETag: " +
manifest.location().eTag());

               // Print out details about the job operation.
               JobOperation operation = response.job().operation();
               if (operation.s3PutObjectTagging() != null) {
                   System.out.println("Operation: S3 Put Object Tagging");
                   System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
               }

               // Print out details about the job report.
```

```
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " +
report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- Einzelheiten zur API finden Sie DescribeJobin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
```

```
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise
```

- Einzelheiten zur API finden Sie DescribeJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3c->describejob(         " oo_result is returned for
testing purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lo_job) = oo_result->get_job( ).
       DATA(lv_status) = lo_job->get_status( ).
       DATA(lv_priority) = lo_job->get_priority( ).
       DATA(lo_progress) = lo_job->get_progresssummary( ).
       IF lo_progress IS NOT INITIAL.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                 |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                 |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
```

```
                  |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
       ELSE.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
       ENDIF.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_srv.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetJobTagging`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetJobTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Einzelheiten zur API finden Sie GetJobTaggingin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise
```

- Einzelheiten zur API finden Sie GetJobTaggingin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3c->getjobtagging(       " oo_result is returned for
testing purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lt_tags) = oo_result->get_tags( ).
       MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|
TYPE 'I'.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_srv.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetJobTaggingin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`PutJobTagging`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `PutJobTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId     the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
   public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
       S3Tag departmentTag = S3Tag.builder()
           .key("department")
           .value("Marketing")
           .build();

       S3Tag fiscalYearTag = S3Tag.builder()
           .key("FiscalYear")
           .value("2020")
           .build();

       PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .tags(departmentTag, fiscalYearTag)
           .build();
```

```
        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Einzelheiten zur API finden Sie PutJobTaggingin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
                ]
            )
```

```
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise
```

- Einzelheiten zur API finden Sie PutJobTaggingin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3c->putjobtagging(
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          it_tags      = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Environment'
                iv_value = 'Development' ) )
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Team'
                iv_value = 'DataProcessing' ) )
          )
        ).
        MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
        MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_tags.
```

```
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Einzelheiten zur API finden Sie PutJobTaggingin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateJobPriority`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateJobPriority` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So aktualisieren Sie die Auftragspriorität eines Auftrags von Amazon S3 Batch Operations

Im folgenden Beispiel für `update-job-priority` wird der angegebene Auftrag auf eine neue Priorität aktualisiert.

```
aws s3control update-job-priority \
    --account-id 123456789012 \
    --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
    --priority 52
```

Ausgabe:

```
{
```

```
    "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
    "Priority": 52
}
```

- Einzelheiten zur API finden Sie UpdateJobPriorityin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the priority of a job asynchronously.
    *
    * @param jobId     the ID of the job to update
    * @param accountId the ID of the account associated with the job
    * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
    */
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
               future.complete(null); // Complete the CompletableFuture on
successful execution
           })
           .exceptionally(ex -> {
```

```
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- Einzelheiten zur API finden Sie UpdateJobPriorityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
```

```
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return
```

- Einzelheiten zur API finden Sie UpdateJobPriorityin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3c->updatejobpriority(   " oo_result is returned for
 testing purposes.
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          iv_priority  = 60
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateJobPriorityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateJobStatus`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateJobStatus` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So aktualisieren Sie den Status eines Auftrags von Amazon S3 Batch Operations

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag gelöscht, der auf Genehmigung wartet.

```
aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
    --requested-job-status Cancelled
```

Ausgabe:

```
{
    "Status": "Cancelled",
    "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag bestätigt und ausgeführt, der auf Genehmigung wartet.

```
aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
    --requested-job-status Ready

Output::

{
```

```
    "Status": "Ready",
    "JobId": "5782949f-3301-4fb3-
be34-8d5bab54dbca"
}
```

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag gelöscht, der bereits ausgeführt wird.

```
 aws s3control update-job-status \
    --account-id 123456789012 \
    --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
    --requested-job-status Cancelled

Output::
{
         "Status": "Cancelling",
         "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

- Einzelheiten zur API finden Sie UpdateJobStatusin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
 been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
 complete exceptionally.
     */
```

```
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Einzelheiten zur API finden Sie UpdateJobStatusin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
```

```
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be
cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise
```

- Einzelheiten zur API finden Sie UpdateJobStatusin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_requested_status = 'Cancelled'
        oo_result = lo_s3c->updatejobstatus(     " oo_result is returned for
 testing purposes.
          iv_accountid          = iv_account_id
          iv_jobid              = iv_job_id
          iv_requestedjobstatus = iv_requested_status
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
        MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_js.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateJobStatusin der API-Referenz zum AWS SDK für SAP ABAP.

Codebeispiele für S3 Directory Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon S3 Directory Buckets mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Benutzerhandbuch zu S3-Verzeichnis-Buckets – Weitere Informationen zu S3-Verzeichnis-Buckets.
- API-Referenz für S3-Verzeichnis-Buckets – Details zu allen verfügbaren Aktionen von S3-Verzeichnis-Buckets.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für S3-Verzeichnis-Buckets mit AWS SDKs
  - Hello Amazon-S3-Verzeichnis-Buckets
  - Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK
  - Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs
    - Verwenden Sie AbortMultipartUploadmit einem AWS SDK
    - Verwenden Sie CompleteMultipartUploadmit einem AWS SDK
    - Verwenden Sie CopyObjectmit einem AWS SDK
    - Verwenden Sie CreateBucketmit einem AWS SDK
    - Verwenden Sie CreateMultipartUploadmit einem AWS SDK
    - Verwenden Sie CreateSessionmit einem AWS SDK
    - Verwenden Sie mit DeleteBucketeinem AWS SDK
    - Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK
    - Verwenden Sie DeleteBucketPolicymit einem AWS SDK
    - Verwenden Sie DeleteObjectmit einem AWS SDK
    - Verwenden Sie DeleteObjectsmit einem AWS SDK
    - Verwenden Sie GetBucketEncryptionmit einem AWS SDK

- Verwenden Sie GetBucketPolicymit einem AWS SDK
- Verwenden Sie GetObjectmit einem AWS SDK
- Verwenden Sie GetObjectAttributesmit einem AWS SDK
- Verwenden Sie HeadBucketmit einem AWS SDK
- Verwenden Sie HeadObjectmit einem AWS SDK
- Verwenden Sie ListDirectoryBucketsmit einem AWS SDK
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK
- Verwenden Sie ListObjectsV2 mit einem AWS SDK
- Verwenden Sie ListPartsmit einem AWS SDK
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK
- Verwenden Sie PutBucketPolicymit einem AWS SDK
- Verwenden Sie PutObjectmit einem AWS SDK
- Verwenden Sie UploadPartmit einem AWS SDK
- Verwenden Sie UploadPartCopymit einem AWS SDK

- Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs
  - Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK

Grundlegende Beispiele für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon-S3-Verzeichnis-Buckets mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon-S3-Verzeichnis-Buckets
- Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK
- Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs
  - Verwenden Sie AbortMultipartUploadmit einem AWS SDK
  - Verwenden Sie CompleteMultipartUploadmit einem AWS SDK
  - Verwenden Sie CopyObjectmit einem AWS SDK
  - Verwenden Sie CreateBucketmit einem AWS SDK
  - Verwenden Sie CreateMultipartUploadmit einem AWS SDK

- Verwenden Sie CreateSessionmit einem AWS SDK
- Verwenden Sie mit DeleteBucketeinem AWS SDK
- Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK
- Verwenden Sie DeleteBucketPolicymit einem AWS SDK
- Verwenden Sie DeleteObjectmit einem AWS SDK
- Verwenden Sie DeleteObjectsmit einem AWS SDK
- Verwenden Sie GetBucketEncryptionmit einem AWS SDK
- Verwenden Sie GetBucketPolicymit einem AWS SDK
- Verwenden Sie GetObjectmit einem AWS SDK
- Verwenden Sie GetObjectAttributesmit einem AWS SDK
- Verwenden Sie HeadBucketmit einem AWS SDK
- Verwenden Sie HeadObjectmit einem AWS SDK
- Verwenden Sie ListDirectoryBucketsmit einem AWS SDK
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK
- Verwenden Sie ListObjectsV2 mit einem AWS SDK
- Verwenden Sie ListPartsmit einem AWS SDK
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK
- Verwenden Sie PutBucketPolicymit einem AWS SDK
- Verwenden Sie PutObjectmit einem AWS SDK
- Verwenden Sie UploadPartmit einem AWS SDK
- Verwenden Sie UploadPartCopymit einem AWS SDK

Hello Amazon-S3-Verzeichnis-Buckets

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Verzeichnis-Buckets von Amazon S3.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.s3.directorybucket;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
```

```
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
 code: {}",
```

```
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
```

```
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
```

```
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String
 bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error
 code: " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateBucket
  - ListDirectoryBuckets

Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.

- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.
- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
 UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";
```

```
    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    // Runs the scenario.
    private static void s3ExpressScenario() {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
        logger.info("""
            Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
            specifically when working in the same Availability Zone (AZ). To see
the best results in this example and when you implement
            directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
            bucket.
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        // Create an optional VPC and create 2 IAM users.
        UserNames userNames = createVpcUsers();
        String expressUserName = userNames.getExpressUserName();
        String regularUserName = userNames.getRegularUserName();

        //  Set up two S3 clients, one regular and one express,
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
        waitForInputToContinue(scanner);
```

```
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
```

```
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few
more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
```

```
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ",
e.getCause().getMessage());
            return;
```

```
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the
object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.

            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
```

```
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}",
downloads);
        }
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
```

```
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
```

```
        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
```

```
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.

            It's important to remember the user permissions when interacting
with
            directory buckets. Instead of validating permissions on every call
as
            regular buckets do, directory buckets utilize the user credentials
and session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
        */
        logger.info(DASHES);
        logger.info("""
```

```
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");
```

```
        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the
stack outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express
permissions
     * @param regularUserName the username for the user with regular S3
permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
```

```
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory
bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory
bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
bucket name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
```

```
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
 bucket."
             """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
 directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
 cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
 cause.getMessage(), ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
 regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
 cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
 cause.getMessage(), ce);
                return;
            }
        }
```

```
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
access key.
     *
     * @param accessKeyId     the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3
client
     * @throws IllegalArgumentException if the access key ID or secret access key
is null
     */
    public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
```

```
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }
```

```
    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it
asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or
its objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
        ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();
```

```
        return s3AsyncClient.listObjectsV2(listRequest)
            .thenCompose(listResponse -> {
                if (!listResponse.contents().isEmpty()) {
                    List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                        .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                        .collect(Collectors.toList());

                    DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                        .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
                        .build();

                    return s3AsyncClient.deleteObjects(deleteRequest)
                        .thenAccept(deleteResponse -> {
                            if (!deleteResponse.errors().isEmpty()) {
                                deleteResponse.errors().forEach(error ->
                                    logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                            }
                        });
                }
                return CompletableFuture.completedFuture(null);
            })
            .thenCompose(ignored -> {
                DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3AsyncClient.deleteBucket(deleteBucketRequest);
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
```

```
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }

    /**
     *  Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
                }
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client   the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
```

```
    */
   public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       // Get the object asynchronously and transform it into a byte array
       return s3Client.getObject(objectRequest,
AsyncResponseTransformer.toBytes())
           .exceptionally(exception -> {
               Throwable cause = exception.getCause();
               if (cause instanceof NoSuchKeyException) {
                   throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
               }
               throw new CompletionException("Failed to get the object",
exception);
           });
   }

   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param s3Client           the S3 async client to use for the copy
operation
    * @param sourceBucket       the name of the source bucket
    * @param sourceKey          the key of the object to be copied in the source
bucket
    * @param destinationBucket  the name of the destination bucket
    * @param destinationKey     the key of the copied object in the destination
bucket
    * @return a {@link CompletableFuture} that completes when the copy operation
is finished
    */
   public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
       CopyObjectRequest copyRequest = CopyObjectRequest.builder()
           .sourceBucket(sourceBucket)
           .sourceKey(sourceKey)
           .destinationBucket(destinationBucket)
           .destinationKey(destinationKey)
```

```
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the
session
     * @param bucketName the name of the S3 bucket for which to create the
session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
```

```
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
```

```
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3Waiter.waitUntilBucketExists(bucketRequestWait);
            })
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                    } else {
```

```
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
                logger.info(bucketName + " is ready");
            });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client     the S3 async client to use for the upload
     * @param bucketName   the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
     * @param text         the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
```

```
    * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
    */
   public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
       CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
           .userName(userName)
           .build();

       return getIAMAsyncClient().createAccessKey(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Access Key Created.");
               } else {
                   if (exception == null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof IamException) {
                           throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                       } else {
                           throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                       }
                   }
               }
           });
   }

   /**
    * Asynchronously selects an Availability Zone ID from the available EC2
zones.
    *
    * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.
    * @throws CompletionException if an error occurs during the request or
processing.
    */
   public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
       DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
           .build();

       return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
           .thenCompose(response -> {
```

```
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return
null if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
```

```
    */
   private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
       Scanner scanner = new Scanner(System.in);
       int index = -1;

       while (index < 0 || index >= zoneIds.size()) {
           logger.info("Select an availability zone:");
           IntStream.range(0, zoneIds.size()).forEach(i ->
               logger.info(i + ": " + zoneIds.get(i))
           );

           logger.info("Enter the number corresponding to your choice: ");
           if (scanner.hasNextInt()) {
               index = scanner.nextInt();
           } else {
               scanner.next();
           }
       }

       AvailabilityZone selectedZone = zonesList.get(index);
       logger.info("You selected: " + selectedZone.zoneId());
       return selectedZone;
   }

   /**
    * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
    * creating a VPC endpoint for the S3 service.
    *
    * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
    *         VPC ID and VPC endpoint ID.
    */
   public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
       String cidr = "10.0.0.0/16";
       CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
           .cidrBlock(cidr)
           .build();

       return getEc2AsyncClient().createVpc(vpcRequest)
           .thenCompose(vpcResponse -> {
```

```
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());
```

```
                CreateVpcEndpointRequest endpointRequest =
 CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
 vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
 vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: "
 + cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject

- GetObject
- ListObjects
- PutObject

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
        echo "\n";
        echo "--------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "--------------------------------------\n";

        // Change these both of these values to use a different region/
availability zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
 in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
 resources in the same AZ as your Directory
bucket.\n
INTRO;
```

```
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
 is running in an EC2 instance in the same AZ as your Directory buckets will be.
\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
 VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
 and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
```

```
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
 $regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
 $expressKey['SecretAccessKey']);
```

```
        // 3. Create an additional client using the credentials with S3 Express
 permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3
 Express permissions.\n";
        echo "This client is created with the credentials associated with the
 user account with the S3 Express policy attached, so it can perform S3 Express
 operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user.
 Then, a new S3 Client and Service were created using that user's credentials.
\n";
        echo "We can now use this client to make calls to S3 Express operations.
 Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
 Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the
 S3 Express One Zone service.\n";
        echo "Directory buckets behave in different ways from regular S3 buckets,
 which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
 bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets
 in subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
 bucket.\n";
```

```
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");
```

```
        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;
```

```
        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
 normal bucket took $normalTimeDiff.\n";
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a
 different manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how
 the output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use
 ListObjects and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);
```

```
        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical
 order, while the directory bucket does not. This is because the normal bucket
 considers the whole \"key\" to be the object identifies, while the directory
 bucket actually creates directories and uses the object \"key\" as a path to the
 object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;
```

```
    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "\nPlease fix error with bucket deletion before continuing.
\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
 $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
```

```
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object downloading before
continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
```

```
                echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);
```

```
        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args
= [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
```

```
    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array
 $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
```

```
    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
```

```
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
 Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
 3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
 in this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute
 resources in the same AZ as your Directory
bucket.
    """
        )
        press_enter_to_continue()
```

```
        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets,
 one regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
 buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express
 buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
 regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
 Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo
 (y/n)? ",
            q.is_yesno,
        ):
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
 without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
 an EC2 instance in the same AZ as your
```

```
Directory buckets will be. Are you running this in an EC2 instance located in the
 same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
 production!")
        print(
            """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
 permissions to do S3 Express One Zone actions.
            """
        )
        press_enter_to_continue()
        stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
        template_as_string = S3ExpressScenario.get_template_as_string()
        self.stack = self.deploy_cloudformation_stack(stack_name,
 template_as_string)
        regular_user_name = None
        express_user_name = None
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
 'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name
```

```
    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str
    ) -> None:
        """
        Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        :param express_user_name: The name of the user with S3 Express
 permissions.
        :param regular_user_name: The name of the user with regular S3
 permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
 permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
 This client is created with the
credentials associated with the user account with the S3 Express policy attached,
 so it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3__client_with_access_key_credentials(
            regular_credentials
        )
        self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
        s3_express_client = self.create_s3__client_with_access_key_credentials(
            express_credentials
        )
        self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
        print(
            """
All the roles and policies were created and attached to the user. Then a new S3
 Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
 operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
 """
        )
        press_enter_to_continue()
        # 4. Create two buckets.
        print(
```

```
            """
3. Create two buckets.
Now we will create a Directory bucket which is the linchpin of the S3 Express One
 Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here.
 We'll also create a normal bucket, put
an object into the normal bucket, and copy it over to the Directory bucket.
"""
        )

        # Create a directory bucket. These are different from normal S3 buckets
 in subtle ways.
        bucket_prefix = q.ask(
            "Enter a bucket name prefix that will be used for both buckets: ",
            q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
        )

        # Some availability zones are not supported for Directory buckets. We'll
 choose one that is supported.
        print(
            "Now, let's choose an availability zone for the Directory bucket.
 We'll choose one that is supported."
        )
        while True:
            availability_zone = self.select_availability_zone_id(self.region)
            # Construct the parts of a directory bucket name that is made unique
 with a UUID string.
            directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
            max_uuid_length = 63 - len(bucket_prefix) -
 len(directory_bucket_suffix) - 1
            bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
            directory_bucket_name = (
                f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
            )
            regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
            configuration = {
                "Bucket": {
                    "Type": "Directory",
                    "DataRedundancy": "SingleAvailabilityZone",
                },
                "Location": {
                    "Name": availability_zone["ZoneId"],
                    "Type": "AvailabilityZone",
                },
```

```
            }
            press_enter_to_continue()
            print(
                "Now, let's create the actual Directory bucket, as well as a
regular bucket."
            )
            press_enter_to_continue()
            try:
                self.s3_express_wrapper.create_bucket(
                    directory_bucket_name, configuration
                )
                break
            except ClientError as client_error:
                if client_error.response["Error"]["Code"] == "InvalidBucketName":
                    print(
                        f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                    )
                    if q.ask(
                        "Would you like to select a different availability zone?
",
                        q.is_yesno,
                    ):
                        continue
                    else:
                        raise
                else:
                    raise
        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
buckets.
        """
        print(
            """
```

```
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
 bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
 operations are not restricted for
Directory buckets.
        """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
 Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the
 user credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls,
 this is low, but for many concurrent calls
this adds up to a lot of time saved.
"""
        )
        press_enter_to_continue()
        return bucket_object

    def demonstrate_performance(self, bucket_object: str) -> None:
        """
        Demonstrate performance differences between regular and Directory
 buckets.
        :param bucket_object: The name of the object to download from each
 bucket.
        """
        print("")
        print("6. Demonstrate performance difference.")
        print(
```

```
            """
Now, let's do a performance test. We'll download the same object from each bucket
 'downloads' times
and compare the total time needed. Note: the performance difference will be much
 more pronounced if this
example is run in an EC2 instance in the same Availability Zone as the bucket.
"""
        )
        downloads = 1000
        print(
            f"The number of downloads of the same object for this example is set
 at {downloads}."
        )
        if q.ask("Would you like to download a different number? (y/n) ",
 q.is_yesno):
            max_downloads = 1000000
            downloads = q.ask(
                f"Enter a number between 1 and {max_downloads} for the number of
 downloads: ",
                q.is_int,
                q.in_range(1, max_downloads),
            )
        # Download the object 'downloads' times from each bucket and time it to
 demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

            self.s3_express_wrapper.get_object(
                self.directory_bucket_name, bucket_object
            )

        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
 bucket_object)
```

```
        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
 while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{(difference) / 1000000000} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are
 not running on the cloud within a vpc."
            )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
 buckets.
        This is done by creating a few objects in each bucket and listing them to
 show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
 regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and
 therefore retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
 efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause
 unexpected results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
        """
        )
        press_enter_to_continue()
        # Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        other_object = f"other/{bucket_object}"
```

```
        alt_object = f"alt/{bucket_object}"
        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name,
 other_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
 other_object, "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
 "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
 alt_object, "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
 directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
 while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
```

```
        """
        if self.directory_bucket_name is not None:
            self.s3_express_wrapper.delete_bucket_and_objects(
                self.directory_bucket_name
            )
            print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
            self.directory_bucket_name = None

        if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
            print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
            self.regular_bucket_name = None

        if self.stack is not None:
            self.destroy_cloudformation_stack(self.stack)
            self.stack = None

        self.tear_done_vpc()

    def create_access_key(self, user_name: str) -> dict[str, any]:
        """
        Creates an access key for the user.
        :param user_name: The name of the user.
        :return: The access key for the user.
        """
        try:
            access_key = self.iam_client.create_access_key(UserName=user_name)
            return access_key["AccessKey"]
        except ClientError as client_error:
            logging.error(
                "Couldn't create the access key. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def create_s3__client_with_access_key_credentials(
        self, access_key: dict[str, any]
    ) -> client:
        """
        Creates an S3 client with access key credentials.
        :param access_key: The access key for the user.
        :return: The S3 Express One Zone client.
        """
```

```
        try:
            s3_express_client = boto3.client(
                "s3",
                aws_access_key_id=access_key["AccessKeyId"],
                aws_secret_access_key=access_key["SecretAccessKey"],
                region_name=self.region,
            )
            return s3_express_client
        except ClientError as client_error:
            logging.error(
                "Couldn't create the S3 Express One Zone client. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def select_availability_zone_id(self, region: str) -> dict[str, any]:
        """
        Selects an availability zone.
        :param region: The region to select the availability zone from.
        :return: The availability zone dictionary.
        """
        try:
            response = self.ec2_client.describe_availability_zones(
                Filters=[{"Name": "region-name", "Values": [region]}]
            )
            availability_zones = response["AvailabilityZones"]
            zone_names = [zone["ZoneName"] for zone in availability_zones]
            index = q.choose("Select an availability zone: ", zone_names)
            return availability_zones[index]
        except ClientError as client_error:
            logging.error(
                "Couldn't describe availability zones. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script
and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
```

```
        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
```

```
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
 "s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
            route_table_id = response["RouteTables"][0]["RouteTableId"]
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
```

```
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
            print(f"Created vpc endpoint {self.vpc_endpoint_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc endpoint. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def tear_done_vpc(self) -> None:
        if self.vpc_endpoint_id is not None:
            try:
                self.ec2_client.delete_vpc_endpoints(
                    VpcEndpointIds=[self.vpc_endpoint_id]
                )
                print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
                self.vpc_endpoint_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc endpoint %s. Here's why: %s",
                    self.vpc_endpoint_id,
                    client_error.response["Error"]["Message"],
                )
        if self.vpc_id is not None:
            try:
                self.ec2_client.delete_vpc(VpcId=self.vpc_id)
                print(f"Deleted vpc {self.vpc_id}")
                self.vpc_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc %s. Here's why: %s",
                    self.vpc_id,
                    client_error.response["Error"]["Message"],
                )
```

Eine Wrapper-Klasse für SDK-Funktionen von Amazon S3 Express.

```
class S3ExpressWrapper:
```

```
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                              access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:
        """
        Creates a bucket.
        :param bucket_name: The name of the bucket.
        :param bucket_configuration: The optional configuration for the bucket.
        """
        try:
            params = {"Bucket": bucket_name}
            if bucket_configuration:
                params["CreateBucketConfiguration"] = bucket_configuration

            self.s3_client.create_bucket(**params)
        except ClientError as client_error:
            # Do not log InvalidBucketName error because it is logged elsewhere.
            if client_error.response["Error"]["Code"] != "InvalidBucketName":
                logging.error(
                    "Couldn't create the bucket %s. Here's why: %s",
```

```
                    bucket_name,
                    client_error.response["Error"]["Message"],
                )
            raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
         :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a
bucket to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
                                error["Key"],
                                error["Message"],
                            )

            self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )

    def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
        """
        Puts an object into a bucket.
```

```
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        :param content: The content of the object.
        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

    def list_objects(self, bucket: str) -> list[str]:
        """
        Lists objects in a bucket.
        :param bucket: The name of the bucket.
        :return: The list of objects in the bucket.
        """
        try:
            response = self.s3_client.list_objects_v2(Bucket=bucket)
            return response.get("Contents", [])
        except ClientError as client_error:
            logging.error(
                "Couldn't list objects in bucket %s. Here's why: %s",
                bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def copy_object(
        self,
        source_bucket: str,
        source_key: str,
        destination_bucket: str,
        destination_key: str,
    ) -> None:
        """
        Copies an object from one bucket to another.
        :param source_bucket: The source bucket.
        :param source_key: The source key.
```

```
        :param destination_bucket: The destination bucket.
        :param destination_key: The destination key.
        :return: None
        """
        try:
            self.s3_client.copy_object(
                CopySource={"Bucket": source_bucket, "Key": source_key},
                Bucket=destination_bucket,
                Key=destination_key,
            )
        except ClientError as client_error:
            logging.error(
                "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
                source_key,
                source_bucket,
                destination_bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise


    def get_object(self, bucket_name: str, object_key: str) -> None:
        """
        Gets an object from a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        """
```

```
        try:
            self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't get the object %s from bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne S3 Directory Buckets-Aktionen mit SDKs ausgeführt werden. AWS Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die S3-Verzeichnis-Buckets-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon-S3-Verzeichnis-Buckets.

Beispiele

- Verwenden Sie AbortMultipartUploadmit einem AWS SDK
- Verwenden Sie CompleteMultipartUploadmit einem AWS SDK
- Verwenden Sie CopyObjectmit einem AWS SDK
- Verwenden Sie CreateBucketmit einem AWS SDK
- Verwenden Sie CreateMultipartUploadmit einem AWS SDK
- Verwenden Sie CreateSessionmit einem AWS SDK
- Verwenden Sie mit DeleteBucketeinem AWS SDK
- Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK
- Verwenden Sie DeleteBucketPolicymit einem AWS SDK
- Verwenden Sie DeleteObjectmit einem AWS SDK
- Verwenden Sie DeleteObjectsmit einem AWS SDK
- Verwenden Sie GetBucketEncryptionmit einem AWS SDK
- Verwenden Sie GetBucketPolicymit einem AWS SDK
- Verwenden Sie GetObjectmit einem AWS SDK
- Verwenden Sie GetObjectAttributesmit einem AWS SDK
- Verwenden Sie HeadBucketmit einem AWS SDK
- Verwenden Sie HeadObjectmit einem AWS SDK
- Verwenden Sie ListDirectoryBucketsmit einem AWS SDK
- Verwenden Sie ListMultipartUploadsmit einem AWS SDK
- Verwenden Sie ListObjectsV2 mit einem AWS SDK
- Verwenden Sie ListPartsmit einem AWS SDK
- Verwenden Sie PutBucketEncryptionmit einem AWS SDK
- Verwenden Sie PutBucketPolicymit einem AWS SDK
- Verwenden Sie PutObjectmit einem AWS SDK
- Verwenden Sie UploadPartmit einem AWS SDK
- Verwenden Sie UploadPartCopymit einem AWS SDK

Verwenden Sie **`AbortMultipartUpload`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`AbortMultipartUpload`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Brechen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;


    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false
 otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
 bucketName);
        try {
            // Abort the multipart upload
```

```
            AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie AbortMultipartUploadin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CompleteMultipartUpload`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CompleteMultipartUpload`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Schließen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client
 s3Client, String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                    .parts(uploadParts)
```

```
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie CompleteMultipartUploadin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CopyObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CopyObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt aus einem Verzeichnis-Bucket in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
```

```
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateBucket`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateBucket`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String
 bucketName, String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);
```

```
        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateMultipartUpload`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateMultipartUpload`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique
 upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
```

```
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code:
 {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie CreateMultipartUploadin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`CreateSession`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateSession`.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                          access to AWS S3 services.
        """
        self.s3_client = s3_client
```

```
    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateSessionin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DeleteBucket`**einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteBucket`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie einen S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);
```

```
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteBucketEncryption`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteBucketEncryption`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration für einen Verzeichnis-Bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
```

```
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code:
 {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteBucketEncryptionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteBucketPolicy`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteBucketPolicy`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Bucket-Richtlinie für einen Verzeichnis-Bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
 DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Objekts in einem Verzeichnis-Bucket

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




   /**
    * Deletes an object from the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey  The key (name) of the object to be deleted
    */
   public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
       logger.info("Deleting object: {} from bucket: {}", objectKey,
 bucketName);

       try {
           // Create a DeleteObjectRequest
           DeleteObjectRequest deleteObjectRequest =
 DeleteObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           // Delete the object
           s3Client.deleteObject(deleteObjectRequest);
           logger.info("Object {} has been deleted", objectKey);

       } catch (S3Exception e) {
           logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode(), e);
           throw e;
       }
   }
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`DeleteObjects`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteObjects`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte in einem Verzeichnis-Bucket

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
```

```
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetBucketEncryption`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetBucketEncryption`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration eines Verzeichnis-Buckets.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
```

```
            GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket encryption configuration
            GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie GetBucketEncryptionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetBucketPolicy`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetBucketPolicy`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Richtlinie eines Verzeichnis-Buckets ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
```

```
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetObject`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen
- Erstellen einer vorsignierten URL, um ein Objekt abzurufen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt aus einem Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();
```

```
            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`GetObjectAttributes`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`GetObjectAttributes`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Objektattribute aus einem eines Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);

        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
 GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
 ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
```

```
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectAttributesin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`HeadBucket`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`HeadBucket`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Prüfen Sie, ob das angegebene S3-Verzeichnis-Bucket existiert und zugänglich ist.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String
 bucketName) {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie HeadBucketin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`HeadObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`HeadObject`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Metadaten eines Objekts in einem Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
```

```
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- Einzelheiten zur API finden Sie HeadObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListDirectoryBuckets`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListDirectoryBuckets`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Verzeichnis-Buckets auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
 ListDirectoryBucketsRequest.builder().build();
```

```
            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
 s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie ListDirectoryBucketsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListMultipartUploads`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListMultipartUploads`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie mehrteilige Uploads in einen Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * Lists multipart uploads for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of MultipartUpload objects representing the multipart
 uploads
     */
    public static List<MultipartUpload>
 listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
        logger.info("Listing in-progress multipart uploads for bucket: {}",
 bucketName);

        try {
            // Create a ListMultipartUploadsRequest
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            // List the multipart uploads
```

```
            ListMultipartUploadsResponse response =
 s3Client.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                logger.info("In-progress multipart upload: Upload ID: {}, Key:
 {}, Initiated: {}", upload.uploadId(),
                        upload.key(), upload.initiated());
            }
            return uploads;

        } catch (S3Exception e) {
            logger.error("Failed to list multipart uploads: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Einzelheiten zur API finden Sie ListMultipartUploadsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListObjectsV2`** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListObjectsV2`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte in einem Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);

        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
 ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
 s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
```

```
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
            logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`ListParts`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListParts`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Teile eines mehrteiligen Uploads in einen Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
```

```
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Lists the parts of a multipart upload for the specified S3 directory
 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object being uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @return A list of Part representing the parts of the multipart upload
     */
    public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
            String objectKey, String uploadId) {
        logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

        try {
            // Create a ListPartsRequest
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                    .bucket(bucketName)
                    .uploadId(uploadId)
                    .key(objectKey)
                    .build();

            // List the parts of the multipart upload
            ListPartsResponse response = s3Client.listParts(listPartsRequest);
            List<Part> parts = response.parts();
            for (Part part : parts) {
                logger.info("Uploaded part: Part number = \"{}\", etag = {}",
 part.partNumber(), part.eTag());
            }
```

```
            return parts;

        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- Einzelheiten zur API finden Sie ListPartsin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`PutBucketEncryption`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PutBucketEncryption`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Bucket-Verschlüsselung auf einen Verzeichnis-Bucket fest.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
 software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
 bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For
 directory
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
 keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
 ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
 enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();

        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
 ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
```

```
        PutBucketEncryptionRequest putRequest =
PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketEncryptionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`PutBucketPolicy`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PutBucketPolicy`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wenden Sie eine Bucket-Richtlinie auf einen Verzeichnis-Bucket an.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the
 same account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
```

```
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`PutObject`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`PutObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie ein Objekts in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey, Path filePath) {
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
 bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
 e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
```

```
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UploadPart`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UploadPart`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie einen Teil eines mehrteiligen Uploads in einen Verzeichnis-Bucket hoch.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
```

```
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the
 completed
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
     */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
 s3Client, String bucketName,
            String objectKey, String uploadId, Path filePath) throws IOException
 {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
 bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
```

```
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
        return uploadedParts;
    }
```

- Einzelheiten zur API finden Sie UploadPartin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`UploadPartCopy`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UploadPartCopy`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Kopierteile basierend auf der Größe des Quellobjekts und kopieren Sie einzelne Teile in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart>
 multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse headObjectResponse =
 s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
```

```
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
    }
```

- Einzelheiten zur API finden Sie UploadPartCopyin der AWS SDK for Java 2.x API-Referenz.

Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in S3 Directory Buckets mit SDKs implementieren. AWS Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von S3-Verzeichnis-Buckets oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK

Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine vorsignierte URL für S3-Verzeichnis-Buckets erstellen und ein Objekt hochladen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte GET-URL für den Zugriff auf ein Objekt in einem S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Generates a presigned URL for accessing an object in the specified S3
     * directory bucket.
     *
     * @param s3Presigner The S3 presigner client used to generate the presigned
 URL
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to access
     * @return A presigned URL for accessing the specified object
     */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
            String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);
```

```
        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                    .getObjectRequest(getObjectRequest)
                    .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für SageMaker KI unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon SageMaker AI mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- SageMaker KI-Entwicklerhandbuch — Weitere Informationen zu SageMaker KI.
- SageMaker KI-API-Referenz — Details zu allen verfügbaren SageMaker KI-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von SageMaker KI AWS SDKs
  - Hallo AI SageMaker
  - Aktionen für SageMaker KI unter Verwendung AWS SDKs
    - Verwenden Sie CreateEndpointmit einem AWS SDK
    - Verwenden Sie CreateModelmit einem AWS SDK
    - Verwenden Sie CreatePipelinemit einem AWS SDK
    - Verwenden Sie CreateTrainingJobmit einem AWS SDK
    - Verwenden Sie CreateTransformJobmit einem AWS SDK
    - Verwenden Sie DeleteEndpointmit einem AWS SDK
    - Verwenden Sie DeleteModelmit einem AWS SDK
    - Verwenden Sie DeletePipelinemit einem AWS SDK
    - Verwenden Sie DescribePipelineExecutionmit einem AWS SDK
    - Verwenden Sie DescribeTrainingJobmit einem AWS SDK
    - Verwenden Sie ListAlgorithmsmit einem AWS SDK
    - Verwenden Sie ListModelsmit einem AWS SDK
    - Verwenden Sie ListNotebookInstancesmit einem AWS SDK
    - Verwenden Sie ListTrainingJobsmit einem AWS SDK

    - Verwenden Sie StartPipelineExecutionmit einem AWS SDK
    - Verwenden Sie UpdatePipelinemit einem AWS SDK
- Szenarien für die Verwendung von SageMaker KI AWS SDKs
  - Beginnen Sie mit SageMaker KI-Geodatenaufträgen in einer Pipeline mithilfe eines AWS SDK
  - Beginnen Sie mit SageMaker KI-Modellen und Endpunkten mithilfe eines AWS SDK
  - Erste Schritte mit Feature-Stores für maschinelles Lernen

Grundlegende Beispiele für die Verwendung von SageMaker KI AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon SageMaker AI mit AWS SDKs verwenden können.

Beispiele

- Hallo AI SageMaker
- Aktionen für SageMaker KI unter Verwendung AWS SDKs
  - Verwenden Sie CreateEndpointmit einem AWS SDK
  - Verwenden Sie CreateModelmit einem AWS SDK
  - Verwenden Sie CreatePipelinemit einem AWS SDK
  - Verwenden Sie CreateTrainingJobmit einem AWS SDK
  - Verwenden Sie CreateTransformJobmit einem AWS SDK
  - Verwenden Sie DeleteEndpointmit einem AWS SDK
  - Verwenden Sie DeleteModelmit einem AWS SDK
  - Verwenden Sie DeletePipelinemit einem AWS SDK
  - Verwenden Sie DescribePipelineExecutionmit einem AWS SDK
  - Verwenden Sie DescribeTrainingJobmit einem AWS SDK
  - Verwenden Sie ListAlgorithmsmit einem AWS SDK
  - Verwenden Sie ListModelsmit einem AWS SDK
  - Verwenden Sie ListNotebookInstancesmit einem AWS SDK
  - Verwenden Sie ListTrainingJobsmit einem AWS SDK
  - Verwenden Sie StartPipelineExecutionmit einem AWS SDK
  - Verwenden Sie UpdatePipelinemit einem AWS SDK

Hallo AI SageMaker

Die folgenden Codebeispiele zeigen, wie Sie mit SageMaker KI beginnen können.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SageMaker;
using Amazon.SageMaker.Model;

namespace SageMakerActions;

public static class HelloSageMaker
{
    static async Task Main(string[] args)
    {
        var sageMakerClient = new AmazonSageMakerClient();

        Console.WriteLine($"Hello Amazon SageMaker! Let's list some of your
 notebook instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five notebook instances.
        var response = await sageMakerClient.ListNotebookInstancesAsync(
            new ListNotebookInstancesRequest()
            {
                MaxResults = 5
            });

        if (!response.NotebookInstances.Any())
        {
            Console.WriteLine($"No notebook instances found.");
```

```
            Console.WriteLine("See https://docs.aws.amazon.com/sagemaker/latest/
dg/howitworks-create-ws.html to create one.");
        }

        foreach (var notebookInstance in response.NotebookInstances)
        {
            Console.WriteLine($"\tInstance:
 {notebookInstance.NotebookInstanceName}");
            Console.WriteLine($"\tArn: {notebookInstance.NotebookInstanceArn}");
            Console.WriteLine($"\tCreation Date:
 {notebookInstance.CreationTime.ToShortDateString()}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
```

```
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
 sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
 notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
 item.notebookInstanceName());
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  SageMakerClient,
  ListNotebookInstancesCommand,
} from "@aws-sdk/client-sagemaker";

const client = new SageMakerClient({
  region: "us-west-2",
});

export const helloSagemaker = async () => {
  const command = new ListNotebookInstancesCommand({ MaxResults: 5 });

  const response = await client.send(command);
  console.log(
    "Hello Amazon SageMaker! Let's list some of your notebook instances:",
  );

  const instances = response.NotebookInstances || [];

  if (instances.length === 0) {
    console.log(
      "• No notebook instances found. Try creating one in the AWS Management
 Console or with the CreateNotebookInstanceCommand.",
    );
  } else {
    console.log(
      instances
        .map(
          (i) =>
            `• Instance: ${i.NotebookInstanceName}\n  Arn:${
              i.NotebookInstanceArn
            } \n  Creation Date: ${i.CreationTime.toISOString()}`,
        )
        .join("\n"),
    );
  }

  return response;
};
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use
 { sageMakerClient ->
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der API-Referenz zum AWS SDK für Kotlin.

Aktionen für SageMaker KI unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne SageMaker KI-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die SageMaker KI-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von SageMaker KI AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon SageMaker AI API-Referenz.

Beispiele

- Verwenden Sie CreateEndpointmit einem AWS SDK

- Verwenden Sie CreateModelmit einem AWS SDK
- Verwenden Sie CreatePipelinemit einem AWS SDK
- Verwenden Sie CreateTrainingJobmit einem AWS SDK
- Verwenden Sie CreateTransformJobmit einem AWS SDK
- Verwenden Sie DeleteEndpointmit einem AWS SDK
- Verwenden Sie DeleteModelmit einem AWS SDK
- Verwenden Sie DeletePipelinemit einem AWS SDK
- Verwenden Sie DescribePipelineExecutionmit einem AWS SDK
- Verwenden Sie DescribeTrainingJobmit einem AWS SDK
- Verwenden Sie ListAlgorithmsmit einem AWS SDK
- Verwenden Sie ListModelsmit einem AWS SDK
- Verwenden Sie ListNotebookInstancesmit einem AWS SDK
- Verwenden Sie ListTrainingJobsmit einem AWS SDK
- Verwenden Sie StartPipelineExecutionmit einem AWS SDK
- Verwenden Sie UpdatePipelinemit einem AWS SDK

Verwenden Sie **`CreateEndpoint`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateEndpoint`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.

    "Create a production variant as an ABAP object."
    "Identifies a model that you want to host and the resources chosen to deploy
 for hosting it."
    lo_production_variants = NEW #( iv_variantname = iv_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
 iv_initial_instance_count
                                    iv_instancetype = iv_instance_type ).

    INSERT lo_production_variants INTO TABLE lt_production_variants.

    "Create an endpoint configuration."
    TRY.
        oo_ep_config_result = lo_sgm->createendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name
          it_productionvariants = lt_production_variants ).
        MESSAGE 'Endpoint configuration created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
 'E'.
    ENDTRY.

    "Create an endpoint."
    TRY.
        oo_result = lo_sgm->createendpoint(     " oo_result is returned for
 testing purposes. "
            iv_endpointconfigname = iv_endpoint_config_name
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateEndpointin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateModel`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateModel`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

   "Create an ABAP object for the container image based on input variables."
   lo_primarycontainer = NEW #( iv_image = iv_container_image
                                iv_modeldataurl = iv_model_data_url ).

   "Create an Amazon SageMaker model."
   TRY.
       oo_result = lo_sgm->createmodel(        " oo_result is returned for
testing purposes. "
         iv_executionrolearn = iv_execution_role_arn
         iv_modelname = iv_model_name
         io_primarycontainer = lo_primarycontainer ).
       MESSAGE 'Model created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateModelin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreatePipeline`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreatePipeline` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Geodatenaufträgen und Pipelines

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
```

```
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Einzelheiten zur API finden Sie CreatePipelinein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
```

```
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest =
CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }
```

- Einzelheiten zur API finden Sie CreatePipelinein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Funktion, die mithilfe einer lokal bereitgestellten JSON-Definition eine SageMaker KI-Pipeline erstellt.

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using
 PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/
client-sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
```

```
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}
```

- Einzelheiten zur API finden Sie CreatePipelineunter AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- Einzelheiten zur API finden Sie CreatePipelinein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateTrainingJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateTrainingJob`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lt_hyperparameters TYPE /aws1/
cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.

    "Create ABAP internal table for hyperparameters based on input variables."
    "These hyperparameters are based on the Amazon SageMaker built-in algorithm,
 XGBoost."
    lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
    INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
    INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
    INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO
 TABLE lt_hyperparameters.
```

```
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP objects for training data sources."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).

   INSERT lo_trn_channel INTO TABLE lt_input_data_config.

   "Create ABAP objects for validation data sources."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).
```

```
   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   "Create a training job."
   TRY.
       oo_result = lo_sgm->createtrainingjob(    " oo_result is returned for
testing purposes. "
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTrainingJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateTransformJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`CreateTransformJob`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
   DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
   DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
   DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
   DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

   "Create an ABAP object for an Amazon Simple Storage Service (Amazon S3) data
source."
   lo_s3datasource = NEW #( iv_s3uri = iv_tf_data_s3uri
                            iv_s3datatype = iv_tf_data_s3datatype ).

   "Create an ABAP object for data source."
   lo_datasource = NEW #( io_s3datasource = lo_s3datasource ).

   "Create an ABAP object for transform data source."
   lo_transforminput = NEW #( io_datasource = lo_datasource
                              iv_contenttype = iv_tf_data_contenttype
                              iv_compressiontype = iv_tf_data_compressiontype ).

   "Create an ABAP object for resource configuration."
   lo_transformresources = NEW #( iv_instancecount = iv_instance_count
                                  iv_instancetype = iv_instance_type ).

   "Create an ABAP object for output data configuration."
```

```
   lo_transformoutput = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create a transform job."
   TRY.
       oo_result = lo_sgm->createtransformjob(     " oo_result is returned for
testing purposes. "
           iv_modelname = iv_tf_model_name
           iv_transformjobname = iv_tf_job_name
           io_transforminput = lo_transforminput
           io_transformoutput = lo_transformoutput
           io_transformresources = lo_transformresources ).
       MESSAGE 'Transform job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTransformJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteEndpoint`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteEndpoint`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Delete an endpoint."
    TRY.
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
        DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
 { lo_endpoint_exception->av_err_msg }|.
        MESSAGE lv_endpoint_error TYPE 'E'.
    ENDTRY.

    "Delete an endpoint configuration."
    TRY.
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name ).
        MESSAGE 'Endpoint deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteEndpointin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteModel`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DeleteModel`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteModelin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeletePipeline`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeletePipeline` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Geodatenaufträgen und Pipelines

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete a SageMaker pipeline by name.
/// </summary>
/// <param name="pipelineName">The name of the pipeline to delete.</param>
/// <returns>The ARN of the pipeline.</returns>
public async Task<string> DeletePipelineByName(string pipelineName)
{
    var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
        new DeletePipelineRequest()
        {
            PipelineName = pipelineName
        });

    return deleteResponse.PipelineArn;
}
```

- Einzelheiten zur API finden Sie DeletePipelinein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Delete a SageMaker pipeline by name.
```

```
    public static void deletePipeline(SageMakerClient sageMakerClient, String
 pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }
```

- Einzelheiten zur API finden Sie DeletePipelinein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Syntax zum Löschen einer SageMaker AI-Pipeline. Dieser Code ist Teil einer größeren Funktion. Weitere Informationen finden Sie unter „Eine Pipeline erstellen“ oder im GitHub Repository.

```
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
```

- Einzelheiten zur API finden Sie unter DeletePipelineAPI-Referenz.AWS SDK für JavaScript

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- Einzelheiten zur API finden Sie DeletePipelinein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribePipelineExecution`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribePipelineExecution` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Geodatenaufträgen und Pipelines

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus>
CheckPipelineExecutionStatus(string pipelineExecutionArn)
    {
        var describeResponse = await
_amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }
```

- Einzelheiten zur API finden Sie DescribePipelineExecutionin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " +
status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }
```

- Einzelheiten zur API finden Sie DescribePipelineExecutionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Warten Sie, bis die Ausführung einer SageMaker KI-Pipeline erfolgreich ist, fehlschlägt oder beendet wird.

```
/**
```

```
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before
 checking again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}
```

- Einzelheiten zur API finden Sie DescribePipelineExecutionunter AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- Einzelheiten zur API finden Sie DescribePipelineExecutionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DescribeTrainingJob`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`DescribeTrainingJob`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Modellen und Endpunkten

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->describetrainingjob(      " oo_result is returned for
 testing purposes. "
          iv_trainingjobname = iv_training_job_name ).
        MESSAGE 'Retrieved description of training job.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeTrainingJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListAlgorithms`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListAlgorithms`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listalgorithms(         " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of algorithms.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListAlgorithmsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListModels`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListModels`.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listmodels(           " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of models.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListModelsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListNotebookInstances`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListNotebookInstances` verwendet wird.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name();

        println!("  Name :          {}", n_name.unwrap_or("Unknown"));
        println!("  Status :        {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listnotebookinstances(        " oo_result is returned
 for testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of notebook instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListNotebookInstancesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTrainingJobs`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListTrainingJobs` verwendet wird.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_jobs(client: &Client) -> Result<(), Error> {
```

```
    let job_details = client.list_training_jobs().send().await?;

    println!("Jobs:");

    for j in job_details.training_job_summaries() {
        let name = j.training_job_name().unwrap_or("Unknown");
        let creation_time = j.creation_time().expect("creation
 time").to_chrono_utc()?;
        let training_end_time = j
            .training_end_time()
            .expect("Training end time")
            .to_chrono_utc()?;

        let status = j.training_job_status().expect("training status");
        let duration = training_end_time - creation_time;

        println!("  Name:               {}", name);
        println!(
            "  Creation date/time: {}",
            creation_time.format("%Y-%m-%d@%H:%M:%S")
        );
        println!("  Duration (seconds): {}", duration.num_seconds());
        println!("  Status:             {:?}", status);

        println!();
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListTrainingJobsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listtrainingjobs(       " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains
          iv_maxresults = iv_max_results ).
        MESSAGE 'Retrieved list of training jobs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTrainingJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartPipelineExecution`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartPipelineExecution` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Geodatenaufträgen und Pipelines

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple
Storage Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
```

```
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-
execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value
 = JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value
 = JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value
 = JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }
```

- Einzelheiten zur API finden Sie StartPipelineExecutionin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
```

```
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();
```

```
        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig
\":{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" +
 gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-
execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }
```

- Einzelheiten zur API finden Sie StartPipelineExecutionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie die Ausführung einer SageMaker KI-Pipeline.

```
/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a
 CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
```

```
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse
 Geocoding requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
```

```
  };
}
```

- Einzelheiten zur API finden Sie StartPipelineExecutionunter AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn:
 String?, pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }
```

```
    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName
\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
```

```
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- Einzelheiten zur API finden Sie StartPipelineExecutionin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`UpdatePipeline`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`UpdatePipeline`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Geodatenaufträgen und Pipelines

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- Einzelheiten zur API finden Sie UpdatePipelinein der AWS SDK für .NET API-Referenz.

Szenarien für die Verwendung von SageMaker KI AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in SageMaker KI mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von SageMaker KI aufrufen oder mit anderen AWS-Services kombinieren. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Beginnen Sie mit SageMaker KI-Geodatenaufträgen in einer Pipeline mithilfe eines AWS SDK
- Beginnen Sie mit SageMaker KI-Modellen und Endpunkten mithilfe eines AWS SDK
- Erste Schritte mit Feature-Stores für maschinelles Lernen

Beginnen Sie mit SageMaker KI-Geodatenaufträgen in einer Pipeline mithilfe eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter SageMaker Pipelines mithilfe von AWS SDKs erstellen und ausführen. Community.aws

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die SageMaker KI-Operationen abdeckt.

```
using System.Text.Json;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

namespace SageMakerActions;

/// <summary>
/// Wrapper class for Amazon SageMaker actions and logic.
/// </summary>
public class SageMakerWrapper
{
    private readonly IAmazonSageMaker _amazonSageMaker;
    public SageMakerWrapper(IAmazonSageMaker amazonSageMaker)
    {
        _amazonSageMaker = amazonSageMaker;
    }

    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
```

```
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }

    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple
Storage Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
```

```
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-
execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
```

```
                    new Parameter() { Name = "parameter_vej_input_config", Value
 = JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value
 = JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value
 = JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }

    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus>
 CheckPipelineExecutionStatus(string pipelineExecutionArn)
    {
        var describeResponse = await
 _amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }

    /// <summary>
    /// Delete a SageMaker pipeline by name.
    /// </summary>
    /// <param name="pipelineName">The name of the pipeline to delete.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public async Task<string> DeletePipelineByName(string pipelineName)
    {
        var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
            new DeletePipelineRequest()
            {
                PipelineName = pipelineName
            });

        return deleteResponse.PipelineArn;
```

```
    }
}
```

Erstellen Sie eine Funktion, die Rückrufe aus der SageMaker AI-Pipeline verarbeitet.

```
using System.Text.Json;
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

// Assembly attribute to enable the AWS Lambda function's JSON input to be
 converted into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SageMakerLambda;

/// <summary>
/// The AWS Lambda function handler for the Amazon SageMaker pipeline.
/// </summary>
public class SageMakerLambdaFunction
{
    /// <summary>
    /// Default constructor. This constructor is used by AWS Lambda to construct
 the instance. When invoked in a Lambda environment
    /// the AWS credentials will come from the AWS Identity and Access Management
 (IAM) role associated with the function. The AWS Region will be set to the
    /// Region that the Lambda function is running in.
    /// </summary>
    public SageMakerLambdaFunction()
    {
    }

    /// <summary>
    /// The AWS Lambda function handler that processes events from the SageMaker
 pipeline and starts a job or export.
    /// </summary>
```

```
    /// <param name="request">The custom SageMaker pipeline request object.</
 param>
    /// <param name="context">The Lambda context.</param>
    /// <returns>The dictionary of output parameters.</returns>
    public async Task<Dictionary<string, string>> FunctionHandler(PipelineRequest
  request, ILambdaContext context)
    {
        var geoSpatialClient = new AmazonSageMakerGeospatialClient();
        var sageMakerClient = new AmazonSageMakerClient();
        var responseDictionary = new Dictionary<string, string>();
        context.Logger.LogInformation("Function handler started with request: " +
  JsonSerializer.Serialize(request));
        if (request.Records != null && request.Records.Any())
        {
            context.Logger.LogInformation("Records found, this is a queue event.
  Processing the queue records.");
            foreach (var message in request.Records)
            {
                await ProcessMessageAsync(message, context, geoSpatialClient,
  sageMakerClient);
            }
        }
        else if (!string.IsNullOrEmpty(request.vej_export_config))
        {
            context.Logger.LogInformation("Export configuration found, this is an
  export. Start the Vector Enrichment Job (VEJ) export.");

            var outputConfig =

  JsonSerializer.Deserialize<ExportVectorEnrichmentJobOutputConfig>(
                    request.vej_export_config);

            var exportResponse = await
  geoSpatialClient.ExportVectorEnrichmentJobAsync(
                new ExportVectorEnrichmentJobRequest()
                {
                    Arn = request.vej_arn,
                    ExecutionRoleArn = request.Role,
                    OutputConfig = outputConfig
                });
            context.Logger.LogInformation($"Export response:
  {JsonSerializer.Serialize(exportResponse)}");
            responseDictionary = new Dictionary<string, string>
            {
```

```
                { "export_eoj_status", exportResponse.ExportStatus.ToString() },
                { "vej_arn", exportResponse.Arn }
            };
        }
        else if (!string.IsNullOrEmpty(request.vej_name))
        {
            context.Logger.LogInformation("Vector Enrichment Job name found,
starting the job.");
            var inputConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobInputConfig>(
                    request.vej_input_config);

            var jobConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobConfig>(
                    request.vej_config);

            var jobResponse = await
geoSpatialClient.StartVectorEnrichmentJobAsync(
                new StartVectorEnrichmentJobRequest()
                {
                    ExecutionRoleArn = request.Role,
                    InputConfig = inputConfig,
                    Name = request.vej_name,
                    JobConfig = jobConfig

                });
            context.Logger.LogInformation("Job response: " +
JsonSerializer.Serialize(jobResponse));
            responseDictionary = new Dictionary<string, string>
            {
                { "vej_arn", jobResponse.Arn },
                { "statusCode", jobResponse.HttpStatusCode.ToString() }
            };
        }
        return responseDictionary;
    }

    /// <summary>
    /// Process a queue message and check the status of a SageMaker job.
    /// </summary>
    /// <param name="message">The queue message.</param>
    /// <param name="context">The Lambda context.</param>
    /// <param name="geoClient">The SageMaker GeoSpatial client.</param>
    /// <param name="sageMakerClient">The SageMaker client.</param>
```

```
    /// <returns>Async task.</returns>
    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
ILambdaContext context,
        AmazonSageMakerGeospatialClient geoClient, AmazonSageMakerClient
sageMakerClient)
    {
        context.Logger.LogInformation($"Processed message {message.Body}");

        // Get information about the SageMaker job.
        var payload = JsonSerializer.Deserialize<QueuePayload>(message.Body);
        context.Logger.LogInformation($"Payload token {payload!.token}");
        var token = payload.token;

        if (payload.arguments.ContainsKey("vej_arn"))
        {
            // Use the job ARN and the token to get the job status.
            var job_arn = payload.arguments["vej_arn"];
            context.Logger.LogInformation($"Token: {token}, arn {job_arn}");

            var jobInfo = geoClient.GetVectorEnrichmentJobAsync(
                new GetVectorEnrichmentJobRequest()
                {
                    Arn = job_arn
                });
            context.Logger.LogInformation("Job info: " +
JsonSerializer.Serialize(jobInfo));
            if (jobInfo.Result.Status == VectorEnrichmentJobStatus.COMPLETED)
            {
                context.Logger.LogInformation($"Status completed, resuming
pipeline...");
                await sageMakerClient.SendPipelineExecutionStepSuccessAsync(
                    new SendPipelineExecutionStepSuccessRequest()
                    {
                        CallbackToken = token,
                        OutputParameters = new List<OutputParameter>()
                        {
                            new OutputParameter()
                                { Name = "export_status", Value =
jobInfo.Result.Status }
                        }
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.FAILED)
            {
```

```
                context.Logger.LogInformation($"Status failed, stopping
 pipeline...");
                await sageMakerClient.SendPipelineExecutionStepFailureAsync(
                    new SendPipelineExecutionStepFailureRequest()
                    {
                        CallbackToken = token,
                        FailureReason = jobInfo.Result.ErrorDetails.ErrorMessage
                    });
            }
            else if (jobInfo.Result.Status ==
 VectorEnrichmentJobStatus.IN_PROGRESS)
            {
                // Put this message back in the queue to reprocess later.
                context.Logger.LogInformation(
                    $"Status still in progress, check back later.");
                throw new("Job still running.");
            }
        }
    }
}
```

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public static class PipelineWorkflow
{
    public static IAmazonIdentityManagementService _iamClient = null!;
    public static SageMakerWrapper _sageMakerWrapper = null!;
    public static IAmazonSQS _sqsClient = null!;
    public static IAmazonS3 _s3Client = null!;
    public static IAmazonLambda _lambdaClient = null!;
    public static IConfiguration _configuration = null!;

    public static string lambdaFunctionName = "SageMakerExampleFunction";
    public static string sageMakerRoleName = "SageMakerExampleRole";
    public static string lambdaRoleName = "SageMakerExampleLambdaRole";

    private static string[] lambdaRolePolicies = null!;
    private static string[] sageMakerRolePolicies = null!;

    static async Task Main(string[] args)
    {
        var options = new AWSOptions() { Region = RegionEndpoint.USWest2 };
```

```
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>(options)
                    .AddAWSService<IAmazonEC2>(options)
                    .AddAWSService<IAmazonSageMaker>(options)
                    .AddAWSService<IAmazonSageMakerGeospatial>(options)
                    .AddAWSService<IAmazonSQS>(options)
                    .AddAWSService<IAmazonS3>(options)
                    .AddAWSService<IAmazonLambda>(options)
                    .AddTransient<SageMakerWrapper>()
        )
        .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);
        string queueUrl = "";
        string queueName = _configuration["queueName"];
        string bucketName = _configuration["bucketName"];
        var pipelineName = _configuration["pipelineName"];

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "Welcome to the Amazon SageMaker pipeline example scenario.");
            Console.WriteLine(
                "\nThis example scenario will guide you through setting up and
running an" +
                "\nAmazon SageMaker pipeline. The pipeline uses an AWS Lambda
function and an" +
```

```
                "\nAmazon SQS Queue. It runs a vector enrichment reverse geocode
 job to" +
                "\nreverse geocode addresses in an input file and store the
 results in an export file.");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "First, we will set up the roles, functions, and queue needed by
 the SageMaker pipeline.");
            Console.WriteLine(new string('-', 80));

            var lambdaRoleArn = await CreateLambdaRole();
            var sageMakerRoleArn = await CreateSageMakerRole();
            var functionArn = await SetupLambda(lambdaRoleArn, true);
            queueUrl = await SetupQueue(queueName);
            await SetupBucket(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Now we can create and run our pipeline.");
            Console.WriteLine(new string('-', 80));

            await SetupPipeline(sageMakerRoleArn, functionArn, pipelineName);
            var executionArn = await ExecutePipeline(queueUrl, sageMakerRoleArn,
 pipelineName, bucketName);
            await WaitForPipelineExecution(executionArn);

            await GetOutputResults(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("The pipeline has completed. To view the pipeline
 and runs " +
                              "in SageMaker Studio, follow these instructions:" +
                              "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/
pipelines-studio.html");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await CleanupResources(true, queueUrl, pipelineName, bucketName);

            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("SageMaker pipeline scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources(true, queueUrl, pipelineName, bucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sageMakerWrapper = host.Services.GetRequiredService<SageMakerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _sqsClient = host.Services.GetRequiredService<IAmazonSQS>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _lambdaClient = host.Services.GetRequiredService<IAmazonLambda>();
    }

    /// <summary>
    /// Set up AWS Lambda, either by updating an existing function or creating a
new function.
    /// </summary>
    /// <param name="roleArn">The role Amazon Resource Name (ARN) to use for the
Lambda function.</param>
    /// <param name="askUser">True to ask the user before updating.</param>
    /// <returns>The ARN of the function.</returns>
    public static async Task<string> SetupLambda(string roleArn, bool askUser)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Setting up the Lambda function for the pipeline.");
        var handlerName =
"SageMakerLambda::SageMakerLambda.SageMakerLambdaFunction::FunctionHandler";
        var functionArn = "";
        try
        {
```

```
            var functionInfo = await _lambdaClient.GetFunctionAsync(new
GetFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });

            var updateFunction = true;
            if (askUser)
            {
                updateFunction = GetYesNoResponse(
                    $"\tThe Lambda function {lambdaFunctionName} already exists,
do you want to update it?");
            }

            if (updateFunction)
            {
                // Update the Lambda function.
                using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
                await _lambdaClient.UpdateFunctionCodeAsync(
                    new UpdateFunctionCodeRequest()
                    {
                        FunctionName = lambdaFunctionName,
                        ZipFile = zipMemoryStream,
                    });
            }

            functionArn = functionInfo.Configuration.FunctionArn;
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"\tThe Lambda function {lambdaFunctionName} was
not found, creating the new function.");

            // Create the function if it does not already exist.
            using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
            var createResult = await _lambdaClient.CreateFunctionAsync(
                new CreateFunctionRequest()
                {
                    FunctionName = lambdaFunctionName,
                    Runtime = Runtime.Dotnet6,
                    Description = "SageMaker example function.",
                    Code = new FunctionCode()
```

```
                    {
                        ZipFile = zipMemoryStream
                    },
                    Handler = handlerName,
                    Role = roleArn,
                    Timeout = 30
                });

            functionArn = createResult.FunctionArn;
        }

        Console.WriteLine($"\tLambda ready with ARN {functionArn}.");
        Console.WriteLine(new string('-', 80));
        return functionArn;
    }

    /// <summary>
    /// Create a role to be used by AWS Lambda. Does not create the role if it
already exists.
    /// </summary>
    /// <returns>The role ARN.</returns>
    public static async Task<string> CreateLambdaRole()
    {
        Console.WriteLine(new string('-', 80));

        lambdaRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSQSFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
"AmazonSageMakerGeospatialFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
"AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy",
            "arn:aws:iam::aws:policy/service-role/" +
"AWSLambdaSQSQueueExecutionRole"
        };

        var roleArn = await GetRoleArnIfExists(lambdaRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to for AWS Lambda to use.");
```

```
        var assumeRolePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": [" +
                                             "\"sagemaker.amazonaws.com\"," +
                                             "\"sagemaker-geospatial.amazonaws.com
\"," +
                                             "\"lambda.amazonaws.com\"," +
                                             "\"s3.amazonaws.com\"" +
                                         "]" +
                                     "}," +
                                     "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = lambdaRoleName
            });
        foreach (var policy in lambdaRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = lambdaRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));

        return roleResult.Role.Arn;
    }


    /// <summary>
```

```
    /// Create a role to be used by SageMaker.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateSageMakerRole()
    {
        Console.WriteLine(new string('-', 80));

        sageMakerRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSageMakerGeospatialFullAccess",
        };

        var roleArn = await GetRoleArnIfExists(sageMakerRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to use with SageMaker.");

        var assumeRolePolicy = "{" +
                                        "\"Version\": \"2012-10-17\"," +
                                        "\"Statement\": [{" +
                                            "\"Effect\": \"Allow\"," +
                                            "\"Principal\": {" +
                                                $"\"Service\": [" +
                                                    "\"sagemaker.amazonaws.com
\"," +
                                                    "\"sagemaker-
geospatial.amazonaws.com\"," +
                                                    "\"lambda.amazonaws.com\"," +
                                                    "\"s3.amazonaws.com\"" +
                                                "]" +
                                            "}," +
                                            "\"Action\": \"sts:AssumeRole\"" +
                                        "}]" +
                                    "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = sageMakerRoleName
```

```
            });

        foreach (var policy in sageMakerRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = sageMakerRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Set up the SQS queue to use with the pipeline.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <returns>The URL for the queue.</returns>
    public static async Task<string> SetupQueue(string queueName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up queue {queueName}.");

        try
        {
            var queueInfo = await _sqsClient.GetQueueUrlAsync(new
GetQueueUrlRequest()
            { QueueName = queueName });
            return queueInfo.QueueUrl;
        }
        catch (QueueDoesNotExistException)
        {
            var attrs = new Dictionary<string, string>
            {
                {
                    QueueAttributeName.DelaySeconds,
                    "5"
                },
```

```
                {
                    QueueAttributeName.ReceiveMessageWaitTimeSeconds,
                    "5"
                },
                {
                    QueueAttributeName.VisibilityTimeout,
                    "300"
                },
            };

            var request = new CreateQueueRequest
            {
                Attributes = attrs,
                QueueName = queueName,
            };

            var response = await _sqsClient.CreateQueueAsync(request);
            Thread.Sleep(10000);
            await ConnectLambda(response.QueueUrl);
            Console.WriteLine($"\tQueue ready with Url {response.QueueUrl}.");
            Console.WriteLine(new string('-', 80));
            return response.QueueUrl;
        }
    }

    /// <summary>
    /// Connect the queue to the Lambda function as an event source.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue.</param>
    /// <returns>Async task.</returns>
    public static async Task ConnectLambda(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Connecting the Lambda function and queue for the
pipeline.");

        var queueAttributes = await _sqsClient.GetQueueAttributesAsync(
            new GetQueueAttributesRequest() { QueueUrl = queueUrl, AttributeNames
= new List<string>() { "All" } });
        var queueArn = queueAttributes.QueueARN;

        var eventSource = await _lambdaClient.ListEventSourceMappingsAsync(
             new ListEventSourceMappingsRequest()
             {
```

```
                 FunctionName = lambdaFunctionName
             });

        if (!eventSource.EventSourceMappings.Any())
        {
            // Only add the event source mapping if it does not already exist.
            await _lambdaClient.CreateEventSourceMappingAsync(
                new CreateEventSourceMappingRequest()
                {
                    EventSourceArn = queueArn,
                    FunctionName = lambdaFunctionName,
                    Enabled = true
                });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the bucket to use for pipeline input and output.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task SetupBucket(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up bucket {bucketName}.");

        var bucketExists = await
 Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            bucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = bucketName,
                BucketRegion = S3Region.USWest2
            });

            Thread.Sleep(5000);

            await _s3Client.PutObjectAsync(new PutObjectRequest()
            {
```

```
                BucketName = bucketName,
                Key = "samplefiles/latlongtest.csv",
                FilePath = "latlongtest.csv"
            });
        }

        Console.WriteLine($"\tBucket {bucketName} ready.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Display some results from the output directory.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<string> GetOutputResults(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting output results {bucketName}.");
        string outputKey = "";
        Thread.Sleep(15000);
        var outputFiles = await _s3Client.ListObjectsAsync(
              new ListObjectsRequest()
              {
                  BucketName = bucketName,
                  Prefix = "outputfiles/"
              });

        if (outputFiles.S3Objects.Any())
        {
            var sampleOutput = outputFiles.S3Objects.OrderBy(s =>
s.LastModified).Last();
            Console.WriteLine($"\tOutput file: {sampleOutput.Key}");
            var outputSampleResponse = await _s3Client.GetObjectAsync(
                new GetObjectRequest()
                {
                    BucketName = bucketName,
                    Key = sampleOutput.Key
                });
            outputKey = sampleOutput.Key;
            StreamReader reader = new
StreamReader(outputSampleResponse.ResponseStream);
            await reader.ReadLineAsync();
            Console.WriteLine("\tOutput file contents: \n");
```

```
            for (int i = 0; i < 10; i++)
            {
                if (!reader.EndOfStream)
                {
                    Console.WriteLine("\t" + await reader.ReadLineAsync());
                }
            }
        }

        Console.WriteLine(new string('-', 80));
        return outputKey;
    }

    /// <summary>
    /// Create a pipeline from the example pipeline JSON
    /// that includes the Lambda, callback, processing, and export jobs.
    /// </summary>
    /// <param name="roleArn">The ARN of the role for the pipeline.</param>
    /// <param name="functionArn">The ARN of the Lambda function for the
pipeline.</param>
    /// <param name="pipelineName">The name for the pipeline.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public static async Task<string> SetupPipeline(string roleArn, string
functionArn, string pipelineName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up the pipeline.");

        var pipelineJson = await
File.ReadAllTextAsync("GeoSpatialPipeline.json");

        // Add the correct function ARN instead of the placeholder.
        pipelineJson = pipelineJson.Replace("*FUNCTION_ARN*", functionArn);

        var pipelineArn = await _sageMakerWrapper.SetupPipeline(pipelineJson,
roleArn, pipelineName,
            "sdk example pipeline", pipelineName);

        Console.WriteLine($"\tPipeline set up with ARN {pipelineArn}.");
        Console.WriteLine(new string('-', 80));

        return pipelineArn;
    }
```

```
    /// <summary>
    /// Start a pipeline run with job configurations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue used in the pipeline.</
param>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>The pipeline run ARN.</returns>
    public static async Task<string> ExecutePipeline(
        string queueUrl,
        string roleArn,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Starting pipeline execution.");

        var input = $"s3://{bucketName}/samplefiles/latlongtest.csv";
        var output = $"s3://{bucketName}/outputfiles/";

        var executionARN =
            await _sageMakerWrapper.ExecutePipeline(queueUrl, input, output,
                pipelineName, roleArn);

        Console.WriteLine($"\tRun started with ARN {executionARN}.");
        Console.WriteLine(new string('-', 80));

        return executionARN;
    }

    /// <summary>
    /// Wait for a pipeline run to complete.
    /// </summary>
    /// <param name="executionArn">The pipeline run ARN.</param>
    /// <returns>Async task.</returns>
    public static async Task WaitForPipelineExecution(string executionArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Waiting for pipeline to finish.");

        PipelineExecutionStatus status;
        do
        {
```

```
            status = await
 _sageMakerWrapper.CheckPipelineExecutionStatus(executionArn);
            Thread.Sleep(30000);
            Console.WriteLine($"\tStatus is {status}.");
        } while (status == PipelineExecutionStatus.Executing);

        Console.WriteLine($"\tPipeline finished with status {status}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="askUser">True to ask the user for cleanup.</param>
    /// <param name="queueUrl">The URL of the queue to clean up.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> CleanupResources(
        bool askUser,
        string queueUrl,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (!askUser || GetYesNoResponse($"\tDelete pipeline {pipelineName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting pipeline.");
            // Delete the pipeline.
            await _sageMakerWrapper.DeletePipelineByName(pipelineName);
        }

        if (!string.IsNullOrEmpty(queueUrl) && (!askUser ||
 GetYesNoResponse($"\tDelete queue {queueUrl}? (y/n)")))
        {
            Console.WriteLine($"\tDeleting queue.");
            // Delete the queue.
            await _sqsClient.DeleteQueueAsync(new DeleteQueueRequest(queueUrl));
        }
```

```
        if (!askUser || GetYesNoResponse($"\tDelete Amazon S3 bucket
 {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
 ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            if (deleteList.KeyCount > 0)
            {
                await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.S3Objects
                        .Select(o => new KeyVersion { Key = o.Key }).ToList()
                });
            }

            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete lambda {lambdaFunctionName}?
 (y/n)"))
        {
            Console.WriteLine($"\tDeleting lambda function.");

            await _lambdaClient.DeleteFunctionAsync(new DeleteFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {lambdaRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in lambdaRolePolicies)
```

```
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = lambdaRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = lambdaRoleName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {sageMakerRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in sageMakerRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = sageMakerRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = sageMakerRoleName
            });
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a role's ARN if it already exists.
    /// </summary>
```

```
    /// <param name="roleName">The name of the AWS Identity and Access Management
 (IAM) Role to look for.</param>
    /// <returns>The role ARN if it exists, otherwise an empty string.</returns>
    private static async Task<string> GetRoleArnIfExists(string roleName)
    {
        Console.WriteLine($"Checking for role named {roleName}.");

        try
        {
            var existingRole = await _iamClient.GetRoleAsync(new GetRoleRequest()
            {
                RoleName = lambdaRoleName
            });
            return existingRole.Role.Arn;
        }
        catch (NoSuchEntityException)
        {
            return string.Empty;
        }
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreatePipeline

- DeletePipeline
- DescribePipelineExecution
- StartPipelineExecution
- UpdatePipeline

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n
\n" +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service
 (Amazon SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
```

```
                "    lnglatData - The file location of the latlongtest.csv file
required for this use case.\n\n" +
                "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for
example, sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName =
"org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
```

```
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store
the results in an export file.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
        String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
        String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

        String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
                handlerName);
        String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
        System.out.println("The queue URL is " + queueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setting up bucket " + bucketName);
        if (!checkBucket(s3Client, bucketName)) {
            setupBucket(s3Client, bucketName);
            System.out.println("Put " + lnglatData + " into " + bucketName);
            putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now we can create and run our pipeline.");
        setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
```

```
        String pipelineExecutionARN = executePipeline(sageMakerClient,
 bucketName, queueUrl, sageMakerRoleArn,
                pipelineName);
        System.out.println("The pipeline execution ARN value is " +
 pipelineExecutionARN);
        waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
        System.out.println("Getting output results " + bucketName);
        getOutputResults(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30
 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }
```

```
   private static void readObject(S3Client s3Client, String bucketName, String
key) {
       System.out.println("Output file contents: \n");
       GetObjectRequest objectRequest = GetObjectRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
       byte[] byteArray = objectBytes.asByteArray();
       String text = new String(byteArray, StandardCharsets.UTF_8);
       System.out.println("Text output: " + text);
   }

   // Display some results from the output directory.
   public static void getOutputResults(S3Client s3Client, String bucketName) {
       System.out.println("Getting output results {bucketName}.");
       ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
               .bucket(bucketName)
               .prefix("outputfiles/")
               .build();

       ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
       List<S3Object> s3Objects = response.contents();
       for (S3Object object : s3Objects) {
           readObject(s3Client, bucketName, object.key());
       }
   }

   // Check the status of a pipeline execution.
   public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
           throws InterruptedException {
       String status;
       int index = 0;
       do {
           DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                   .pipelineExecutionArn(executionArn)
                   .build();

           DescribePipelineExecutionResponse response = sageMakerClient
```

```
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " +
status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest =
CreatePipelineRequest.builder()
```

```
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
```

```
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();
```

```
        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig
\":{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" +
 gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-
execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
 DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
```

```
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
```

```
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName, String
 objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
    public static String setupQueue(SqsClient sqsClient, LambdaClient
lambdaClient, String queueName,
            String lambdaName) {
        System.out.println("Setting up queue named " + queueName);
        try {
            Map<QueueAttributeName, String> queueAtt = new HashMap<>();
            queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
            queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS,
"5");
            queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(queueAtt)
                    .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            TimeUnit.SECONDS.sleep(15);

            connectLambda(sqsClient, lambdaClient,
getQueueUrlResponse.queueUrl(), lambdaName);
            System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
```

```
    }

    // Connect the queue to the Lambda function as an event source.
    public static void connectLambda(SqsClient sqsClient, LambdaClient
lambdaClient, String queueUrl,
            String lambdaName) {
        System.out.println("Connecting the Lambda function and queue for the
pipeline.");
        String queueArn = "";

        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda
function was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
```

```
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
```

```
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }
```

```
            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this
workflow.");
        String functionArn;
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName,
filePath, role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists,
this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
```

```
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates
a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient
lambdaClient, String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName,
lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
```

```
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates
 it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
```

```
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Der folgende Dateiauszug enthält Funktionen, die den SageMaker AI-Client zur Verwaltung einer Pipeline verwenden.

```
import { readFileSync } from "node:fs";

import {
  CreateRoleCommand,
  DeleteRoleCommand,
  CreatePolicyCommand,
  DeletePolicyCommand,
  AttachRolePolicyCommand,
  DetachRolePolicyCommand,
  GetRoleCommand,
  ListPoliciesCommand,
} from "@aws-sdk/client-iam";

import {
  PublishLayerVersionCommand,
  DeleteLayerVersionCommand,
  CreateFunctionCommand,
  Runtime,
  DeleteFunctionCommand,
  CreateEventSourceMappingCommand,
  DeleteEventSourceMappingCommand,
  GetFunctionCommand,
} from "@aws-sdk/client-lambda";

import {
  PutObjectCommand,
  CreateBucketCommand,
  DeleteBucketCommand,
  DeleteObjectCommand,
  GetObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  CreatePipelineCommand,
  DeletePipelineCommand,
  DescribePipelineCommand,
  DescribePipelineExecutionCommand,
  PipelineExecutionStatus,
  StartPipelineExecutionCommand,
} from "@aws-sdk/client-sagemaker";
```

```
import { VectorEnrichmentJobDocumentType } from "@aws-sdk/client-sagemaker-
geospatial";

import {
  CreateQueueCommand,
  DeleteQueueCommand,
  GetQueueAttributesCommand,
  GetQueueUrlCommand,
} from "@aws-sdk/client-sqs";

import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * Create the AWS IAM role that will be assumed by AWS Lambda.
 * @param {{ name: string, iamClient: import('@aws-sdk/client-iam').IAMClient }}
 props
 */
export async function createLambdaExecutionRole({ name, iamClient }) {
  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: { Service: ["lambda.amazonaws.com"] },
            },
          ],
        }),
      }),
    );

  let role = null;

  try {
    const { Role } = await createRole();
    role = Role;
  } catch (caught) {
    if (
      caught instanceof Error &&
```

```
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create an AWS IAM policy that will be attached to the AWS IAM role assumed by
 the AWS Lambda function.
 * The policy grants permission to work with Amazon SQS, Amazon CloudWatch, and
 Amazon SageMaker.
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 pipelineExecutionRoleArn: string}} props
 */
export async function createLambdaExecutionPolicy({
  name,
  iamClient,
  pipelineExecutionRoleArn,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: [
          "sqs:ReceiveMessage",
          "sqs:DeleteMessage",
          "sqs:GetQueueAttributes",
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
          "logs:PutLogEvents",
```

```
          "sagemaker-geospatial:StartVectorEnrichmentJob",
          "sagemaker-geospatial:GetVectorEnrichmentJob",
          "sagemaker:SendPipelineExecutionStepFailure",
          "sagemaker:SendPipelineExecutionStepSuccess",
          "sagemaker-geospatial:ExportVectorEnrichmentJob",
        ],
        Resource: "*",
      },
      {
        Effect: "Allow",
        // The AWS Lambda function needs permission to pass the pipeline
 execution role to
        // the StartVectorEnrichmentCommand. This restriction prevents an AWS
 Lambda function
        // from elevating privileges. For more information, see:
        // https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_use_passrole.html
        Action: ["iam:PassRole"],
        Resource: `${pipelineExecutionRoleArn}`,
        Condition: {
          StringEquals: {
            "iam:PassedToService": [
              "sagemaker.amazonaws.com",
              "sagemaker-geospatial.amazonaws.com",
            ],
          },
        },
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
```

```
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}

/**
 * Attach an AWS IAM policy to an AWS IAM role.
 * @param {{roleName: string, policyArn: string, iamClient: import('@aws-sdk/
client-iam').IAMClient}} props
 */
export async function attachPolicy({ roleName, policyArn, iamClient }) {
  const attachPolicyCommand = new AttachRolePolicyCommand({
    RoleName: roleName,
    PolicyArn: policyArn,
  });

  await iamClient.send(attachPolicyCommand);
  return {
    cleanUp: async () => {
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: roleName,
          PolicyArn: policyArn,
        }),
```

```
      );
    },
  };
}

/**
 * Create an AWS Lambda layer that contains the Amazon SageMaker and Amazon
 SageMaker Geospatial clients
 * in the runtime. The default runtime supports v3.188.0 of the JavaScript SDK.
 The Amazon SageMaker
 * Geospatial client wasn't introduced until v3.221.0.
 * @param {{ name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient }} props
 */
export async function createLambdaLayer({ name, lambdaClient }) {
  const layerPath = `${dirnameFromMetaUrl(import.meta.url)}lambda/nodejs.zip`;
  const { LayerVersionArn, Version } = await lambdaClient.send(
    new PublishLayerVersionCommand({
      LayerName: name,
      Content: {
        ZipFile: Uint8Array.from(readFileSync(layerPath)),
      },
    }),
  );

  return {
    versionArn: LayerVersionArn,
    version: Version,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteLayerVersionCommand({
          LayerName: name,
          VersionNumber: Version,
        }),
      );
    },
  };
}

/**
 * Deploy the AWS Lambda function that will be used to respond to Amazon
 SageMaker pipeline
 * execution steps.
```

```
 * @param {{roleArn: string, name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient, layerVersionArn: string}} props
 */
export async function createLambdaFunction({
  name,
  roleArn,
  lambdaClient,
  layerVersionArn,
}) {
  const lambdaPath = `${dirnameFromMetaUrl(
    import.meta.url,
  )}lambda/dist/index.mjs.zip`;

  // If a function of the same name already exists, return that
  // function's ARN instead. By default this is
  // "sagemaker-wkflw-lambda-function", so collisions are
  // unlikely.
  const createFunction = async () => {
    try {
      return await lambdaClient.send(
        new CreateFunctionCommand({
          Code: {
            ZipFile: Uint8Array.from(readFileSync(lambdaPath)),
          },
          Runtime: Runtime.nodejs18x,
          Handler: "index.handler",
          Layers: [layerVersionArn],
          FunctionName: name,
          Role: roleArn,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "ResourceConflictException"
      ) {
        const { Configuration } = await lambdaClient.send(
          new GetFunctionCommand({ FunctionName: name }),
        );
        return Configuration;
      }
      throw caught;
    }
  };
```

```
  // Function creation fails if the Role is not ready. This retries
  // function creation until it succeeds or it times out.
  const { FunctionArn } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    createFunction,
  );

  return {
    arn: FunctionArn,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteFunctionCommand({ FunctionName: name }),
      );
    },
  };
}

/**
 * This uploads some sample coordinate data to an Amazon S3 bucket.
 * The Amazon SageMaker Geospatial vector enrichment job will take the simple
 Lat/Long
 * coordinates in this file and augment them with more detailed location data.
 * @param {{bucketName: string, s3Client: import('@aws-sdk/client-s3').S3Client}}
 props
 */
export async function uploadCSVDataToS3({ bucketName, s3Client }) {
  const s3Path = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../../scenarios/features/sagemaker_pipelines/resources/
latlongtest.csv`;

  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Key: "input/sample_data.csv",
      Body: readFileSync(s3Path),
    }),
  );
}

/**
 * Create the AWS IAM role that will be assumed by the Amazon SageMaker pipeline.
```

```
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 wait: (ms: number) => Promise<void>}} props
 */
export async function createSagemakerRole({ name, iamClient, wait }) {
  let role = null;

  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: {
                Service: [
                  "sagemaker.amazonaws.com",
                  "sagemaker-geospatial.amazonaws.com",
                ],
              },
            },
          ],
        }),
      }),
    );

  try {
    const { Role } = await createRole();
    role = Role;
    // Wait for the role to be ready.
    await wait(10);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
```

```
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create the Amazon SageMaker execution policy. This policy grants permission to
 * invoke the AWS Lambda function, read/write to the Amazon S3 bucket, and send
 messages to
 * the Amazon SQS queue.
 * @param {{ name: string, sqsQueueArn: string, lambdaArn: string, iamClient:
 import('@aws-sdk/client-iam').IAMClient, s3BucketName: string}} props
 */
export async function createSagemakerExecutionPolicy({
  sqsQueueArn,
  lambdaArn,
  iamClient,
  name,
  s3BucketName,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: ["lambda:InvokeFunction"],
        Resource: lambdaArn,
      },
      {
        Effect: "Allow",
        Action: ["s3:*"],
        Resource: [
          `arn:aws:s3:::${s3BucketName}`,
          `arn:aws:s3:::${s3BucketName}/*`,
        ],
      },
      {
        Effect: "Allow",
```

```
        Action: ["sqs:SendMessage"],
        Resource: sqsQueueArn,
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}
```

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using
 PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/
client-sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
```

```
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}

/**
 * Create an Amazon SQS queue. The Amazon SageMaker pipeline will send messages
 * to this queue that are then processed by the AWS Lambda function.
 * @param {{name: string, sqsClient: import('@aws-sdk/client-sqs').SQSClient}}
 props
 */
export async function createSQSQueue({ name, sqsClient }) {
  const createSqsQueue = () =>
    sqsClient.send(
      new CreateQueueCommand({
        QueueName: name,
        Attributes: {
          DelaySeconds: "5",
          ReceiveMessageWaitTimeSeconds: "5",
          VisibilityTimeout: "300",
        },
      }),
    );
```

```
  let queueUrl = null;
  try {
    const { QueueUrl } = await createSqsQueue();
    queueUrl = QueueUrl;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "QueueNameExists") {
      const { QueueUrl } = await sqsClient.send(
        new GetQueueUrlCommand({ QueueName: name }),
      );
      queueUrl = QueueUrl;
    } else {
      throw caught;
    }
  }

  const { Attributes } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    () =>
      sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: queueUrl,
          AttributeNames: ["QueueArn"],
        }),
      ),
  );

  return {
    queueUrl,
    queueArn: Attributes.QueueArn,
    cleanUp: async () => {
      await sqsClient.send(new DeleteQueueCommand({ QueueUrl: queueUrl }));
    },
  };
}

/**
 * Configure the AWS Lambda function to long poll for messages from the Amazon
 SQS
 * queue.
 * @param {{
 *   paginateListEventSourceMappings: () => Generator<import('@aws-sdk/client-
lambda').ListEventSourceMappingsCommandOutput>,
 *   lambdaName: string,
 *   queueArn: string,
```

```
 *   lambdaClient: import('@aws-sdk/client-lambda').LambdaClient}} props
 */
export async function configureLambdaSQSEventSource({
  lambdaName,
  queueArn,
  lambdaClient,
  paginateListEventSourceMappings,
}) {
  let uuid = null;
  const createEvenSourceMapping = () =>
    lambdaClient.send(
      new CreateEventSourceMappingCommand({
        EventSourceArn: queueArn,
        FunctionName: lambdaName,
      }),
    );

  try {
    const { UUID } = await createEvenSourceMapping();
    uuid = UUID;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceConflictException"
    ) {
      const paginator = paginateListEventSourceMappings(
        { client: lambdaClient },
        {},
      );
      /**
       * @type {import('@aws-sdk/client-
lambda').EventSourceMappingConfiguration[]}
       */
      const eventSourceMappings = [];
      for await (const page of paginator) {
        eventSourceMappings.concat(page.EventSourceMappings || []);
      }

      const { Configuration } = await lambdaClient.send(
        new GetFunctionCommand({ FunctionName: lambdaName }),
      );

      uuid = eventSourceMappings.find(
        (mapping) =>
```

```
          mapping.EventSourceArn === queueArn &&
          mapping.FunctionArn === Configuration.FunctionArn,
      ).UUID;
    } else {
      throw caught;
    }
  }

  return {
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteEventSourceMappingCommand({
          UUID: uuid,
        }),
      );
    },
  };
}

/**
 * Create an Amazon S3 bucket that will store the simple coordinate file as input
 * and the output of the Amazon SageMaker Geospatial vector enrichment job.
 * @param {{
 *   s3Client: import('@aws-sdk/client-s3').S3Client,
 *   name: string,
 *   paginateListObjectsV2: () => Generator<import('@aws-sdk/client-
s3').ListObjectsCommandOutput>
 * }} props
 */
export async function createS3Bucket({
  name,
  s3Client,
  paginateListObjectsV2,
}) {
  await s3Client.send(new CreateBucketCommand({ Bucket: name }));

  return {
    cleanUp: async () => {
      const paginator = paginateListObjectsV2(
        { client: s3Client },
        { Bucket: name },
      );
      for await (const page of paginator) {
        const objects = page.Contents;
```

```
        if (objects) {
          for (const object of objects) {
            await s3Client.send(
              new DeleteObjectCommand({ Bucket: name, Key: object.Key }),
            );
          }
        }
      }
      await s3Client.send(new DeleteBucketCommand({ Bucket: name }));
    },
  };
}

/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a
 CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
```

```
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse
 Geocoding requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
```

```
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}

/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before
 checking again.`,
      );
```

```
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}

/**
 * Return the string value of an Amazon S3 object.
 * @param {{ bucket: string, key: string, s3Client: import('@aws-sdk/client-
s3').S3Client}} param0
 */
export async function getObject({ bucket, s3Client }) {
  const prefix = "output/";
  const { Contents } = await s3Client.send(
    new ListObjectsV2Command({ MaxKeys: 1, Bucket: bucket, Prefix: prefix }),
  );

  if (!Contents.length) {
    throw new Error("No objects found in bucket.");
  }

  // Find the CSV file.
  const outputObject = Contents.find((obj) => obj.Key.endsWith(".csv"));

  if (!outputObject) {
    throw new Error(`No CSV file found in bucket with the prefix "${prefix}".`);
  }

  const { Body } = await s3Client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: outputObject.Key,
    }),
  );

  return Body.transformToString();
}
```

Diese Funktion ist ein Auszug aus einer Datei, die die vorherigen Bibliotheksfunktionen verwendet, um eine SageMaker AI-Pipeline einzurichten, auszuführen und alle erstellten Ressourcen zu löschen.

```
import { retry, wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  attachPolicy,
  configureLambdaSQSEventSource,
  createLambdaExecutionPolicy,
  createLambdaExecutionRole,
  createLambdaFunction,
  createLambdaLayer,
  createS3Bucket,
  createSQSQueue,
  createSagemakerExecutionPolicy,
  createSagemakerPipeline,
  createSagemakerRole,
  getObject,
  startPipelineExecution,
  uploadCSVDataToS3,
  waitForPipelineComplete,
} from "./lib.js";
import { MESSAGES } from "./messages.js";

export class SageMakerPipelinesWkflw {
  names = {
    LAMBDA_EXECUTION_ROLE: "sagemaker-wkflw-lambda-execution-role",
    LAMBDA_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-lambda-execution-role-policy",
    LAMBDA_FUNCTION: "sagemaker-wkflw-lambda-function",
    LAMBDA_LAYER: "sagemaker-wkflw-lambda-layer",
    SAGE_MAKER_EXECUTION_ROLE: "sagemaker-wkflw-pipeline-execution-role",
    SAGE_MAKER_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-pipeline-execution-role-policy",
    SAGE_MAKER_PIPELINE: "sagemaker-wkflw-pipeline",
    SQS_QUEUE: "sagemaker-wkflw-sqs-queue",
    S3_BUCKET: `sagemaker-wkflw-s3-bucket-${Date.now()}`,
  };

  cleanUpFunctions = [];

  /**
   * @param {import("@aws-doc-sdk-examples/lib/prompter.js").Prompter} prompter
```

```
   * @param {import("@aws-doc-sdk-examples/lib/logger.js").Logger} logger
   * @param {{ IAM: import("@aws-sdk/client-iam").IAMClient, Lambda:
 import("@aws-sdk/client-lambda").LambdaClient, SageMaker: import("@aws-sdk/
client-sagemaker").SageMakerClient, S3: import("@aws-sdk/client-s3").S3Client,
 SQS: import("@aws-sdk/client-sqs").SQSClient }} clients
   */
  constructor(prompter, logger, clients) {
    this.prompter = prompter;
    this.logger = logger;
    this.clients = clients;
  }

  async run() {
    try {
      await this.startWorkflow();
    } catch (err) {
      console.error(err);
      throw err;
    } finally {
      this.logger.logSeparator();
      const doCleanUp = await this.prompter.confirm({
        message: "Clean up resources?",
      });
      if (doCleanUp) {
        await this.cleanUp();
      }
    }
  }

  async cleanUp() {
    // Run all of the clean up functions. If any fail, we log the error and
 continue.
    // This ensures all clean up functions are run.
    for (let i = this.cleanUpFunctions.length - 1; i >= 0; i--) {
      await retry(
        { intervalInMs: 1000, maxRetries: 60, swallowError: true },
        this.cleanUpFunctions[i],
      );
    }
  }

  async startWorkflow() {
    this.logger.logSeparator(MESSAGES.greetingHeader);
    await this.logger.log(MESSAGES.greeting);
```

```
   this.logger.logSeparator();
   await this.logger.log(
     MESSAGES.creatingRole.replace(
       "${ROLE_NAME}",
       this.names.LAMBDA_EXECUTION_ROLE,
     ),
   );

   // Create an IAM role that will be assumed by the AWS Lambda function. This
function
   // is triggered by Amazon SQS messages and calls SageMaker and SageMaker
GeoSpatial actions.
   const { arn: lambdaExecutionRoleArn, cleanUp: lambdaExecutionRoleCleanUp } =
     await createLambdaExecutionRole({
       name: this.names.LAMBDA_EXECUTION_ROLE,
       iamClient: this.clients.IAM,
     });
   // Add a clean up step to a stack for every resource created.
   this.cleanUpFunctions.push(lambdaExecutionRoleCleanUp);

   await this.logger.log(
     MESSAGES.roleCreated.replace(
       "${ROLE_NAME}",
       this.names.LAMBDA_EXECUTION_ROLE,
     ),
   );

   this.logger.logSeparator();

   await this.logger.log(
     MESSAGES.creatingRole.replace(
       "${ROLE_NAME}",
       this.names.SAGE_MAKER_EXECUTION_ROLE,
     ),
   );

   // Create an IAM role that will be assumed by the SageMaker pipeline. The
pipeline
   // sends messages to an Amazon SQS queue and puts/retrieves Amazon S3
objects.
   const {
     arn: pipelineExecutionRoleArn,
     cleanUp: pipelineExecutionRoleCleanUp,
```

```
    } = await createSagemakerRole({
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE,
      wait,
    });
    this.cleanUpFunctions.push(pipelineExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    // Create an IAM policy that allows the AWS Lambda function to invoke
  SageMaker APIs.
    const {
      arn: lambdaExecutionPolicyArn,
      policy: lambdaPolicy,
      cleanUp: lambdaExecutionPolicyCleanUp,
    } = await createLambdaExecutionPolicy({
      name: this.names.LAMBDA_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
      iamClient: this.clients.IAM,
      pipelineExecutionRoleArn,
    });
    this.cleanUpFunctions.push(lambdaExecutionPolicyCleanUp);

    console.log(JSON.stringify(lambdaPolicy, null, 2), "\n");

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.LAMBDA_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.LAMBDA_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the Lambda execution policy to the execution role.
    const { cleanUp: lambdaExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.LAMBDA_EXECUTION_ROLE,
      policyArn: lambdaExecutionPolicyArn,
```

```
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(lambdaExecutionRolePolicyCleanUp);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    // Create Lambda layer for SageMaker packages.
    const { versionArn: layerVersionArn, cleanUp: lambdaLayerCleanUp } =
      await createLambdaLayer({
        name: this.names.LAMBDA_LAYER,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaLayerCleanUp);

    await this.logger.log(
      MESSAGES.creatingFunction.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    // Create the Lambda function with the execution role.
    const { arn: lambdaArn, cleanUp: lambdaCleanUp } =
      await createLambdaFunction({
        roleArn: lambdaExecutionRoleArn,
        lambdaClient: this.clients.Lambda,
        name: this.names.LAMBDA_FUNCTION,
        layerVersionArn,
      });
    this.cleanUpFunctions.push(lambdaCleanUp);

    await this.logger.log(
      MESSAGES.functionCreated.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingSQSQueue.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
```

```
    );

    // Create an SQS queue for the SageMaker pipeline.
    const {
      queueUrl,
      queueArn,
      cleanUp: queueCleanUp,
    } = await createSQSQueue({
      name: this.names.SQS_QUEUE,
      sqsClient: this.clients.SQS,
    });
    this.cleanUpFunctions.push(queueCleanUp);

    await this.logger.log(
      MESSAGES.sqsQueueCreated.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.configuringLambdaSQSEventSource
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Configure the SQS queue as an event source for the Lambda.
    const { cleanUp: lambdaSQSEventSourceCleanUp } =
      await configureLambdaSQSEventSource({
        lambdaArn,
        lambdaName: this.names.LAMBDA_FUNCTION,
        queueArn,
        sqsClient: this.clients.SQS,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaSQSEventSourceCleanUp);

    await this.logger.log(
      MESSAGES.lambdaSQSEventSourceConfigured
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();
```

```
    // Create an IAM policy that allows the SageMaker pipeline to invoke AWS
 Lambda
    // and send messages to the Amazon SQS queue.
    const {
      arn: pipelineExecutionPolicyArn,
      policy: sagemakerPolicy,
      cleanUp: pipelineExecutionPolicyCleanUp,
    } = await createSagemakerExecutionPolicy({
      sqsQueueArn: queueArn,
      lambdaArn,
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
    });
    this.cleanUpFunctions.push(pipelineExecutionPolicyCleanUp);

    console.log(JSON.stringify(sagemakerPolicy, null, 2));

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the SageMaker execution policy to the execution role.
    const { cleanUp: pipelineExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.SAGE_MAKER_EXECUTION_ROLE,
      policyArn: pipelineExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(pipelineExecutionRolePolicyCleanUp);
    // Wait for the role to be ready. If the role is used immediately,
    // the pipeline will fail.
    await wait(5);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingPipeline.replace(
        "${PIPELINE_NAME}",
```

```
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    // Create the SageMaker pipeline.
    const { cleanUp: pipelineCleanUp } = await createSagemakerPipeline({
      roleArn: pipelineExecutionRoleArn,
      functionArn: lambdaArn,
      sagemakerClient: this.clients.SageMaker,
      name: this.names.SAGE_MAKER_PIPELINE,
    });
    this.cleanUpFunctions.push(pipelineCleanUp);

    await this.logger.log(
      MESSAGES.pipelineCreated.replace(
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingS3Bucket.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    // Create an S3 bucket for storing inputs and outputs.
    const { cleanUp: s3BucketCleanUp } = await createS3Bucket({
      name: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });
    this.cleanUpFunctions.push(s3BucketCleanUp);

    await this.logger.log(
      MESSAGES.s3BucketCreated.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.uploadingInputData.replace(
        "${BUCKET_NAME}",
        this.names.S3_BUCKET,
      ),
```

```
    );

    // Upload CSV Lat/Long data to S3.
    await uploadCSVDataToS3({
      bucketName: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });

    await this.logger.log(MESSAGES.inputDataUploaded);

    this.logger.logSeparator();

    await this.prompter.checkContinue(MESSAGES.executePipeline);

    // Execute the SageMaker pipeline.
    const { arn: pipelineExecutionArn } = await startPipelineExecution({
      name: this.names.SAGE_MAKER_PIPELINE,
      sagemakerClient: this.clients.SageMaker,
      roleArn: pipelineExecutionRoleArn,
      bucketName: this.names.S3_BUCKET,
      queueUrl,
    });

    // Wait for the pipeline execution to finish.
    await waitForPipelineComplete({
      arn: pipelineExecutionArn,
      sagemakerClient: this.clients.SageMaker,
      wait,
    });

    this.logger.logSeparator();

    await this.logger.log(MESSAGES.outputDelay);

    // The getOutput function will throw an error if the output is not
    // found. The retry function will retry a failed function call once
    // ever 10 seconds for 2 minutes.
    const output = await retry({ intervalInMs: 10000, maxRetries: 12 }, () =>
      getObject({
        bucket: this.names.S3_BUCKET,
        s3Client: this.clients.S3,
      }),
    );
```

```
    this.logger.logSeparator();
    await this.logger.log(MESSAGES.outputDataRetrieved);
    console.log(output.split("\n").slice(0, 6).join("\n"));
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
```

```
        functionName - The name of the AWS Lambda function (for
example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the
Lambda function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS)
queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the
Lambda ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for
this use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json
file required for this use case.
        pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and
an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
```

```
        reverse geocode addresses in an input file and store the results in an
export file.
       """.trimIndent(),
   )
   println(DASHES)

   println(DASHES)
   println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
   val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
   val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
   val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
   val queueUrl = checkQueue(queueName, functionName)
   println(DASHES)

   println(DASHES)
   println("Setting up bucket $bucketName")
   if (!checkBucket(bucketName)) {
       setupBucket(bucketName)
       println("Put $lnglatData into $bucketName")
       val objectKey = "samplefiles/latlongtest.csv"
       putS3Object(bucketName, objectKey, lnglatData)
   }
   println(DASHES)

   println(DASHES)
   println("Now we can create and run our pipeline.")
   setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn,
pipelineName)
   val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
   println("The pipeline execution ARN value is $pipelineExecutionARN")
   waitForPipelineExecution(pipelineExecutionARN)
   println("Wait 30 secs to get output results $bucketName")
   TimeUnit.SECONDS.sleep(30)
   getOutputResults(bucketName)
   println(DASHES)

   println(DASHES)
   println(
       """
            The pipeline has completed. To view the pipeline and runs in
SageMaker Studio, follow these instructions:
```

```
            https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/
n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

```
suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
```

```
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
    }
```

```
    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
    }
```

```
    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
    }
}
```

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn:
 String?, pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }
```

```
    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName
\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
```

```
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }
```

```
        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath:
 String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
    }
```

```
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
    queueAtt.put("DelaySeconds", "5")
```

```
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a
 new queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}
```

```
suspend fun createNewFunction(myFunctionName: String, s3BucketName: String,
 myS3Key: String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
    } catch (e: LambdaException) {
```

```
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be
 created for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
```

```
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be
 created for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
```

```
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
```

```
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Beginnen Sie mit SageMaker KI-Modellen und Endpunkten mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen Schulungsjob und erstellen Sie ein SageMaker KI-Modell.
- Eine Endpunktkonfiguration erstellen.
- Erstellen Sie einen Endpunkt und bereinigen Sie dann die Ressourcen.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
    DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
    DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescrtrnjobrsp.
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lt_hyperparameters TYPE /aws1/
cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lv_model_data_url TYPE /aws1/sgmurl.

    lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

    "Create ABAP internal table for hyperparameters based on input variables."
```

```
   "These hyperparameters are based on Amazon SageMaker built-in algorithm -
XGBoost"
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
   INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP internal table for data based on input variables."
   "Training data."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).
   INSERT lo_trn_channel INTO TABLE lt_input_data_config.
```

```
   "Validation data."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).
   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification based on input variables."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   TRY.
       lo_sgm->createtrainingjob(
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
```

```
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
    ENDTRY.

    "Wait for training job to be completed."
    lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
    WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
    ENDWHILE.

    "Create ABAP object for the container image based on input variables."
    lo_primarycontainer = NEW #( iv_image = iv_training_image
                                 iv_modeldataurl = lv_model_data_url ).

    "Create an Amazon SageMaker model."
    TRY.
        lo_sgm->createmodel(
          iv_executionrolearn = iv_role_arn
          iv_modelname = iv_model_name
          io_primarycontainer = lo_primarycontainer ).
        MESSAGE 'Model created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
    ENDTRY.

    "Create an endpoint production variant."
    lo_production_variants = NEW #( iv_variantname = iv_ep_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
iv_ep_initial_instance_count
                                    iv_instancetype = iv_ep_instance_type ).
    INSERT lo_production_variants INTO TABLE lt_production_variants.
```

```
   TRY.
       "Create an endpoint configuration."
       lo_ep_config_result = lo_sgm->createendpointconfig(
         iv_endpointconfigname = iv_ep_cfg_name
         it_productionvariants = lt_production_variants ).
       MESSAGE 'Endpoint configuration created.' TYPE 'I'.

       "Create an endpoint."
       oo_ep_output = lo_sgm->createendpoint(        " oo_ep_output is returned
for testing purposes. "
           iv_endpointconfigname = iv_ep_cfg_name
           iv_endpointname = iv_ep_name ).
       MESSAGE 'Endpoint created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE
'E'.
   ENDTRY.

   "Wait for endpoint creation to be completed."
   DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
   WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
     IF sy-index = 30.
       EXIT.               "Maximum 900 seconds."
     ENDIF.
     WAIT UP TO 30 SECONDS.
     lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
   ENDWHILE.

   TRY.
       "Delete an endpoint."
       lo_sgm->deleteendpoint(
           iv_endpointname = iv_ep_name ).
       MESSAGE 'Endpoint deleted' TYPE 'I'.

       "Delete an endpoint configuration."
       lo_sgm->deleteendpointconfig(
         iv_endpointconfigname = iv_ep_cfg_name ).
       MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.

       "Delete model."
       lo_sgm->deletemodel(
                 iv_modelname = iv_model_name ).
```

```
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CreateEndpoint
  - CreateEndpointConfig
  - CreateModel
  - CreateTrainingJob
  - DeleteEndpoint
  - DeleteEndpointConfig
  - DeleteModel
  - DescribeEndpoint
  - DescribeTrainingJob

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}
```

```
# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
```

```
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)
```

```
    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}
```

```
# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
```

```
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
```

```
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
```

```
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
```

```
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
```

```
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)
```

```
if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
```

```
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
```

```
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup

- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Codebeispiele für Secrets Manager mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Secrets Manager mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Secrets-Manager-Benutzerhandbuch – Weitere Informationen zu Secrets Manager.
- API-Referenz für Secrets Manager – Details zu allen verfügbaren Secrets-Manager-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Secrets Manager AWS SDKs
  - Aktionen für Secrets Manager mit AWS SDKs
    - Verwenden Sie BatchGetSecretValuemit einem AWS SDK oder CLI
    - Verwenden Sie GetSecretValuemit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Secrets Manager AWS SDKs
  - Leihbibliothek-REST-API erstellen

- Eine relationale Datenbankinstanz erstellen
- Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
- Erste Schritte mit Dokumentendatenbanken
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher

Grundlegende Beispiele für die Verwendung von Secrets Manager AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von AWS Secrets Manager mit AWS SDKs verwenden können.

Beispiele

- Aktionen für Secrets Manager mit AWS SDKs
  - Verwenden Sie BatchGetSecretValuemit einem AWS SDK oder CLI
  - Verwenden Sie GetSecretValuemit einem AWS SDK oder CLI

Aktionen für Secrets Manager mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Secrets Manager Manager-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Secrets-Manager-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Secrets Manager AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Secrets Manager -API-Referenz.

Beispiele

- Verwenden Sie BatchGetSecretValuemit einem AWS SDK oder CLI
- Verwenden Sie GetSecretValuemit einem AWS SDK oder CLI

Verwenden Sie **`BatchGetSecretValue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `BatchGetSecretValue` verwendet wird.

CLI

AWS CLI

Beispiel 1: So rufen Sie den Secrets-Wert für eine Gruppe von Secrets ab, die nach Namen aufgelistet sind

Im folgenden Beispiel für `batch-get-secret-value` wird der aktuelle Secret-Wert für drei Secrets abgerufen.

```
aws secretsmanager batch-get-secret-value \
    --secret-id-list MySecret1 MySecret2 MySecret3
```

Ausgabe:

```
{
    "SecretValues": [
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret1-a1b2c3",
            "Name": "MySecret1",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "SecretString": "{\"username\":\"diego_ramirez\",
\"password\":\"EXAMPLE-PASSWORD\",\"engine\":\"mysql\",\"host\":
\"secretsmanagertutorial.cluster.us-west-2.rds.amazonaws.com\",\"port\":3306,
\"dbClusterIdentifier\":\"secretsmanagertutorial\"}",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1523477145.729"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret2-a1b2c3",
            "Name": "MySecret2",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1673477781.275"
        },
```

```
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret3-a1b2c3",
            "Name": "MySecret3",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1373477721.124"
        }
    ],
    "Errors": []
}
```

Weitere Informationen finden Sie unter Abrufen einer Gruppe von Secret in einem Batch im Benutzerhandbuch für AWS Secrets Manager.

Beispiel 2: So rufen Sie den Secrets-Wert für eine Gruppe von Secrets ab, die durch Filter ausgewählt sind

Das folgende Beispiel für `batch-get-secret-value` erhält eine Liste der Secrets-Werte in Ihrem Konto, deren Name `MySecret` enthält. Beim Filtern nach Namen muss die Groß- und Kleinschreibung beachtet werden.

```
aws secretsmanager batch-get-secret-value \
    --filters Key="name",Values="MySecret"
```

Ausgabe:

```
{
    "SecretValues": [
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret1-a1b2c3",
            "Name": "MySecret1",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "SecretString": "{\"username\":\"diego_ramirez\",
\"password\":\"EXAMPLE-PASSWORD\",\"engine\":\"mysql\",\"host\":
\"secretsmanagertutorial.cluster.us-west-2.rds.amazonaws.com\",\"port\":3306,
\"dbClusterIdentifier\":\"secretsmanagertutorial\"}",
```

```
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1523477145.729"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret2-a1b2c3",
            "Name": "MySecret2",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "SecretString": "{\"username\":\"akua_mansa\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1673477781.275"
        },
        {
            "ARN": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MySecret3-a1b2c3",
            "Name": "MySecret3",
            "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "SecretString": "{\"username\":\"jie_liu\",\"password\":\"EXAMPLE-
PASSWORD\"",
            "VersionStages": [
                "AWSCURRENT"
            ],
            "CreatedDate": "1373477721.124"
        }
    ],
    "Errors": []
}
```

Weitere Informationen finden Sie unter Abrufen einer Gruppe von Secret in einem Batch im Benutzerhandbuch für AWS Secrets Manager.

- Einzelheiten zur API finden Sie BatchGetSecretValuein der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BatchGetSecretsWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def batch_get_secrets(self, filter_name):
        """
        Retrieve multiple secrets from AWS Secrets Manager using the
 batch_get_secret_value API.
        This function assumes the stack mentioned in the source code README has
 been successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param filter_name: The full or partial name of secrets to be fetched.
        :type filter_name: str
        """
        try:
            secrets = []
            response = self.client.batch_get_secret_value(
                Filters=[{"Key": "name", "Values": [f"{filter_name}"]}]
            )
            for secret in response["SecretValues"]:
                secrets.append(json.loads(secret["SecretString"]))
            if secrets:
                logger.info("Secrets retrieved successfully.")
            else:
                logger.info("Zero secrets returned without error.")
            return secrets
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"One or more requested secrets were not found with filter:
 {filter_name}"
```

```
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred:\n{str(e)}.")
            raise
```

- Einzelheiten zur API finden Sie BatchGetSecretValuein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_filter_name = 'mySecret'
        DATA(lo_result) = lo_smr->batchgetsecretvalue(
          it_filters = VALUE /aws1/cl_smrfilter=>tt_filterslisttype(
            (
              NEW /aws1/cl_smrfilter(
                iv_key = 'name'
                it_values = VALUE /aws1/
cl_smrfiltvalsstrlist_w=>tt_filtervaluesstringlist(
                  ( NEW /aws1/cl_smrfiltvalsstrlist_w( iv_value =
 iv_filter_name ) )
                )
              )
            )
          )
        ).
        ot_secret_values = lo_result->get_secretvalues( ).
        MESSAGE 'Secrets retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
```

```
        MESSAGE 'One or more requested secrets were not found.' TYPE 'E'.
      CATCH /aws1/cx_smrdecryptionfailure.
        MESSAGE 'Failed to decrypt one or more secrets.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie BatchGetSecretValuein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetSecretValue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSecretValue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Eine relationale Datenbankinstanz erstellen
- Erste Schritte mit Serverless-Data Warehouses
- Erste Schritte mit Dokumentendatenbanken
- Übertragung von hartcodierten Geheimnissen in einen sicheren geheimen Speicher

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.SecretsManager;
```

```
using Amazon.SecretsManager.Model;

/// <summary>
/// This example uses the Amazon Web Service Secrets Manager to retrieve
/// the secret value for the provided secret name.
/// </summary>
public class GetSecretValue
{
    /// <summary>
    /// The main method initializes the necessary values and then calls
    /// the GetSecretAsync and DecodeString methods to get the decoded
    /// secret value for the secret named in secretName.
    /// </summary>
    public static async Task Main()
    {
        string secretName = "<<{{MySecretName}}>>";
        string secret;

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
            secret = DecodeString(response);

            if (!string.IsNullOrEmpty(secret))
            {
                Console.WriteLine($"The decoded secret value is: {secret}.");
            }
            else
            {
                Console.WriteLine("No secret value was returned.");
            }
        }
    }

    /// <summary>
    /// Retrieves the secret value given the name of the secret to
    /// retrieve.
    /// </summary>
    /// <param name="client">The client object used to retrieve the secret
    /// value for the given secret name.</param>
```

```
        /// <param name="secretName">The name of the secret value to retrieve.</
param>
        /// <returns>The GetSecretValueReponse object returned by
        /// GetSecretValueAsync.</returns>
        public static async Task<GetSecretValueResponse> GetSecretAsync(
            IAmazonSecretsManager client,
            string secretName)
        {
            GetSecretValueRequest request = new GetSecretValueRequest()
            {
                SecretId = secretName,
                VersionStage = "AWSCURRENT", // VersionStage defaults to
 AWSCURRENT if unspecified.
            };

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }

        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
```

```
            {
                var secret = response.SecretString;
                return secret;
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
 System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetSecretValuein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                            const
 Aws::Client::ClientConfiguration &clientConfiguration) {
```

```
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() <<
 std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So rufen Sie den verschlüsselten Secrets-Wert eines Secrets ab

Im folgenden Beispiel für `get-secret-value` wird der aktuelle Secret-Wert abgerufen.

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
```

Ausgabe:

```
{
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
    "Name": "MyTestSecret",
```

```
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "SecretString": "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1523477145.713
}
```

Weitere Informationen finden Sie unter Abrufen eines Secrets im Benutzerhandbuch für Secrets Manager.

Beispiel 2: So rufen Sie den vorherigen Secret-Wert ab

Im folgenden Beispiel für `get-secret-value` wird der vorherige Secret-Wert abgerufen:

```
aws secretsmanager get-secret-value \
    --secret-id MyTestSecret
    --version-stage AWSPREVIOUS
```

Ausgabe:

```
{
    "ARN": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyTestSecret-
a1b2c3",
    "Name": "MyTestSecret",
    "VersionId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "SecretString": "{\"user\":\"diegor\",\"password\":\"PREVIOUS-EXAMPLE-
PASSWORD\"}",
    "VersionStages": [
        "AWSPREVIOUS"
    ],
    "CreatedDate": 1523477145.713
}
```

Weitere Informationen finden Sie unter Abrufen eines Secrets im Benutzerhandbuch für Secrets Manager.

- Einzelheiten zur API finden Sie GetSecretValuein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import
 software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import
 software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s
```

```
                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
 secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
```

```
  //   VersionStages: [ 'AWSCURRENT' ]
  // }

  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

- Einzelheiten zur API finden Sie GetSecretValuein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GetSecretWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def get_secret(self, secret_name):
        """
        Retrieve individual secrets from AWS Secrets Manager using the
 get_secret_value API.
        This function assumes the stack mentioned in the source code README has
 been successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
```

```
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

- Einzelheiten zur API finden Sie GetSecretValuein AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_secret_name = 'MySecretName'
        DATA(lo_result) = lo_smr->getsecretvalue( iv_secretid = iv_secret_name ).
        ov_secret_value = lo_result->get_secretstring( ).
        MESSAGE 'Secret value retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
        MESSAGE 'The requested secret was not found.' TYPE 'E'.
      CATCH /aws1/cx_smrdecryptionfailure.
        MESSAGE 'Failed to decrypt the secret.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'E'.
      CATCH /aws1/cx_smrinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetSecretValuein der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Secrets Manager AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Secrets Manager mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Secrets Manager oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Leihbibliothek-REST-API erstellen
- Eine relationale Datenbankinstanz erstellen
- Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI
- Erste Schritte mit Dokumentendatenbanken
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

Python

SDK für Python (Boto3)

Zeigt, wie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) API und AWS Chalice verwendet wird, um eine REST-API zu erstellen, die von einer Amazon Aurora Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldedaten verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Eine relationale Datenbankinstanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Netzwerkkomponenten einrichten

- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/
latest/UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier
 "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ];
 then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ];
 then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
```

```
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -
f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found
 ${#SUBNET_IDS[@]} subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -
f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"
```

```
echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP
 ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"
```

```
SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id
 "$SECRET_NAME" --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -
d'"' -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -
d'"' -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?
```

```
if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret

- DeleteSecurityGroup
- DescribeDbInstances
- DescribeSubnets
- DescribeVpcs
- GetSecretValue
- Wait

Erste Schritte mit serverlosen Data Warehouses mithilfe der CLI

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace
- Benutze Redshift CreateWorkgroup
- Benutze Redshift DeleteNamespace
- Benutze iam CreateRole

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
```

```
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
```

```
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
```

```
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
```

```
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "==========================================="
  echo "CLEANUP CONFIRMATION"
```

```
  echo "==========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"
```

```
    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
```

```
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
```

```
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
```

```
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
```

```
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)
```

```
# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
```

```
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
```

```
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
```

```
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
```

```
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
```

```
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole

- DeleteRolePolicy
- DeleteSecret
- DeleteWorkgroup
- GetNamespace
- GetRole
- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Erste Schritte mit Dokumentendatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.
```

```
set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

################################################################################
# Logging
################################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

################################################################################
# Region pre-check
################################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""
```

```
##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
```

```
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
```

```
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound
\|not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
```

```
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG
 ingress rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed
 to delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete
 cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}
```

```
##############################################################################
# Step 1: Generate password and store in Secrets Manager
##############################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "==========================================="
echo "Step 2: Find default VPC and subnets"
```

```
echo "==========================================="
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi
```

```
# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

##############################################################################
# Step 3: Create subnet group
##############################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "==========================================="
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
echo ""
```

```
# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
```

```
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

##############################################################################
# Step 6: Get cluster endpoint and security group
##############################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

##############################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
```

```
##############################################################################
echo "============================================="
echo "Step 7: Add security group ingress rule"
echo "============================================="
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null
 || true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
```

```
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "============================================="
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================="
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

################################################################################
# Step 9: Display connection information
################################################################################
echo "============================================="
```

```
echo "CONNECTION INFORMATION"
echo "==========================================="
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH}
 \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString
 --output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisiere das Geheimnis
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
```

```
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
```

```
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs
```

```
# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
```

```
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)
```

```
check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
```

```
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy

- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Codebeispiele für Amazon SES mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Simple Email Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon SES – Weitere Informationen zu Amazon SES.
- API-Referenz für Amazon SES – Details zu allen verfügbaren Amazon-SES-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon SES AWS SDKs
  - Aktionen für Amazon SES mit AWS SDKs
    - Verwenden Sie CreateReceiptFiltermit einem AWS SDK
    - Verwenden Sie CreateReceiptRulemit einem AWS SDK
    - Verwenden Sie CreateReceiptRuleSetmit einem AWS SDK
    - Verwenden Sie CreateTemplatemit einem AWS SDK
    - Verwenden Sie DeleteIdentitymit einem AWS SDK oder CLI
    - Verwenden Sie DeleteReceiptFiltermit einem AWS SDK

- Verwenden Sie DeleteReceiptRulemit einem AWS SDK
- Verwenden Sie DeleteReceiptRuleSetmit einem AWS SDK
- Verwenden Sie DeleteTemplatemit einem AWS SDK
- Verwenden Sie DescribeReceiptRuleSetmit einem AWS SDK
- Verwenden Sie GetIdentityVerificationAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetSendQuotamit einem AWS SDK oder CLI
- Verwendung von GetSendStatistics mit einer CLI
- Verwenden Sie mit einem GetTemplate AWS SDK
- Verwenden Sie ListIdentitiesmit einem AWS SDK oder CLI
- Verwenden Sie ListReceiptFiltersmit einem AWS SDK
- Verwenden Sie ListTemplatesmit einem AWS SDK
- Verwenden Sie SendBulkTemplatedEmailmit einem AWS SDK
- Verwenden Sie SendEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendRawEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendTemplatedEmailmit einem AWS SDK
- Verwenden Sie UpdateTemplatemit einem AWS SDK
- Verwenden Sie VerifyDomainIdentitymit einem AWS SDK oder CLI
- Verwenden Sie VerifyEmailIdentitymit einem AWS SDK oder CLI

- Szenarien für die Verwendung von Amazon SES AWS SDKs
  - Erstellen einer Amazon-Transcribe-Streaming-App
  - Kopieren Sie Amazon SES SES-E-Mail- und Domainidentitäten von einer AWS Mit einem von einer Region in eine andere AWS SDK
  - Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten
  - Erstellen eines Amazon-Redshift-Element-Trackers
  - Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben
  - Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
  - Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
  - Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

- Generieren von Anmeldeinformationen für die Verbindung mit einem Amazon-SES-SMTP-Endpunkt
- E-Mail-Versanddienste einrichten
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Überprüfen Sie eine E-Mail-Identität und senden Sie Nachrichten mit Amazon SES mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon SES AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Email Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon SES mit AWS SDKs
  - Verwenden Sie CreateReceiptFiltermit einem AWS SDK
  - Verwenden Sie CreateReceiptRulemit einem AWS SDK
  - Verwenden Sie CreateReceiptRuleSetmit einem AWS SDK
  - Verwenden Sie CreateTemplatemit einem AWS SDK
  - Verwenden Sie DeleteIdentitymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteReceiptFiltermit einem AWS SDK
  - Verwenden Sie DeleteReceiptRulemit einem AWS SDK
  - Verwenden Sie DeleteReceiptRuleSetmit einem AWS SDK
  - Verwenden Sie DeleteTemplatemit einem AWS SDK
  - Verwenden Sie DescribeReceiptRuleSetmit einem AWS SDK
  - Verwenden Sie GetIdentityVerificationAttributesmit einem AWS SDK oder CLI
  - Verwenden Sie GetSendQuotamit einem AWS SDK oder CLI
  - Verwendung von GetSendStatistics mit einer CLI
  - Verwenden Sie mit einem GetTemplate AWS SDK
  - Verwenden Sie ListIdentitiesmit einem AWS SDK oder CLI
  - Verwenden Sie ListReceiptFiltersmit einem AWS SDK
  - Verwenden Sie ListTemplatesmit einem AWS SDK

  - Verwenden Sie SendBulkTemplatedEmailmit einem AWS SDK

- Verwenden Sie SendEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendRawEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendTemplatedEmailmit einem AWS SDK
- Verwenden Sie UpdateTemplatemit einem AWS SDK
- Verwenden Sie VerifyDomainIdentitymit einem AWS SDK oder CLI
- Verwenden Sie VerifyEmailIdentitymit einem AWS SDK oder CLI

Aktionen für Amazon SES mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon SES SES-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon–SES-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon SES AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Simple Email Service.

Beispiele

- Verwenden Sie CreateReceiptFiltermit einem AWS SDK
- Verwenden Sie CreateReceiptRulemit einem AWS SDK
- Verwenden Sie CreateReceiptRuleSetmit einem AWS SDK
- Verwenden Sie CreateTemplatemit einem AWS SDK
- Verwenden Sie DeleteIdentitymit einem AWS SDK oder CLI
- Verwenden Sie DeleteReceiptFiltermit einem AWS SDK
- Verwenden Sie DeleteReceiptRulemit einem AWS SDK
- Verwenden Sie DeleteReceiptRuleSetmit einem AWS SDK
- Verwenden Sie DeleteTemplatemit einem AWS SDK
- Verwenden Sie DescribeReceiptRuleSetmit einem AWS SDK
- Verwenden Sie GetIdentityVerificationAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetSendQuotamit einem AWS SDK oder CLI
- Verwendung von GetSendStatistics mit einer CLI

- Verwenden Sie mit einem GetTemplate AWS SDK
- Verwenden Sie ListIdentitiesmit einem AWS SDK oder CLI
- Verwenden Sie ListReceiptFiltersmit einem AWS SDK
- Verwenden Sie ListTemplatesmit einem AWS SDK
- Verwenden Sie SendBulkTemplatedEmailmit einem AWS SDK
- Verwenden Sie SendEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendRawEmailmit einem AWS SDK oder CLI
- Verwenden Sie SendTemplatedEmailmit einem AWS SDK
- Verwenden Sie UpdateTemplatemit einem AWS SDK
- Verwenden Sie VerifyDomainIdentitymit einem AWS SDK oder CLI
- Verwenden Sie VerifyEmailIdentitymit einem AWS SDK oder CLI

Verwenden Sie **`CreateReceiptFilter`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateReceiptFilter` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
 (CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
```

```
                                        Aws::SES::Model::ReceiptFilterPolicy
 policy,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
 sesClient.CreateReceiptFilter(
            createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
                  createReceiptFilterOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateReceiptFilterin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
```

```
  CreateReceiptFilterCommand,
  ReceiptFilterPolicy,
} from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const createCreateReceiptFilterCommand = ({ policy, ipOrRange, name }) => {
  return new CreateReceiptFilterCommand({
    Filter: {
      IpFilter: {
        Cidr: ipOrRange, // string, either a single IP address (10.0.0.1) or an
 IP address range in CIDR notation (10.0.0.1/24)).
        Policy: policy, // enum ReceiptFilterPolicy, email traffic from the
 filtered addressesOptions.
      },
      /*
        The name of the IP address filter. Only ASCII letters, numbers,
 underscores, or dashes.
        Must be less than 64 characters and start and end with a letter or
 number.
       */
      Name: name,
    },
  });
};

const FILTER_NAME = getUniqueName("ReceiptFilter");

const run = async () => {
  const createReceiptFilterCommand = createCreateReceiptFilterCommand({
    policy: ReceiptFilterPolicy.Allow,
    ipOrRange: "10.0.0.1",
    name: FILTER_NAME,
  });

  try {
    return await sesClient.send(createReceiptFilterCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
```

```
  }
};
```

- Einzelheiten zur API finden Sie CreateReceiptFilterin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
        """
        Creates a filter that allows or blocks incoming mail from an IP address
 or
        range.

        :param filter_name: The name to give the filter.
        :param ip_address_or_range: The IP address or range to block or allow.
        :param allow: When True, incoming mail is allowed from the specified IP
                      address or range; otherwise, it is blocked.
        """
```

```
        try:
            policy = "Allow" if allow else "Block"
            self.ses_client.create_receipt_filter(
                Filter={
                    "Name": filter_name,
                    "IpFilter": {"Cidr": ip_address_or_range, "Policy": policy},
                }
            )
            logger.info(
                "Created receipt filter %s to %s IP of %s.",
                filter_name,
                policy,
                ip_address_or_range,
            )
        except ClientError:
            logger.exception("Couldn't create receipt filter %s.", filter_name)
            raise
```

- Einzelheiten zur API finden Sie CreateReceiptFilterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_allow = abap_true means 'Allow', abap_false means 'Block'
DATA(lv_policy) = COND /aws1/sesreceiptfilterpolicy(
  WHEN iv_allow = abap_true THEN 'Allow'
  ELSE 'Block'
).

DATA(lo_ip_filter) = NEW /aws1/cl_sesreceiptipfilter(
  iv_policy = lv_policy
```

```
      iv_cidr = iv_ip_address_or_range
    ).

    DATA(lo_filter) = NEW /aws1/cl_sesreceiptfilter(
      iv_name = iv_filter_name
      io_ipfilter = lo_ip_filter
    ).

    TRY.
        lo_ses->createreceiptfilter( io_filter = lo_filter ).
        MESSAGE 'Receipt filter created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Filter already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateReceiptFilterin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateReceiptRule`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateReceiptRule` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
```

```
  \param receiptRuleName: The name for the receipt rule.
  \param s3BucketName: The name of the S3 bucket for incoming mail.
  \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
  \param ruleSetName: The name of the rule set where the receipt rule is added.
  \param recipients: Aws::Vector of recipients.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);

    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
    else {
```

```
        std::cerr << "Error creating receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateReceiptRulein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateReceiptRuleCommand, TlsPolicy } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");
const RULE_NAME = getUniqueName("RuleName");
const S3_BUCKET_NAME = getUniqueName("S3BucketName");

const createS3ReceiptRuleCommand = ({
  bucketName,
  emailAddresses,
  name,
  ruleSet,
}) => {
  return new CreateReceiptRuleCommand({
    Rule: {
      Actions: [
        {
          S3Action: {
            BucketName: bucketName,
```

```
            ObjectKeyPrefix: "email",
          },
        },
      ],
      Recipients: emailAddresses,
      Enabled: true,
      Name: name,
      ScanEnabled: false,
      TlsPolicy: TlsPolicy.Optional,
    },
    RuleSetName: ruleSet, // Required
  });
};

const run = async () => {
  const s3ReceiptRuleCommand = createS3ReceiptRuleCommand({
    bucketName: S3_BUCKET_NAME,
    emailAddresses: ["email@example.com"],
    name: RULE_NAME,
    ruleSet: RULE_SET_NAME,
  });

  try {
    return await sesClient.send(s3ReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to create S3 receipt rule.", err);
    throw err;
  }
};
```

- Einzelheiten zur API finden Sie CreateReceiptRulein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Amazon-S3-Bucket, in dem Amazon SES Kopien eingehender E-Mails ablegen und eine Regel erstellen kann, die eingehende E-Mails für eine bestimmte Empfängerliste in den Bucket kopiert.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_bucket_for_copy(self, bucket_name):
        """
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.

        :param bucket_name: The name of the bucket to create.
        :return: The newly created bucket.
        """
        allow_ses_put_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowSESPut",
                    "Effect": "Allow",
                    "Principal": {"Service": "ses.amazonaws.com"},
```

```
                    "Action": "s3:PutObject",
                    "Resource": f"arn:aws:s3:::{bucket_name}/*",
                }
            ],
        }
        bucket = None
        try:
            bucket = self.s3_resource.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    "LocationConstraint":
self.s3_resource.meta.client.meta.region_name
                },
            )
            bucket.wait_until_exists()
            bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
            logger.info("Created bucket %s to receive copies of emails.",
bucket_name)
        except ClientError:
            logger.exception("Couldn't create bucket to receive copies of
emails.")
            if bucket is not None:
                bucket.delete()
            raise
        else:
            return bucket


    def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix
    ):
        """
        Creates a rule so that all emails received by the specified recipients
are
        copied to an Amazon S3 bucket.

        :param rule_set_name: The name of a previously created rule set to
contain
                              this rule.
        :param rule_name: The name to give the rule.
        :param recipients: When an email is received by one of these recipients,
it
                           is copied to the Amazon S3 bucket.
        :param bucket_name: The name of the bucket to receive email copies. This
```

```
                                bucket must allow Amazon SES to put objects into it.
        :param prefix: An object key prefix to give the emails copied to the
bucket.
        """
        try:
            self.ses_client.create_receipt_rule(
                RuleSetName=rule_set_name,
                Rule={
                    "Name": rule_name,
                    "Enabled": True,
                    "Recipients": recipients,
                    "Actions": [
                        {
                            "S3Action": {
                                "BucketName": bucket_name,
                                "ObjectKeyPrefix": prefix,
                            }
                        }
                    ],
                },
            )
            logger.info(
                "Created rule %s to copy mail received by %s to bucket %s.",
                rule_name,
                recipients,
                bucket_name,
            )
        except ClientError:
            logger.exception("Couldn't create rule %s.", rule_name)
            raise
```

- Einzelheiten zur API finden Sie CreateReceiptRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Create S3 action for copying emails to S3
DATA(lo_s3_action) = NEW /aws1/cl_sess3action(
  iv_bucketname = iv_bucket_name
  iv_objectkeyprefix = iv_prefix
).

" Create receipt action with S3 action
DATA(lo_action) = NEW /aws1/cl_sesreceiptaction(
  io_s3action = lo_s3_action
).

" Create list of actions
DATA lt_actions TYPE /aws1/cl_sesreceiptaction=>tt_receiptactionslist.
APPEND lo_action TO lt_actions.

" Create receipt rule
DATA(lo_rule) = NEW /aws1/cl_sesreceiptrule(
  iv_name = iv_rule_name
  iv_enabled = abap_true
  it_recipients = it_recipients
  it_actions = lt_actions
).

TRY.
    lo_ses->createreceiptrule(
      iv_rulesetname = iv_rule_set_name
      io_rule = lo_rule
    ).
    MESSAGE 'Receipt rule created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesinvalids3confex INTO DATA(lo_ex1).
    DATA(lv_error) = |Invalid S3 configuration: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
```

```
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateReceiptRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateReceiptRuleSet`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateReceiptRuleSet` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);
```

```
    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
 sesClient.CreateReceiptRuleSet(
            createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateReceiptRuleSetin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createCreateReceiptRuleSetCommand = (ruleSetName) => {
  return new CreateReceiptRuleSetCommand({ RuleSetName: ruleSetName });
};

const run = async () => {
```

```
  const createReceiptRuleSetCommand =
    createCreateReceiptRuleSetCommand(RULE_SET_NAME);

  try {
    return await sesClient.send(createReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to create receipt rule set", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie CreateReceiptRuleSetin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_rule_set(self, rule_set_name):
        """
        Creates an empty rule set. Rule sets contain individual rules and can be
        used to organize rules.
```

```
        :param rule_set_name: The name to give the rule set.
        """
        try:
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Created receipt rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't create receipt rule set %s.",
rule_set_name)
            raise
```

- Einzelheiten zur API finden Sie CreateReceiptRuleSetin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ses->createreceiptruleset( iv_rulesetname = iv_rule_set_name ).
        MESSAGE 'Receipt rule set created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Rule set already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateReceiptRuleSetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateTemplate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateTemplate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an email template.
    /// </summary>
    /// <param name="name">Name of the template.</param>
    /// <param name="subject">Email subject.</param>
    /// <param name="text">Email body text.</param>
    /// <param name="html">Email HTML body text.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
        string html)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.CreateTemplateAsync(
                new CreateTemplateRequest
```

```
                {
                    Template = new Template
                    {
                        TemplateName = name,
                        SubjectPart = subject,
                        TextPart = text,
                        HtmlPart = html
                    }
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("CreateEmailTemplateAsync failed with exception: "
 + ex.Message);
        }

        return success;
    }
```

- Einzelheiten zur API finden Sie CreateTemplatein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;

    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
            createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
    else {
        std::cerr << "Error creating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTemplatein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateTemplateCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const TEMPLATE_NAME = getUniqueName("TestTemplateName");

const createCreateTemplateCommand = () => {
  return new CreateTemplateCommand({
    /**
     * The template feature in Amazon SES is based on the Handlebars template
 system.
     */
    Template: {
      /**
       * The name of an existing template in Amazon SES.
       */
      TemplateName: TEMPLATE_NAME,
      HtmlPart: `
        <h1>Hello, {{contact.firstName}}!</h1>
        <p>
        Did you know Amazon has a mascot named Peccy?
        </p>
      `,
      SubjectPart: "Amazon Tip",
    },
  });
};

const run = async () => {
  const createTemplateCommand = createCreateTemplateCommand();
```

```
  try {
    return await sesClient.send(createTemplateCommand);
  } catch (err) {
    console.log("Failed to create template.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie CreateTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
```

```
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise
```

- Einzelheiten zur API finden Sie CreateTemplatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_template) = NEW /aws1/cl_sestemplate(
  iv_templatename = iv_name
  iv_subjectpart = iv_subject
  iv_textpart = iv_text
  iv_htmlpart = iv_html
).

TRY.
    lo_ses->createtemplate( io_template = lo_template ).
    MESSAGE 'Template created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
    DATA(lv_error) = |Template already exists: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
    lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteIdentity`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteIdentity` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an email identity.
    /// </summary>
    /// <param name="identityEmail">The identity email to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteIdentityAsync(string identityEmail)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
                new DeleteIdentityRequest
                {
                    Identity = identityEmail
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteIdentityin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete the specified identity (an email address or a domain).
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
            deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteIdentityin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine Identität

Im folgenden Beispiel wird mit dem `delete-identity`-Befehl eine Identität aus der Liste der mit Amazon SES verifizierten Identitäten gelöscht:

```
aws ses delete-identity --identity user@example.com
```

Weitere Informationen zu verifizierten Identitäten finden Sie unter „Verifizieren von E-Mail-Adressen und Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie DeleteIdentityin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const IDENTITY_EMAIL = "fake@example.com";
```

```
const createDeleteIdentityCommand = (identityName) => {
  return new DeleteIdentityCommand({
    Identity: identityName,
  });
};

const run = async () => {
  const deleteIdentityCommand = createDeleteIdentityCommand(IDENTITY_EMAIL);

  try {
    return await sesClient.send(deleteIdentityCommand);
  } catch (err) {
    console.log("Failed to delete identity.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie DeleteIdentityin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def delete_identity(self, identity):
```

```
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise
```

- Einzelheiten zur API finden Sie DeleteIdentityin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ses->deleteidentity( iv_identity = iv_identity ).
        MESSAGE 'Identity deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteReceiptFilter`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteReceiptFilter` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
 sesClient.DeleteReceiptFilter(
            deleteReceiptFilterRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error deleting receipt filter. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Einzelheiten zur API finden Sie DeleteReceiptFilterin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptFilterCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RECEIPT_FILTER_NAME = getUniqueName("ReceiptFilterName");

const createDeleteReceiptFilterCommand = (filterName) => {
  return new DeleteReceiptFilterCommand({ FilterName: filterName });
};

const run = async () => {
  const deleteReceiptFilterCommand =
    createDeleteReceiptFilterCommand(RECEIPT_FILTER_NAME);

  try {
    return await sesClient.send(deleteReceiptFilterCommand);
  } catch (err) {
    console.log("Error deleting receipt filter.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie DeleteReceiptFilterin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

- Einzelheiten zur API finden Sie DeleteReceiptFilterin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ses->deletereceiptfilter( iv_filtername = iv_filter_name ).
        MESSAGE 'Receipt filter deleted successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteReceiptFilterin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteReceiptRule`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteReceiptRule` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
```

```
  \param receiptRuleName: The name for the receipt rule.
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &receiptRuleSetName,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome =
 sesClient.DeleteReceiptRule(
            deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {
        std::cout << "Error deleting receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteReceiptRulein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptRuleCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_NAME = getUniqueName("RuleName");
const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleCommand = () => {
  return new DeleteReceiptRuleCommand({
    RuleName: RULE_NAME,
    RuleSetName: RULE_SET_NAME,
  });
};

const run = async () => {
  const deleteReceiptRuleCommand = createDeleteReceiptRuleCommand();
  try {
    return await sesClient.send(deleteReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie DeleteReceiptRulein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.

        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name
            )
            logger.info("Removed rule %s from rule set %s.", rule_name,
 rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
 rule_set_name
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteReceiptRulein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletereceiptrule(
      iv_rulesetname = iv_rule_set_name
      iv_rulename = iv_rule_name
    ).
    MESSAGE 'Receipt rule deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
    DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteReceiptRulein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteReceiptRuleSet`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteReceiptRuleSet` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
 sesClient.DeleteReceiptRuleSet(
            deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteReceiptRuleSetin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleSetCommand = () => {
  return new DeleteReceiptRuleSetCommand({ RuleSetName: RULE_SET_NAME });
};

const run = async () => {
  const deleteReceiptRuleSetCommand = createDeleteReceiptRuleSetCommand();

  try {
    return await sesClient.send(deleteReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule set.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie DeleteReceiptRuleSetin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule_set(self, rule_set_name):
        """
        Deletes a rule set. When a rule set is deleted, all of the rules it
 contains
        are also deleted.

        :param rule_set_name: The name of the rule set to delete.
        """
        try:
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Deleted rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't delete rule set %s.", rule_set_name)
            raise
```

- Einzelheiten zur API finden Sie DeleteReceiptRuleSetin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletereceiptruleset( iv_rulesetname = iv_rule_set_name ).
    MESSAGE 'Receipt rule set deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_sescannotdeleteex INTO DATA(lo_ex1).
    DATA(lv_error) = |Cannot delete rule set: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteReceiptRuleSetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteTemplate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteTemplate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an email template.
    /// </summary>
    /// <param name="templateName">Name of the template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
                new DeleteTemplateRequest
                {
                    TemplateName = templateName
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteEmailTemplateAsync failed with exception: "
+ ex.Message);
        }

        return success;
    }
```

- Einzelheiten zur API finden Sie DeleteTemplatein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
            deleteTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTemplatein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createDeleteTemplateCommand = (templateName) =>
  new DeleteTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const deleteTemplateCommand = createDeleteTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(deleteTemplateCommand);
  } catch (err) {
    console.log("Failed to delete template.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie DeleteTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

 self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
```

```
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteTemplatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletetemplate( iv_templatename = iv_template_name ).
    MESSAGE 'Template deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeReceiptRuleSet`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeReceiptRuleSet` verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def describe_receipt_rule_set(self, rule_set_name):
        """
        Gets data about a rule set.

        :param rule_set_name: The name of the rule set to retrieve.
        :return: Data about the rule set.
        """
        try:
            response = self.ses_client.describe_receipt_rule_set(
                RuleSetName=rule_set_name
            )
            logger.info("Got data for rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DescribeReceiptRuleSetin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ses->describereceiptruleset(
      iv_rulesetname = iv_rule_set_name
    ).
    MESSAGE 'Receipt rule set described successfully' TYPE 'I'.
  CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
    DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeReceiptRuleSetin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetIdentityVerificationAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetIdentityVerificationAttributes` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get identity verification status for an email.
    /// </summary>
    /// <returns>The verification status of the email.</returns>
    public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
    {
        var result = VerificationStatus.TemporaryFailure;
        try
        {
            var response =
                await
_amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
                    new GetIdentityVerificationAttributesRequest
                    {
                        Identities = new List<string> { email }
                    });

            if (response.VerificationAttributes.ContainsKey(email))
                result =
response.VerificationAttributes[email].VerificationStatus;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
        }
```

```
        return result;
    }
```

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den Bestätigungsstatus von Amazon SES für eine Liste der Identitäten ab

Im folgenden Beispiel wird der `get-identity-verification-attributes`-Befehl verwendet, um den Amazon-SES-Bestätigungsstatus für eine Liste der Identitäten abzurufen:

```
aws ses get-identity-verification-attributes --
identities "user1@example.com" "user2@example.com"
```

Ausgabe:

```
{
   "VerificationAttributes": {
       "user1@example.com": {
           "VerificationStatus": "Success"
       },
       "user2@example.com": {
           "VerificationStatus": "Pending"
       }
   }
}
```

Wenn Sie diesen Befehl mit einer Identität aufrufen, die Sie noch nie zur Überprüfung eingereicht haben, wird diese Identität nicht in der Ausgabe angezeigt.

Weitere Informationen zu verifizierten Identitäten finden Sie unter „Verifizieren von E-Mail-Adressen und Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lt_identities TYPE /aws1/cl_sesidentitylist_w=>tt_identitylist.
   APPEND NEW /aws1/cl_sesidentitylist_w( iv_value = iv_identity ) TO
lt_identities.

   TRY.
       DATA(lo_result) = lo_ses->getidentityverificationattrs(
         it_identities = lt_identities
       ).

       DATA(lt_attrs) = lo_result->get_verificationattributes( ).
       IF lt_attrs IS NOT INITIAL.
         LOOP AT lt_attrs ASSIGNING FIELD-SYMBOL(<ls_attr>).
           ov_status = <ls_attr>-value->get_verificationstatus( ).
           EXIT.
         ENDLOOP.
       ELSE.
         ov_status = 'NotFound'.
       ENDIF.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
       DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetIdentityVerificationAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetSendQuota`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSendQuota` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- E-Mail-Versanddienste einrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information on the current account's send quota.
    /// </summary>
    /// <returns>The send quota response data.</returns>
    public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
    {
        var result = new GetSendQuotaResponse();
        try
        {
            var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
                new GetSendQuotaRequest());
            result = response;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetSendQuotaAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- Einzelheiten zur API finden Sie GetSendQuotain der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So verwalten Sie Ihre Amazon-SES-Sendelimits

Im folgenden Beispiel wird der `get-send-quota`-Befehl verwendet, um Ihre Amazon-SES-Sendelimits zurückzugeben:

```
aws ses get-send-quota
```

Ausgabe:

```
{
   "Max24HourSend": 200.0,
   "SentLast24Hours": 1.0,
   "MaxSendRate": 1.0
}
```

Max24HourSend ist Ihr Versandkontingent, d. h. die maximale Anzahl an E-Mails, die Sie innerhalb von 24 Stunden versenden können. Die Sendequote bezieht sich auf einen gleitenden Zeitraum. Wenn Sie versuchen eine, E-Mail zu senden, überprüft Amazon SES, wie viele E-Mails Sie in den letzten 24 Stunden gesendet haben. Solange die Gesamtzahl der von Ihnen gesendeten E-Mails unter Ihrer Quote liegt, wird Ihre Sendeanforderung akzeptiert und Ihre E-Mail versendet.

SentLast24Hours ist die Anzahl der E-Mails, die Sie in den letzten 24 Stunden gesendet haben.

MaxSendRate ist die maximale Anzahl von E-Mails, die Sie pro Sekunde senden können.

Beachten Sie, dass Sendelimits auf der Anzahl der Empfänger, nicht der Anzahl der Nachrichten basieren. Beispielsweise zählt eine E-Mail mit 10 Empfängern bei Ihrem Sendekontingent als 10.

Weitere Informationen finden Sie unter „Verwalten Ihrer Amazon-SES-Sendelimits“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie GetSendQuotain der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die aktuellen Sende-Limits des Benutzers zurück.

```
Get-SESSendQuota
```

- Einzelheiten zur API finden Sie unter GetSendQuota AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die aktuellen Sende-Limits des Benutzers zurück.

```
Get-SESSendQuota
```

- Einzelheiten zur API finden Sie unter GetSendQuota AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetSendStatistics`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetSendStatistics` verwendet wird.

CLI

AWS CLI

So rufen Sie Ihre Amazon-SES-Sendestatistiken ab

Im folgenden Beispiel wird der `get-send-statistics`-Befehl verwendet, um Ihre Amazon-SES-Sendestatistiken zurückzugeben

```
aws ses get-send-statistics
```

Ausgabe:

```
{
   "SendDataPoints": [
       {
```

```
            "Complaints": 0,
            "Timestamp": "2013-06-12T19:32:00Z",
            "DeliveryAttempts": 2,
            "Bounces": 0,
            "Rejects": 0
        },
        {
            "Complaints": 0,
            "Timestamp": "2013-06-12T00:47:00Z",
            "DeliveryAttempts": 1,
            "Bounces": 0,
            "Rejects": 0
        }
    ]
}
```

Das Ergebnis ist eine Liste mit Datenpunkten, die die letzten zwei Wochen Sendeaktivität repräsentieren. Jeder Datenpunkt in der Liste enthält Statistiken für ein Intervall von 15 Minuten.

In diesem Beispiel gibt es nur zwei Datenpunkte, da die einzigen E-Mails, die der Benutzer in den letzten zwei Wochen gesendet hat, in zwei 15-Minuten-Intervalle fielen.

Weitere Informationen finden Sie unter „Verwalten Ihrer Amazon SES-Nutzungsstatistiken“ im Entwicklerhandbuch für Amazon Simple Email Service.

- Einzelheiten zur API finden Sie GetSendStatisticsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Sendestatistik des Benutzers zurück. Das Ergebnis ist eine Liste mit Datenpunkten, die die letzten zwei Wochen Sendeaktivität repräsentieren. Jeder Datenpunkt in der Liste enthält Statistiken für ein Intervall von 15 Minuten.

```
Get-SESSendStatistic
```

- Einzelheiten zur API finden Sie unter GetSendStatistics AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Sendestatistik des Benutzers zurück. Das Ergebnis ist eine Liste mit Datenpunkten, die die letzten zwei Wochen Sendeaktivität repräsentieren. Jeder Datenpunkt in der Liste enthält Statistiken für ein Intervall von 15 Minuten.

```
Get-SESSendStatistic
```

- Einzelheiten zur API finden Sie unter GetSendStatistics AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetTemplate`** AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetTemplate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get a template's attributes.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
            getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetTemplatein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { GetTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
```

```
const createGetTemplateCommand = (templateName) =>
  new GetTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const getTemplateCommand = createGetTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(getTemplateCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie GetTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()
```

```
    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

- Einzelheiten zur API finden Sie GetTemplatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lo_result) = lo_ses->gettemplate( iv_templatename =
iv_template_name ).
       oo_template = lo_result->get_template( ).
       MESSAGE 'Template retrieved successfully' TYPE 'I'.
     CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
       DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex1.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
       lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex_generic.
   ENDTRY.
```

- Einzelheiten zur API finden Sie GetTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListIdentities`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListIdentities` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kopieren von E-Mail- und Domain-Identitäten von Region zu Region
- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Get the identities of a specified type for the current account.
   /// </summary>
   /// <param name="identityType">IdentityType to list.</param>
   /// <returns>The list of identities.</returns>
   public async Task<List<string>> ListIdentitiesAsync(IdentityType
identityType)
   {
       var result = new List<string>();
       try
       {
           var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
               new ListIdentitiesRequest
               {
                   IdentityType = identityType
               });
           result = response.Identities;
       }
       catch (Exception ex)
       {
           Console.WriteLine("ListIdentitiesAsync failed with exception: " +
ex.Message);
       }

       return result;
   }
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the identities associated with this account.
/*!
  \param identityType: The identity type enum. "NOT_SET" is a valid option.
  \param identities; A vector to receive the retrieved identities.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                 Aws::Vector<Aws::String> &identities,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome =
 sesClient.ListIdentities(
                listIdentitiesRequest);

        if (outcome.IsSuccess()) {
            const auto &retrievedIdentities =
 outcome.GetResult().GetIdentities();
            if (!retrievedIdentities.empty()) {
```

```
                identities.insert(identities.cend(),
 retrievedIdentities.cbegin(),
                                  retrievedIdentities.cend());
            }
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error listing identities. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    return true;
}
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um alle Identitäten (E-Mail-Adressen und Domains) für ein bestimmtes AWS Konto aufzulisten

Im folgenden Beispiel wird der `list-identities`-Befehl verwendet, um alle Identitäten aufzulisten, die zur Überprüfung bei Amazon SES eingereicht wurden:

```
aws ses list-identities
```

Ausgabe:

```
{
    "Identities": [
      "user@example.com",
      "example.com"
    ]
}
```

Die zurückgegebene Liste enthält alle Identitäten unabhängig vom Überprüfungsstatus (verifiziert, Überprüfung ausstehend, fehlgeschlagen usw.).

In diesem Beispiel werden E-Mail-Adressen und Domains zurückgegeben, weil wir den Parameter identity-type nicht angegeben haben.

Weitere Informationen zu verifizierten Identitäten finden Sie unter „Verifizieren von E-Mail-Adressen und Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
```

```
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });
```

```
const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt eine Liste zurück, die alle Identitäten (E-Mail-Adressen und Domains) für ein bestimmtes AWS Konto enthält, unabhängig vom Bestätigungsstatus.

```
Get-SESIdentity
```

- Einzelheiten zur API finden Sie unter ListIdentities AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt eine Liste zurück, die alle Identitäten (E-Mail-Adressen und Domains) für ein bestimmtes AWS Konto enthält, unabhängig vom Bestätigungsstatus.

```
Get-SESIdentity
```

- Einzelheiten zur API finden Sie unter ListIdentities AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

- Einzelheiten zur API finden Sie ListIdentitiesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_ses->listidentities(
      iv_identitytype = iv_identity_type
      iv_maxitems = iv_max_items
    ).
    ot_identities = lo_result->get_identities( ).
    MESSAGE 'Identities retrieved successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie ListIdentitiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListReceiptFilters`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListReceiptFilters` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the receipt filters associated with this account.
/*!
  \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
 &filters,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
 sesClient.ListReceiptFilters(
            listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie ListReceiptFiltersin der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListReceiptFiltersCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListReceiptFiltersCommand = () => new ListReceiptFiltersCommand({});

const run = async () => {
  const listReceiptFiltersCommand = createListReceiptFiltersCommand();

  return await sesClient.send(listReceiptFiltersCommand);
};
```

- Einzelheiten zur API finden Sie ListReceiptFiltersin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
```

```
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.

        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response["Filters"]
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

- Einzelheiten zur API finden Sie ListReceiptFiltersin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ses->listreceiptfilters( ).
        ot_filters = lo_result->get_filters( ).
```

```
        MESSAGE 'Receipt filters retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListReceiptFiltersin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTemplates`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListTemplates` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List email templates for the current account.
    /// </summary>
    /// <returns>A list of template metadata.</returns>
    public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
    {
        var result = new List<TemplateMetadata>();
        try
        {
```

```
            var response = await _amazonSimpleEmailService.ListTemplatesAsync(
                new ListTemplatesRequest());
            result = response.TemplatesMetadata;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
 ex.Message);
        }

        return result;
    }
```

- Einzelheiten zur API finden Sie ListTemplatesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        listAllTemplates(sesv2Client);
```

```
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTemplatesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListTemplatesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListTemplatesCommand = (maxItems) =>
  new ListTemplatesCommand({ MaxItems: maxItems });

const run = async () => {
```

```
  const listTemplatesCommand = createListTemplatesCommand(10);

  try {
    return await sesClient.send(listTemplatesCommand);
  } catch (err) {
    console.log("Failed to list templates.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie ListTemplatesin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
```

```
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

- Einzelheiten zur API finden Sie ListTemplatesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ses->listtemplates( iv_maxitems = iv_max_items ).
        ot_templates = lo_result->get_templatesmetadata( ).
        MESSAGE 'Templates retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
```

```
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTemplatesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendBulkTemplatedEmail`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`SendBulkTemplatedEmail`.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendBulkTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL_1 = postfix(getUniqueName("Bilbo"), "@example.com");
const VERIFIED_EMAIL_2 = postfix(getUniqueName("Frodo"), "@example.com");
```

```
const USERS = [
  { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL_1 },
  { firstName: "Frodo", emailAddress: VERIFIED_EMAIL_2 },
];

/**
 *
 * @param { { emailAddress: string, firstName: string }[] } users
 * @param { string } templateName the name of an existing template in SES
 * @returns { SendBulkTemplatedEmailCommand }
 */
const createBulkReminderEmailCommand = (users, templateName) => {
  return new SendBulkTemplatedEmailCommand({
    /**
     * Each 'Destination' uses a corresponding set of replacement data. We can
 map each user
     * to a 'Destination' and provide user specific replacement data to create
 personalized emails.
     *
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{name}},</h1><p>Don't forget about the party gifts!</
p>
     * Destination 1: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!
</p>
     * Destination 2: <h1>Hello Frodo,</h1><p>Don't forget about the party gifts!
</p>
     */
    Destinations: users.map((user) => ({
      Destination: { ToAddresses: [user.emailAddress] },
      ReplacementTemplateData: JSON.stringify({ name: user.firstName }),
    })),
    DefaultTemplateData: JSON.stringify({ name: "Shireling" }),
    Source: VERIFIED_EMAIL_1,
    Template: templateName,
  });
};

const run = async () => {
  const sendBulkTemplateEmailCommand = createBulkReminderEmailCommand(
    USERS,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendBulkTemplateEmailCommand);
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie SendBulkTemplatedEmailin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`SendEmail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendEmail` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    ///  Send an email by using Amazon SES.
    /// </summary>
    /// <param name="toAddresses">List of recipients.</param>
    /// <param name="ccAddresses">List of cc recipients.</param>
```

```
    /// <param name="bccAddresses">List of bcc recipients.</param>
    /// <param name="bodyHtml">Body of the email in HTML.</param>
    /// <param name="bodyText">Body of the email in plain text.</param>
    /// <param name="subject">Subject line of the email.</param>
    /// <param name="senderAddress">From address.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendEmailAsync(List<string> toAddresses,
        List<string> ccAddresses, List<string> bccAddresses,
        string bodyHtml, string bodyText, string subject, string senderAddress)
    {
        var messageId = "";
        try
        {
            var response = await _amazonSimpleEmailService.SendEmailAsync(
                new SendEmailRequest
                {
                    Destination = new Destination
                    {
                        BccAddresses = bccAddresses,
                        CcAddresses = ccAddresses,
                        ToAddresses = toAddresses
                    },
                    Message = new Message
                    {
                        Body = new Body
                        {
                            Html = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyHtml
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyText
                            }
                        },
                        Subject = new Content
                        {
                            Charset = "UTF-8",
                            Data = subject
                        }
                    },
                    Source = senderAddress
```

```
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendEmailAsync failed with exception: " +
 ex.Message);
        }

        return messageId;
    }
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send an email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param subject: Email subject.
  \param htmlBody: Email body as HTML. At least one body data is required.
  \param textBody: Email body as plain text. At least one body data is required.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                            const Aws::String &subject,
                            const Aws::String &htmlBody,
                            const Aws::String &textBody,
```

```
                            const Aws::String &senderEmailAddress,
                            const Aws::Vector<Aws::String> &ccAddresses,
                            const Aws::String &replyToAddress,
                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::SES::SESClient sesClient(clientConfiguration);

   Aws::SES::Model::Destination destination;
   if (!ccAddresses.empty()) {
       destination.WithCcAddresses(ccAddresses);
   }
   if (!recipients.empty()) {
       destination.WithToAddresses(recipients);
   }

   Aws::SES::Model::Body message_body;
   if (!htmlBody.empty()) {
       message_body.SetHtml(

Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
   }

   if (!textBody.empty()) {
       message_body.SetText(

Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
   }

   Aws::SES::Model::Message message;
   message.SetBody(message_body);
   message.SetSubject(
           Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

   Aws::SES::Model::SendEmailRequest sendEmailRequest;
   sendEmailRequest.SetDestination(destination);
   sendEmailRequest.SetMessage(message);
   if (!senderEmailAddress.empty()) {
       sendEmailRequest.SetSource(senderEmailAddress);
   }
   if (!replyToAddress.empty()) {
       sendEmailRequest.AddReplyToAddresses(replyToAddress);
   }

   auto outcome = sesClient.SendEmail(sendEmailRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So senden Sie eine formatierte E-Mail mit Amazon SES

Im folgenden Beispiel wird der `send-email`-Befehl verwendet, um eine formatierte E-Mail zu senden:

```
aws ses send-email --from sender@example.com --destination file://
destination.json --message file://message.json
```

Ausgabe:

```
{
   "MessageId": "EXAMPLEf3a5efcd1-51adec81-d2a4-4e3f-9fe2-5d85c1b23783-000000"
}
```

Das Ziel und die Nachricht sind JSON-Datenstrukturen, die in JSON-Dateien im aktuellen Verzeichnis gespeichert sind. Es handelt sich dabei um die folgenden Dateien:

`destination.json`:

```
{
  "ToAddresses":  ["recipient1@example.com", "recipient2@example.com"],
```

```
  "CcAddresses":  ["recipient3@example.com"],
  "BccAddresses": []
}
```

`message.json`:

```
{
   "Subject": {
       "Data": "Test email sent using the AWS CLI",
       "Charset": "UTF-8"
   },
   "Body": {
       "Text": {
           "Data": "This is the message body in text format.",
           "Charset": "UTF-8"
       },
       "Html": {
           "Data": "This message body contains HTML formatting. It can, for
 example, contain links like this one: <a class=\"ulink\" href=\"http://
docs.aws.amazon.com/ses/latest/DeveloperGuide\" target=\"_blank\">Amazon SES
 Developer Guide</a>.",
           "Charset": "UTF-8"
       }
   }
}
```

Ersetzen Sie die Absender- und Empfänger-E-Mail-Adressen durch die Adressen, die Sie verwenden möchten. Beachten Sie, dass die E-Mail-Adresse des Absenders mit Amazon SES verifiziert werden muss. Bis Ihnen Produktionszugriff auf Amazon SES gewährt wird, müssen Sie auch die E-Mail-Adresse jedes Empfängers verifizieren, es sei denn, es handelt sich bei dem Empfänger um den Amazon-SES-Postfachsimulator. Weitere Informationen zu verifizierten Identitäten finden Sie unter „Verifizieren von E-Mail-Adressen und Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

Die Nachrichten-ID in der Ausgabe gibt an, dass der Aufruf von send-email erfolgreich war.

Wenn Sie die E-Mail nicht erhalten, überprüfen Sie Ihr Junk-Postfach.

Weitere Informationen zum Senden formatierter E-Mails finden Sie unter „Senden formatierter E-Mails mit der Amazon-SES-API“ im Entwicklerhandbuch von Amazon Simple Email Service.

- Einzelheiten zur API finden Sie SendEmailin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
```

```
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();
```

```
        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
```

```
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use
 as an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
```

```
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list
 "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p>Please see the attached file for a " + "list of customers
 to contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
 bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
 IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());
```

```
        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
 InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));
```

```
        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendEmailCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createSendEmailCommand = (toAddress, fromAddress) => {
  return new SendEmailCommand({
    Destination: {
      /* required */
      CcAddresses: [
        /* more items */
      ],
      ToAddresses: [
        toAddress,
        /* more To-email addresses */
      ],
    },
    Message: {
      /* required */
      Body: {
        /* required */
        Html: {
          Charset: "UTF-8",
          Data: "HTML_FORMAT_BODY",
        },
        Text: {
          Charset: "UTF-8",
          Data: "TEXT_FORMAT_BODY",
        },
      },
      Subject: {
        Charset: "UTF-8",
        Data: "EMAIL_SUBJECT",
      },
```

```
    },
    Source: fromAddress,
    ReplyToAddresses: [
      /* more items */
    ],
  });
};

const run = async () => {
  const sendEmailCommand = createSendEmailCommand(
    "recipient@example.com",
    "sender@example.com",
  );

  try {
    return await sesClient.send(sendEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""
```

```
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html,
reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the
recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source,
destination.tos
            )
        except ClientError:
```

```
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id
```

- Einzelheiten zur API finden Sie SendEmailin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
```

```
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )
```

```
  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create message object
    DATA(lo_subject) = NEW /aws1/cl_sescontent( iv_data = iv_subject ).
    DATA(lo_text_body) = NEW /aws1/cl_sescontent( iv_data = iv_text ).
    DATA(lo_html_body) = NEW /aws1/cl_sescontent( iv_data = iv_html ).
    DATA(lo_body) = NEW /aws1/cl_sesbody(
      io_text = lo_text_body
      io_html = lo_html_body
    ).
    DATA(lo_message) = NEW /aws1/cl_sesmessage(
      io_subject = lo_subject
      io_body = lo_body
    ).

    TRY.
        " Send email
        DATA(lo_result) = lo_ses->sendemail(
          iv_source = iv_source
          io_destination = io_destination
          io_message = lo_message
          it_replytoaddresses = it_reply_tos
        ).
        ov_msg_id = lo_result->get_messageid( ).
```

```
        MESSAGE 'Email sent successfully' TYPE 'I'.
      CATCH /aws1/cx_sesacctsendingpause00 INTO DATA(lo_ex1).
        DATA(lv_error) = |Account sending paused: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_sesmessagerejected INTO DATA(lo_ex2).
        lv_error = |Message rejected: { lo_ex2->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex2.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendEmailin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendRawEmail`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendRawEmail` verwendet wird.

CLI

AWS CLI

So senden Sie eine RAW-E-Mail mit Amazon SES

Im folgenden Beispiel wird der `send-raw-email`-Befehl verwendet, um eine E-Mail mit einem TXT-Anhang zu senden:

```
aws ses send-raw-email --raw-message file://message.json
```

Ausgabe:

```
{
   "MessageId": "EXAMPLEf3f73d99b-c63fb06f-d263-41f8-a0fb-d0dc67d56c07-000000"
}
```

Die RAW-Nachricht ist eine JSON-Datenstruktur, die in einer Datei mit dem Namen `message.json` im aktuellen Verzeichnis gespeichert ist. Sie enthält Folgendes:

```
{
   "Data": "From: sender@example.com\nTo: recipient@example.com\nSubject:
 Test email sent using the AWS CLI (contains an attachment)\nMIME-Version:
 1.0\nContent-type: Multipart/Mixed; boundary=\"NextPart\"\n\n--NextPart
\nContent-Type: text/plain\n\nThis is the message body.\n\n--NextPart\nContent-
Type: text/plain;\nContent-Disposition: attachment; filename=\"attachment.txt\"\n
\nThis is the text in the attachment.\n\n--NextPart--"
}
```

Wie Sie sehen, ist „Data“ eine lange Zeichenfolge, die den gesamten RAW-E-Mail-Inhalt im MIME-Format enthält, einschließlich eines Anhangs namens attachment.txt.

Ersetzen Sie sender@example.com und recipient@example.com durch die Adressen, die Sie verwenden möchten. Beachten Sie, dass die E-Mail-Adresse des Absenders mit Amazon SES verifiziert werden muss. Bis Ihnen Produktionszugriff auf Amazon SES gewährt wird, müssen Sie auch die E-Mail-Adresse des Empfängers verifizieren, es sei denn, es handelt sich bei dem Empfänger um den Amazon-SES-Postfachsimulator. Weitere Informationen zu verifizierten Identitäten finden Sie unter „Verifizieren von E-Mail-Adressen und Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

Die Nachrichten-ID in der Ausgabe gibt an, dass der Aufruf von send-raw-email erfolgreich war.

Wenn Sie die E-Mail nicht erhalten, überprüfen Sie Ihr Junk-Postfach.

Weitere Informationen zum Senden von RAW-E-Mails finden Sie unter „Senden von RAW-E-Mails mit der Amazon-SES-API“ im Entwicklerhandbuch von Amazon Simple Email Service.

- Einzelheiten zur API finden Sie SendRawEmailin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie nodemailer, um eine E-Mail mit einem Anhang zu senden.

```
import sesClientModule from "@aws-sdk/client-ses";
/**
 * nodemailer wraps the SES SDK and calls SendRawEmail. Use this for more
 advanced
 * functionality like adding attachments to your email.
 *
 * https://nodemailer.com/transports/ses
 */
import nodemailer from "nodemailer";

/**
 * @param {string} from An Amazon SES verified email address.
 * @param {*} to An Amazon SES verified email address.
 */
export const sendEmailWithAttachments = (
  from = "from@example.com",
  to = "to@example.com",
) => {
  const ses = new sesClientModule.SESClient({});
  const transporter = nodemailer.createTransport({
    SES: { ses, aws: sesClientModule },
  });

  return new Promise((resolve, reject) => {
    transporter.sendMail(
      {
        from,
        to,
        subject: "Hello World",
        text: "Greetings from Amazon SES!",
        attachments: [{ content: "Hello World!", filename: "hello.txt" }],
      },
      (err, info) => {
        if (err) {
          reject(err);
        } else {
          resolve(info);
        }
      },
    );
  });
};
```

- Einzelheiten zur API finden Sie SendRawEmailin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`SendTemplatedEmail`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SendTemplatedEmail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Send an email using a template.
    /// </summary>
    /// <param name="sender">Address of the sender.</param>
    /// <param name="recipients">Addresses of the recipients.</param>
    /// <param name="templateName">Name of the email template.</param>
    /// <param name="templateDataObject">Data for the email template.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
        string templateName, object templateDataObject)
    {
        var messageId = "";
        try
        {
            // Template data should be serialized JSON from either a class or a
dynamic object.
            var templateData = JsonSerializer.Serialize(templateDataObject);
```

```
            var response = await
 _amazonSimpleEmailService.SendTemplatedEmailAsync(
                new SendTemplatedEmailRequest
                {
                    Source = sender,
                    Destination = new Destination
                    {
                        ToAddresses = recipients
                    },
                    Template = templateName,
                    TemplateData = templateData
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
 ex.Message);
        }

        return messageId;
    }
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send a templated email to a list of recipients.
/*!
```

```
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
 &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "\":\"" << pair.second <<
 "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";
```

```
    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
```

```
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }

    public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        /*
         * Specify both name and favorite animal (favoriteanimal) in your code
when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");
```

```
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL = postfix(getUniqueName("Bilbo"), "@example.com");

const USER = { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL };
```

```
/**
 *
 * @param { { emailAddress: string, firstName: string } } user
 * @param { string } templateName - The name of an existing template in Amazon
 SES.
 * @returns { SendTemplatedEmailCommand }
 */
const createReminderEmailCommand = (user, templateName) => {
  return new SendTemplatedEmailCommand({
    /**
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{contact.firstName}},</h1><p>Don't forget about the
 party gifts!</p>
     * Destination: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</
p>
     */
    Destination: { ToAddresses: [user.emailAddress] },
    TemplateData: JSON.stringify({ contact: { firstName: user.firstName } }),
    Source: VERIFIED_EMAIL,
    Template: templateName,
  });
};

const run = async () => {
  const sendReminderEmailCommand = createReminderEmailCommand(
    USER,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendReminderEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
 passed
        in this function contains key-value pairs that define the values to
 insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
 values
                              that are inserted in the template before it is
 sent.
        :return: The ID of the message, assigned by Amazon SES.
```

```
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Send templated email
        DATA(lo_result) = lo_ses->sendtemplatedemail(
          iv_source = iv_source
          io_destination = io_destination
          iv_template = iv_template_name
          iv_templatedata = iv_template_data
          it_replytoaddresses = it_reply_tos
        ).
        ov_msg_id = lo_result->get_messageid( ).
        MESSAGE 'Templated email sent successfully' TYPE 'I'.
      CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
        DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateTemplate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `UpdateTemplate` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
 sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
```

```
    } else {
        std::cerr << "Error updating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie UpdateTemplatein der AWS SDK für C++ API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
const HTML_PART = "<h1>Hello, World!</h1>";

const createUpdateTemplateCommand = () => {
  return new UpdateTemplateCommand({
    Template: {
      TemplateName: TEMPLATE_NAME,
      HtmlPart: HTML_PART,
      SubjectPart: "Example",
      TextPart: "Updated template text.",
    },
  });
};

const run = async () => {
  const updateTemplateCommand = createUpdateTemplateCommand();
```

```
  try {
    return await sesClient.send(updateTemplateCommand);
  } catch (err) {
    console.log("Failed to update template.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie UpdateTemplatein der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
```

```
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

- Einzelheiten zur API finden Sie UpdateTemplatein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_template) = NEW /aws1/cl_sestemplate(
  iv_templatename = iv_name
  iv_subjectpart = iv_subject
  iv_textpart = iv_text
  iv_htmlpart = iv_html
).

TRY.
    lo_ses->updatetemplate( io_template = lo_template ).
    MESSAGE 'Template updated successfully' TYPE 'I'.
  CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
    DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
    lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`VerifyDomainIdentity`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `VerifyDomainIdentity` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kopieren von E-Mail- und Domain-Identitäten von Region zu Region
- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

CLI

AWS CLI

So verifizieren Sie eine Domain mit Amazon SES

Im folgenden Beispiel wird der `verify-domain-identity`-Befehl verwendet, um eine Domain zu verifizieren:

```
aws ses verify-domain-identity --domain example.com
```

Ausgabe:

```
{
   "VerificationToken": "eoEmxw+YaYhb3h3iVJHuXMJXqeu1q1/wwmvjuEXAMPLE"
}
```

Um die Domain-Verifizierung abzuschließen, müssen Sie den DNS-Einstellungen Ihrer Domain einen TXT-Eintrag mit dem zurückgegebenen Bestätigungstoken hinzufügen. Weitere Informationen finden Sie unter „Verifizieren von Domains in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie VerifyDomainIdentityin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { VerifyDomainIdentityCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * You must have access to the domain's DNS settings to complete the
 * domain verification process.
 */
const DOMAIN_NAME = postfix(getUniqueName("Domain"), ".example.com");

const createVerifyDomainIdentityCommand = () => {
  return new VerifyDomainIdentityCommand({ Domain: DOMAIN_NAME });
};

const run = async () => {
  const VerifyDomainIdentityCommand = createVerifyDomainIdentityCommand();

  try {
    return await sesClient.send(VerifyDomainIdentityCommand);
  } catch (err) {
    console.log("Failed to verify domain.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie VerifyDomainIdentityin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

- Einzelheiten zur API finden Sie VerifyDomainIdentityin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lo_result) = lo_ses->verifydomainidentity( iv_domain =
iv_domain_name ).
       ov_token = lo_result->get_verificationtoken( ).
       MESSAGE 'Domain verification initiated' TYPE 'I'.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
       DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie VerifyDomainIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`VerifyEmailIdentity`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `VerifyEmailIdentity` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kopieren von E-Mail- und Domain-Identitäten von Region zu Region
- E-Mail-Versanddienste einrichten
- Verifizieren einer E-Mail-Identität und Senden von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Starts verification of an email identity. This request sends an email
   /// from Amazon SES to the specified email address. To complete
   /// verification, follow the instructions in the email.
   /// </summary>
   /// <param name="recipientEmailAddress">Email address to verify.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> VerifyEmailIdentityAsync(string
recipientEmailAddress)
   {
       var success = false;
       try
       {
           var response = await
_amazonSimpleEmailService.VerifyEmailIdentityAsync(
               new VerifyEmailIdentityRequest
               {
                   EmailAddress = recipientEmailAddress
               });

           success = response.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (Exception ex)
       {
           Console.WriteLine("VerifyEmailIdentityAsync failed with exception: "
+ ex.Message);
       }

       return success;
   }
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add an email address to the list of identities associated with this account
 and
//! initiate verification.
/*!
  \param emailAddress; The email address to add.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                        const Aws::Client::ClientConfiguration
 &clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
 sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }

    else
    {
```

```
        std::cerr << "Error initiating email verification. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So fügen Sie eine E-Mail-Adresse mit Amazon SES hinzu und verifizieren sie

Im folgenden Beispiel wird der `verify-email-identity`-Befehl verwendet, um eine E-Mail-Adresse zu verifizieren:

```
aws ses verify-email-identity --email-address user@example.com
```

Bevor Sie E-Mails mit Amazon SES versenden können, müssen Sie die Adresse oder Domain verifizieren, von denen Sie die E-Mail senden, um zu beweisen, dass sie Ihnen gehören. Ist Sie noch keinen Produktionszugriff haben, müssen Sie außerdem alle E-Mail-Adresse verifizieren, an die Sie E-Mails senden, mit Ausnahme derer, die vom Amazon-SES-Postfachsimulator bereitgestellt werden.

Nach dem Aufruf von verify-email-identity erhält die E-Mail-Adresse eine Bestätigungs-E-Mail. Der Benutzer muss auf den Link in der E-Mail klicken, um den Verifizierungsvorgang abzuschließen.

Weitere Informationen finden Sie unter „Verifizieren von E-Mail-Adressen in Amazon SES“ im Entwicklerhandbuch zu Amazon Simple Email Service.

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Import required AWS SDK clients and commands for Node.js
import { VerifyEmailIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const EMAIL_ADDRESS = "name@example.com";

const createVerifyEmailIdentityCommand = (emailAddress) => {
  return new VerifyEmailIdentityCommand({ EmailAddress: emailAddress });
};

const run = async () => {
  const verifyEmailIdentityCommand =
    createVerifyEmailIdentityCommand(EMAIL_ADDRESS);
  try {
    return await sesClient.send(verifyEmailIdentityCommand);
  } catch (err) {
    console.log("Failed to verify email identity.", err);
    return err;
  }
};
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->verifyemailidentity( iv_emailaddress = iv_email_address ).
    MESSAGE 'Email verification initiated' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon SES AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon SES mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon SES oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen einer Amazon-Transcribe-Streaming-App
- Kopieren Sie Amazon SES SES-E-Mail- und Domainidentitäten von einer AWS Mit einem von einer Region in eine andere AWS SDK

- Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten
- Erstellen eines Amazon-Redshift-Element-Trackers
- Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben
- Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Generieren von Anmeldeinformationen für die Verbindung mit einem Amazon-SES-SMTP-Endpunkt
- E-Mail-Versanddienste einrichten
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen
- Überprüfen Sie eine E-Mail-Identität und senden Sie Nachrichten mit Amazon SES mithilfe eines AWS SDK

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Kopieren Sie Amazon SES SES-E-Mail- und Domainidentitäten von einer AWS Mit einem von einer Region in eine andere AWS SDK

Das folgende Codebeispiel zeigt, wie Amazon SES SES-E-Mail- und Domainidentitäten von einer AWS Region in eine andere kopiert werden. Wenn Domänenidentitäten von Route 53 verwaltet werden, werden Überprüfungsdatensätze in die Domäne für die Zielregion kopiert.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identity_paginator = ses_client.get_paginator("list_identities")
        identity_iterator = identity_paginator.paginate(
            PaginationConfig={"PageSize": 20}
        )
```

```
        for identity_page in identity_iterator:
            for identity in identity_page["Identities"]:
                if "@" in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.",
            len(email_identities),
            len(domain_identities),
        )
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities


def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an
 email
    to be sent to each address. To complete verification, the recipient must
 follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for
 verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails


def verify_domains(domain_list, ses_client):
    """
```

```
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider
 for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response["VerificationToken"]
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token,
 domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.",
 domain)
    return domain_tokens


def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zone_paginator = route53_client.get_paginator("list_hosted_zones")
        zone_iterator = zone_paginator.paginate(PaginationConfig={"PageSize":
 20})
        zones = [
            zone for zone_page in zone_iterator for zone in
 zone_page["HostedZones"]
        ]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones
```

```
def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are
 taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
             domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain
 until a
        # zone match is found.
        domain_split = domain.split(".")
        for index in range(0, len(domain_split) - 1):
            sub_domain = ".".join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
 '.'.
                zone_name = zone["Name"][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones


def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
```

```
    domain_token_record_set_name = f"_amazonses.{domain}"
    record_set_paginator =
route53_client.get_paginator("list_resource_record_sets")
    record_set_iterator = record_set_paginator.paginate(
        HostedZoneId=zone["Id"], PaginationConfig={"PageSize": 20}
    )
    records = []
    for record_set_page in record_set_iterator:
        try:
            txt_record_set = next(
                record_set
                for record_set in record_set_page["ResourceRecordSets"]
                if record_set["Name"][:-1] == domain_token_record_set_name
                and record_set["Type"] == "TXT"
            )
            records = txt_record_set["ResourceRecords"]
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name,
                zone["Name"],
            )
            break
        except StopIteration:
            pass
    records.append({"Value": json.dumps(token)})
    changes = [
        {
            "Action": "UPSERT",
            "ResourceRecordSet": {
                "Name": domain_token_record_set_name,
                "Type": "TXT",
                "TTL": 1800,
                "ResourceRecords": records,
            },
        }
    ]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name,
            zone["Name"],
```

```
        )
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone
 %s.",
            err.response["Error"]["Code"],
            zone["Name"],
        )


def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to
 the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """
    dkim_tokens = []
    try:
        dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)["DkimTokens"]
        logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
 domain)
    except ClientError:
        logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
    return dkim_tokens


def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
    """
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

    :param hosted_zone: The hosted zone where the records are added.
    :param domain: The domain to add.
    :param tokens: The DKIM tokens for the domain to add.
    :param route53_client: A Boto3 Route 53 client.
    """
    try:
        changes = [
            {
                "Action": "UPSERT",
                "ResourceRecordSet": {
```

```
                    "Name": f"{token}._domainkey.{domain}",
                    "Type": "CNAME",
                    "TTL": 1800,
                    "ResourceRecords": [{"Value":
 f"{token}.dkim.amazonses.com"}],
                },
            }
            for token in tokens
        ]
        route53_client.change_resource_record_sets(
            HostedZoneId=hosted_zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Added %s DKIM CNAME records to %s in zone %s.",
            len(tokens),
            domain,
            hosted_zone["Name"],
        )
    except ClientError:
        logger.warning(
            "Couldn't add DKIM CNAME records for %s to zone %s.",
            domain,
            hosted_zone["Name"],
        )


def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
 Delivery,
                   or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press
 "
            f"Enter to skip: "
        )
        if topic_arn != "":
```

```
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn
                )
                logger.info("Configured %s for %s notifications.", identity,
 topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity,
 topic
                )


def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print(
        f"Replicating Amazon SES identities and other configuration from "
        f"{source_client.meta.region_name} to
 {destination_client.meta.region_name}."
    )
    print("-" * 88)

    print(f"Retrieving identities from {source_client.meta.region_name}.")
    source_emails, source_domains = get_identities(source_client)
    print("Email addresses found:")
    print(*source_emails)
    print("Domains found:")
    print(*source_domains)

    print("Starting verification for email identities.")
    dest_emails = verify_emails(source_emails, destination_client)
    print("Getting domain tokens for domain identities.")
    dest_domain_tokens = verify_domains(source_domains, destination_client)

    # Get Route 53 hosted zones and match them with Amazon SES domains.
    answer = input(
        "Is the DNS configuration for your domains managed by Amazon Route 53 (y/
n)? "
    )
    use_route53 = answer.lower() == "y"
    hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
    if use_route53:
```

```
        print("Adding or updating Route 53 TXT records for your domains.")
        domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
 hosted_zones)
        for domain in domain_zones:
            add_route53_verification_record(
                domain, dest_domain_tokens[domain], domain_zones[domain],
 route53_client
            )
    else:
        print(
            "Use these verification tokens to create TXT records through your DNS
 "
            "provider:"
        )
        pprint(dest_domain_tokens)

    answer = input("Do you want to configure DKIM signing for your identities (y/
n)? ")
    if answer.lower() == "y":
        # Build a set of unique domains from email and domain identities.
        domains = {email.split("@")[1] for email in dest_emails}
        domains.update(dest_domain_tokens)
        domain_zones = find_domain_zone_matches(domains, hosted_zones)
        for domain, zone in domain_zones.items():
            answer = input(
                f"Do you want to configure DKIM signing for {domain} (y/n)? "
            )
            if answer.lower() == "y":
                dkim_tokens = generate_dkim_tokens(domain, destination_client)
                if use_route53 and zone is not None:
                    add_dkim_domain_tokens(zone, domain, dkim_tokens,
 route53_client)
                else:
                    print(
                        "Add the following DKIM tokens as CNAME records through
 your "
                        "DNS provider:"
                    )
                    print(*dkim_tokens, sep="\n")

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities
 (y/n)? "
    )
```

```
    if answer.lower() == "y":
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/
n)? "
            )
            if answer.lower() == "y":
                configure_sns_topics(
                    identity, ["Bounce", "Delivery", "Complaint"],
 destination_client
                )

    print(f"Replication complete for {destination_client.meta.region_name}.")
    print("-" * 88)


def main():
    boto3_session = boto3.Session()
    ses_regions = boto3_session.get_available_regions("ses")
    parser = argparse.ArgumentParser(
        description="Copies email address and domain identities from one AWS
 Region to "
        "another. Optionally adds records for domain verification and DKIM "
        "signing to domains that are managed by Amazon Route 53, "
        "and sets up Amazon SNS notifications for events of interest."
    )
    parser.add_argument(
        "source_region", choices=ses_regions, help="The region to copy from."
    )
    parser.add_argument(
        "destination_region", choices=ses_regions, help="The region to copy to."
    )
    args = parser.parse_args()
    source_client = boto3.client("ses", region_name=args.source_region)
    destination_client = boto3.client("ses", region_name=args.destination_region)
    route53_client = boto3.client("route53")
    replicate(source_client, destination_client, route53_client)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie man die Amazon-DynamoDB-.NET-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in Amazon DynamoDB verfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer DynamoDB-Tabelle gespeichert sind.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel im AWS Code Examples Repository unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB

- Amazon SES

Erstellen eines Amazon-Redshift-Element-Trackers

Die folgenden Code-Beispiele zeigen, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Redshift Redshift-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Die folgenden Code-Beispiele zeigen, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

.NET

SDK für .NET

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für .NET In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend entwickelt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript (v3) eine Webanwendung erstellen, die Arbeitsaufgaben in einer Amazon Aurora Aurora-Datenbank verfolgt und Berichte mithilfe von

Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank verfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PHP-Backend entwickelt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) einen REST-Service erstellen, der Arbeitselemente in einer Amazon Aurora Aurora-Serverless-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für Python (Boto3) In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.

- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldedaten enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

.NET

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition zusammen mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie mit AWS SDK für Python (Boto3) dem eine Webanwendung erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingeschriebene Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.

- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Generieren von Anmeldeinformationen für die Verbindung mit einem Amazon-SES-SMTP-Endpunkt

Das folgende Code-Beispiel zeigt, wie Sie Anmeldeinformationen erzeugen, um eine Verbindung mit einem Amazon-SES-SMTP-Endpunkt herzustellen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
    "us-gov-east-1",  # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04


def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")
```

```
    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))


if __name__ == "__main__":
    main()
```

E-Mail-Versanddienste einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verifizieren Sie eine E-Mail-Adresse
- Verifizieren Sie eine Domain (optional)
- Überprüfe deine Sendelimits
- Senden Sie eine Test-E-Mail
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SES Setup Script (v2)
# This script helps you set up Amazon SES for sending emails

# Initialize log file
LOG_FILE="ses-setup.log"
echo "Starting Amazon SES setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local ignore_error="${4:-false}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        if [[ "$ignore_error" != "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"

    # No physical resources to clean up for SES setup
    # Email identities can be deleted if needed
    if [[ -n "$EMAIL_ADDRESS" ]]; then
        echo "Deleting email identity: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$EMAIL_ADDRESS\""
    fi

    if [[ -n "$RECIPIENT_EMAIL" && "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]];
 then
        echo "Deleting recipient email identity: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$RECIPIENT_EMAIL\""
    fi

    if [[ -n "$DOMAIN_NAME" ]]; then
        echo "Deleting domain identity: $DOMAIN_NAME" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$DOMAIN_NAME\""
    fi
}

# Track created resources
CREATED_RESOURCES=()

# Welcome message
echo "============================================="
echo "Amazon SES Setup Script"
echo "============================================="
echo "This script will help you set up Amazon SES for sending emails."
echo "You'll need to verify at least one email address that you own."
echo ""
echo "NOTE: New SES accounts are placed in the sandbox environment."
echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
echo ""

# Get email address to verify
echo "Please enter an email address that you own and can access:"
EMAIL_ADDRESS="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"
```

```
echo "$EMAIL_ADDRESS"

# Verify email identity
echo "Verifying email address: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address \"$EMAIL_ADDRESS
\"")
check_error "$OUTPUT" $? "Failed to verify email address"

CREATED_RESOURCES+=("Email identity: $EMAIL_ADDRESS")
echo "A verification email has been sent to $EMAIL_ADDRESS."
echo "Please check your inbox and click the verification link before continuing."
echo ""
echo "Press Enter after you've verified your email address..."
sleep 1

# Check verification status
echo "Checking verification status..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses list-identities --identity-type EmailAddress")
check_error "$OUTPUT" $? "Failed to list identities"

OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --identities
 \"$EMAIL_ADDRESS\"")
check_error "$OUTPUT" $? "Failed to get verification attributes"

# Check if the email is verified
VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o '"VerificationStatus": "[^"]*' |
 cut -d'"' -f4)
if [[ "$VERIFICATION_STATUS" != "Success" ]]; then
    echo "Email address $EMAIL_ADDRESS is not verified yet. Please check your
 inbox and verify before continuing."
    echo "Exiting script..."
    exit 1
fi

# Ask if user wants to verify a domain
echo ""
echo "Do you want to verify a domain for sending emails? (y/n):"
VERIFY_DOMAIN="y"
echo "$VERIFY_DOMAIN"

if [[ "$VERIFY_DOMAIN" =~ ^[Yy] ]]; then
    echo "Please enter the domain name you want to verify:"
    DOMAIN_NAME="test-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1).example.com"
```

```
    echo "$DOMAIN_NAME"

    # Verify domain identity
    echo "Verifying domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-identity --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to verify domain identity"

    # Extract verification token
    VERIFICATION_TOKEN=$(echo "$OUTPUT" | grep -o '"VerificationToken": "[^"]*' |
 cut -d'"' -f4)

    CREATED_RESOURCES+=("Domain identity: $DOMAIN_NAME")

    echo ""
    echo "==============================================="
    echo "Domain Verification Instructions"
    echo "==============================================="
    echo "To verify your domain ownership, you need to add a TXT record"
    echo "to your domain's DNS settings with the following values:"
    echo ""
    echo "Record Type: TXT"
    echo "Record Name: _amazonses.$DOMAIN_NAME"
    echo "Record Value: $VERIFICATION_TOKEN"
    echo ""
    echo "After adding this DNS record, verification may take up to 72 hours."
    echo ""

    # Set up DKIM for the domain
    echo "Setting up DKIM for domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-dkim --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to set up DKIM"

    # Extract DKIM tokens
    DKIM_TOKENS=$(echo "$OUTPUT" | grep -o '"DkimTokens": \[[^]]*\]' | sed
 's/"DkimTokens": \[\|\]//g' | sed 's/,//g' | sed 's/"//g')

    echo "==============================================="
    echo "DKIM Configuration Instructions"
    echo "==============================================="
    echo "To configure DKIM for your domain, add the following CNAME records"
    echo "to your domain's DNS settings:"
    echo ""

    for token in $DKIM_TOKENS; do
```

```
        echo "Record Type: CNAME"
        echo "Record Name: ${token}._domainkey.$DOMAIN_NAME"
        echo "Record Value: ${token}.dkim.amazonses.com"
        echo ""
    done

    echo "After adding these DNS records, DKIM verification may take up to 72
 hours."
    echo ""
fi

# Check sending limits
echo "Checking your SES sending limits..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses get-send-quota")
check_error "$OUTPUT" $? "Failed to get sending quota"

# Extract quota information
MAX_SEND_RATE=$(echo "$OUTPUT" | grep -o '"MaxSendRate": [0-9.]*' | cut -d' ' -
f2)
MAX_24_HOUR_SEND=$(echo "$OUTPUT" | grep -o '"Max24HourSend": [0-9.]*' | cut -d'
 ' -f2)
SENT_LAST_24_HOURS=$(echo "$OUTPUT" | grep -o '"SentLast24Hours": [0-9.]*' | cut
 -d' ' -f2)

echo ""
echo "============================================="
echo "Your SES Sending Limits"
echo "============================================="
echo "Maximum send rate: $MAX_SEND_RATE emails/second"
echo "Maximum 24-hour send: $MAX_24_HOUR_SEND emails"
echo "Sent in the last 24 hours: $SENT_LAST_24_HOURS emails"
echo ""

# Ask if user wants to send a test email
echo "Do you want to send a test email? (y/n):"
SEND_TEST="y"
echo "$SEND_TEST"

if [[ "$SEND_TEST" =~ ^[Yy] ]]; then
    echo ""
    echo "============================================="
    echo "SANDBOX ENVIRONMENT NOTICE"
    echo "============================================="
    echo "Your account is likely in the SES sandbox environment."
```

```
    echo "In the sandbox, you can only send emails to verified email addresses."
    echo ""
    echo "Do you want to:"
    echo "1. Send a test email to the same verified address ($EMAIL_ADDRESS)"
    echo "2. Verify another email address to use as recipient"
    echo ""
    echo "Enter your choice (1 or 2):"
    RECIPIENT_CHOICE="1"
    echo "$RECIPIENT_CHOICE"

    if [[ "$RECIPIENT_CHOICE" == "1" ]]; then
        RECIPIENT_EMAIL="$EMAIL_ADDRESS"
    else
        echo "Please enter the recipient email address you want to verify:"
        RECIPIENT_EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 |
 head -n 1)@example.com"
        echo "$RECIPIENT_EMAIL"

        # Verify recipient email identity if different from sender
        if [[ "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
            echo "Verifying recipient email address: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
            OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address
 \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to verify recipient email address"

            CREATED_RESOURCES+=("Email identity: $RECIPIENT_EMAIL")
            echo "A verification email has been sent to $RECIPIENT_EMAIL."
            echo "Please check the inbox and click the verification link before
 continuing."
            echo ""
            echo "Press Enter after you've verified the recipient email
 address..."
            sleep 1

            # Check recipient verification status
            OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --
identities \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to get recipient verification
 attributes"

            # Check if the recipient email is verified
            RECIPIENT_VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o
 '"VerificationStatus": "[^"]*' | cut -d'"' -f4)
```

```
            if [[ "$RECIPIENT_VERIFICATION_STATUS" != "Success" ]]; then
                echo "Recipient email address $RECIPIENT_EMAIL is not verified
 yet."
                echo "You can try sending the email anyway, but it may fail."
            fi
        fi
    fi

    echo "Sending test email from $EMAIL_ADDRESS to $RECIPIENT_EMAIL..." | tee -a
 "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses send-email \
        --from \"$EMAIL_ADDRESS\" \
        --destination \"ToAddresses=$RECIPIENT_EMAIL\" \
        --message \"Subject={Data=SES Test
 Email,Charset=UTF-8},Body={Text={Data=This is a test email sent from Amazon SES
 using the AWS CLI,Charset=UTF-8}}\"")

    # Don't exit on send email error, just report it
    check_error "$OUTPUT" $? "Failed to send test email" "true"

    # Check if the email was sent successfully
    if [[ "$OUTPUT" =~ "MessageId" ]]; then
        # Extract message ID
        MESSAGE_ID=$(echo "$OUTPUT" | grep -o '"MessageId": "[^"]*' | cut -d'"' -
f4)
        echo "Test email sent successfully! Message ID: $MESSAGE_ID"
    else
        echo "Failed to send test email. This is likely because your account is
 in the sandbox environment."
        echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
    fi
    echo ""
fi

# Summary of created resources
echo ""
echo "==============================================="
echo "Setup Complete - Resources Created"
echo "==============================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo ""
```

```
# Ask if user wants to clean up resources
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"
echo "$CLEANUP_CHOICE"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
fi

echo ""
echo "==============================================="
echo "Amazon SES Setup Complete"
echo "==============================================="
echo "For production use, you may need to request to be moved out of the SES
 sandbox."
echo "Visit the SES console and navigate to 'Account dashboard' to request
 production access."
echo ""
echo "Log file: $LOG_FILE"
echo "==============================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - DeleteIdentity
  - GetIdentityVerificationAttributes
  - GetSendQuota
  - ListIdentities
  - SendEmail
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie Sie eine AWS Step Functions Zustandsmaschine erstellen, die nacheinander AWS Lambda Funktionen aufruft.

Java

SDK für Java 2.x

Zeigt, wie Sie einen AWS serverlosen Workflow mithilfe von AWS Step Functions und erstellen. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Überprüfen Sie eine E-Mail-Identität und senden Sie Nachrichten mit Amazon SES mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Hinzufügen und verifizieren einer E-Mail-Adresse mit Amazon SES.
- Senden einer standardmäßigen E-Mail-Nachricht.
- Erstellen Sie eine Vorlage und senden Sie eine E-Mail-Nachricht mit Vorlage.
- Senden einer Nachricht mithilfe eines Amazon-SES-SMTP-Servers.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verifizieren einer E-Mail-Adresse mit Amazon SES und senden von Nachrichten.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    ses_client = boto3.client("ses")
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input("Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == "Success"
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your
 "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for
 use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): "
        )
        if answer.lower() == "y":
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete
 verification.")
            print("Waiting for verification...")
            try:
                ses_identity.wait_until_identity_exists(email)
```

```
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(
                    f"Verification timeout exceeded. You must complete the "
                    f"steps in the email sent to {email} to verify the address."
                )

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail
demo!</p>"

        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email,
            SesDestination([email]),
            "Amazon SES demo",
            test_message_text,
            test_message_html,
        )
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            "name": "doc-example-template",
            "subject": "Example of an email template.",
            "text": "This is what {{name}} will {{action}} if {{name}} can't
display "
            "HTML.",
            "html": "<p><i>This</i> is what {{name}} will {{action}} if {{name}}
"
            "<b>can</b> display HTML.</p>",
        }
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {"name": email.split("@")[0], "action": "read"}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(),
template_data
            )
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
```

```
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f"email-smtp.{region}.amazonaws.com"
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == "y":
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print("-" * 88)
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-Identitätsvorgängen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
```

```
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise


    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
```

```
        identity has been successfully verified or until it exceeds its maximum
time.

        :param identity: The identity to wait for.
        """
        try:
            waiter = self.ses_client.get_waiter("identity_exists")
            logger.info("Waiting until %s exists.", identity)
            waiter.wait(Identities=[identity])
        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.",
identity)
            raise


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
```

```
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-Vorlagenvorgängen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
```

```
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise


    def delete_template(self):
        """
        Deletes an email template.
```

```
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
```

```
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-E-Mail-Vorgängen.

```
class SesDestination:
    """Contains data about an email destination."""

    def __init__(self, tos, ccs=None, bccs=None):
```

```
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {"ToAddresses": self.tos}
        if self.ccs is not None:
            svc_format["CcAddresses"] = self.ccs
        if self.bccs is not None:
            svc_format["BccAddresses"] = self.bccs
        return svc_format



class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html,
 reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
```

```
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the
recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source,
destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to
insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
```

```
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
values
                                that are inserted in the template before it is
sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTemplate

- DeleteIdentity
- DeleteTemplate
- GetIdentityVerificationAttributes
- GetTemplate
- ListIdentities
- ListTemplates
- SendEmail
- SendTemplatedEmail
- UpdateTemplate
- VerifyDomainIdentity
- VerifyEmailIdentity

Codebeispiele für Amazon SES API v2 mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Simple Email Service API v2 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon SES API v2 – Weitere Informationen zu Amazon SES API v2.
- API-Referenz für Amazon SES API v2 – Details zu allen verfügbaren Aktionen von Amazon SES API v2.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon SES API v2 AWS SDKs
  - Hallo Amazon SES API v2
  - Aktionen für Amazon SES API v2 unter Verwendung AWS SDKs
    - Verwenden Sie CreateContactmit einem AWS SDK
    - Verwenden Sie CreateContactListmit einem AWS SDK
    - Verwenden Sie CreateEmailIdentitymit einem AWS SDK
    - Verwenden Sie CreateEmailTemplatemit einem AWS SDK
    - Verwenden Sie DeleteContactListmit einem AWS SDK
    - Verwenden Sie DeleteEmailIdentitymit einem AWS SDK
    - Verwenden Sie DeleteEmailTemplatemit einem AWS SDK
    - Verwenden Sie GetEmailIdentitymit einem AWS SDK
    - Verwenden Sie ListContactListsmit einem AWS SDK
    - Verwenden Sie ListContactsmit einem AWS SDK
    - Verwenden Sie SendBulkEmailmit einem AWS SDK
    - Verwenden Sie SendEmailmit einem AWS SDK
- Szenarien für die Verwendung von Amazon SES API v2 AWS SDKs
  - Senden Sie E-Mails mit Anhängen mithilfe der Amazon SES API v2
  - Ein vollständiges Amazon SES API v2-Newsletter-Szenario mit einem AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon SES API v2 AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Email Service API v2 mit AWS SDKs verwendet werden.

Beispiele

- Hallo Amazon SES API v2
- Aktionen für Amazon SES API v2 unter Verwendung AWS SDKs
  - Verwenden Sie CreateContactmit einem AWS SDK
  - Verwenden Sie CreateContactListmit einem AWS SDK
  - Verwenden Sie CreateEmailIdentitymit einem AWS SDK
  - Verwenden Sie CreateEmailTemplatemit einem AWS SDK

- Verwenden Sie DeleteContactListmit einem AWS SDK
- Verwenden Sie DeleteEmailIdentitymit einem AWS SDK
- Verwenden Sie DeleteEmailTemplatemit einem AWS SDK
- Verwenden Sie GetEmailIdentitymit einem AWS SDK
- Verwenden Sie ListContactListsmit einem AWS SDK
- Verwenden Sie ListContactsmit einem AWS SDK
- Verwenden Sie SendBulkEmailmit einem AWS SDK
- Verwenden Sie SendEmailmit einem AWS SDK

Hallo Amazon SES API v2

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von Amazon SES API v2 beginnen können.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_sesv2(sesv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon SESv2 client and
    list the email identities in your account. This example uses the default
    settings specified in your shared credentials and config files.

    :param sesv2_client: A Boto3 SESv2 client object.
    """
    print("Hello, Amazon SESv2. Let's list up to 5 email identities:\n")

    try:
        response = sesv2_client.list_email_identities(PageSize=5)
        identities = response["EmailIdentities"]
```

```
        if not identities:
            print(
                "No email identities found. "
                "Use CreateEmailIdentity to add one."
            )
        else:
            for identity in identities:
                print(
                    f"  Identity: {identity['IdentityName']}"
                    f"  Type: {identity['IdentityType']}"
                    f"  Status: {identity['VerificationStatus']}"
                    f"  Sending: {'Enabled' if identity['SendingEnabled'] else
 'Disabled'}"
                )
            print(f"\nShowing {len(identities)} email identity(ies).")

    except ClientError as err:
        logger.error(
            "Couldn't list email identities. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Einzelheiten zur API finden Sie ListEmailIdentitiesin AWS SDK for Python (Boto3) API Reference.

Aktionen für Amazon SES API v2 unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon SES API v2-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die API „Amazon-SES-API v2“ auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon SES API v2 AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Simple Email Service API v2.

Beispiele

- Verwenden Sie CreateContactmit einem AWS SDK
- Verwenden Sie CreateContactListmit einem AWS SDK
- Verwenden Sie CreateEmailIdentitymit einem AWS SDK
- Verwenden Sie CreateEmailTemplatemit einem AWS SDK
- Verwenden Sie DeleteContactListmit einem AWS SDK
- Verwenden Sie DeleteEmailIdentitymit einem AWS SDK
- Verwenden Sie DeleteEmailTemplatemit einem AWS SDK
- Verwenden Sie GetEmailIdentitymit einem AWS SDK
- Verwenden Sie ListContactListsmit einem AWS SDK
- Verwenden Sie ListContactsmit einem AWS SDK
- Verwenden Sie SendBulkEmailmit einem AWS SDK
- Verwenden Sie SendEmailmit einem AWS SDK

Verwenden Sie **`CreateContact`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateContact` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Creates a contact and adds it to the specified contact list.
/// </summary>
```

```
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }
```

- Einzelheiten zur API finden Sie CreateContactin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
```

```
                                .html(Content.builder().data(welcomeHtml).build())
                                .build())
                            .build())
                    .build())
                .build();
            SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
            System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
         } catch (AlreadyExistsException e) {
            // If the contact already exists, skip this step for that contact and
 proceed
            // with the next contact
            System.out.println("Contact already exists, skipping creation...");
         } catch (Exception e) {
            System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
            throw e;
         }
      }
```

- Einzelheiten zur API finden Sie CreateContactin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
```

```
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
 Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
```

```
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
 Skipping...")
                else:
                    raise e
```

- Einzelheiten zur API finden Sie CreateContactin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(),
 Error> {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateContactin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->createcontact(
      iv_contactlistname = iv_contact_list_name
      iv_emailaddress = iv_email_address ).
    MESSAGE 'Contact created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex.
    MESSAGE 'Bad request.' TYPE 'E'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateContactin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateContactList`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateContactList` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }
```

- Einzelheiten zur API finden Sie CreateContactListin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
```

```
        throw e;
    }
```

- Einzelheiten zur API finden Sie CreateContactListin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
```

```
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e
```

- Einzelheiten zur API finden Sie CreateContactListin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateContactListin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->createcontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact list already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request - contact list limit may be reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
    MESSAGE 'Limit exceeded - contact list limit reached.' TYPE 'I'.
    " Re-raise the exception so the caller can handle it
    RAISE EXCEPTION lo_limit_exceeded.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateContactListin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateEmailIdentity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateEmailIdentity` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Szenario mit E-Mail-Anhängen
- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
```

```
            Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   try {
     CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
         .emailIdentity(verifiedEmail)
         .build();
     sesClient.createEmailIdentity(createEmailIdentityRequest);
     System.out.println("Email identity created: " + verifiedEmail);
   } catch (AlreadyExistsException e) {
     System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
   } catch (NotFoundException e) {
     System.err.println("The provided email address is not verified: " +
verifiedEmail);
     throw e;
   } catch (LimitExceededException e) {
     System.err
         .println("You have reached the limit for email identities. Please
remove some identities and try again.");
     throw e;
   } catch (SesV2Exception e) {
     System.err.println("Error creating email identity: " + e.getMessage());
     throw e;
   }
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Starts the process of verifying an email identity (email address or
 domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
```

```
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin AWS SDK for Python (Boto3) API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
```

```
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
 successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->createemailidentity(
      iv_emailidentity = iv_email_identity ).
    MESSAGE 'Email identity created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Email identity already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
  CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
    MESSAGE lo_limit_exceeded TYPE 'I' DISPLAY LIKE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateEmailIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateEmailTemplate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateEmailTemplate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Szenario mit E-Mail-Anhängen
- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
 exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
```

```
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and
 inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
 remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
 e.getMessage());
      throw e;
    }
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.
```

```
        :param template_name: The name for the new template.
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein AWS SDK for Python (Boto3) API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
```

```
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();
```

```
        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_template_content) = NEW /aws1/cl_se2emailtmplcontent(
          iv_subject = iv_subject
          iv_html = iv_html
          iv_text = iv_text ).
```

```
        lo_se2->createemailtemplate(
          iv_templatename = iv_template_name
          io_templatecontent = lo_template_content ).
        MESSAGE 'Email template created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Email template already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex.
        MESSAGE 'Bad request.' TYPE 'E'.
      CATCH /aws1/cx_se2limitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateEmailTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteContactList`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteContactList` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
list: {ex.Message}");
        }

        return false;
    }
```

- Einzelheiten zur API finden Sie DeleteContactList in der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Einzelheiten zur API finden Sie DeleteContactListin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
```

```
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)
```

- Einzelheiten zur API finden Sie DeleteContactListin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }
```

- Einzelheiten zur API finden Sie DeleteContactListin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->deletecontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request.' TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteContactListin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteEmailIdentity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteEmailIdentity` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Szenario mit E-Mail-Anhängen
- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
identity: {ex.Message}");
        }

        return false;
    }
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
```

```
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin AWS SDK for Python (Boto3) API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
```

```
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:

 self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
 successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
 exist.")
                else:
                    print(e)
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        lo_se2->deleteemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE 'Email identity deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Email identity not found.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteEmailIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteEmailTemplate`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteEmailTemplate` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Szenario mit E-Mail-Anhängen
- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
template: {ex.Message}");
        }

        return false;
    }
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   try {
     // Delete the template
     DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
         .templateName(TEMPLATE_NAME)
         .build();

     sesClient.deleteEmailTemplate(deleteTemplateRequest);

     System.out.println("Email template deleted: " + TEMPLATE_NAME);
   } catch (NotFoundException e) {
     // If the email template does not exist, log the error and proceed
     System.out.println("Email template not found. Skipping deletion...");
   } catch (Exception e) {
     System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
     e.printStackTrace();
   }
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_template(
                TemplateName=template_name
```

```
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein AWS SDK for Python (Boto3) API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
```

```
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
```

```
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_se2->deleteemailtemplate(
          iv_templatename = iv_template_name ).
        MESSAGE 'Email template deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE 'Email template not found.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteEmailTemplatein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetEmailIdentity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetEmailIdentity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Szenario mit E-Mail-Anhängen

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification
status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie GetEmailIdentityin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Bestimmt, ob eine E-Mail-Adresse überprüft wurde.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie GetEmailIdentityin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_se2->getemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE |Identity type: { oo_result->get_identitytype( ) }, | &&
                |verified for sending: { oo_result-
>get_verifiedforsendingstatus( ) }| TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE |Email identity { iv_email_identity } not found.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetEmailIdentityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListContactLists`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`ListContactLists`.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
```

```
}
```

- Einzelheiten zur API finden Sie ListContactListsin der API-Referenz zum AWS SDK für Rust.

Verwenden Sie **`ListContacts`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListContacts` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Newsletter-Szenario

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
```

```
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }
```

- Einzelheiten zur API finden Sie ListContactsin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    ListContactsRequest contactListRequest = ListContactsRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    List<String> contactEmails;
    try {
```

```
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }
```

- Einzelheiten zur API finden Sie ListContactsin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()
```

```
class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e
```

- Einzelheiten zur API finden Sie ListContactsin AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;
```

```
    println!("Contacts:");

    for contact in resp.contacts() {
        println!("  {}", contact.email_address().unwrap_or_default());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListContactsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_se2->listcontacts(
          iv_contactlistname = iv_contact_list_name ).
        DATA(lv_count) = lines( oo_result->get_contacts( ) ).
        MESSAGE |Retrieved { lv_count } contacts from list.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE 'Contact list not found.' TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListContactsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendBulkEmail`** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SendBulkEmail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Szenario mit E-Mail-Anhängen

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def send_bulk_email(
```

```
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
            return results
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie SendBulkEmailin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Build the default template content used for all bulk recipients
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_default_content) = NEW /aws1/cl_se2bulkemailcontent(
          io_template = lo_template ).

        DATA(lo_result) = lo_se2->sendbulkemail(
```

```
          iv_fromemailaddress = iv_from_address
          io_defaultcontent   = lo_default_content
          it_bulkemailentries = it_bulk_entries ).

        ot_results = lo_result->get_bulkemailentryresults( ).
        MESSAGE |Bulk email sent to { lines( it_bulk_entries ) } recipient(s).|
TYPE 'I'.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_rejected).
        MESSAGE lo_rejected TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_rejected.
      CATCH /aws1/cx_se2mailfrmdomnotver00 INTO DATA(lo_not_verified).
        MESSAGE lo_not_verified TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_verified.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendBulkEmailin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendEmail`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SendEmail` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Szenario mit E-Mail-Anhängen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email
from.</param>
    /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
(optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
    /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
    public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
        string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
    {
        var request = new SendEmailRequest
        {
            FromEmailAddress = fromEmailAddress
        };

        if (toEmailAddresses.Any())
        {
            request.Destination = new Destination { ToAddresses =
toEmailAddresses };
        }
```

```
        if (!string.IsNullOrEmpty(templateName))
        {
            request.Content = new EmailContent()
            {
                Template = new Template
                {
                    TemplateName = templateName,
                    TemplateData = templateData
                }
            };
        }
        else
        {
            request.Content = new EmailContent
            {
                Simple = new Message
                {
                    Subject = new Content { Data = subject },
                    Body = new Body
                    {
                        Html = new Content { Data = htmlContent },
                        Text = new Content { Data = textContent }
                    }
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
```

```
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
        }

        return string.Empty;
    }
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
```

```
                                recipient - An email address that represents the
 recipient.\s
                                subject - The subject line.\s
                            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML) {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();
```

```
        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
                    + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Sendet eine Nachricht unter Verwendung einer Vorlage.

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
```

```
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
     }
```

Sendet eine Nachricht mit Header-Informationen.

```
public class SendwithHeader {

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
```

```
                .build();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c ->
 c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
```

```
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " +
 response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht mit optionalen Anhängen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""
```

```
    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
```

```
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
            )
            return message_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- Einzelheiten zur API finden Sie SendEmailin AWS SDK for Python (Boto3) API Reference.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht an alle Mitglieder der Kontaktliste.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
```

```
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
 Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
```

Sendet mithilfe einer Vorlage eine Nachricht an alle Mitglieder der Kontaktliste.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )
```

- Einzelheiten zur API finden Sie SendEmailin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
      destination: {
        to_addresses: [recipient_email]
```

```
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht an alle Mitglieder der Kontaktliste.

```
async fn send_message(
    client: &Client,
    list: &str,
    from: &str,
    subject: &str,
    message: &str,
```

```
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts();

    let cs: Vec<String> = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default().to_string())
        .collect();

    let mut dest: Destination = Destination::builder().build();
    dest.to_addresses = Some(cs);
    let subject_content = Content::builder()
        .data(subject)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body_content = Content::builder()
        .data(message)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
        .send()
        .await?;
```

```
    println!("Email sent to list");

    Ok(())
}
```

Sendet mithilfe einer Vorlage eine Nachricht an alle Mitglieder der Kontaktliste.

```
            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
```

```
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
 {}", email, e)),
            }
```

- Einzelheiten zur API finden Sie SendEmailin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value =
 iv_to_email_address ) TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create message content
        DATA(lo_subject) = NEW /aws1/cl_se2content( iv_data = iv_subject ).
        DATA(lo_text_body) = NEW /aws1/cl_se2content( iv_data = iv_text_body ).
        DATA(lo_html_body) = NEW /aws1/cl_se2content( iv_data = iv_html_body ).
        DATA(lo_body) = NEW /aws1/cl_se2body(
          io_text = lo_text_body
          io_html = lo_html_body ).
        DATA(lo_message) = NEW /aws1/cl_se2message(
          io_subject = lo_subject
          io_body = lo_body ).
```

```
        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_simple = lo_message ).

        " Send the email
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content ).
        MESSAGE 'Email sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

Sendet eine Nachricht unter Verwendung einer Vorlage.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value =
 iv_to_email_address ) TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create template reference
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_template = lo_template ).

        " Create list management options
        DATA(lo_list_mgmt) = NEW /aws1/cl_se2listmanagementopts(
          iv_contactlistname = iv_contact_list_name ).
```

```
      " Send the email using template
      lo_se2->sendemail(
        iv_fromemailaddress = iv_from_email_address
        io_destination = lo_destination
        io_content = lo_content
        io_listmanagementoptions = lo_list_mgmt ).
      MESSAGE 'Email sent using template successfully.' TYPE 'I'.
    CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
      MESSAGE 'Account suspended.' TYPE 'I'.
      RAISE EXCEPTION lo_account_suspended.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE 'Bad request.' TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
      MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
      RAISE EXCEPTION lo_message_rejected.
  ENDTRY.
```

- Einzelheiten zur API finden Sie SendEmailin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon SES API v2 AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon SES API v2 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon SES API v2 oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Senden Sie E-Mails mit Anhängen mithilfe der Amazon SES API v2
- Ein vollständiges Amazon SES API v2-Newsletter-Szenario mit einem AWS SDK

Senden Sie E-Mails mit Anhängen mithilfe der Amazon SES API v2

Das folgende Codebeispiel zeigt, wie E-Mails mit Anhängen mithilfe der Amazon SES API v2 gesendet werden.

- Erstellen Sie eine E-Mail-Vorlage für Massensendungen.
- Senden Sie eine einfache E-Mail mit einem Dateianhang.
- Senden Sie eine einfache E-Mail mit einem Inline-Bild.
- Senden Sie Massen-E-Mails mit Vorlagen und Anhängen.
- Bereinigen Sie die Ressourcen.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem E-Mail-Anhänge demonstriert werden.

```
class SESv2EmailAttachmentsScenario:
    """
    Demonstrates sending emails with attachments using Amazon SESv2.

    This scenario demonstrates:
    1. Setting up an email identity and template.
    2. Sending a simple email with a file attachment.
    3. Sending a simple email with an inline image.
    4. Sending bulk templated emails with attachments.
    5. Cleaning up created resources.
    """

    TEMPLATE_NAME = "AttachmentDemoTemplate"

    def __init__(self, sesv2_wrapper: SESv2Wrapper) -> None:
        """
        :param sesv2_wrapper: An instance of the SESv2Wrapper class.
```

```
        """
        self.sesv2_wrapper = sesv2_wrapper
        self.sender_email = ""
        self.recipient_emails: list = []
        self.identity_was_created = False

    def run_scenario(self) -> None:
        """Runs the SESv2 email attachments scenario."""
        print("-" * 88)
        print("Welcome to the Amazon SESv2 Email Attachments Scenario.")
        print("-" * 88)
        print(
            "This scenario demonstrates how to send emails with attachments\n"
            "using SESv2 attachment support. SES handles MIME\n"
            "construction automatically, so you don't need to build raw\n"
            "MIME messages.\n"
        )

        try:
            self._setup()
            self._step1_send_email_with_attachment()
            self._step2_send_email_with_inline_image()
            self._step3_send_bulk_email_with_attachments()
        except Exception as e:
            logger.error("Scenario failed: %s", e)
            print(f"\nThe scenario encountered an error: {e}")
        finally:
            self._cleanup()

    # ---------- Setup ----------

    def _setup(self) -> None:
        """
        Prompts for configuration, verifies the sender identity, prepares a
        sample attachment, and creates an email template.
        """
        print("\n--- Setup ---\n")

        # Prompt for sender and recipient addresses.
        print(
            "Both sender and recipient addresses must be verified if your\n"
            "account is in the SES sandbox.\n"
        )
        self.sender_email = q.ask(
```

```
            "Enter a verified sender email address: "
        )
        recipient_input = q.ask(
            "Enter one or more recipient email addresses (comma-separated): "
        )
        self.recipient_emails = [
            addr.strip() for addr in recipient_input.split(",") if addr.strip()
        ]

        # Verify the sender identity.
        print(f"\nChecking identity for {self.sender_email}...")
        try:
            identity_info = self.sesv2_wrapper.get_email_identity(
                self.sender_email
            )
            verified = identity_info.get("VerifiedForSendingStatus", False)
            if verified:
                print(f"  {self.sender_email} is verified and ready to send.")
            else:
                print(
                    f"  {self.sender_email} exists but is not yet verified."
                )
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Identity {self.sender_email} not found. "
                    "Creating it now..."
                )
                result = self.sesv2_wrapper.create_email_identity(
                    self.sender_email
                )
                self.identity_was_created = True
                print(
                    f"  Identity created. Verification status: "
                    f"{result.get('VerifiedForSendingStatus', False)}"
                )
                print(
                    "  Check your inbox and click the verification link "
                    "before continuing."
                )
                q.ask("Press Enter when you have verified the address...")
            else:
                raise
```

```
        # Create the email template for the bulk-send step.
        print("\nCreating email template for the bulk email step...")
        try:
            self.sesv2_wrapper.create_email_template(
                template_name=self.TEMPLATE_NAME,
                subject="Bulk Email with Attachment for {{name}}",
                html_body=(
                    "<h1>Hello {{name}}</h1>"
                    "<p>Please find the attached document.</p>"
                ),
                text_body=(
                    "Hello {{name}}, Please find the attached document."
                ),
            )
            print(f"  Template '{self.TEMPLATE_NAME}' created.\n")
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' already exists. "
                    "Using it.\n"
                )
            else:
                raise

    # ---------- Step 1: Simple email with file attachment ----------

    def _step1_send_email_with_attachment(self) -> None:
        """Sends a simple email with a text file attachment."""
        print("\n--- Step 1: Send a Simple Email with a File Attachment ---\n")
        print(
            "Creating a sample text file attachment and sending it with\n"
            "the Simple email content type. SES constructs the MIME message\n"
            "automatically.\n"
        )

        # Prepare a sample text file as bytes.
        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report text file",
```

```
            "ContentTransferEncoding": "BASE64",
        }

        print(
            "Note: When using an AWS SDK, the SDK handles base64 encoding\n"
            "automatically. Direct API callers must encode content themselves.\n"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Simple Email with Attachment",
            html_body=(
                "<h1>Attachment Demo</h1>"
                "<p>Please see the attached <b>report document</b>.</p>"
            ),
            text_body="Please see the attached report document.",
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  SES automatically constructed the MIME message with the "
            "attachment.\n"
        )

    # ---------- Step 2: Simple email with inline image ----------

    def _step2_send_email_with_inline_image(self) -> None:
        """Sends a simple email with an inline image that renders in HTML."""
        print("\n--- Step 2: Send a Simple Email with an Inline Image ---\n")
        print(
            "This step demonstrates INLINE disposition. The image renders\n"
            "directly in the HTML body using a 'cid:' reference instead of\n"
            "appearing as a downloadable attachment.\n"
        )

        # Create a minimal 1x1 red PNG (valid PNG file).
        sample_image = (
            b"\x89PNG\r\n\x1a\n"  # PNG signature
            b"\x00\x00\x00\rIHDR\x00\x00\x00\x01\x00\x00\x00\x01"
            b"\x08\x02\x00\x00\x00\x90wS\xde"  # 1x1 RGB
            b"\x00\x00\x00\x0cIDATx\x9cc\xf8\x0f\x00\x00\x01\x01"
            b"\x00\x05\x18\xd8N"  # compressed data
```

```
            b"\x00\x00\x00\x00IEND\xaeB`\x82"  # IEND
        )

        attachment = {
            "RawContent": sample_image,
            "FileName": "logo.png",
            "ContentType": "image/png",
            "ContentDisposition": "INLINE",
            "ContentId": "logo123",
            "ContentDescription": "Company logo",
            "ContentTransferEncoding": "BASE64",
        }

        html_body = (
            "<html><body>"
            "<h1>Inline Image Demo</h1>"
            "<p>Here is our logo:</p>"
            '<img src="cid:logo123" alt="Company Logo">'
            "</body></html>"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Inline Image",
            html_body=html_body,
            text_body=(
                "Inline Image Demo - Please view this email in an "
                "HTML-capable client to see the embedded image."
            ),
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  The ContentId 'logo123' is referenced in the HTML body via\n"
            "  'cid:logo123', which lets the image render inline.\n"
        )

    # ---------- Step 3: Bulk templated email with attachments ----------

    def _step3_send_bulk_email_with_attachments(self) -> None:
        """Sends bulk templated emails with attachments to multiple
recipients."""
```

```
        print("\n--- Step 3: Send Bulk Templated Emails with Attachments ---\n")
        print(
            "Using SendBulkEmail to send a templated email with an attachment\n"
            "to multiple recipients in a single API call. Each recipient gets\n"
            "personalized content via template data.\n"
        )

        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report for bulk recipients",
            "ContentTransferEncoding": "BASE64",
        }

        # Build one entry per recipient with personalized names.
        names = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
        bulk_entries = []
        for i, email in enumerate(self.recipient_emails):
            name = names[i] if i < len(names) else f"Recipient{i + 1}"
            bulk_entries.append(
                {
                    "Destination": {"ToAddresses": [email]},
                    "ReplacementEmailContent": {
                        "ReplacementTemplate": {
                            "ReplacementTemplateData": json.dumps(
                                {"name": name}
                            )
                        }
                    },
                }
            )

        results = self.sesv2_wrapper.send_bulk_email(
            from_address=self.sender_email,
            template_name=self.TEMPLATE_NAME,
            default_template_data='{"name": "Valued Customer"}',
            bulk_entries=bulk_entries,
            attachments=[attachment],
        )
```

```
        print("  Bulk email results:")
        for idx, result in enumerate(results):
            status = result.get("Status", "Unknown")
            msg_id = result.get("MessageId", "N/A")
            error = result.get("Error", "")
            recipient = (
                self.recipient_emails[idx]
                if idx < len(self.recipient_emails)
                else "Unknown"
            )
            print(f"    {recipient}: Status={status}, MessageId={msg_id}")
            if error:
                print(f"      Error: {error}")

        print(
            "\n  All recipients receive the same attachment(s) defined in\n"
            "  DefaultContent. Template data is personalized per recipient.\n"
        )

    # ---------- Cleanup ----------

    def _cleanup(self) -> None:
        """Deletes the email template and optionally the email identity."""
        print("\n--- Cleanup ---\n")

        # Delete the email template.
        try:
            self.sesv2_wrapper.delete_email_template(self.TEMPLATE_NAME)
            print(f"  Template '{self.TEMPLATE_NAME}' deleted.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' was already deleted."
                )
            else:
                logger.error("Failed to delete template: %s", err)

        # Optionally delete the email identity.
        if self.identity_was_created and self.sender_email:
            delete_identity = q.ask(
                f"Delete the email identity '{self.sender_email}'? (y/n) ",
                q.is_yesno,
            )
            if delete_identity:
```

```
                try:
                    self.sesv2_wrapper.delete_email_identity(
                        self.sender_email
                    )
                    print(
                        f"  Email identity '{self.sender_email}' deleted."
                    )
                except ClientError as err:
                    if err.response["Error"]["Code"] == "NotFoundException":
                        print(
                            f"  Identity '{self.sender_email}' was "
                            "already deleted."
                        )
                    else:
                        logger.error(
                            "Failed to delete identity: %s", err
                        )
            else:
                print(
                    f"  Skipping identity deletion for {self.sender_email}."
                )
        else:
            print(
                "  Sender identity was pre-existing. Skipping deletion."
            )

        print("\nAll resources have been cleaned up.")
        print("-" * 88)
```

Erstellen Sie eine SESv2-Wrapper-Klasse zur Verwaltung von Vorgängen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
```

```
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification
status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Starts the process of verifying an email identity (email address or
domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.
```

```
        :param template_name: The name for the new template.
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
```

```
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
```

```
            )
            return message_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
```

```
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
```

```
        try:
            self.sesv2_client.delete_email_template(
                TemplateName=template_name
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - GetEmailIdentity
  - SendBulkEmail
  - SendEmail

Ein vollständiges Amazon SES API v2-Newsletter-Szenario mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie das Newsletter-Szenario von Amazon SES API v2 ausführen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.SimpleEmailV2;
```

```
using Amazon.SimpleEmailV2.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace Sesv2Scenario;

public static class NewsletterWorkflow
{
    /*
      This scenario demonstrates how to use the Amazon Simple Email Service (SES)
 v2 to send a coupon newsletter to a list of subscribers.
      The scenario performs the following tasks:

      1. Prepare the application:
         - Create a verified email identity for sending and replying to emails.
         - Create a contact list to store the subscribers' email addresses.
         - Create an email template for the coupon newsletter.

      2. Gather subscriber email addresses:
         - Prompt the user for a base email address.
         - Create 3 variants of the email address using subaddress extensions
 (e.g., user+ses-weekly-newsletter-1@example.com).
         - Add each variant as a contact to the contact list.
         - Send a welcome email to each new contact.

      3. Send the coupon newsletter:
         - Retrieve the list of contacts from the contact list.
         - Send the coupon newsletter using the email template to each contact.

      4. Monitor and review:
         - Provide instructions for the user to review the sending activity and
 metrics in the AWS console.

      5. Clean up resources:
         - Delete the contact list (which also deletes all contacts within it).
         - Delete the email template.
         - Optionally delete the verified email identity.

    */

    public static SESv2Wrapper _sesv2Wrapper;
```

```
    public static string? _baseEmailAddress = null;
    public static string? _verifiedEmail = null;
    private static string _contactListName = "weekly-coupons-newsletter";
    private static string _templateName = "weekly-coupons";
    private static string _subject = "Weekly Coupons Newsletter";
    private static string _htmlContentFile = "coupon-newsletter.html";
    private static string _textContentFile = "coupon-newsletter.txt";
    private static string _htmlWelcomeFile = "welcome.html";
    private static string _textWelcomeFile = "welcome.txt";
    private static string _couponsDataFile = "sample_coupons.json";

    // Relative location of the resources folder.
    private static string _resourcesFilePathLocation = "../../../../resources/";

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSimpleEmailServiceV2>()
                    .AddTransient<SESv2Wrapper>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon SES v2 Coupon Newsletter
Scenario.");
            Console.WriteLine("This scenario demonstrates how to use the Amazon
Simple Email Service (SES) v2 " +
                              "\r\nto send a coupon newsletter to a list of
subscribers.");

            // Prepare the application.
```

```
            var emailIdentity = await PrepareApplication();

            // Gather subscriber email addresses.
            await GatherSubscriberEmailAddresses(emailIdentity);

            // Send the coupon newsletter.
            await SendCouponNewsletter(emailIdentity);

            // Monitor and review.
            MonitorAndReview(true);

            // Clean up resources.
            await Cleanup(emailIdentity, true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon SES v2 Coupon Newsletter scenario is
complete.");
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sesv2Wrapper = host.Services.GetRequiredService<SESv2Wrapper>();
    }

    /// <summary>
    /// Set up the resources for the scenario.
    /// </summary>
    /// <returns>The email address of the verified identity.</returns>
    public static async Task<string?> PrepareApplication()
    {
        var htmlContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _htmlContentFile);
```

```
        var textContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _textContentFile);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("1. In this step, we will prepare the application:" +
                          "\r\n  - Create a verified email identity for sending
and replying to emails." +
                          "\r\n  - Create a contact list to store the
subscribers' email addresses." +
                          "\r\n  - Create an email template for the coupon
newsletter.\r\n");

        // Prompt the user for a verified email address.
        while (!IsEmail(_verifiedEmail))
        {
            Console.Write("Enter a verified email address or an email to verify:
");
            _verifiedEmail = Console.ReadLine();
        }

        try
        {
            // Create an email identity and start the verification process.
            await _sesv2Wrapper.CreateEmailIdentityAsync(_verifiedEmail);
            Console.WriteLine($"Identity {_verifiedEmail} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Identity {_verifiedEmail} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email identity: {ex.Message}");
        }

        // Create a contact list.
        try
        {
            await _sesv2Wrapper.CreateContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} created.");
        }
        catch (AlreadyExistsException)
        {
```

```
            Console.WriteLine($"Contact list {_contactListName} already
exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating contact list: {ex.Message}");
        }

        // Create an email template.
        try
        {
            await _sesv2Wrapper.CreateEmailTemplateAsync(_templateName, _subject,
htmlContent, textContent);
            Console.WriteLine($"Email template {_templateName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Email template {_templateName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email template: {ex.Message}");
        }

        return _verifiedEmail;
    }

    /// <summary>
    /// Generate subscriber addresses and send welcome emails.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> GatherSubscriberEmailAddresses(string
fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("2. In Step 2, we will gather subscriber email
addresses:" +
                          "\r\n  - Prompt the user for a base email address." +
                          "\r\n  - Create 3 variants of the email address using
subaddress extensions (e.g., user+ses-weekly-newsletter-1@example.com)." +
                          "\r\n  - Add each variant as a contact to the contact
list." +
```

```
                            "\r\n  - Send a welcome email to each new contact.\r
\n");

        // Prompt the user for a base email address.
        while (!IsEmail(_baseEmailAddress))
        {
            Console.Write("Enter a base email address (e.g., user@example.com):
 ");
            _baseEmailAddress = Console.ReadLine();
        }

        // Create 3 variants of the email address using +ses-weekly-newsletter-1,
 +ses-weekly-newsletter-2, etc.
        var baseEmailAddressParts = _baseEmailAddress!.Split("@");
        for (int i = 1; i <= 3; i++)
        {
            string emailAddress = $"{baseEmailAddressParts[0]}+ses-weekly-
newsletter-{i}@{baseEmailAddressParts[1]}";

            try
            {
                // Create a contact with the email address in the contact list.
                await _sesv2Wrapper.CreateContactAsync(emailAddress,
 _contactListName);
                Console.WriteLine($"Contact {emailAddress} added to the
 {_contactListName} contact list.");
            }
            catch (AlreadyExistsException)
            {
                Console.WriteLine($"Contact {emailAddress} already exists in the
 {_contactListName} contact list.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error creating contact {emailAddress}:
 {ex.Message}");
                return false;
            }

            // Send a welcome email to the new contact.
            try
            {
                string subject = "Welcome to the Weekly Coupons Newsletter";
```

```
                string htmlContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _htmlWelcomeFile);
                string textContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _textWelcomeFile);

                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress, new
List<string> { emailAddress }, subject, htmlContent, textContent);
                Console.WriteLine($"Welcome email sent to {emailAddress}.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending welcome email to
{emailAddress}: {ex.Message}");
                return false;
            }

            // Wait 2 seconds before sending the next email (if the account is in
the SES Sandbox).
            await Task.Delay(2000);
        }

        return true;
    }

    /// <summary>
    ///  Send the coupon newsletter to the subscribers in the contact list.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> SendCouponNewsletter(string fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("3. In this step, we will send the coupon newsletter:"
+
                          "\r\n  - Retrieve the list of contacts from the contact
list." +
                          "\r\n  - Send the coupon newsletter using the email
template to each contact.\r\n");


        // Retrieve the list of contacts from the contact list.
        var contacts = await _sesv2Wrapper.ListContactsAsync(_contactListName);
        if (!contacts.Any())
```

```
        {
            Console.WriteLine($"No contacts found in the {_contactListName}
contact list.");
            return false;
        }

        // Load the coupon data from the sample_coupons.json file.
        string couponsData = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _couponsDataFile);

        // Send the coupon newsletter to each contact using the email template.
        try
        {
            foreach (var contact in contacts)
            {
                // To use the Contact List for list management, send to only one
address at a time.
                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress,
                    new List<string> { contact.EmailAddress },
                    null, null, null, _templateName, couponsData,
_contactListName);
            }

            Console.WriteLine($"Coupon newsletter sent to contact list
{_contactListName}.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending coupon newsletter to contact list
{_contactListName}: {ex.Message}");
            return false;
        }

        return true;
    }

    /// <summary>
    /// Provide instructions for monitoring sending activity and metrics.
    /// </summary>
    /// <param name="interactive">True to run in interactive mode.</param>
    /// <returns>True if successful.</returns>
    public static bool MonitorAndReview(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("4. In step 4, we will monitor and review:" +
                          "\r\n  - Provide instructions for the user to review
the sending activity and metrics in the AWS console.\r\n");

        Console.WriteLine("Review your sending activity using the SES Homepage in
the AWS console.");
        Console.WriteLine("Press Enter to open the SES Homepage in your default
browser...");
        if (interactive)
        {
            Console.ReadLine();
            try
            {
                // Open the SES Homepage in the default browser.
                Process.Start(new ProcessStartInfo
                {
                    FileName = "https://console.aws.amazon.com/ses/home",
                    UseShellExecute = true
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error opening the SES Homepage:
{ex.Message}");
                return false;
            }
        }

        Console.WriteLine("Review the sending activity and email metrics, then
press Enter to continue...");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Clean up the resources used in the scenario.
    /// </summary>
    /// <param name="verifiedEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <param name="interactive">True if interactive.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Cleanup(string verifiedEmailAddress, bool
interactive)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("5. Finally, we clean up resources:" +
                          "\r\n  - Delete the contact list (which also deletes
 all contacts within it)." +
                          "\r\n  - Delete the email template." +
                          "\r\n  - Optionally delete the verified email identity.
\r\n");

        Console.WriteLine("Cleaning up resources...");

        // Delete the contact list (this also deletes all contacts in the list).
        try
        {
            await _sesv2Wrapper.DeleteContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Contact list {_contactListName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting contact list {_contactListName}:
 {ex.Message}");
            return false;
        }

        // Delete the email template.
        try
        {
            await _sesv2Wrapper.DeleteEmailTemplateAsync(_templateName);
            Console.WriteLine($"Email template {_templateName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Email template {_templateName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting email template {_templateName}:
 {ex.Message}");
            return false;
        }
```

```
        // Ask the user if they want to delete the email identity.
        var deleteIdentity = !interactive ||
            GetYesNoResponse(
                $"Do you want to delete the email identity
 {verifiedEmailAddress}? (y/n) ");
        if (deleteIdentity)
        {
            try
            {
                await
 _sesv2Wrapper.DeleteEmailIdentityAsync(verifiedEmailAddress);
                Console.WriteLine($"Email identity {verifiedEmailAddress}
 deleted.");
            }
            catch (NotFoundException)
            {
                Console.WriteLine(
                    $"Email identity {verifiedEmailAddress} not found.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"Error deleting email identity {verifiedEmailAddress}:
 {ex.Message}");
                return false;
            }
        }
        else
        {
            Console.WriteLine(
                $"Skipping deletion of email identity {verifiedEmailAddress}.");
        }

        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
```

```
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Simple check to verify a string is an email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email.</returns>
    private static bool IsEmail(string? email)
    {
        if (string.IsNullOrEmpty(email))
            return false;
        return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$",
 RegexOptions.IgnoreCase);
    }
}
```

Wrapper für Service-Vorgänge.

```
using System.Net;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;

namespace Sesv2Scenario;

/// <summary>
/// Wrapper class for Amazon Simple Email Service (SES) v2 operations.
/// </summary>
public class SESv2Wrapper
{

    private readonly IAmazonSimpleEmailServiceV2 _sesClient;

    /// <summary>
    /// Constructor for the SESv2Wrapper.
    /// </summary>
    /// <param name="sesClient">The injected SES v2 client.</param>
```

```
    public SESv2Wrapper(IAmazonSimpleEmailServiceV2 sesClient)
    {
        _sesClient = sesClient;
    }

    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
 contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
 exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
```

```
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
```

```
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
 exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
 exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
```

```
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
 list: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
```

```
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 identity: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
```

```
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };
```

```
        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }

    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email
from.</param>
    /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
(optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
    /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
```

```
   public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
               }
           };
       }

       if (!string.IsNullOrEmpty(contactListName))
```

```
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
```

```
        }

        return string.Empty;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
```

```
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
```

```
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
```

```
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }

    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
 verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please
 remove some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }

    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");
```

```
     CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
         .templateName(TEMPLATE_NAME)
         .templateContent(EmailTemplateContent.builder()
             .subject("Weekly Coupons Newsletter")
             .html(newsletterHtml)
             .text(newsletterText)
             .build())
         .build();

     sesClient.createEmailTemplate(templateRequest);

     System.out.println("Email template created: " + TEMPLATE_NAME);
   } catch (AlreadyExistsException e) {
     // If the template already exists, skip this step and proceed with the next
     // operation
     System.out.println("Email template already exists, skipping creation...");
   } catch (LimitExceededException e) {
     // If the limit for email templates is exceeded, fail the workflow and
inform
     // the user
     System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
     throw e;
   } catch (Exception e) {
     System.err.println("Error occurred while creating email template: " +
e.getMessage());
     throw e;
   }

   try {
     // Delete the contact list
     DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
         .contactListName(CONTACT_LIST_NAME)
         .build();

     sesClient.deleteContactList(deleteContactListRequest);

     System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
   } catch (NotFoundException e) {
     // If the contact list does not exist, log the error and proceed
     System.out.println("Contact list not found. Skipping deletion...");
```

```
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }

      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
```

```
        e.printStackTrace();
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e

            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
```

```
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e

        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
```

```
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )

        try:

self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e

        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
```

```
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e

        try:

self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)

            try:

self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
exist.")
                else:
                    print(e)

        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateContact

- CreateContactList
- CreateEmailIdentity
- CreateEmailTemplate
- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .create_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateContactListError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Contact list already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating contact list: {}", e)),
            },
```

```
        }

            match self
                .client
                .create_contact()
                .contact_list_name(CONTACT_LIST_NAME)
                .email_address(email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Contact created for {}", email)?,
                Err(e) => match e.into_service_error() {
                    CreateContactError::AlreadyExistsException(_) => writeln!(
                        self.stdout,
                        "Contact already exists for {}, skipping creation.",
                        email
                    )?,
                    e => return Err(anyhow!("Error creating contact for {}: {}",
 email, e)),
                },
            }

        let contacts: Vec<Contact> = match self
            .client
            .list_contacts()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(list_contacts_output) => {
                list_contacts_output.contacts.unwrap().into_iter().collect()
            }
            Err(e) => {
                return Err(anyhow!(
                    "Error retrieving contact list {}: {}",
                    CONTACT_LIST_NAME,
                    e
                ))
            }
        };

            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
```

```
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

.destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
{}", email, e)),
            }

        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
```

```
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }

        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
```

```
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }

        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }

            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }

        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
```

```
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Codebeispiele für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Simple Notification Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon SNS – Weitere Informationen zu Amazon SNS.
- API-Referenz für Amazon SNS – Details zu allen verfügbaren Amazon-SNS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon SNS AWS SDKs
  - Hello Amazon SNS
  - Aktionen für Amazon SNS mit AWS SDKs
    - Verwenden Sie CheckIfPhoneNumberIsOptedOutmit einem AWS SDK oder CLI
    - Verwenden Sie ConfirmSubscriptionmit einem AWS SDK oder CLI
    - Verwenden Sie CreateTopicmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteTopicmit einem AWS SDK oder CLI
    - Verwenden Sie GetSMSAttributes mit einem AWS SDK oder CLI
    - Verwenden Sie GetTopicAttributesmit einem AWS SDK oder CLI
    - Verwenden Sie ListPhoneNumbersOptedOutmit einem AWS SDK oder CLI
    - Verwenden Sie ListSubscriptionsmit einem AWS SDK oder CLI
    - Verwenden Sie ListTopicsmit einem AWS SDK oder CLI
    - Verwenden Sie Publish mit einem AWS SDK oder CLI
    - Verwenden Sie SetSMSAttributes mit einem AWS SDK oder CLI
    - Verwenden Sie SetSubscriptionAttributesmit einem AWS SDK oder CLI
    - Verwenden Sie SetSubscriptionAttributesRedrivePolicymit einem AWS SDK
    - Verwenden Sie SetTopicAttributesmit einem AWS SDK oder CLI
    - Verwenden Sie Subscribe mit einem AWS SDK oder CLI
    - Verwenden Sie TagResourcemit einem AWS SDK oder CLI
    - Verwenden Sie Unsubscribe mit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon SNS AWS SDKs
  - Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
  - Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
  - Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
  - Erstellen Sie einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen mit einem AWS SDK

- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Veröffentlichen Sie SMS-Nachrichten zu einem Amazon SNS SNS-Thema mithilfe eines AWS SDK
- Veröffentlichen Sie mit Amazon S3 eine große Nachricht in Amazon SNS mithilfe eines AWS SDK
- Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem AWS SDK
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

- Serverless-Beispiele für Amazon SNS
  - Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Grundlegende Beispiele für die Verwendung von Amazon SNS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Notification Service mit AWS SDKs verwendet werden.

Beispiele

- Hello Amazon SNS
- Aktionen für Amazon SNS mit AWS SDKs
  - Verwenden Sie CheckIfPhoneNumberIsOptedOutmit einem AWS SDK oder CLI
  - Verwenden Sie ConfirmSubscriptionmit einem AWS SDK oder CLI
  - Verwenden Sie CreateTopicmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTopicmit einem AWS SDK oder CLI

- Verwenden Sie GetSMSAttributes mit einem AWS SDK oder CLI
- Verwenden Sie GetTopicAttributesmit einem AWS SDK oder CLI
- Verwenden Sie ListPhoneNumbersOptedOutmit einem AWS SDK oder CLI
- Verwenden Sie ListSubscriptionsmit einem AWS SDK oder CLI
- Verwenden Sie ListTopicsmit einem AWS SDK oder CLI
- Verwenden Sie Publish mit einem AWS SDK oder CLI
- Verwenden Sie SetSMSAttributes mit einem AWS SDK oder CLI
- Verwenden Sie SetSubscriptionAttributesmit einem AWS SDK oder CLI
- Verwenden Sie SetSubscriptionAttributesRedrivePolicymit einem AWS SDK
- Verwenden Sie SetTopicAttributesmit einem AWS SDK oder CLI
- Verwenden Sie Subscribe mit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie Unsubscribe mit einem AWS SDK oder CLI

Hello Amazon SNS

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon SNS beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
```

```
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)
```

```
# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_sns.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
```

```
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
```

```
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Einzelheiten zur API finden Sie ListTopicsunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen SNS-Client und listen Sie Themen in Ihrem Konto auf.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});
```

```
  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
```

```
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Package.swift Akte.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
```

```
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Das Hauptprogramm von Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
```

```
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}
```

```
/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Swift.

Aktionen für Amazon SNS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon SNS SNS-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-SNS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon SNS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz zu Amazon Simple Notification Service.

Beispiele

- Verwenden Sie CheckIfPhoneNumberIsOptedOutmit einem AWS SDK oder CLI
- Verwenden Sie ConfirmSubscriptionmit einem AWS SDK oder CLI
- Verwenden Sie CreateTopicmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTopicmit einem AWS SDK oder CLI
- Verwenden Sie GetSMSAttributes mit einem AWS SDK oder CLI
- Verwenden Sie GetTopicAttributesmit einem AWS SDK oder CLI

- Verwenden Sie ListPhoneNumbersOptedOutmit einem AWS SDK oder CLI
- Verwenden Sie ListSubscriptionsmit einem AWS SDK oder CLI
- Verwenden Sie ListTopicsmit einem AWS SDK oder CLI
- Verwenden Sie Publish mit einem AWS SDK oder CLI
- Verwenden Sie SetSMSAttributes mit einem AWS SDK oder CLI
- Verwenden Sie SetSubscriptionAttributesmit einem AWS SDK oder CLI
- Verwenden Sie SetSubscriptionAttributesRedrivePolicymit einem AWS SDK
- Verwenden Sie SetTopicAttributesmit einem AWS SDK oder CLI
- Verwenden Sie Subscribe mit einem AWS SDK oder CLI
- Verwenden Sie TagResourcemit einem AWS SDK oder CLI
- Verwenden Sie Unsubscribe mit einem AWS SDK oder CLI

Verwenden Sie **`CheckIfPhoneNumberIsOptedOut`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CheckIfPhoneNumberIsOptedOut` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out.
/// </summary>
public class IsPhoneNumOptedOut
{
```

```
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
```

```
    }
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So überprüfen Sie SMS-Nachrichten-Abmeldungen für eine Telefonnummer

Im folgenden `check-if-phone-number-is-opted-out` Beispiel wird geprüft, ob die angegebene Telefonnummer den Empfang von SMS-Nachrichten vom AWS Girokonto deaktiviert hat.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Ausgabe:

```
{
    "isOptedOut": false
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }
```

```
    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CheckIfPhoneNumberIsOptedOutin der AWS SDK für PHP API-Referenz.

Verwenden Sie **`ConfirmSubscription`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ConfirmSubscription` verwendet wird.

CLI

AWS CLI

So bestätigen Sie ein Abonnement

Mit dem folgenden `confirm-subscription`-Befehl wird der Bestätigungsvorgang abgeschlossen, der gestartet wurde, als Sie ein SNS-Thema mit dem Namen `my-topic` abonniert haben. Der --token-Parameter stammt aus der Bestätigungsnachricht, die an den im Abonnementaufruf angegebenen Benachrichtigungsendpunkt gesendet wurde.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Ausgabe:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
```

```
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ConfirmSubscriptionin der AWS SDK für PHP API-Referenz.

Verwenden Sie **`CreateTopic`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTopic` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten

- Erstellen und veröffentlichen zu einem FIFO-Thema
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
```

```
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };
```

```
        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
```

```
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein SNS-Thema

Das folgende `create-topic`-Beispiel erstellt ein SNS-Thema namens `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Ausgabe:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Weitere Informationen finden Sie unter Verwenden der AWS Befehlszeilenschnittstelle mit Amazon SQS und Amazon SNS im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- Einzelheiten zur API finden Sie unter CreateTopic AWS CLIBefehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
```

```
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
```

```
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
 {e}")
            raise
```

- Einzelheiten zur API finden Sie CreateTopicin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
```

```
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- Einzelheiten zur API finden Sie CreateTopicin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteTopic`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTopic` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
- Erste Schritte mit dem Konfigurationsmanagement
- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-ARN.

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
```

```
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie das SNS-Thema

Das folgende `delete-topic`-Beispiel löscht die angegebene SNS-Thema.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie DeleteTopicin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
```

```
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }
```

```
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
```

```
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
```

```
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- Einzelheiten zur API finden Sie DeleteTopicin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- Einzelheiten zur API finden Sie DeleteTopicin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSMSAttributes` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

So führen Sie die Standard-SMS-Nachrichtenattribute auf

Das folgende `get-sms-attributes`-Beispiel führt die Standardattribute für das Senden von SMS-Nachrichten auf.

```
aws sns get-sms-attributes
```

Ausgabe:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS CLI - Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
```

```
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
```

```
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für JavaScript -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für PHP - API-Referenz.

Verwenden Sie **`GetTopicAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetTopicAttributes` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
```

```
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die Attribute eines Themas ab

Im folgenden Beispiel für `get-topic-attributes` werden die Attribute für das angegebene Thema angezeigt.

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " +
 topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
 topicArn) {
        try {
            GetTopicAttributesRequest request =
 GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();
```

```
// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für PHP API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
 oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListPhoneNumbersOptedOut`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListPhoneNumbersOptedOut` verwendet wird.

CLI

AWS CLI

So führen Sie Abmeldungen für SMS-Nachrichten auf

Das folgende `list-phone-numbers-opted-out`-Beispiel listet die Telefonnummern auf, bei denen der Empfang von SMS-Nachrichten abbestellt wurde.

```
aws sns list-phone-numbers-opted-out
```

Ausgabe:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
```

```
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK for Java 2.x API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListPhoneNumbersOptedOutin der AWS SDK für PHP API-Referenz.

Verwenden Sie **`ListSubscriptions`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListSubscriptions` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example will retrieve a list of the existing Amazon Simple
```

```
    /// Notification Service (Amazon SNS) subscriptions.
    /// </summary>
    public class ListSubscriptions
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
 specific topic, " +
                              "or press Enter to list subscriptions for all
 topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
 topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
```

```
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
```

```
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Abonnements auf

Im folgenden `list-subscriptions` Beispiel wird eine Liste der SNS-Abonnements in Ihrem AWS Konto angezeigt.

```
aws sns list-subscriptions
```

Ausgabe:

```
{
    "Subscriptions": [
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
```

```
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
 returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Einzelheiten zur API finden Sie ListSubscriptionsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
```

```
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTopics`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTopics` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit dem Geräteschutz für das Internet der Dinge

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
```

```
        public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So führen Sie Ihre SNS-Themen auf

Das folgende `list-topics` Beispiel listet alle SNS-Themen in Ihrem AWS Konto auf.

```
aws sns list-topics
```

Ausgabe:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
```

```
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Einzelheiten zur API finden Sie ListTopicsin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
 testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
```

```
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

- Einzelheiten zur API finden Sie ListTopicsin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`Publish`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Publish` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen einer SMS-Nachricht
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

```
    using System;
    using System.Threading.Tasks;
```

```
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
 {response.MessageId}");
```

```
        }
    }
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");
```

```
                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }
```

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
```

```
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                        "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
```

```
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

Beispiel 1: So veröffentlichen Sie eine Nachricht für ein Thema

Das folgende `publish`-Beispiel veröffentlicht die angegebene Nachricht im angegebenen SNS-Thema. Die Nachricht stammt aus einer Textdatei, in der Sie Zeilenumbrüche einfügen können.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

Inhalt von `message.txt`:

```
Hello World
Second Line
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Beispiel 2: So veröffentlichen Sie eine SMS-Nachricht an eine Telefonnummer

Im folgenden Beispiel für `publish` wird Nachricht `Hello world!` an Telefonnummer `+1-555-555-0100` veröffentlicht.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Ausgabe:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
```

```
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
```

```
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                    if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
```

```
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für PHP -API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Weitere API-Informationen finden Sie unter Publish in der AWS -Tools für PowerShell -Cmdlet-Referenz (V4).

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt das Veröffentlichen einer Nachricht mit einer einzigen MessageAttribute deklarierten Inline.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
```

```
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere Nachrichten im Voraus MessageAttributes deklariert wurden.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Weitere API-Informationen finden Sie unter Publish in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
 is
                                sent to subscribers that have protocols that are
 not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
 subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
```

```
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
 MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
```

```
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            logger.error(f"Error publishing message to topic: {error_code} -
 {e}")
            raise
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zu AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
```

```
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message
 content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);
```

```
    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
```

```
          iv_message = iv_message ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

Veröffentlichen Sie eine Nachricht mit Attributen zu einem Thema.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
          it_messageattributes = it_msg_attrs ).
        MESSAGE 'Message with attributes published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

Veröffentlichen Sie eine Nachricht in mehreren Formaten zu einem Thema.

```
    " Build JSON message structure for multi-format message
    DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.

    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = lv_json_message
          iv_subject = iv_subject
          iv_messagestructure = 'json' ).
        MESSAGE 'Multi-format message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
        }

        print("Published message with ID \(messageId)")
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS -SDK für Swift.

Verwenden Sie **`SetSMSAttributes`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetSMSAttributes` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

So verwenden Sie Amazon SNS zum Festlegen des DefaultSMSType-Attributs.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für C++ - API-Referenz.

CLI

AWS CLI

So legen Sie SMS-Nachrichtenattribute fest

Im folgenden Beispiel für `set-sms-attributes` wird die standardmäßige Absender-ID für SMS-Nachrichten auf `MyName` festgelegt.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS CLI - Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK for Java 2.x - API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
 messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für JavaScript -API-Referenz.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für PHP - API-Referenz.

Verwenden Sie **`SetSubscriptionAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetSubscriptionAttributes` verwendet wird.

CLI

AWS CLI

So legen Sie Abonnementattribute fest

Im folgenden Beispiel für `set-subscription-attributes` wird das `RawMessageDelivery`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden Beispiel für `set-subscription-attributes` wird ein `FilterPolicy`-Attribut auf ein SQS-Abonnement festgelegt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Im folgenden Beispiel für `set-subscription-attributes` wird das `FilterPolicy`-Attribut von einem SQS-Abonnement entfernt.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>
```

```
                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
```

```
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
```

```
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->setsubscriptionattributes(
            iv_subscriptionarn = iv_subscription_arn
            iv_attributename  = 'FilterPolicy'
```

```
            iv_attributevalue = iv_filter_policy ).
        MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SetSubscriptionAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SetSubscriptionAttributesRedrivePolicy`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
```

```
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Verwenden Sie **`SetTopicAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetTopicAttributes` verwendet wird.

CLI

AWS CLI

So legen Sie ein Attribut für ein Thema fest

Im folgenden Beispiel für `set-topic-attributes` wird das `DisplayName`-Attribute für das angegebene Thema festgelegt.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
```

```
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für PHP API-Referenz.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SetTopicAttributesin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie SetTopicAttributesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`Subscribe`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Subscribe` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
- Erstellen und veröffentlichen zu einem FIFO-Thema
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::SNS::SNSClient snsClient(clientConfiguration);

           Aws::SNS::Model::SubscribeRequest request;
           request.SetTopicArn(topicARN);
           request.SetProtocol("sqs");
           request.SetEndpoint(queueARN);

           Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

           if (outcome.IsSuccess()) {
               Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
               std::cout << "The queue '" << queueName
                         << "' has been subscribed to the topic '"
                         << "'" << topicName << "'" << std::endl;
               std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                         << std::endl;
               subscriptionARNS.push_back(subscriptionARN);
           }
           else {
               std::cerr << "Error with TopicsAndQueues::Subscribe. "
                         << outcome.GetError().GetMessage()
                         << std::endl;

               cleanUp(topicARN,
                       queueURLS,
                       subscriptionARNS,
                       snsClient,
                       sqsClient);

               return false;
           }
```

Abonnieren Sie ein Thema mit einem Filter.

```
       static const Aws::String TONE_ATTRIBUTE("tone");
       static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                      "sincere"};
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;
```

```
                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
                                << "will be added to this subscription." <<
 std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
```

```
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
```

```
        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So abonnieren Sie ein Thema

Der folgende `subscribe`-Befehl abonniert das angegebene Thema mit eine E-Mail-Adresse.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Ausgabe:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS CLI -Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
```

```
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für Go -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie einen HTTP-Endpunkt für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();
```

```
            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
```

```
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
```

```
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für JavaScript -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
```

```
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Abonnieren Sie einen HTTP-Endpunkt für ein Thema.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
```

```
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zu AWS SDK für Python (Boto3).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'aws-sdk-sns'
require 'logger'

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info('Subscription created successfully.')
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
```

```
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = 'email'
  endpoint = 'EMAIL_ADDRESS' # Should be replaced with a real email address
  topic_arn = 'TOPIC_ARN'    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info('Creating the subscription.')
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error('Subscription creation failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für Ruby -API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);
```

```
    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
    TRY.
        oo_result = lo_sns->subscribe(                      "oo_result is
returned for testing purposes."
```

```
                iv_topicarn = iv_topic_arn
                iv_protocol = 'email'
                iv_endpoint = iv_email_address
                iv_returnsubscriptionarn = abap_true ).
        MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )
```

```
        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnieren Sie eine Telefonnummer für ein Thema, um Benachrichtigungen per SMS zu erhalten.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS -SDK für Swift.

Verwenden Sie **`TagResource`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TagResource` verwendet wird.

CLI

AWS CLI

So fügen Sie einem Thema ein Tag hinzu

Das folgende `tag-resource`-Beispiel fügt dem angegebenen Amazon-SNS-Thema ein Metadaten-Tag hinzu.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie TagResourcein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);
```

```
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
```

```
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Einzelheiten zur API finden Sie TagResourcein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`Unsubscribe`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `Unsubscribe` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

```
    /// <summary>
```

```
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
```

```
    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für C++ -API-Referenz.

CLI

AWS CLI

So melden Sie sich von einem Thema ab

Im folgenden Beispiel für `unsubscribe` wird das angegebene Abonnement aus einem Thema gelöscht.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS CLI -Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK for Java 2.x -API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für JavaScript - API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.
```

```
        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zu AWS SDK für Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
       MESSAGE 'Subscription deleted.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Subscription does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snsinvalidparameterex.
       MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
   ENDTRY.
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS - SDK für Swift.

Szenarien für die Verwendung von Amazon SNS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon SNS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon SNS oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle
- Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
- Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten
- Erstellen Sie einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen mit einem AWS SDK
- Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Erste Schritte mit dem Konfigurationsmanagement

- Erste Schritte mit dem Geräteschutz für das Internet der Dinge
- Veröffentlichen Sie SMS-Nachrichten zu einem Amazon SNS SNS-Thema mithilfe eines AWS SDK
- Veröffentlichen Sie mit Amazon S3 eine große Nachricht in Amazon SNS mithilfe eines AWS SDK
- Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem AWS SDK
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK
- Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion
- Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Erstellen Sie eine Anwendung zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Code-Beispiele zeigen, wie man eine Anwendung erstellt, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

Java

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt

Die folgende Code-Beispiele zeigen, wie man eine Anwendung erstellt, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

.NET

SDK für .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Java

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung des Beispiels, das die Java Async API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Kotlin

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen Sie ein Messaging-Thema und veröffentlichen Sie Nachrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon SNS-Thema.
- Abonnieren Sie das Thema über einen E-Mail-Endpunkt
- Bestätigen Sie Ihr Abonnement
- Veröffentlichen Sie eine Nachricht zum Thema
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SNS Getting Started Script
# This script demonstrates how to create an SNS topic, subscribe to it, publish a
 message,
# and clean up resources.

set -euo pipefail

# Set up logging with secure file permissions
LOG_FILE="sns-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon SNS Getting Started Tutorial..."
echo "$(date)"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?
```

```
    if [ -n "${SUBSCRIPTION_ARN:-}" ] && [ "$SUBSCRIPTION_ARN" != "pending
 confirmation" ] && [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ]; then
        echo "Deleting subscription: $SUBSCRIPTION_ARN"
        if ! aws sns unsubscribe --subscription-arn "$SUBSCRIPTION_ARN" --region
 "$AWS_REGION" 2>/dev/null; then
            echo "Warning: Failed to delete subscription" >&2
        fi
    fi

    if [ -n "${TOPIC_ARN:-}" ]; then
        echo "Deleting topic: $TOPIC_ARN"
        if ! aws sns delete-topic --topic-arn "$TOPIC_ARN" --region "$AWS_REGION"
 2>/dev/null; then
            echo "Warning: Failed to delete topic" >&2
        fi
    fi

    return $exit_code
}

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Set trap to cleanup on exit
trap cleanup_resources EXIT

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random topic name suffix using secure method
RANDOM_SUFFIX=$(openssl rand -hex 4)
```

```
TOPIC_NAME="my-topic-${RANDOM_SUFFIX}"

# Validate topic name length (max 256 characters)
if [ ${#TOPIC_NAME} -gt 256 ]; then
    handle_error "Topic name exceeds maximum length of 256 characters"
fi

# Step 1: Create an SNS topic
echo "Creating SNS topic: $TOPIC_NAME"
TOPIC_RESULT=$(aws sns create-topic \
    --name "$TOPIC_NAME" \
    --region "$AWS_REGION" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
    --output json) || handle_error "Failed to create SNS topic"

# Extract the topic ARN using jq for reliable parsing
TOPIC_ARN=$(echo "$TOPIC_RESULT" | jq -r '.TopicArn // empty') || handle_error
 "Failed to parse topic result"

if [ -z "$TOPIC_ARN" ]; then
    handle_error "Failed to extract topic ARN from result: $TOPIC_RESULT"
fi

# Validate ARN format
if [[ ! "$TOPIC_ARN" =~ ^arn:aws:sns:[a-z0-9-]+:[0-9]{12}:[a-zA-Z0-9_-]+$ ]];
 then
    handle_error "Invalid SNS topic ARN format: $TOPIC_ARN"
fi

echo "Successfully created topic with ARN: $TOPIC_ARN"

# Step 2: Subscribe to the topic using Email-JSON protocol to reduce costs
echo ""
echo "==============================================="
echo "EMAIL SUBSCRIPTION"
echo "==============================================="
EMAIL_ADDRESS="test-${RANDOM_SUFFIX}@example.com"

# Validate email format (basic validation)
if [[ ! "$EMAIL_ADDRESS" =~ ^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];
 then
    handle_error "Invalid email format: $EMAIL_ADDRESS"
fi
```

```
echo "Subscribing email: $EMAIL_ADDRESS to topic using Email-JSON protocol"
SUBSCRIPTION_RESULT=$(aws sns subscribe \
    --topic-arn "$TOPIC_ARN" \
    --protocol email-json \
    --notification-endpoint "$EMAIL_ADDRESS" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to create subscription"

# Extract the subscription ARN using jq
SUBSCRIPTION_ARN=$(echo "$SUBSCRIPTION_RESULT" | jq -r '.SubscriptionArn //
 empty') || handle_error "Failed to parse subscription result"

echo "Subscription created: $SUBSCRIPTION_ARN"
echo "A confirmation email has been sent to $EMAIL_ADDRESS"
echo ""

# Tag the subscription
if [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ] && [ "$SUBSCRIPTION_ARN" !=
 "pending confirmation" ]; then
    aws sns tag-resource \
        --resource-arn "$SUBSCRIPTION_ARN" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
        --region "$AWS_REGION" 2>/dev/null || echo "Warning: Failed to tag
 subscription"
fi

# Step 3: List subscriptions to verify
echo "Listing subscriptions for topic: $TOPIC_ARN"
SUBSCRIPTIONS=$(aws sns list-subscriptions-by-topic --topic-arn "$TOPIC_ARN"
 --region "$AWS_REGION" --output json) || handle_error "Failed to list
 subscriptions"

echo "Current subscriptions:"
echo "$SUBSCRIPTIONS" | jq '.'

# Get the confirmed subscription ARN with optimized jq query and improved error
 handling
CONFIRMED_SUBSCRIPTION=$(echo "$SUBSCRIPTIONS" | jq -r '.Subscriptions[]? |
 select(.SubscriptionArn != "PendingConfirmation") | .SubscriptionArn' 2>/dev/
null | head -n 1)

if [ -n "$CONFIRMED_SUBSCRIPTION" ]; then
```

```
    SUBSCRIPTION_ARN="$CONFIRMED_SUBSCRIPTION"
else
    echo "Warning: No confirmed subscription found. You may not have confirmed
 the subscription yet."
    echo "The script will continue, but you may not receive the test message."
fi

# Step 4: Publish a message to the topic
echo ""
echo "Publishing a test message to the topic"
MESSAGE="Hello from Amazon SNS! This is a test message sent at $(date)."

# Validate message length (max 256 KB for SNS)
if [ ${#MESSAGE} -gt 262144 ]; then
    handle_error "Message exceeds maximum size of 256 KB"
fi

PUBLISH_RESULT=$(aws sns publish \
    --topic-arn "$TOPIC_ARN" \
    --message "$MESSAGE" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to publish message"

MESSAGE_ID=$(echo "$PUBLISH_RESULT" | jq -r '.MessageId // empty') ||
 handle_error "Failed to parse publish result"

if [ -z "$MESSAGE_ID" ]; then
    handle_error "No message ID returned from publish operation"
fi

# Validate message ID format (UUID v4)
if [[ ! "$MESSAGE_ID" =~ ^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-
f0-9]{12}$ ]]; then
    handle_error "Unexpected message ID format: $MESSAGE_ID"
fi

echo "Message published successfully with ID: $MESSAGE_ID"
echo "Check your email for the message."

# Pause to allow the user to check their email
echo ""
echo "Pausing for 3 seconds to allow message delivery..."
sleep 3
```

```
# Step 5: Clean up resources
echo ""
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Resources created:"
echo "- SNS Topic: $TOPIC_ARN"
echo "- Subscription: ${SUBSCRIPTION_ARN:-N/A}"
echo ""
echo "Cleaning up resources to avoid unnecessary charges..."

echo ""
echo "Tutorial completed successfully!"
echo "$(date)"
echo "==============================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateTopic
  - DeleteTopic
  - ListSubscriptionsByTopic
  - Veröffentlichen
  - Abonnieren
  - Unsubscribe

Erstellen Sie einen Plattformendpunkt für Amazon SNS SNS-Push-Benachrichtigungen mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie ein Plattformendpunkt für Amazon-SNS-Push-Benachrichtigungen erstellt werden.

CLI

AWS CLI

So erstellen Sie ein Plattformanwendungsendpunkt

Im folgenden Beispiel für `create-platform-endpoint` wird mithilfe des angegebenen Tokens ein Endpunkt für die angegebene Plattformanwendung erstellt.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Ausgabe:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
```

```
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
```

```
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS

- Amazon Textract

Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie man ein FIFO-Amazon-SNS-Thema erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
```

```
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
```

```
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
```

```
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
```

```
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")
```

```
    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
```

```
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
```

```
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
```

```
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
```

```
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'

 iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
```

```
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
```

```
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=========================================="
    echo "CREATED RESOURCES"
    echo "=========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
```

```
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Erste Schritte mit dem Geräteschutz für das Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- IoT Device Defender Defender-Audit-Checks aktivieren
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Abhilfemaßnahme
- Wenden Sie Maßnahmen zur Risikominderung auf die Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- IoT-Protokollierung aktivieren

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "==================================================="
```

```
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
```

```
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true
```

```
        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
```

```
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
```

```
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
```

```
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "======================================================"
echo "Step 1: Creating required IAM roles"
echo "======================================================"

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
```

```
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)
```

```
if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "==================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "==================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi
```

```
echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
echo "Step 3: Running an on-demand audit"
echo "==================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
```

```
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
```

```
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "==================================================="
echo "Step 4: Creating a mitigation action"
echo "==================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
```

```
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
```

```
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="
```

```
# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)
```

```
    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "==================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "==================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
```

```
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
```

```
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
```

```
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
```

```
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics

- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Veröffentlichen Sie SMS-Nachrichten zu einem Amazon SNS SNS-Thema mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.
- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }
```

```
    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen Sie mit Amazon S3 eine große Nachricht in Amazon SNS mithilfe eines AWS SDK

Das folgende Code-Beispiel zeigt, wie man mit Amazon S3 eine große Nachricht an Amazon SNS veröffentlicht, um die Nachrichtennutzlast zu speichern.

Java

SDK für Java 1.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um eine große Nachricht zu veröffentlichen, verwenden Sie die Amazon SNS Extended Client Library für Java. Die Nachricht, die Sie senden, verweist auf ein Amazon S3-Objekt, das den tatsächlichen Nachrichteninhalt enthält.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will
 be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
 AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
 AmazonSQSClientBuilder.standard().withRegion(region).build();
```

```
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
```

```
                final String message = "This message is stored in S3 as it
 exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
 = new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

Veröffentlichen Sie eine Amazon SNS SNS-SMS-Textnachricht mit einem AWS SDK

In den folgenden Codebeispielen wird veranschaulicht, wie Sie SMS-Nachrichten über Amazon SNS veröffentlichen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
```

```
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };
```

```
            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für .NET -API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
```

```
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK for Java 2.x -API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
```

```
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für PHP -API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zu AWS SDK für Python (Boto3).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_phone_number = '+12065550101' - Phone number in E.164 format
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_phonenumber = iv_phone_number
         iv_message = iv_message ).
       MESSAGE 'Message published to phone number.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Phone number does not exist.' TYPE 'E'.
   ENDTRY.
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für SAP ABAP.

Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
```

```
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
```

```
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
```

```
                            $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                            $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
```

```
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
```

```
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
```

```
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
```

```
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }
```

```
        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
```

```
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
```

```
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;
```

```
    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
```

```
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
```

```
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                            $"\"Service\": " +
                                                "\"sns.amazonaws.com\"" +
                                               "}," +
                                        "\"Action\": \"sqs:SendMessage\"," +
                                        $"\"Resource\": \"{queueArn}\"," +
                                         "\"Condition\": {" +
                                              "\"ArnEquals\": {" +
                                                  $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                                "}" +
                                            "}" +
                                    "}]" +
                                "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
```

```
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
```

```
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
```

```
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
```

```
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
```

```
   printAsterisksLine();
   std::cout << "In this workflow, you will create an SNS topic and subscribe "
             << NUMBER_OF_QUEUES <<
             " SQS queues to the topic." << std::endl;
   std::cout
           << "You can select from several options for configuring the topic and
the subscriptions for the "
           << NUMBER_OF_QUEUES << " queues." << std::endl;
   std::cout << "You can then post to the topic and see the results in the
queues."
             << std::endl;

   Aws::SNS::SNSClient snsClient(clientConfiguration);

   printAsterisksLine();

   std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
             << std::endl;
   std::cout
           << "FIFO topics deliver messages in order and support deduplication
and message filtering."
           << std::endl;
   bool isFifoTopic = askYesNoQuestion(
           "Would you like to work with FIFO topics? (y/n) ");

   bool contentBasedDeduplication = false;
   Aws::String topicName;
   if (isFifoTopic) {
       printAsterisksLine();
       std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                 << std::endl;
       std::cout
               << "Deduplication IDs are either set in the message or
automatically generated "
               << "from content using a hash function." << std::endl;
       std::cout
               << "If a message is successfully published to an SNS FIFO topic,
any message "
               << "published and determined to have the same deduplication ID, "
               << std::endl;
       std::cout
               << "within the five-minute deduplication interval, is accepted
but not delivered."
```

```
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
```

```
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                        "true");
```

```
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
```

```
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
```

```
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;
```

```
                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
```

```
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
```

```
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
```

```
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);
```

```
                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

```
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
```

```
    "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
   contentBasedDeduplication = runner.questioner.AskBool(
    "\nDo you want to use content-based deduplication instead of entering a
  deduplication ID? (y/n) ", "y")
  }
  log.Println(strings.Repeat("-", 88))

  topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
  if isFifoTopic {
   topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
   log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
    "the topic name.", FIFO_SUFFIX)
  }

  topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
  contentBasedDeduplication)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
  \n"+
   "'%v' has been created.", topicName, topicArn)

  return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
  isFifoTopic bool) (string, string) {
  queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
  queue. ", ordinal))
  if isFifoTopic {
   queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
   if ordinal == "first" {
    log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
     "be appended to the queue name.\n", FIFO_SUFFIX)
   }
  }
  queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
  if err != nil {
   panic(err)
  }
  log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
   "'%v' has been created.", queueName, queueUrl)
```

```
 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
```

```
    for askAboutTones {
     toneIndex := runner.questioner.AskChoice(
      "Enter the number of the tone you want to filter by:\n", ToneChoices)
     toneSelections = append(toneSelections, ToneChoices[toneIndex])
     askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
  the filter? (y/n) ", "y")
    }
    log.Printf("Your subscription will be filtered to only pass the following
  tones: %v\n", toneSelections)
    filterPolicy = map[string][]string{TONE_KEY: toneSelections}
   }
  }

  subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
  filterPolicy)
  if err != nil {
   panic(err)
  }
  log.Printf("The queue %v is now subscribed to the topic %v with the subscription
  ARN %v.\n",
   queueName, topicName, subscriptionArn)

  return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
  string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
  var message string
  var groupId string
  var dedupId string
  var toneSelection string
  publishMore := true
  for publishMore {
   groupId = ""
   dedupId = ""
   toneSelection = ""
   message = runner.questioner.Ask("Enter a message to publish: ")
   if isFifoTopic {
    log.Println("Because you are using a FIFO topic, you must set a message group
  ID.\n" +
     "All messages within the same group will be received in the order they were
  published.")
    groupId = runner.questioner.Ask("Enter a message group ID: ")
    if !contentBasedDeduplication {
```

```
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
```

```
    log.Printf("One message was received by queue %v:\n", queueUrl)

   } else {
    log.Printf("%v messages were received by queue %v:\n", len(messages),
  queueUrl)
   }
   for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
     panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
   }

   if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
```

```
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
```

```
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SNS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
```

```
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

```
// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
```

```
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SQS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
```

```
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
```

```
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
```

```
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
```

```
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
```

```
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
```

```
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
```

```
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
 must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
```

```
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
```

```
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
```

```
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
```

```
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
```

```
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
```

```
                                        String msgAttValue,
                                        String duplication,
                                        String groupId,
                                        String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
```

```
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
```

```
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
```

```
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt für dieses Szenario.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Der obige Code stellt die erforderlichen Abhängigkeiten bereit und startet das Szenario. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
  }
```

```
  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
```

```
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });
```

```
    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
    const policy = JSON.stringify(
      {
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              Service: "sns.amazonaws.com",
            },
            Action: "sqs:SendMessage",
            Resource: queue.queueArn,
            Condition: {
              ArnEquals: {
                "aws:SourceArn": this.topicArn,
              },
            },
          },
        ],
      },
      null,
      2,
    );

    if (index !== 0) {
      this.logger.logSeparator();
    }

    await this.logger.log(MESSAGES.attachPolicyNotice);
    console.log(policy);
    const addPolicy = await this.prompter.confirm({
      message: MESSAGES.addPolicyConfirmation.replace(
        "${QUEUE_NAME}",
        queue.queueName,
```

```
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
```

```
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });
```

```
      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });
```

```
    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });
```

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
```

```
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns
```

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
```

```
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
```

```
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)
```

```
    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
```

```
            println("You can filter messages by one or more of the following
\"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
```

```
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }
```

```
        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
```

```
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
```

```
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")
```

```
            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
```

```
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
```

```
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
```

```
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe
{self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)
```

```
            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication
 and message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
 q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name.")
            print()

            print(self.DASHES)
            print("""
    Because you have chosen a FIFO topic, deduplication is supported.
    Deduplication IDs are either set in the message or automatically generated
    from content using a hash function.

    If a message is successfully published to an SNS FIFO topic, any message
    published and determined to have the same deduplication ID,
    within the five-minute deduplication interval, is accepted but not delivered.
```

```
    For more information about deduplication,
    see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a
deduplication ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)
```

```
            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
 attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
 self.topic_arn, queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
```

```
                print()
                print("For this example, you can filter messages by a TONE
attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
to the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")
```

```
        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based
deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")
```

```
                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
 len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i
+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message',
 message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
```

```
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
(y/n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
```

```
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Erstellen Sie Klassen, die Amazon SNS- und Amazon SQS SQS-Operationen für die Verwendung im Szenario zusammenfassen.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
```

```
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
            raise
```

```
    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
```

```
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
```

```
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} -
{e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
 [])])

            logger.info(f"Found {len(topics)} topics")
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
```

```
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
```

```
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
```

```
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
```

```
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
```

```
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM
permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
```

```
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
```

```
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }
```

```
        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
```

```
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
 attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the
 'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
```

```
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
 throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
 name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
 String? {
```

```
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
```

```
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.
```

```
        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "...")")
            }
        }
```

```
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues
 example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well
 as
              the queue subscriptions. You can then post messages to the topic
 and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
 region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
```

```
                  supported.

                  Deduplication IDs are either set in the message or are
 automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic,
 any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
 create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.
```

```
        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }
```

```
        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
```

```
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil
```

```
        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.
```

```
            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
 message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
 deduplication, you
                              must enter a deduplication ID. If other messages
 with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
 "Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
 = [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
 this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
 "String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }
```

```
            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.
```

```
        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine von Amazon API Gateway aufgerufene AWS Lambda Funktion erstellt wird.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der

Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie eine AWS Lambda Funktion erstellt wird, die durch ein von Amazon EventBridge geplantes Ereignis aufgerufen wird.

Java

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie so EventBridge , dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der JavaScript Lambda-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel einer geplanten EventBridge Amazon-Veranstaltung registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten für den späteren Abruf in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erzeugt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten auf ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele für Amazon SNS

Die folgenden Codebeispiele zeigen, wie Amazon SNS mit AWS SDKs verwendet wird.

Beispiele

- Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }
```

```
    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SNS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
```

```
  raise
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Codebeispiele für Amazon SQS mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Simple Queue Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Amazon-SQS-Entwicklerhandbuch – Weitere Informationen zu Amazon SQS.
- API-Referenz für Amazon SQS – Details zu allen verfügbaren Amazon-SQS-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon SQS AWS SDKs

- Hello Amazon SQS
- Aktionen für Amazon SQS mit AWS SDKs
  - Verwendung von AddPermission mit einer CLI
  - Verwenden Sie mit einem ChangeMessageVisibility AWS SDK oder CLI
  - Verwendung von ChangeMessageVisibilityBatch mit einer CLI
  - Verwenden Sie mit einem CreateQueue AWS SDK oder CLI
  - Verwenden Sie DeleteMessagemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteMessageBatchmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteQueuemit einem AWS SDK oder CLI
  - Verwenden Sie GetQueueAttributesmit einem AWS SDK oder CLI
  - Verwenden Sie GetQueueUrlmit einem AWS SDK oder CLI
  - Verwendung von ListDeadLetterSourceQueues mit einer CLI
  - Verwenden Sie mit einem ListQueues AWS SDK oder CLI
  - Verwendung von PurgeQueue mit einer CLI
  - Verwenden Sie mit einem ReceiveMessage AWS SDK oder CLI
  - Verwendung von RemovePermission mit einer CLI
  - Verwenden Sie mit einem SendMessage AWS SDK oder CLI
  - Verwenden Sie SendMessageBatchmit einem AWS SDK oder CLI
  - Verwenden Sie SetQueueAttributesmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon SQS AWS SDKs
  - Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft
  - Erstellen einer Messenger-Anwendung mit Step Functions
  - Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
  - Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK
  - Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
  - Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK
  - Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK

- Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK
- Senden und Empfangen von Nachrichtenstapeln mit Amazon SQS unter Verwendung eines AWS SDK
- Benutze die AWS Message Processing Framework für.NET zum Veröffentlichen und Empfangen von Amazon SQS SQS-Nachrichten
- Verwenden der Amazon SQS Java Messaging Library, um mit der Java-Message-Service(JMS)-Schnittstelle zu arbeiten
- Arbeiten Sie mit Warteschlangen-Tags und Amazon SQS mithilfe eines AWS SDK
- Serverless-Beispiele für Amazon SQS
  - Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser
  - Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Grundlegende Beispiele für die Verwendung von Amazon SQS AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Queue Service mit AWS SDKs verwendet werden können.

Beispiele

- Hello Amazon SQS
- Aktionen für Amazon SQS mit AWS SDKs
  - Verwendung von AddPermission mit einer CLI
  - Verwenden Sie mit einem ChangeMessageVisibility AWS SDK oder CLI
  - Verwendung von ChangeMessageVisibilityBatch mit einer CLI
  - Verwenden Sie mit einem CreateQueue AWS SDK oder CLI
  - Verwenden Sie DeleteMessagemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteMessageBatchmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteQueuemit einem AWS SDK oder CLI
  - Verwenden Sie GetQueueAttributesmit einem AWS SDK oder CLI
  - Verwenden Sie GetQueueUrlmit einem AWS SDK oder CLI
  - Verwendung von ListDeadLetterSourceQueues mit einer CLI
  - Verwenden Sie mit einem ListQueues AWS SDK oder CLI

- Verwendung von PurgeQueue mit einer CLI
- Verwenden Sie mit einem ReceiveMessage AWS SDK oder CLI
- Verwendung von RemovePermission mit einer CLI
- Verwenden Sie mit einem SendMessage AWS SDK oder CLI
- Verwenden Sie SendMessageBatchmit einem AWS SDK oder CLI
- Verwenden Sie SetQueueAttributesmit einem AWS SDK oder CLI

Hello Amazon SQS

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon SQS beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SQS;
using Amazon.SQS.Model;

namespace SQSActions;

public static class HelloSQS
{
    static async Task Main(string[] args)
    {
        var sqsClient = new AmazonSQSClient();

        Console.WriteLine($"Hello Amazon SQS! Following are some of your
 queues:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five queues.
```

```
        var response = await sqsClient.ListQueuesAsync(
            new ListQueuesRequest()
            {
                MaxResults = 5
            });

        foreach (var queue in response.QueueUrls)
        {
            Console.WriteLine($"\tQueue Url: {queue}");
            Console.WriteLine();
        }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
        hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_sqs.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 *  A "Hello SQS" starter application that initializes an Amazon Simple Queue
 Service
 *  (Amazon SQS) client and lists the SQS queues in the current account.
```

```
 *
 *  main function
 *
 *  Usage: 'hello_sqs'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SQS::Model::ListQueuesRequest request;

            Aws::SQS::Model::ListQueuesOutcome outcome =
 sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &pageOfQueueUrls =
 outcome.GetResult().GetQueueUrls();
                if (!pageOfQueueUrls.empty()) {
                    allQueueUrls.insert(allQueueUrls.cend(),
 pageOfQueueUrls.cbegin(),
                                        pageOfQueueUrls.cend());
                }
            }
            else {
                std::cerr << "Error with SQS::ListQueues. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
            nextToken = outcome.GetResult().GetNextToken();
```

```
        } while (!nextToken.empty());


        std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
 queue"
                  << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allQueueUrls.empty()) {
            std::cout << "Here are your queue URLs." << std::endl;
            for (const Aws::String &queueUrl: allQueueUrls) {
                std::cout << "  * " << queueUrl << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Einzelheiten zur API finden Sie ListQueuesunter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                    .flatMap(r -> r.queueUrls().stream())
                    .forEach(content -> System.out.println(" Queue URL: " +
 content.toLowerCase()));
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen Amazon-SQS-Client und listen Sie Warteschlangen auf.

```
import { SQSClient, paginateListQueues } from "@aws-sdk/client-sqs";

export const helloSqs = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SQSClient({});

  // You can also use `ListQueuesCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListQueuesCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedQueues = paginateListQueues({ client }, {});
  const queues = [];

  for await (const page of paginatedQueues) {
    if (page.QueueUrls?.length) {
      queues.push(...page.QueueUrls);
    }
  }
```

```
  const suffix = queues.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SQS! You have ${queues.length} queue${suffix} in your
 account.`,
  );
  console.log(queues.map((t) => `  * ${t}`).join("\n"));
};
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Kotlin.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
import PackageDescription

let package = Package(
    name: "sqs-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sqs-basics",
```

```
            dependencies: [
                .product(name: "AWSSQS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Der Swift-Quellcode, `entry.swift`.

```
import ArgumentParser
import AWSClientRuntime
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sqs-basics",
        abstract: """
        This example shows how to list all of your available Amazon SQS queues.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.
```

```
        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }

        print("You have \(queues.count) queues:")
        for queue in queues {
            print("   \(queue)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Swift.

Aktionen für Amazon SQS mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon SQS-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-SQS-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon SQS AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Simple Email Service.

Beispiele

- Verwendung von AddPermission mit einer CLI
- Verwenden Sie mit einem ChangeMessageVisibility AWS SDK oder CLI
- Verwendung von ChangeMessageVisibilityBatch mit einer CLI
- Verwenden Sie mit einem CreateQueue AWS SDK oder CLI
- Verwenden Sie DeleteMessagemit einem AWS SDK oder CLI
- Verwenden Sie DeleteMessageBatchmit einem AWS SDK oder CLI
- Verwenden Sie DeleteQueuemit einem AWS SDK oder CLI
- Verwenden Sie GetQueueAttributesmit einem AWS SDK oder CLI
- Verwenden Sie GetQueueUrlmit einem AWS SDK oder CLI
- Verwendung von ListDeadLetterSourceQueues mit einer CLI
- Verwenden Sie mit einem ListQueues AWS SDK oder CLI
- Verwendung von PurgeQueue mit einer CLI
- Verwenden Sie mit einem ReceiveMessage AWS SDK oder CLI
- Verwendung von RemovePermission mit einer CLI
- Verwenden Sie mit einem SendMessage AWS SDK oder CLI
- Verwenden Sie SendMessageBatchmit einem AWS SDK oder CLI
- Verwenden Sie SetQueueAttributesmit einem AWS SDK oder CLI

Verwendung von **`AddPermission`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddPermission` verwendet wird.

CLI

AWS CLI

So fügen Sie einer Warteschlange ein Thema hinzu

In diesem Beispiel kann das angegebene AWS Konto Nachrichten an die angegebene Warteschlange senden.

Befehl:

```
aws sqs add-permission --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessagesFromMyQueue --aws-account-ids 12345EXAMPLE --actions SendMessage
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie AddPermissionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel kann der angegebene AWS-Konto Benutzer Nachrichten aus der angegebenen Warteschlange senden.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE
 -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter AddPermission AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel können die angegebenen Benutzer AWS-Konto Nachrichten aus der angegebenen Warteschlange senden.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE
 -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter AddPermission AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ChangeMessageVisibility`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ChangeMessageVisibility` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Changes the visibility timeout of a message in an Amazon Simple Queue Service
//! (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param visibilityTimeoutSeconds: Visibility timeout in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
    Aws::SQS::Model::ChangeMessageVisibilityRequest request;
    request.SetQueueUrl(queue_url);
    request.SetReceiptHandle(messageReceiptHandle);
    request.SetVisibilityTimeout(visibilityTimeoutSeconds);

    auto outcome = sqsClient.ChangeMessageVisibility(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully changed visibility of message " <<
                  messageReceiptHandle << " from queue " << queue_url <<
 std::endl;
    }
    else {
        std::cout << "Error changing visibility of message from queue "
                  << queue_url << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So ändern Sie die Timeout-Sichtbarkeit einer Nachricht

In diesem Beispiel wird die Timeout-Sichtbarkeit der angegebenen Nachricht auf 10 Stunden (10 Stunden x 60 Minuten x 60 Sekunden) geändert.

Befehl:

```
aws sqs change-message-visibility --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBTpyI...t6HyQg== --
visibility-timeout 36000
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Amazon-SQS-Nachricht und ändern Sie deren Timeout-Sichtbarkeit.

```
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
  SQSClient,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 1,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 1,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  const response = await client.send(
    new ChangeMessageVisibilityCommand({
      QueueUrl: queueUrl,
      ReceiptHandle: Messages[0].ReceiptHandle,
      VisibilityTimeout: 20,
```

```
    }),
  );
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Amazon-SQS-Nachricht und ändern Sie deren Timeout-Sichtbarkeit.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region to us-west-2
AWS.config.update({ region: "us-west-2" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else {
    // Make sure we have a message
    if (data.Messages != null) {
```

```
      var visibilityParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
        VisibilityTimeout: 20, // 20 second timeout
      };
      sqs.changeMessageVisibility(visibilityParams, function (err, data) {
        if (err) {
          console.log("Delete Error", err);
        } else {
          console.log("Timeout Changed", data);
        }
      });
    } else {
      console.log("No messages to change");
    }
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für die Nachricht mit dem angegebenen Empfangs-Handle in der angegebenen Warteschlange auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-
east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -
VisibilityTimeout 36000
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibility AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für die Nachricht mit dem angegebenen Empfangs-Handle in der angegebenen Warteschlange auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-
east-1.amazonaws.com/8039EXAMPLE/MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -
VisibilityTimeout 36000
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibility AWS -Tools für PowerShellCmdlet-Referenz (V5).

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages:
 10
                                                      })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."
```

```
  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
                                    visibility_timeout: 30 # This message will
 not be visible for 30 seconds after first receipt.
                                  })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                       queue_url: queue_url,
                                                       max_number_of_messages: 10
                                                     })

  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für Ruby API-Referenz.

Verwendung von **`ChangeMessageVisibilityBatch`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ChangeMessageVisibilityBatch` verwendet wird.

CLI

AWS CLI

So ändern Sie die Timeout-Sichtbarkeit mehrerer Nachrichten als Batch

In diesem Beispiel wird die Timeout-Sichtbarkeit der beiden angegebenen Nachrichten auf 10 Stunden (10 Stunden x 60 Minuten x 60 Sekunden) geändert.

Befehl:

```
aws sqs change-message-visibility-batch --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://change-message-
visibility-batch.json
```

Eingabedatei (change-message-visibility-batch.json):

```
[
  {
    "Id": "FirstMessage",
        "ReceiptHandle": "AQEBhz2q...Jf3kaw==",
        "VisibilityTimeout": 36000
  },
  {
    "Id": "SecondMessage",
        "ReceiptHandle": "AQEBkTUH...HifSnw==",
        "VisibilityTimeout": 36000
  }
]
```

Ausgabe:

```
{
  "Successful": [
    {
      "Id": "SecondMessage"
    },
    {
      "Id": "FirstMessage"
    }
  ]
}
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityBatchin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für zwei Nachrichten mit den angegebenen Empfangs-Handles in der angegebenen Warteschlange geändert. Das

Sichtbarkeits-Timeout der ersten Nachricht wird auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden). Das Sichtbarkeits-Timeout der zweiten Nachricht wird auf 5 Stunden geändert (5 Stunden * 60 Minuten * 60 Sekunden = 18 000 Sekunden).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibilityBatch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für zwei Nachrichten mit den angegebenen Empfangs-Handles in der angegebenen Warteschlange geändert. Das Sichtbarkeits-Timeout der ersten Nachricht wird auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden). Das Sichtbarkeits-Timeout der zweiten Nachricht wird auf 5 Stunden geändert (5 Stunden * 60 Minuten * 60 Sekunden = 18 000 Sekunden).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- Einzelheiten zur API finden Sie unter ChangeMessageVisibilityBatch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateQueue`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateQueue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen
- Senden und Empfangen von Stapeln von Nachrichten
- Verwenden der Amazon SQS Java Messaging Library, um mit der JMS-Schnittstelle zu arbeiten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Warteschlage mit einem bestimmten Namen erstellen

```
    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }
```

```
            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }
```

Erstellen Sie eine Amazon-SQS-Warteschlange und senden Sie eine Nachricht an diese.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
        private static readonly RegionEndpoint ServiceRegion =
 RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
 Dictionary<string, MessageAttributeValue>
            {
```

```
                { "Title",   new MessageAttributeValue { DataType = "String",
 StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
 StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
 StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
 bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl,
 messageBody, messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
 client, string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }
```

```
        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Create an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::CreateQueueRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::CreateQueueOutcome outcome =
 sqsClient.CreateQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created queue " << queueName << " with a queue
 URL "
                  << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Warteschlange

In diesem Beispiel wird eine Warteschlange mit dem angegebenen Namen erstellt, die Aufbewahrungsdauer für Nachrichten auf 3 Tage (3 Tage * 24 Stunden * 60 Minuten * 60 Sekunden) festgelegt, und die Warteschlange für unzustellbare Nachrichten der Warteschlange wird auf die angegebene Warteschlange mit einer maximalen Empfangszahl von 1 000 Nachrichten festgelegt.

Befehl:

```
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
```

Eingabedatei (create-queue.json):

```
{
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "MessageRetentionPeriod": "259200"
}
```

Ausgabe:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
```

```
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello
from msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
```

```
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (sqsQueueName = SQS_QUEUE_NAME) => {
  const command = new CreateQueueCommand({
    QueueName: sqsQueueName,
    Attributes: {
      DelaySeconds: "60",
      MessageRetentionPeriod: "86400",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Erstellen einer Amazon-SQS-Warteschlange mit Langabfrage

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "queue_name";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const response = await client.send(
    new CreateQueueCommand({
      QueueName: queueName,
      Attributes: {
        // When the wait time for the ReceiveMessage API action is greater than
 0,
        // long polling is in effect. The maximum long polling wait time is 20
        // seconds. Long polling helps reduce the cost of using Amazon SQS by,
        // eliminating the number of empty responses and false empty responses.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        ReceiveMessageWaitTimeSeconds: "20",
      },
    }),
  );
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    DelaySeconds: "60",
    MessageRetentionPeriod: "86400",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Erstellen Sie eine Amazon-SQS-Warteschlange, die auf eine Nachricht wartet.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
```

```
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- Einzelheiten zur API finden Sie CreateQueuein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird eine Warteschlange mit dem angegebenen Namen erstellt.

```
New-SQSQueue -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter CreateQueue AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Warteschlange mit dem angegebenen Namen erstellt.

```
New-SQSQueue -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter CreateQueue AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the
 queue.
    :param attributes: The attributes of the queue, such as maximum message size
 or
                       whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can
 be used
             to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(QueueName=name, Attributes=attributes)
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise
```

- Einzelheiten zur API finden Sie CreateQueuein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require 'aws-sdk-sqs'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)
```

```
  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
    TRY.
        oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).        "
oo_result is returned for testing purposes. "
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

Erstellen Sie eine Amazon-SQS-Warteschlange, die auf eine Nachricht wartet.

```
    TRY.
        DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
        DATA ls_attribute TYPE /aws1/
cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
        ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.               " Time
 in seconds for long polling, such as how long the call waits for a message to
 arrive in the queue before returning. "
        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
 iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
        oo_result = lo_sqs->createqueue(                  " oo_result is returned
 for testing purposes. "
                iv_queuename = iv_queue_name
                it_attributes = lt_attributes ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
 queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateQueuein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)
```

```
        let output = try await sqsClient.createQueue(
            input: CreateQueueInput(
                queueName: queueName
            )
        )

        guard let queueUrl = output.queueUrl else {
            print("No queue URL returned.")
            return
        }
```

- Einzelheiten zur API finden Sie CreateQueuein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteMessage`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteMessage` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Senden und Empfangen von Stapeln von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht aus einer Amazon-SQS-Warteschlange und löschen Sie sie anschließend.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
```

```
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
 queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await
 client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</
param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
 ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
```

```
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome =
 sqsClient.DeleteMessage(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
                  << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Um eine Nachricht zu löschen

In diesem Beispiel wird die angegebene Nachricht gelöscht.

Befehl:

```
aws sqs delete-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --receipt-handle AQEBRXTo...q2doVA==
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Amazon-SQS-Nachrichten ab und löschen Sie sie.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);
```

```
  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Amazon-SQS-Nachrichten ab und löschen Sie sie.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle,
    };
    sqs.deleteMessage(deleteParams, function (err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteMessagein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die Nachricht mit dem angegebenen Empfangs-Handle aus der angegebenen Warteschlange gelöscht.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Einzelheiten zur API finden Sie unter DeleteMessage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Nachricht mit dem angegebenen Empfangs-Handle aus der angegebenen Warteschlange gelöscht.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -ReceiptHandle AQEBd329...v6gl8Q==
```

- Einzelheiten zur API finden Sie unter DeleteMessage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained
 in
```

```
                    the message's metadata.
    :return: None
    """
    try:
        message.delete()
        logger.info("Deleted message: %s", message.message_id)
    except ClientError as error:
        logger.exception("Couldn't delete message: %s", message.message_id)
        raise error
```

- Einzelheiten zur API finden Sie DeleteMessagein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sqs->deletemessage(
           iv_queueurl = iv_queue_url
           iv_receipthandle = iv_receipt_handle ).
        MESSAGE 'Message deleted from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsinvalididformat.
        MESSAGE 'The specified receipt handle is not valid.' TYPE 'E'.
      CATCH /aws1/cx_sqsreceipthandleisinv.
        MESSAGE 'The specified receipt handle is not valid for the current
version.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteMessagein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteMessageBatch`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteMessageBatch` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Process-S3-Ereignisbenachrichtigungen
- Veröffentlichen Sie Nachrichten in Warteschlangen
- Senden und Empfangen von Stapeln von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Delete a batch of messages from a queue by its url.
   /// </summary>
   /// <param name="queueUrl">The url of the queue.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
   {
       var deleteRequest = new DeleteMessageBatchRequest()
       {
           QueueUrl = queueUrl,
           Entries = new List<DeleteMessageBatchRequestEntry>()
       };
       foreach (var message in messages)
       {
           deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
           {
               ReceiptHandle = message.ReceiptHandle,
               Id = message.MessageId
```

```
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
```

```
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie mehrere Nachrichten als Batch

In diesem Beispiel werden die angegebenen Nachrichten gelöscht.

Befehl:

```
aws sqs delete-message-batch --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://delete-message-batch.json
```

Eingabedatei (delete-message-batch.json):

```
[
  {
```

```
        "Id": "FirstMessage",
        "ReceiptHandle": "AQEB1mgl...Z4GuLw=="
  },
  {
    "Id": "SecondMessage",
        "ReceiptHandle": "AQEBLsYM...VQubAA=="
  }
]
```

Ausgabe:

```
{
  "Successful": [
    {
      "Id": "FirstMessage"
    },
    {
      "Id": "SecondMessage"
    }
  ]
}
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
```

```
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Empfangs-Handles aus der angegebenen Warteschlange gelöscht.

```
$deleteMessageRequest1 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
 $deleteMessageRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Einzelheiten zur API finden Sie unter DeleteMessageBatch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Empfangs-Handles aus der angegebenen Warteschlange gelöscht.

```
$deleteMessageRequest1 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object
 Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $deleteMessageRequest1,
 $deleteMessageRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- Einzelheiten zur API finden Sie unter DeleteMessageBatch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and
 failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
 bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
```

```
                'ReceiptHandle': message['ReceiptHandle']
            })

        # Delete messages in batches of 10 (SQS limit)
        batch_size = 10
        for i in range(0, len(delete_entries), batch_size):
            batch = delete_entries[i:i + batch_size]

            response = self.sqs_client.delete_message_batch(
                QueueUrl=queue_url,
                Entries=batch
            )

            # Check for failures
            if 'Failed' in response and response['Failed']:
                for failed in response['Failed']:
                    logger.warning(f"Failed to delete message: {failed}")

        logger.info(f"Deleted {len(messages)} messages from {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error deleting messages: {error_code} - {e}")
        raise
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->deletemessagebatch(       " oo_result is returned for
 testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_entries ).
        MESSAGE 'Messages deleted from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsbtcentidsnotdist00.
        MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
 'E'.
      CATCH /aws1/cx_sqsemptybatchrequest.
        MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
      CATCH /aws1/cx_sqsinvbatchentryid.
        MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
 'E'.
      CATCH /aws1/cx_sqstoomanyentriesin00.
        MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        // Create the list of message entries.

        var entries: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []
        var messageNumber = 1
```

```
        for handle in handles {
            let entry = SQSClientTypes.DeleteMessageBatchRequestEntry(
                id: "\(messageNumber)",
                receiptHandle: handle
            )
            entries.append(entry)
            messageNumber += 1
        }

        // Delete the messages.

        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(
                entries: entries,
                queueUrl: queue
            )
        )

        // Get the lists of failed and successful deletions from the output.

        guard let failedEntries = output.failed else {
            print("Failed deletion list is missing!")
            return
        }
        guard let successfulEntries = output.successful else {
            print("Successful deletion list is missing!")
            return
        }

        // Display a list of the failed deletions along with their
        // corresponding explanation messages.

        if failedEntries.count != 0 {
            print("Failed deletions:")

            for entry in failedEntries {
                print("Message #\(entry.id ?? "<unknown>") failed:
\(entry.message ?? "<unknown>")")
            }
        } else {
            print("No failed deletions.")
        }
```

```
        // Output a list of the message numbers that were successfully deleted.

        if successfulEntries.count != 0 {
            var successes = ""

            for entry in successfulEntries {
                if successes.count == 0 {
                    successes = entry.id ?? "<unknown>"
                } else {
                    successes = "\(successes), \(entry.id ?? "<unknown>")"
                }
            }
            print("Succeeded: ", successes)
        } else {
            print("No successful deletions.")
        }
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`DeleteQueue`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteQueue` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen
- Senden und Empfangen von Stapeln von Nachrichten
- Verwenden der Amazon SQS Java Messaging Library, um mit der JMS-Schnittstelle zu arbeiten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Warteschlange unter Verwendung ihrer URL löschen

```
/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueURL: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::SQS::Model::DeleteQueueRequest request;
    request.SetQueueUrl(queueURL);

    const Aws::SQS::Model::DeleteQueueOutcome outcome =
 sqsClient.DeleteQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted queue with url " << queueURL <<
                  std::endl;
    }
    else {
        std::cerr << "Error deleting queue " << queueURL << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So löschen Sie eine Warteschlange

In diesem Beispiel wird die angegebene Warteschlange gelöscht.

Befehl:

```
aws sqs delete-queue --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewerQueue
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteQueue deletes an Amazon SQS queue.
```

```
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <queueName>

                Where:
                   queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-SQS-Warteschlange.

```
import { DeleteQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "test-queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new DeleteQueueCommand({ QueueUrl: queueUrl });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.deleteQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }
```

```
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Warteschlange gelöscht.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Einzelheiten zur API finden Sie unter DeleteQueue AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Warteschlange gelöscht.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- Einzelheiten zur API finden Sie unter DeleteQueue AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
```

```
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise
```

- Einzelheiten zur API finden Sie DeleteQueuein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
        MESSAGE 'SQS queue deleted' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteQueuein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.deleteQueue(
                input: DeleteQueueInput(
                    queueUrl: queueUrl
                )
            )
        } catch _ as AWSSQS.QueueDoesNotExist {
            print("Error: The specified queue doesn't exist.")
            return
        }
```

- Einzelheiten zur API finden Sie DeleteQueuein der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetQueueAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetQueueAttributes` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Process-S3-Ereignisbenachrichtigungen
- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }

            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
            }
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie die Attribute einer Warteschlange ab

In diesem Beispiel werden alle Attribute der angegebenen Warteschlange abgerufen.

Befehl:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All
```

Ausgabe:

```
{
  "Attributes": {
    "ApproximateNumberOfMessagesNotVisible": "0",
    "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":1000}",
    "MessageRetentionPeriod": "345600",
    "ApproximateNumberOfMessagesDelayed": "0",
    "MaximumMessageSize": "262144",
    "CreatedTimestamp": "1442426968",
    "ApproximateNumberOfMessages": "0",
    "ReceiveMessageWaitTimeSeconds": "0",
    "DelaySeconds": "0",
    "VisibilityTimeout": "30",
    "LastModifiedTimestamp": "1442426968",
    "QueueArn": "arn:aws:sqs:us-east-1:80398EXAMPLE:MyNewQueue"
  }
}
```

In diesem Beispiel werden ausschließlich die maximalen Timeout-Attribute für Nachrichtengrößen und Sichtbarkeit der angegebenen Warteschlange abgerufen.

Befehl:

```
aws sqs get-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attribute-
names MaximumMessageSize VisibilityTimeout
```

Ausgabe:

```
{
  "Attributes": {
    "VisibilityTimeout": "30",
    "MaximumMessageSize": "262144"
  }
}
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
```

```
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS SDK für Go API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { GetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";
```

```
const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const getQueueAttributes = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new GetQueueAttributesCommand({
    QueueUrl: queueUrl,
    AttributeNames: ["DelaySeconds"],
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '747a1192-c334-5682-a508-4cd5e8dc4e79',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: { DelaySeconds: '1' }
  // }
  return response;
};
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zu den Attributen für die angegebene Warteschlange aufgelistet.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                       : 30
```

```
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                          [ApproximateNumberOfMessagesNotVisible,
 0], [ApproximateNumberOfMessagesDelayed, 0]...}
```

Beispiel 2: In diesem Beispiel werden nur die angegebenen Attribute für die angegebene Warteschlange separat aufgelistet.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
```

```
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Einzelheiten zur API finden Sie unter GetQueueAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zu den Attributen für die angegebene Warteschlange aufgelistet.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
```

```
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                        [ApproximateNumberOfMessagesNotVisible,
 0], [ApproximateNumberOfMessagesDelayed, 0]...}
```

Beispiel 2: In diesem Beispiel werden nur die angegebenen Attribute für die angegebene Warteschlange separat aufgelistet.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -
QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue
```

```
Policy                                  : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14

 495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                              : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- Einzelheiten zur API finden Sie unter GetQueueAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client
```

```
    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise
```

- Einzelheiten zur API finden Sie GetQueueAttributesin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [
                    .approximatenumberofmessages,
                    .maximummessagesize
                ],
                queueUrl: url
            )
        )

        guard let attributes = output.attributes else {
            print("No queue attributes returned.")
            return
        }

        for (attr, value) in attributes {
            switch(attr) {
            case "ApproximateNumberOfMessages":
                print("Approximate message count: \(value)")
            case "MaximumMessageSize":
                print("Maximum message size: \(value)kB")
            default:
                continue
            }
        }
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der API-Referenz zum AWS SDK für Swift.

Verwenden Sie **`GetQueueUrl`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetQueueUrl` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueUrl
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
    /// queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as
your
        // default user, you need to provide the AWS Region as a parameter to
the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueName = "New-Example-Queue";

        try
        {
```

```
                var response = await client.GetQueueUrlAsync(queueName);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine($"The URL for {queueName} is:
 {response.QueueUrl}");
                }
            }
            catch (QueueDoesNotExistException ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine($"The queue {queueName} was not found.");
            }
        }
    }
```

- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
 sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
                  outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie eine Warteschlangen-URL ab

In diesem Beispiel wird die URL der angegebenen Warteschlange abgerufen.

Befehl:

```
aws sqs get-queue-url --queue-name MyQueue
```

Ausgabe:

```
{
  "QueueUrl": "https://queue.amazonaws.com/80398EXAMPLE/MyQueue"
}
```

- Einzelheiten zur API finden Sie GetQueueUrlin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die URL für eine Amazon-SQS-Warteschlange ab.

```
import { GetQueueUrlCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const command = new GetQueueUrlCommand({ QueueName: queueName });
```

```
  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die URL für eine Amazon-SQS-Warteschlange ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
};

sqs.getQueueUrl(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die URL der Warteschlange mit dem angegebenen Namen aufgelistet.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter GetQueueUrl AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die URL der Warteschlange mit dem angegebenen Namen aufgelistet.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter GetQueueUrl AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
        logger.info("Got queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't get queue named %s.", name)
        raise error
    else:
        return queue
```

- Einzelheiten zur API finden Sie GetQueueUrlin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).        "
 oo_result is returned for testing purposes. "
        MESSAGE 'Queue URL retrieved.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedoesnotexist.
        MESSAGE 'The requested queue does not exist.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetQueueUrlin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`ListDeadLetterSourceQueues`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDeadLetterSourceQueues` verwendet wird.

CLI

AWS CLI

So listen Sie Quellwarteschlangen für unzustellbare Nachrichten auf

In diesem Beispiel werden die Warteschlangen aufgeführt, die der angegebenen Quellwarteschlange für unzustellbare Nachrichten zugeordnet sind.

Befehl:

```
aws sqs list-dead-letter-source-queues --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Ausgabe:

```
{
  "queueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- Einzelheiten zur API finden Sie ListDeadLetterSourceQueuesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die URLs aller Warteschlangen aufgelistet, die die angegebene Warteschlange als Warteschlange für unzustellbare Nachrichten verwenden.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Einzelheiten zur API finden Sie unter ListDeadLetterSourceQueues AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die URLs aller Warteschlangen aufgelistet, die die angegebene Warteschlange als Warteschlange für unzustellbare Nachrichten verwenden.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- Einzelheiten zur API finden Sie unter ListDeadLetterSourceQueues AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListQueues`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListQueues` verwendet wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS
 account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
                listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
 outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());
```

```
    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." <<
 std::endl;
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So listen Sie Warteschlangen auf

In diesem Beispiel werden alle Warteschlangen aufgelistet.

Befehl:

```
aws sqs list-queues
```

Ausgabe:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/TestQueue1",
        "https://queue.amazonaws.com/80398EXAMPLE/TestQueue2"
  ]
}
```

In diesem Beispiel werden nur Warteschlangen aufgelistet, die mit „My“ beginnen.

Befehl:

```
aws sqs list-queues --queue-name-prefix My
```

Ausgabe:

```
{
  "QueueUrls": [
    "https://queue.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyQueue",
    "https://queue.amazonaws.com/80398EXAMPLE/MyOtherQueue"
  ]
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
```

```
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
 ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
 sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Ihre Amazon-SQS-Warteschlangen auf.

```
import { paginateListQueues, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});

export const main = async () => {
  const paginatedListQueues = paginateListQueues({ client }, {});

  /** @type {string[]} */
  const urls = [];
  for await (const page of paginatedListQueues) {
```

```
    const nextUrls = page.QueueUrls?.filter((qurl) => !!qurl) || [];
    urls.push(...nextUrls);
    for (const url of urls) {
      console.log(url);
    }
  }

  return urls;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Ihre Amazon-SQS-Warteschlangen auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {};

sqs.listQueues(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel werden alle Parameter aufgelistet.

```
Get-SQSQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Beispiel 2: In diesem Beispiel werden alle Warteschlangen auflistet, die mit dem angegebenen Namen beginnen.

```
Get-SQSQueue -QueueNamePrefix My
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Einzelheiten zur API finden Sie unter ListQueues AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Parameter aufgelistet.

```
Get-SQSQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Beispiel 2: In diesem Beispiel werden alle Warteschlangen auflistet, die mit dem angegebenen Namen beginnen.

```
Get-SQSQueue -QueueNamePrefix My
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- Einzelheiten zur API finden Sie unter ListQueues AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ", ".join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- Einzelheiten zur API finden Sie ListQueuesin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
```

```
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die erste Amazon-SQS-Warteschlange ab, die in der Region aufgeführt ist.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to
 proceed.")
        .to_string())
}
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->listqueues( ).        " oo_result is returned for
 testing purposes. "
        MESSAGE 'Retrieved list of queues.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }
```

- Einzelheiten zur API finden Sie ListQueuesin der API-Referenz zum AWS SDK für Swift.

Verwendung von **PurgeQueue** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PurgeQueue` verwendet wird.

CLI

AWS CLI

So bereinigen Sie eine Warteschlange

In diesem Beispiel werden alle Nachrichten in der angegebenen Warteschlange gelöscht.

Befehl:

```
aws sqs purge-queue --queue-url https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyNewQueue
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie PurgeQueuein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Nachrichten aus der angegebenen Warteschlange gelöscht.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter PurgeQueue AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Nachrichten aus der angegebenen Warteschlange gelöscht.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter PurgeQueue AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ReceiveMessage`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ReceiveMessage` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten großer Nachrichten mit S3
- Process-S3-Ereignisbenachrichtigungen
- Veröffentlichen Sie Nachrichten in Warteschlangen
- Senden und Empfangen von Stapeln von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Nachrichten aus einer Warteschlange unter Verwendung ihrer URL empfangen

```
    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
```

```
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }
```

Empfangen Sie eine Nachricht aus einer Amazon-SQS-Warteschlange und löschen Sie sie anschließend.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
 queueName)
```

```
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await
 client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</
param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
 ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
 client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };
```

```
            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);

    const Aws::SQS::Model::ReceiveMessageOutcome outcome =
 sqsClient.ReceiveMessage(
            request);
```

```
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
                outcome.GetResult().GetMessages();
        if (!messages.empty()) {
            const Aws::SQS::Model::Message &message = messages[0];
            std::cout << "Received message:" << std::endl;
            std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
            std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
 std::endl;
            std::cout << "  Body: " << message.GetBody() << std::endl <<
 std::endl;
        }
        else {
            std::cout << "No messages received from queue " << queueUrl <<
                      std::endl;

        }
    }
    else {
        std::cerr << "Error receiving message from queue " << queueUrl << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So empfangen Sie eine Nachricht

In diesem Beispiel werden bis zu 10 verfügbare Nachrichten empfangen und alle verfügbaren Attribute zurückgegeben.

Befehl:

```
aws sqs receive-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-names All --message-
attribute-names All --max-number-of-messages 10
```

Ausgabe:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEBzbVv...fqNzFw==",
      "MD5OfBody": "1000f835...a35411fa",
      "MD5OfMessageAttributes": "9424c491...26bc3ae7",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "ApproximateFirstReceiveTimestamp": "1442428276921",
        "SenderId": "AIDAIAZKMSNQ7TEXAMPLE",
        "ApproximateReceiveCount": "5",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        },
        "City": {
          "DataType": "String",
          "StringValue": "Any City"
        }
      }
    }
  ]
}
```

In diesem Beispiel wird die nächste verfügbare Nachricht empfangen und nur die SentTimestamp Attribute SenderId und sowie das PostalCode Nachrichtenattribut zurückgegeben.

Befehl:

```
aws sqs receive-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --attribute-
names SenderId SentTimestamp --message-attribute-names PostalCode
```

Ausgabe:

```
{
  "Messages": [
    {
      "Body": "My first message.",
      "ReceiptHandle": "AQEB6nR4...HzlvZQ==",
      "MD5OfBody": "1000f835...a35411fa",
      "MD5OfMessageAttributes": "b8e89563...e088e74f",
      "MessageId": "d6790f8d-d575-4f01-bc51-40122EXAMPLE",
      "Attributes": {
        "SenderId": "AIDAIAZKMSNQ7TEXAMPLE",
        "SentTimestamp": "1442428276921"
      },
      "MessageAttributes": {
        "PostalCode": {
          "DataType": "String",
          "StringValue": "ABC123"
        }
      }
    }
  ]
}
```

- Einzelheiten zur API finden Sie ReceiveMessageunter AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht von einer Amazon-SQS-Warteschlange.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
```

```
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
import { ReceiveMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new ReceiveMessageCommand({
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueUrl,
    // The duration (in seconds) for which the call waits for a message
    // to arrive in the queue before returning. If a message is available,
    // the call returns sooner than WaitTimeSeconds. If no messages are
    // available and the wait time expires, the call returns successfully
    // with an empty list of messages.
    // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/
API_ReceiveMessage.html#API_ReceiveMessage_RequestSyntax
    WaitTimeSeconds: 20,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen für die nächsten 10 Nachrichten aufgelistet, die in der angegebenen Warteschlange empfangen werden sollen. Die Informationen enthalten Werte für die angegebenen Nachrichtenattribute, sofern sie existieren.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Einzelheiten zur API finden Sie unter ReceiveMessage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen für die nächsten 10 Nachrichten aufgelistet, die in der angegebenen Warteschlange empfangen werden sollen. Die Informationen enthalten Werte für die angegebenen Nachrichtenattribute, sofern sie existieren.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- Einzelheiten zur API finden Sie unter ReceiveMessage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual
 number
                        of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
 When
                       this number is greater than zero, long polling is used.
 This
                       can result in reduced costs and fewer false empty
 responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
```

```
    else:
        return messages
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
 List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
```

```
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise
```

- Einzelheiten zur API finden Sie ReceiveMessagein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
```

```
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end

  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
```

```
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
 client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht von einer Amazon-SQS-Warteschlange.

```
   TRY.
       oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).    "
oo_result is returned for testing purposes. "
       DATA(lt_messages) = oo_result->get_messages( ).
       MESSAGE 'Message received from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsoverlimit.
       MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
   ENDTRY.
```

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
   TRY.
       oo_result = lo_sqs->receivemessage(           " oo_result is returned for
testing purposes. "
               iv_queueurl = iv_queue_url
               iv_waittimeseconds = iv_wait_time ).    " Time in seconds for
long polling, such as how long the call waits for a message to arrive in the
queue before returning. " ).
       DATA(lt_messages) = oo_result->get_messages( ).
       MESSAGE 'Message received from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsoverlimit.
       MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ReceiveMessagein der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: maxMessages,
                queueUrl: url
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return
        }

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print(message.body ?? "<body missing>")
            print("---")
        }
```

- Einzelheiten zur API finden Sie ReceiveMessagein der API-Referenz zum AWS SDK für Swift.

Verwendung von **`RemovePermission`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemovePermission` verwendet wird.

CLI

AWS CLI

So entfernen Sie eine Berechtigung

In diesem Beispiel wird die Berechtigung mit dem angegebenen Label aus der angegebenen Warteschlange entfernt.

Befehl:

```
aws sqs remove-permission --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --label SendMessagesFromMyQueue
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie RemovePermissionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Berechtigungseinstellungen mit der angegebenen Bezeichnung aus der angegebenen Warteschlange entfernt.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Berechtigungseinstellungen mit der angegebenen Bezeichnung aus der angegebenen Warteschlange entfernt.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter RemovePermission AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`SendMessage`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendMessage` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Verwalten großer Nachrichten mit S3
- Senden und Empfangen von Stapeln von Nachrichten

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Amazon-SQS-Warteschlange und senden Sie eine Nachricht an diese.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;
```

```
    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
        private static readonly RegionEndpoint ServiceRegion =
 RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
 Dictionary<string, MessageAttributeValue>
            {
                { "Title",   new MessageAttributeValue { DataType = "String",
 StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
 StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
 StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
 bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl,
 messageBody, messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
```

```
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
 client, string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
```

```
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageBody: A message body.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                              const Aws::String &messageBody,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
```

```
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
 sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
                  std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So senden Sie eine Nachricht

In diesem Beispiel wird eine Nachricht mit dem angegebenen Nachrichtentext, der angegebenen Verzögerungszeit und Nachrichtenattributen an die angegebene Warteschlange gesendet.

Befehl:

```
aws sqs send-message --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --message-body "Information about the
 largest city in Any Region." --delay-seconds 10 --message-attributes file://
send-message.json
```

Eingabedatei (send-message.json):

```
{
  "City": {
    "DataType": "String",
    "StringValue": "Any City"
```

```
  },
  "Greeting": {
    "DataType": "Binary",
    "BinaryValue": "Hello, World!"
  },
  "Population": {
    "DataType": "Number",
    "StringValue": "1250800"
  }
}
```

Ausgabe:

```
{
  "MD5OfMessageBody": "51b0a325...39163aa0",
  "MD5OfMessageAttributes": "00484c68...59e48f06",
  "MessageId": "da68f62c-0c07-4bee-bf5f-7e856EXAMPLE"
}
```

- Einzelheiten zur API finden Sie SendMessagein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Es folgen zwei Beispiele für die `SendMessage`-Operation:

- Senden einer Nachricht mit Text und einer Verzögerung
- Senden einer Nachricht mit Text und Nachrichtenattributen

Senden einer Nachricht mit Text und einer Verzögerung.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
```

```
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName> <message>

                Where:
                   queueName - The name of the queue.
                   message - The message to send.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
 message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
```

```
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(message)
                    .delaySeconds(5)
                    .build();

            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Senden einer Nachricht mit Text und Nachrichtenattributen.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java
 Map to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary".
 You can optionally
     * define a custom extension by using a "." and your extension.</p>
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
```

```
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

.binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}",
e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Nachricht an eine Amazon-SQS-Warteschlange.

```
import { SendMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (sqsQueueUrl = SQS_QUEUE_URL) => {
  const command = new SendMessageCommand({
    QueueUrl: sqsQueueUrl,
    DelaySeconds: 10,
    MessageAttributes: {
      Title: {
        DataType: "String",
        StringValue: "The Whistler",
      },
      Author: {
        DataType: "String",
        StringValue: "John Grisham",
      },
      WeeksOn: {
        DataType: "Number",
        StringValue: "6",
      },
    },
    MessageBody:
      "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
```

```
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Nachricht an eine Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  // Remove DelaySeconds parameter and value for FIFO queues
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
```

```
    "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  // MessageDeduplicationId: "TheWhistler",  // Required for FIFO queues
  // MessageGroupId: "Group1",  // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.sendMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.MessageId);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
            messageBody = message
            delaySeconds = 10
        }
```

```
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- Einzelheiten zur API finden Sie SendMessagein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: In diesem Beispiel wird eine Nachricht mit den angegebenen Attributen und dem angegebenen Nachrichtentext an die angegebene Warteschlange gesendet, wobei die Zustellung der Nachricht um 10 Sekunden verzögert wird.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Einzelheiten zur API finden Sie unter SendMessage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Nachricht mit den angegebenen Attributen und dem angegebenen Nachrichtentext an die angegebene Warteschlange gesendet, wobei die Zustellung der Nachricht um 10 Sekunden verzögert wird.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
```

```
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- Einzelheiten zur API finden Sie unter SendMessage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.
```

```
    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-
value
                               pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body, MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- Einzelheiten zur API finden Sie SendMessagein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
```

```
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end
```

```
# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
 Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // and message_group_id or configure the queue for
 ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:#?}", rsp);

    Ok(())
}
```

- Einzelheiten zur API finden Sie SendMessagein der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_sqs->sendmessage(              " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          iv_messagebody = iv_message ).
       MESSAGE 'Message sent to SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsinvalidmsgconts.
       MESSAGE 'Message contains non-valid characters.' TYPE 'E'.
     CATCH /aws1/cx_sqsunsupportedop.
       MESSAGE 'Operation not supported.' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie SendMessagein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendMessageBatch`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendMessageBatch` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Senden und Empfangen von Stapeln von Nachrichten

CLI

AWS CLI

So senden Sie mehrere Nachrichten als Batch

In diesem Beispiel werden 2 Nachrichten mit den angegebenen Nachrichtentexten, Verzögerungszeiten und Nachrichtenattributen an die angegebene Warteschlange gesendet.

Befehl:

```
aws sqs send-message-batch --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue --entries file://send-message-
batch.json
```

Eingabedatei (send-message-batch.json):

```
[
  {
    "Id": "FuelReport-0001-2015-09-16T140731Z",
        "MessageBody": "Fuel report for account 0001 on 2015-09-16 at 02:07:31
 PM.",
        "DelaySeconds": 10,
        "MessageAttributes": {
          "SellerName": {
            "DataType": "String",
                "StringValue": "Example Store"
      },
          "City": {
        "DataType": "String",
        "StringValue": "Any City"
      },
          "Region": {
            "DataType": "String",
                "StringValue": "WA"
      },
          "PostalCode": {
            "DataType": "String",
                "StringValue": "99065"
          },
          "PricePerGallon": {
            "DataType": "Number",
                "StringValue": "1.99"
      }
        }
  },
  {
    "Id": "FuelReport-0002-2015-09-16T140930Z",
```

```
        "MessageBody": "Fuel report for account 0002 on 2015-09-16 at 02:09:30
 PM.",
        "DelaySeconds": 10,
        "MessageAttributes": {
          "SellerName": {
            "DataType": "String",
                "StringValue": "Example Fuels"
      },
          "City": {
        "DataType": "String",
        "StringValue": "North Town"
      },
          "Region": {
            "DataType": "String",
                "StringValue": "WA"
      },
          "PostalCode": {
            "DataType": "String",
                "StringValue": "99123"
          },
          "PricePerGallon": {
            "DataType": "Number",
                "StringValue": "1.87"
      }
        }
  }
]
```

Ausgabe:

```
{
  "Successful": [
    {
      "MD5OfMessageBody": "203c4a38...7943237e",
      "MD5OfMessageAttributes": "10809b55...baf283ef",
      "Id": "FuelReport-0001-2015-09-16T140731Z",
      "MessageId": "d175070c-d6b8-4101-861d-adeb3EXAMPLE"
    },
    {
      "MD5OfMessageBody": "2cf0159a...c1980595",
      "MD5OfMessageAttributes": "55623928...ae354a25",
      "Id": "FuelReport-0002-2015-09-16T140930Z",
      "MessageId": "f9b7d55d-0570-413e-b9c5-a9264EXAMPLE"
```

```
    }
  ]
}
```

- Einzelheiten zur API finden Sie SendMessageBatchin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello
from msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- Einzelheiten zur API finden Sie SendMessageBatchin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Attributen und Nachrichtentexten an die angegebene Warteschlange gesendet. Die Zustellung verzögert sich um 15 Sekunden für die erste Nachricht und um 10 Sekunden für die zweite Nachricht.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Einzelheiten zur API finden Sie unter SendMessageBatch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Attributen und Nachrichtentexten an die angegebene Warteschlange gesendet. Die Zustellung verzögert sich um 15 Sekunden für die erste Nachricht und um 10 Sekunden für die zweite Nachricht.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- Einzelheiten zur API finden Sie unter SendMessageBatch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not
 sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and
 failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
```

```
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response
```

- Einzelheiten zur API finden Sie SendMessageBatchin AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
```

```
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
```

```
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- Einzelheiten zur API finden Sie SendMessageBatchin der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->sendmessagebatch(         " oo_result is returned for
 testing purposes. "
          iv_queueurl = iv_queue_url
```

```
          it_entries = it_messages ).
        MESSAGE 'Messages sent to SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsbtcentidsnotdist00.
        MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
      CATCH /aws1/cx_sqsbatchreqtoolong.
        MESSAGE 'The length of all the messages put together is more than the
limit.' TYPE 'E'.
      CATCH /aws1/cx_sqsemptybatchrequest.
        MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
      CATCH /aws1/cx_sqsinvbatchentryid.
        MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE
'E'.
      CATCH /aws1/cx_sqstoomanyentriesin00.
        MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
      CATCH /aws1/cx_sqsunsupportedop.
        MESSAGE 'Operation not supported.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendMessageBatchin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SetQueueAttributes`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SetQueueAttributes` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Veröffentlichen Sie Nachrichten in Warteschlangen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richtlinienattribut einer Warteschlange für ein Thema festlegen

```
    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
 topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                               $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK für .NET API-Referenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
 queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param attributeName: An attribute name enum.
  \param attribute: The attribute value as a string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
 attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            attributeName,
            attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute  " <<
```

```
 Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
                         attributeName)
                << " with value " << attribute << " in queue " <<
                queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for  queue " <<
                queueURL << ": " << outcome.GetError().GetMessage() <<
                std::endl;
    }

    return outcome.IsSuccess();
}
```

Konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
//! dead-letter queue.
/*!
  \param srcQueueUrl: An Amazon SQS queue URL.
  \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS
 dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                     const Aws::String &deadLetterQueueARN,
                                     int maxReceiveCount,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
 maxReceiveCount);

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(srcQueueUrl);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set dead letter queue for queue  " <<
                  srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
    }
    else {
        std::cerr << "Error setting dead letter queue for queue " <<
                  srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
  \param queueArn: An Amazon SQS ARN for the dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \return Aws::String: Policy as JSON string.
 */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}
```

Konfigurieren Sie eine Amazon-SQS-Warteschlange für die Verwendung einer Langabfrage.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param pollTimeSeconds: The receive message wait time in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                               const Aws::String
 &pollTimeSeconds,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
                  queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So legen Sie Warteschlangenattribute fest

In diesem Beispiel wird für die angegebene Warteschlange eine Zustellungsverzögerung von 10 Sekunden, eine maximale Nachrichtengröße von 128 KB (128 KB x 1 024 Byte), eine Aufbewahrungszeit für Nachrichten von 3 Tagen (3 Tage x 24 Stunden x 60 Minuten x 60 Sekunden), eine Wartezeit für den Empfang von Nachrichten von 20 Sekunden, und ein standardmäßiges Sichtbarkeits-Timeout von 60 Sekunden festgelegt. In diesem Beispiel wird der angegebenen Warteschlange für unzustellbare Nachrichten außerdem eine maximale Empfangszahl von 1 000 Nachrichten zugewiesen.

Befehl:

```
aws sqs set-queue-attributes --queue-url https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyNewQueue --attributes file://set-queue-
attributes.json
```

Eingabedatei (set-queue-attributes.json):

```
{
  "DelaySeconds": "10",
  "MaximumMessageSize": "131072",
  "MessageRetentionPeriod": "259200",
  "ReceiveMessageWaitTimeSeconds": "20",
  "RedrivePolicy": "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-1:80398EXAMPLE:MyDeadLetterQueue\",\"maxReceiveCount\":\"1000\"}",
  "VisibilityTimeout": "60"
}
```

Ausgabe:

```
None.
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
```

```
    Principal: map[string]string{"Service": "sns.amazonaws.com"},
    Resource:  aws.String(queueArn),
    Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie ein Amazon SQS für die serverseitige Verschlüsselung (SSE) mit einem benutzerdefinierten KMS-Schlüssel.

```
   public static void addEncryption(String queueName, String kmsMasterKeyAlias)
{
       SqsClient sqsClient = SqsClient.create();

       GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
               .queueName(queueName)
               .build();

       GetQueueUrlResponse getQueueUrlResponse;
       try {
           getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
       } catch (QueueDoesNotExistException e) {
           LOGGER.error(e.getMessage(), e);
           throw new RuntimeException(e);
       }
       String queueUrl = getQueueUrlResponse.queueUrl();


       Map<QueueAttributeName, String> attributes = Map.of(
               QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
               QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" //
Set the data key reuse period to 140 seconds.
       );                                                                  //
This is how long SQS can reuse the data key before requesting a new one from
KMS.
```

```
        SetQueueAttributesRequest attRequest =
 SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
        } catch (InvalidAttributeNameException | InvalidAttributeValueException
 e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    QueueUrl: queueUrl,
    Attributes: {
```

```
      DelaySeconds: "1",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Konfigurieren Sie eine Amazon-SQS-Warteschlange für die Verwendung einer Langabfrage.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      ReceiveMessageWaitTimeSeconds: "20",
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
const DEAD_LETTER_QUEUE_ARN = "dead_letter_queue_arn";

export const main = async (
  queueUrl = SQS_QUEUE_URL,
  deadLetterQueueArn = DEAD_LETTER_QUEUE_ARN,
) => {
```

```
  const command = new SetQueueAttributesCommand({
    Attributes: {
      RedrivePolicy: JSON.stringify({
        // Amazon SQS supports dead-letter queues (DLQ), which other
        // queues (source queues) can target for messages that can't
        // be processed (consumed) successfully.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-dead-letter-queues.html
        deadLetterTargetArn: deadLetterQueueArn,
        maxReceiveCount: "10",
      }),
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt, wie Sie eine Richtlinie festlegen, mit der eine Warteschlange für ein SNS-Thema abonniert wird. Wenn eine Nachricht in einem Thema veröffentlicht wird, wird eine Nachricht an die abonnierte Warteschlange gesendet.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
 "QueueArn").QueueARN
```

```
# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Attribute für die angegebene Warteschlange festgelegt.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter SetQueueAttributes AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt, wie Sie eine Richtlinie festlegen, mit der eine Warteschlange für ein SNS-Thema abonniert wird. Wenn eine Nachricht in einem Thema veröffentlicht wird, wird eine Nachricht an die abonnierte Warteschlange gesendet.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
```

```
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName
 "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Attribute für die angegebene Warteschlange festgelegt.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- Einzelheiten zur API finden Sie unter SetQueueAttributes AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richtlinienattribut einer Warteschlange für ein Thema festlegen

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
 queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
```

```
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise
```

- Einzelheiten zur API finden Sie SetQueueAttributesin AWS SDK for Python (Boto3) API Reference.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.setQueueAttributes(
                input: SetQueueAttributesInput(
                    attributes: [
                        "MaximumMessageSize": "\(maxSize)"
                    ],
                    queueUrl: url
                )
            )
        } catch _ as AWSSQS.InvalidAttributeValue {
            print("Invalid maximum message size: \(maxSize) kB.")
        }
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der API-Referenz zum AWS SDK für Swift.

Szenarien für die Verwendung von Amazon SQS AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon SQS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon SQS oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft
- Erstellen einer Messenger-Anwendung mit Step Functions
- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK
- Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK
- Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK
- Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK
- Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK
- Senden und Empfangen von Nachrichtenstapeln mit Amazon SQS unter Verwendung eines AWS SDK
- Benutze die AWS Message Processing Framework für.NET zum Veröffentlichen und Empfangen von Amazon SQS SQS-Nachrichten
- Verwenden der Amazon SQS Java Messaging Library, um mit der Java-Message-Service(JMS)-Schnittstelle zu arbeiten
- Arbeiten Sie mit Warteschlangen-Tags und Amazon SQS mithilfe eines AWS SDK

Entwickeln einer Webanwendung, die Nachrichten mithilfe von Amazon SQS sendet und abruft

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

Java

SDK für Java 2.x

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Kotlin

SDK für Kotlin

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Erstellen einer Messenger-Anwendung mit Step Functions

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

Python

SDK für Python (Boto3)

Zeigt, wie AWS SDK für Python (Boto3) mit AWS Step Functions dem with eine Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. Die Zustandsmaschine ist mit einer AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht werden kann.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.

- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3

- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Erstellen und veröffentlichen Sie es in einem FIFO-Amazon SNS-Thema mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie man ein FIFO-Amazon-SNS-Thema erstellt.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created
 for the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
 created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
```

```
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
```

```
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")
```

```
    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
```

```
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
```

```
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error
```

```
    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
```

```
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
```

```
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'

 iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie die erweiterte Clientbibliothek von Amazon SQS verwenden, um mit großen Amazon-SQS-Nachrichten zu arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
```

```
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
```

```
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));
```

```
            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(),
e);
            throw e;
        }
    }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit S3-Ereignisbenachrichtigungen auf objektorientierte Weise arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie ein S3-Benachrichtigungsereignis mithilfe von Amazon SQS verarbeiten.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
```

```
    * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
    * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
    * the JSON payload and access the messages in an object-oriented way.
    *
    * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
    * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
 the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String
 queueArn) {
       try {
           // Configure the bucket to send Object Created and Object Tagging
 notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();
```

```
            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the
queue.
                sqsClient.getQueueAttributes(b -> b
                                .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
```

```
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Veröffentlichen Sie Amazon SNS SNS-Nachrichten in Amazon SQS SQS-Warteschlangen mithilfe eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
```

```
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();
```

```
            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
```

```
                            $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                            $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                            $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                                $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                                $"\r\nfrom content using a hash function.\r\n" +
                                $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                                $"\r\npublished and determined to have the same
 deduplication ID, " +
                                $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
```

```
                                $"\r\nFor more information about deduplication, " +
                                $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
 Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
 queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
 appended to the queue name.");
                }
```

```
                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
```

```
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
```

```
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }
```

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
```

```
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
```

```
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
```

```
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
```

```
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
```

```
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\":
\"{topicArn}\"" +
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
```

```
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
```

```
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
```

```
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
 subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
 filterPolicy, string queueArn)
```

```
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
```

```
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
```

```
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
```

```
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
```

```
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
```

```
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                  "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
```

```
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;
```

```
                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }
```

```
                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
```

```
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));
```

```
    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
```

```
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
```

```
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);
```

```
        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                           const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
```

```
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
```

```
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
```

```
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
```

```
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
```

```
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
   for askAboutTones {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelections = append(toneSelections, ToneChoices[toneIndex])
    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
   }
   log.Printf("Your subscription will be filtered to only pass the following
 tones: %v\n", toneSelections)
   filterPolicy = map[string][]string{TONE_KEY: toneSelections}
  }
 }

 subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
 string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
```

```
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
```

```
    if err != nil {
     panic(err)
    }
    if len(currentMsgs) == 0 {
     break
    }
    messages = append(messages, currentMsgs...)
   }
   if len(messages) == 0 {
    log.Printf("No messages were received by queue %v.\n", queueUrl)
   } else if len(messages) == 1 {
    log.Printf("One message was received by queue %v:\n", queueUrl)

   } else {
    log.Printf("%v messages were received by queue %v:\n", len(messages),
  queueUrl)
   }
   for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
     panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
   }

   if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
```

```
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))
```

```
 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SNS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

```
// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
```

```
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SQS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }
```

```
 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
```

```
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
```

```
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
```

```
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management
 Console\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
```

```
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
```

```
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
```

```
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
 the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
```

```
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,
\n"
                +
                "        within the five-minute deduplication interval, is
 accepted but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
 must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);
```

```
        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
```

```
                    "ArnEquals": {
                        "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                    }
                }
            }
            ]
        }
       """.formatted(accountId, sqsQueueName, accountId, topicName);

       setQueueAttr(sqsClient, sqsQueueUrl, policy);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("6. Subscribe to the SQS queue.");
       if (selectFIFO) {
           System.out.println(
               "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                   +
                   "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                   +
                   "For this example, you can filter messages by a \"tone\"
 attribute.");
           System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
               + topicName + "?  (y/n)");
           String filterAns = in.nextLine();
           if (filterAns.compareTo("y") == 0) {
               boolean moreAns = false;
               System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
               System.out.println("1. cheerful");
               System.out.println("2. funny");
               System.out.println("3. serious");
               System.out.println("4. sincere");
               while (!moreAns) {
                   System.out.println("Select a number or choose 0 to end.");
                   String ans = in.nextLine();
                   switch (ans) {
                       case "1":
                           filterList.add("cheerful");
                           break;
                       case "2":
```

```
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
```

```
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
```

```
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());
```

```
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();
```

```
                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();
```

```
        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt für dieses Szenario.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
```

```
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Der obige Code stellt die erforderlichen Abhängigkeiten bereit und startet das Szenario. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
```

```
 * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
 * @param {import('../../libs/prompter.js').Prompter} prompter
 * @param {import('../../libs/logger.js').Logger} logger
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }

  const response = await this.snsClient.send(
    new CreateTopicCommand({
```

```
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
```

```
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );
```

```
      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
```

```
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });
```

```
    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
```

```
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
```

```
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
```

```
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

```
This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
```

```
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)
```

```
    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
```

```
        },
        "Action": "sqs:SendMessage",
                "Resource": "$sqsQueueArn",
                "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "$topicArn"
            }
        }
    }
    ]
    }"""
   setQueueAttr(sqsQueueUrl, policy)
   println(DASHES)

   println(DASHES)
   println("6. Subscribe to the SQS queue.")
   if (selectFIFO) {
       println(
           """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
       )
       println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
       val filterAns: String = input.nextLine()
       if (filterAns.compareTo("y") == 0) {
           var moreAns = false
           println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
           println("1. cheerful")
           println("2. funny")
           println("3. serious")
           println("4. sincere")
           while (!moreAns) {
               println("Select a number or choose 0 to end.")
               val ans: String = input.nextLine()
               when (ans) {
                   "1" -> filterList.add("cheerful")
                   "2" -> filterList.add("funny")
                   "3" -> filterList.add("serious")
                   "4" -> filterList.add("sincere")
                   else -> moreAns = true
```

```
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
 deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
```

```
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
```

```
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
```

```
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
```

```
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
```

```
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }
```

```
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
```

```
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
```

```
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe
 {self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
 subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
```

```
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication
 and message filtering.")
        print()

        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
 q.non_empty)
```

```
            print("Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name.")
            print()

            print(self.DASHES)
            print("""
   Because you have chosen a FIFO topic, deduplication is supported.
   Deduplication IDs are either set in the message or automatically generated
   from content using a hash function.

   If a message is successfully published to an SNS FIFO topic, any message
   published and determined to have the same deduplication ID,
   within the five-minute deduplication interval, is accepted but not delivered.

   For more information about deduplication,
   see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a
deduplication ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )

        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
```

```
        print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
self.topic_arn, queue_url)

            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
queue_name)
            self.subscription_arns.append(subscription_arn)
```

```
    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
 to the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
 {self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
        print("You can filter messages by one or more of the following TONE
 attributes.")

        filter_selections = []
        selection_number = 0
```

```
        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)
```

```
                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based
deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")

            messages = self._poll_queue_for_messages(queue_url)

            if messages:
```

```
                print(f"{len(messages)} message(s) were received by queue #{i
+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message',
 message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
 1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")
```

```
        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
 (y/n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Erstellen Sie Klassen, die Amazon SNS- und Amazon SQS SQS-Operationen für die Verwendung im Szenario zusammenfassen.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""
```

```
    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
```

```
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )
```

```
            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }
```

```
            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} -
{e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
```

```
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

            logger.info(f"Found {len(topics)} topics")
```

```
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
```

```
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} -
{e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
```

```
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )
```

```
            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
```

```
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
```

```
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
```

```
            return queue_urls

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM
permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
```

```
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
            str = readLine()

            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
```

```
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }
```

```
    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }

        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
 attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true
```

```
        while let ans = askForFilter(message: "Choose a value to apply to the
'tone' attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
```

```
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
String? {
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
```

```
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
```

```
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
                                ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
```

```
   ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
   ///     describing the messages to delete.
   ///
   /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
   func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                          deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
       let output = try await sqsClient.deleteMessageBatch(
           input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
       )

       if let failed = output.failed {
           print("\(failed.count) errors occurred deleting messages from the
queue.")
           for message in failed {
               print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
           }
       }
   }

   /// Called by ``main()`` to run the bulk of the example.
   func runAsync() async throws {
       let rowOfStars = String(repeating: "*", count: 75)

       print("""
             \(rowOfStars)
             Welcome to the cross-service messaging with topics and queues
example.
             In this workflow, you'll create an SNS topic, then create two SQS
             queues which will be subscribed to that topic.

             You can specify several options for configuring the topic, as well
as
             the queue subscriptions. You can then post messages to the topic
and
             receive the results on the queues.
             \(rowOfStars)\n
             """
       )

       // 0. Create SNS and SQS clients.
```

```
        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
 region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
 region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
                  supported.

                  Deduplication IDs are either set in the message or are
 automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic,
 any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
 deduplication ID (Y/N)? ")
            print(rowOfStars)
        }
```

```
        var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )
```

```
        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil
```

```
        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
 messages will
                be received in the queue. For information about message
 filtering, see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's
 subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
 queue: ",
```

```
                                sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil

        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }
```

```
        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
 publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.
\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a
 message group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based
 deduplication, you
                              must enter a deduplication ID. If other messages
 with the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
```

```
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue]
= [:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType:
"String", stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
```

```
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
```

```
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Senden und Empfangen von Nachrichtenstapeln mit Amazon SQS unter Verwendung eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen einer Amazon SQS-Warteschlange
- Senden Sie Stapel von Nachrichten an die Warteschlange.
- Empfangen Sie Stapel von Nachrichten aus der Warteschlange.
- Löschen Sie Stapel von Nachrichten aus der Warteschlange.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wie in den folgenden Beispielen gezeigt, können Sie Batch-Nachrichtenoperationen mit Amazon SQS auf zwei verschiedene Arten mit dem AWS SDK for Java 2.x durchführen.

SendRecvBatch.javaverwendet explizite Batch-Operationen. Sie erstellen Nachrichtenstapel manuell und rufen `sendMessageBatch()` und `deleteMessageBatch()` direkt auf. Sie bearbeiten auch Batch-Antworten, einschließlich fehlgeschlagener Nachrichten. Dieser Ansatz gibt Ihnen die volle Kontrolle über die Batchgröße und die Fehlerbehandlung. Es erfordert jedoch mehr Code, um die Logik der Batchverarbeitung zu verwalten.

SimpleProducerConsumer.javaverwendet die `SqsAsyncBatchManager` High-Level-Bibliothek für die automatische Batchverarbeitung von Anfragen. Sie führen individuelle Aufrufe von `sendMessage()` und `deleteMessage()` durch und verwenden dieselben Methodensignaturen wie der Standardclient. Das SDK puffert diese Aufrufe automatisch und sendet sie als Batch-Operationen. Dieser Ansatz erfordert nur minimale Codeänderungen und bietet gleichzeitig Leistungsvorteile bei der Batchverarbeitung.

Verwenden Sie die explizite Batchverarbeitung, wenn Sie eine genaue Kontrolle über die Batchzusammensetzung und die Fehlerbehandlung benötigen. Verwenden Sie die

automatisches Batchverarbeitung, wenn Sie die Leistung mit minimalen Codeänderungen optimieren möchten.

SendRecvBatch.java - Verwendet explizite Batch-Operationen mit Nachrichten.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
 sent.
     * The caller must inspect the Successful and Failed lists in the response
 and
     * resend any failed messages.
     *
     * @param queueUrl  The URL of the queue to receive the messages.
     * @param messages  The messages to send to the queue. Each message contains
 a body and attributes.
     * @return The response from SQS that contains the list of successful and
 failed messages.
     */
    public static SendMessageBatchResponse sendMessages(
            String queueUrl, List<MessageEntry> messages) {

        try {
            List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                MessageEntry message = messages.get(i);
                entries.add(SendMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .messageBody(message.getBody())
                        .messageAttributes(message.getAttributes())
                        .build());
            }
```

```
            SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

            if (!response.successful().isEmpty()) {
                for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                    LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
                }
            }

            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10
by SQS).
     *                   The actual number of messages received might be less.
     * @param waitTime   The maximum time to wait (in seconds) before returning.
When
```

```
    *                   this number is greater than zero, long polling is used.
This
    *                   can result in reduced costs and fewer false empty
responses.
    * @return The list of Message objects received. These each contain the body
    *         of the message and metadata and custom attributes.
    */
   public static List<Message> receiveMessages(String queueUrl, int maxNumber,
int waitTime) {
       try {
           ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                   .queueUrl(queueUrl)
                   .maxNumberOfMessages(maxNumber)
                   .waitTimeSeconds(waitTime)
                   .messageAttributeNames("All")
                   .build();

           List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

           for (Message message : messages) {
               LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
           }

           return messages;

       } catch (SqsException e) {
           LOGGER.error("Couldn't receive messages from queue: {}", queueUrl,
e);
           throw e;
       }
   }

   /**
    * Delete a batch of messages from a queue in a single request.
    *
    * @param queueUrl  The URL of the queue from which to delete the messages.
    * @param messages  The list of messages to delete.
    * @return The response from SQS that contains the list of successful and
failed
    *         message deletions.
    */
```

```
   public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
       try {
           List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               entries.add(DeleteMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .receiptHandle(messages.get(i).receiptHandle())
                       .build());
           }

           DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

           if (!response.successful().isEmpty()) {
               for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
               }
           }

           if (!response.failed().isEmpty()) {
               for (BatchResultErrorEntry errorEntry : response.failed()) {
                   LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
               }
           }

           return response;

       } catch (SqsException e) {
           LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
           throw e;
       }
   }
```

```
    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }

    /**
     * Shows how to:
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
     * * Reassemble the lines of the file and verify they match the original
file.
     */
    public static void usageDemo() {
        LOGGER.info("-".repeat(88));
        LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
        LOGGER.info("-".repeat(88));

        String queueUrl = null;
        try {
            // Create a queue for the demo.
            String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
            CreateQueueRequest createRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
```

```
            queueUrl = sqsClient.createQueue(createRequest).queueUrl();
            LOGGER.info("Created queue: {}", queueUrl);

            try (InputStream inputStream =
 SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                 BufferedReader reader = new BufferedReader(new
 InputStreamReader(inputStream))) {

                List<String> lines = reader.lines().toList();

                // Send file lines in batches.
                int batchSize = 10;
                LOGGER.info("Sending file lines in batches of {} as messages.",
 batchSize);

                for (int i = 0; i < lines.size(); i += batchSize) {
                    List<MessageEntry> messageBatch = new ArrayList<>();

                    for (int j = i; j < Math.min(i + batchSize, lines.size()); j+
+) {
                        String line = lines.get(j);
                        if (line == null || line.trim().isEmpty()) {
                            continue; // Skip empty lines.
                        }

                        Map<String, MessageAttributeValue> attributes = new
 HashMap<>();
                        attributes.put("line", MessageAttributeValue.builder()
                                .dataType("String")
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
 attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
```

```
                LOGGER.info("Receiving, handling, and deleting messages in
batches of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");

                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
```

```
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
 DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
 }
```

SimpleProducerConsumer.java - Verwendet automatisches Batching von Nachrichten.

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;
```

```
import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for
 Amazon SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that
 provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
 * - Automatic batching: The SDK automatically buffers individual requests and
 sends them
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-
blocking execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better
 connection reuse
 *
 * This example compares:
```

```
 * 1. Single-message operations using SqsAsyncClient directly
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance
 measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch
 processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
 LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
     * Prompts for queue name, thread counts, batch size, message size, and
 runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
```

```
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl,
batchSize,
                        messageSizeByte, producedCount, stop);
```

```
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl,
consumedCount, stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl,
batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }

        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }
```

```
    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS,
providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility
while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
```

```
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends messages until the stop flag is set.
         *
         * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call
per message.
         * This approach provides baseline performance metrics for comparison
with batching.
         * Each call blocks until the individual message is sent, demonstrating
traditional
         * one-request-per-operation behavior.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
```

```
    * Sends messages using SqsAsyncBatchManager for automatic request batching
and optimization.
    *
    * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
    * The SqsAsyncBatchManager automatically buffers individual sendMessage()
calls and
    * sends them as batches when thresholds are reached:
    * - maxBatchSize: Maximum 10 messages per batch (default)
    * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
    *
    * Key advantages of the batching approach:
    * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
    * - Automatic optimization: No code changes needed to benefit from batching
    * - Transparent buffering: The SDK handles batching logic internally
    * - Reduced API calls: Multiple messages sent in single batch requests
    * - Lower costs: Fewer API calls result in reduced SQS charges
    * - Higher throughput: Batch operations process significantly more messages
per second
    */
   private static class BatchProducer extends Thread {
       final SqsAsyncBatchManager batchManager;
       final String queueUrl;
       final int batchSize;
       final AtomicInteger producedCount;
       final AtomicBoolean stop;
       final String theMessage;

       /**
        * Creates a producer thread for batch operations.
        *
        * @param batchManager the batch manager for efficient message sending
        * @param queueUrl the URL of the target queue
        * @param batchSize the number of messages to send per batch
        * @param messageSizeByte the size of messages to generate
        * @param producedCount shared counter for tracking sent messages
        * @param stop shared flag to signal thread termination
        */
       BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                     int messageSizeByte, AtomicInteger producedCount,
                     AtomicBoolean stop) {
```

```
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
library.
         *
         * Notice how batchManager.sendMessage() uses the exact same method
signature
         * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
         * the drop-in replacement capability of the SqsAsyncBatchManager.
         *
         * The SDK automatically:
         * - Buffers individual sendMessage() calls internally
         * - Groups them into batch requests when thresholds are met
         * - Sends SendMessageBatchRequest operations to SQS
         * - Returns individual CompletableFuture responses for each message
         *
         * This transparent batching provides significant performance
improvements
         * without requiring changes to application logic or error handling
patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Send multiple messages using the high-level batch manager.
                    // Each batchManager.sendMessage() call uses identical syntax
to
                    // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                    // The SDK automatically buffers these calls and sends them
as
                    // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                    // thresholds are reached, significantly improving
throughput.
                    for (int i = 0; i < batchSize; i++) {
```

```
                        CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                                SendMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .messageBody(theMessage)
                                        .build());

                        // Handle the response asynchronously
                        future.whenComplete((response, throwable) -> {
                            if (throwable == null) {
                                producedCount.incrementAndGet();
                            } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                      !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                log.error("BatchProducer: Failed to send
message", throwable);
                            }
                            // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                        });
                    }

                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Receives and deletes messages individually using SqsAsyncClient for
baseline measurement.
     *
     * This consumer demonstrates traditional single-message operations without
batching.
     * Each receiveMessage() and deleteMessage() call results in separate API
requests,
```

```
     * providing a performance baseline for comparison with batch operations.
     *
     * The method signatures are identical to SqsAsyncBatchManager methods:
     * - receiveMessage() matches batchManager.receiveMessage()
     * - deleteMessage() matches batchManager.deleteMessage()
     *
     * This API consistency allows easy migration to the high-level batching
 library.
     */
    private static class Consumer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for receiving messages
         * @param queueUrl the URL of the source queue
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
 consumedCount,
                 AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using traditional single-
request operations.
         *
         * Uses SqsAsyncClient methods directly:
         * - receiveMessage(): One API call per receive operation
         * - deleteMessage(): One API call per delete operation
         *
         * This approach demonstrates the baseline performance without batching
 optimization.
         * Compare these method calls with the identical signatures used in
 BatchConsumer
```

```
        * to see how the high-level batching library maintains API
compatibility.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   try {
                       final ReceiveMessageResponse result =
sqsAsyncClient.receiveMessage(
                               ReceiveMessageRequest.builder()
                                       .queueUrl(queueUrl)
                                       .build()).join();

                       if (!result.messages().isEmpty()) {
                           final Message m = result.messages().get(0);
                           // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                   .queueUrl(queueUrl)
                                   .receiptHandle(m.receiptHandle())
                                   .build()).join();
                           consumedCount.incrementAndGet();
                       }
                   } catch (SdkException |
java.util.concurrent.CompletionException e) {
                       log.error(e.getMessage());
                   }
               }
           } catch (SdkException | java.util.concurrent.CompletionException e) {
               // Handle both SdkException and CompletionException from async
operations.
               // If this unlikely condition occurs, stop.
               log.error("Consumer: " + e.getMessage());
               System.exit(1);
           }
       }
   }

   /**
    * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
    *
```

```
    * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
    * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
    *
    * 1. Receive optimization: Maintains an internal buffer of messages fetched
in the
    *    background, so receiveMessage() calls return immediately from the
buffer
    * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
    *    as DeleteMessageBatchRequest operations when thresholds are reached
    *
    * Key features:
    * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
    *   as SqsAsyncClient methods, making this a true drop-in replacement
    * - Background fetching: The batch manager continuously fetches messages to
keep
    *   the internal buffer populated, reducing receive latency
    * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
    *   sent as batch operations (up to 10 per batch, 200ms frequency)
    * - Transparent optimization: No application logic changes needed to benefit
    *
    * Performance benefits:
    * - Reduced API calls through automatic batching of delete operations
    * - Lower latency for receives due to background message buffering
    * - Higher overall throughput with fewer network round trips
    */
   private static class BatchConsumer extends Thread {
       final SqsAsyncBatchManager batchManager;
       final String queueUrl;
       final int batchSize;
       final AtomicInteger consumedCount;
       final AtomicBoolean stop;

       /**
        * Creates a consumer thread for batch operations.
        *
        * @param batchManager the batch manager for efficient message processing
        * @param queueUrl the URL of the source queue
        * @param batchSize the maximum number of messages to receive per batch
        * @param consumedCount shared counter for tracking processed messages
```

```
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level
batching library.
         *
         * Demonstrates the key advantage of SqsAsyncBatchManager: identical
method signatures
         * with automatic optimization. Notice how:
         *
         * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
         * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
         *
         * Behind the scenes, the batch manager:
         * 1. Maintains an internal message buffer populated by background
fetching
         * 2. Returns messages immediately from the buffer (reduced latency)
         * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
         * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
         *
         * This provides significant performance improvements with zero code
changes
         * compared to traditional SqsAsyncClient usage patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Receive messages using the high-level batch manager.
                    // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
```

```
                    // but benefits from internal message buffering for improved
performance.
                    final ReceiveMessageResponse result =
batchManager.receiveMessage(
                            ReceiveMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .maxNumberOfMessages(Math.min(batchSize, 10))
                                    .build()).join();

                    if (!result.messages().isEmpty()) {
                        final List<Message> messages = result.messages();

                        // Delete messages using the batch manager.
                        // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                        // but the SDK automatically buffers these calls and
sends them
                        // as DeleteMessageBatchRequest operations for optimal
performance.
                        for (Message message : messages) {
                            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                    DeleteMessageRequest.builder()
                                            .queueUrl(queueUrl)

.receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                            !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
                    }
```

```
                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Logs throughput statistics every second until stopped.
         *
```

```
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching
 strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, um Amazon-SQS-Nachrichtenfunktionen zu umschließen.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError

import queue_wrapper

logger = logging.getLogger(__name__)
sqs = boto3.resource("sqs")

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not
 sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and
 failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
```

```
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response



def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual
 number
                        of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning.
 When
                       this number is greater than zero, long polling is used.
 This
                       can result in reduced costs and fewer false empty
 responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
```

```
    else:
        return messages



def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and
 failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

Verwenden Sie die Wrapper-Funktionen, um Nachrichten stapelweise zu senden und zu empfangen.

```
def usage_demo():
```

```
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """

    def pack_message(msg_path, msg_body, msg_line):
        return {
            "body": msg_body,
            "attributes": {
                "path": {"StringValue": msg_path, "DataType": "String"},
                "line": {"StringValue": str(msg_line), "DataType": "String"},
            },
        }

    def unpack_message(msg):
        return (
            msg.message_attributes["path"]["StringValue"],
            msg.body,
            int(msg.message_attributes["line"]["StringValue"]),
        )

    print("-" * 88)
    print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
    print("-" * 88)

    queue = queue_wrapper.create_queue("sqs-usage-demo-message-wrapper")

    with open(__file__) as file:
        lines = file.readlines()

    line = 0
    batch_size = 10
    received_lines = [None] * len(lines)
    print(f"Sending file lines in batches of {batch_size} as messages.")
    while line < len(lines):
        messages = [
            pack_message(__file__, lines[index], index)
            for index in range(line, min(line + batch_size, len(lines)))
        ]
        line = line + batch_size
        send_messages(queue, messages)
```

```
        print(".", end="")
        sys.stdout.flush()
    print(f"Done. Sent {len(lines) - 1} messages.")

    print(f"Receiving, handling, and deleting messages in batches of
{batch_size}.")
    more_messages = True
    while more_messages:
        received_messages = receive_messages(queue, batch_size, 2)
        print(".", end="")
        sys.stdout.flush()
        for message in received_messages:
            path, body, line = unpack_message(message)
            received_lines[line] = body
        if received_messages:
            delete_messages(queue, received_messages)
        else:
            more_messages = False
    print("Done.")

    if all([lines[index] == received_lines[index] for index in
range(len(lines))]):
        print(f"Successfully reassembled all file lines!")
    else:
        print(f"Uh oh, some lines were missed!")

    queue.delete()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage

- SendMessage
- SendMessageBatch

Benutze die AWS Message Processing Framework für.NET zum Veröffentlichen und Empfangen von Amazon SQS SQS-Nachrichten

Das folgende Codebeispiel zeigt, wie Sie mithilfe des AWS Message Processing Framework for.NET Anwendungen erstellen, die Amazon SQS SQS-Nachrichten veröffentlichen und empfangen.

.NET

SDK für .NET

Stellt ein Tutorial für das AWS Message Processing Framework für .NET bereit. Das Tutorial erstellt eine Webanwendung, mit der der Benutzer Amazon-SQS-Nachricht veröffentlichen kann, sowie eine Befehlszeilenanwendung, die die Nachricht empfängt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Tutorial im AWS SDK für .NET Entwicklerhandbuch und im Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SQS

Verwenden der Amazon SQS Java Messaging Library, um mit der Java-Message-Service(JMS)-Schnittstelle zu arbeiten

Das folgende Codebeispiel zeigt, wie Sie die Amazon SQS Java Messaging Library verwenden, um mit der JMS-Schnittstelle zu arbeiten.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die folgenden Beispiele funktionieren mit standardmäßigen Amazon-SQS-Warteschlangen und beinhalten:

- Senden einer Testnachricht
- Synchrones Empfangen von Nachrichten
- Asynchrones Empfangen von Nachrichten
- Empfangen von Nachrichten im CLIENT_ACKNOWLEDGE-Modus.
- Empfangen von Nachrichten im UNORDERED_ACKNOWLEDGE-Modus.
- Verwenden von Spring, um Abhängigkeiten einzufügen.
- Eine Dienstprogramm-Klasse, die allgemeine Methoden bereitstellt, die in den anderen Beispielen verwendet werden.

Weitere Informationen zur Verwendung von JMS mit Amazon SQS finden Sie im Entwicklerhandbuch für Amazon SQS.

Senden einer Testnachricht

```
   /**
    * This method establishes a connection to a standard Amazon SQS queue using
the Amazon SQS
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery,
and automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending
messages to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection()) {

           // Create the queue if needed.
```

```
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session that uses the JMS auto-acknowledge mode.
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

            createAndSendMessages(session, producer);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method reads text input from the keyboard and sends each line as a
separate message
     * to a standard Amazon SQS queue using the Amazon SQS Java Messaging
Library. It continues
     * to accept input until the user enters an empty line, using JMS (Java
Message Service) API to
     * handle the message delivery.
     *
     * @param session The JMS session used to create messages
     * @param producer The JMS message producer used to send messages to the
queue
     */
    private static void createAndSendMessages(Session session, MessageProducer
producer) {
        BufferedReader inputReader = new BufferedReader(
                new InputStreamReader(System.in, Charset.defaultCharset()));

        try {
            String input;
            while (true) {
                LOGGER.info("Enter message to send (leave empty to exit): ");
                input = inputReader.readLine();
                if (input == null || input.isEmpty()) break;

                TextMessage message = session.createTextMessage(input);
                producer.send(message);
                LOGGER.info("Send message {}", message.getJMSMessageID());
            }
        } catch (EOFException e) {
```

```
            // Just return on EOF
        } catch (IOException e) {
            LOGGER.error("Failed reading input: {}", e.getMessage(), e);
        } catch (JMSException e) {
            LOGGER.error("Failed sending message: {}", e.getMessage(), e);
        }
    }
```

Synchrones Empfangen von Nachrichten

```
    /**
     * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
     * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
     * waits for messages to arrive, and processes them one at a time. The method
handles all
     * necessary setup and cleanup of messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageSync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
```

```
            connection.start();

            receiveMessages(consumer);
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method continuously checks for new messages from a standard Amazon
SQS queue using
     * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
     * it using JMS (Java Message Service), and confirms receipt. The method
stops checking for
     * messages after 20 seconds of no activity.
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}",
message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

Asynchrones Empfangen von Nachrichten

```
   /**
    * This method sets up automatic message handling for a standard Amazon SQS
queue using the
    * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
    * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
    * messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
   public static void doReceiveMessageAsync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           try {
               // Create a consumer for the queue.
               MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
               // Provide an implementation of the MessageListener interface,
which has a single 'onMessage' method.
               // We use a lambda expression for the implementation.
               consumer.setMessageListener(message -> {
                   try {
                       SqsJmsExampleUtils.handleMessage(message);
                       message.acknowledge();
                   } catch (JMSException e) {
```

```
                        LOGGER.error("Error processing message: {}",
e.getMessage());
                    }
                });
                // Start receiving incoming messages.
                connection.start();
                LOGGER.info("Waiting for messages...");
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

Empfangen von Nachrichten im CLIENT_ACKNOWLEDGE-Modus.

```
    /**
     * This method demonstrates how message acknowledgment affects message
processing in a standard
     * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
     * then shows how JMS (Java Message Service) client acknowledgment mode
handles both explicit
     * and implicit message confirmations, including how acknowledging one
message can automatically
     * acknowledge previous messages.
     *
     * @throws JMSException If there is a problem with the messaging operations
     */
    public static void doReceiveMessagesSyncClientAcknowledge() throws
JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );
```

```
        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with client acknowledge mode.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            // Create a producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open the connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout",
e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the
second one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based
on the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from
the queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after
processing
```

```
    * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
    */
   private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
       // Receive a message.
       Message message = consumer.receive(TIME_OUT_MILLIS);

       if (message == null) {
           LOGGER.info("Queue is empty!");
       } else {
           // Since this queue has only text messages, cast the message object
and print the text.
           LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

           // Acknowledge the message if asked.
           if (acknowledge) message.acknowledge();
       }
   }
```

Empfangen von Nachrichten im UNORDERED_ACKNOWLEDGE-Modus.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message
acknowledges all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws
JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
```

```
        );

        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try( SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with unordered acknowledge mode.
            Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

            // Create the producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open a connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout",
e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, and we'll get the
first message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its
own separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS
API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
```

```
            // Since this queue has only text messages, cast the message object
 and print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
 message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Verwenden von Spring, um Abhängigkeiten einzufügen.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;

import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java
 Messaging Library
 * with Spring Framework integration. This example connects to a standard Amazon
 SQS message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);
```

```
   /**
    * Demonstrates sending and receiving messages through a standard Amazon SQS
message queue
    * using Spring Framework configuration. This method loads connection
settings from an XML file,
    * establishes a messaging session using the Amazon SQS Java Messaging
Library, and processes
    * messages using JMS (Java Message Service) operations. If the queue doesn't
exist, it will
    * be created automatically.
    *
    * @param args Command line arguments (not used)
    */
   public static void main(String[] args) {

       URL resource =
SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
       File springFile = new File(resource.getFile());
       if (!springFile.exists() || !springFile.canRead()) {
           LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or
isn't readable.");
           System.exit(1);
       }

       try (FileSystemXmlApplicationContext context =
                   new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

           Connection connection;
           try {
               connection = context.getBean(Connection.class);
           } catch (NoSuchBeanDefinitionException e) {
               LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
               System.exit(2);
               return;
           }

           String queueName;
           try {
               queueName = context.getBean("queueName", String.class);
           } catch (NoSuchBeanDefinitionException e) {
```

```
                LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
                System.exit(3);
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection)
connection, queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
(Java Message
     * Service) operations. The method will stop checking for messages after 15
seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
```

```
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Spring-Bean-Definitionen.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/
schema/beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>
```

```
    <bean id="clientBuilder"
 class="software.amazon.awssdk.services.sqs.SqsClient" factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

Eine Dienstprogramm-Klasse, die allgemeine Methoden bereitstellt, die in den anderen Beispielen verwendet werden.

```
package com.example.sqs.jms;
```

```
import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple
 Queue Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common
 operations for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if
 necessary. The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
     * @param queueName The name of the queue to verify or create
     * @param visibilityTimeout The duration in seconds that messages will be
 hidden after being received
     * @throws JMSException If there is an error accessing or creating the queue
     */
    public static void ensureQueueExists(SQSConnection connection, String
 queueName, Long visibilityTimeout) throws JMSException {
        AmazonSQSMessagingClientWrapper client =
 connection.getWrappedAmazonSQSClient();
```

```
      /* In most cases, you can do this with just a 'createQueue' call, but
'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where
the queue
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
    * @param session The active messaging session used to create and send the
message
    * @param queueName The name of the queue where the message will be sent
    */
   public static void sendTextMessage(Session session, String queueName) {
       // Rest of implementation...

       try {
           MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
           Message message = session.createTextMessage("Hello world!");
           producer.send(message);
       } catch (JMSException e) {
           LOGGER.error( "Error receiving from SQS", e );
       }
   }

   /**
    * This method processes incoming messages and logs their content based on
the message type.
    * It supports text messages, binary data, and Java objects.
```

```
     *
     * @param message The message to be processed and logged
     * @throws JMSException If there is an error reading the message content
     */
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to
256k so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
    public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
        // Rest of implementation...
        try {
            // Create a consumer for the queue.
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
            // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
            // We use a lambda expression for the implementation.
            consumer.setMessageListener(message -> {
```

```
                try {
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSException e) {
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * This method performs cleanup operations after message processing is
complete. It receives
     * any messages in the specified queue, removes the message queue and closes
all
     * active connections to prevent resource leaks.
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
```

```
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);

            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain
hidden after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long
secondsToWait, Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
```

```
                SQSConnectionFactory connectionFactory = new
 SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
 connectionFactory.createConnection()) {
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " +
 i + "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - DeleteQueue

Arbeiten Sie mit Warteschlangen-Tags und Amazon SQS mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Tagging-Operationen mit Amazon SQS durchführen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im folgenden Beispiel werden Tags für eine Warteschlange erstellt, Tags aufgelistet und ein Tag entfernt.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER =
 LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
```

```
            queueUrl = sqsClient.createQueue(b ->
b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
```

```
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a
Consumer<UntagQueueRequest.Builder> parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListQueueTags
  - TagQueue
  - UntagQueue

Serverless-Beispiele für Amazon SQS

Die folgenden Codebeispiele zeigen, wie Amazon SQS mit AWS SDKs verwendet wird.

Beispiele

- Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser
- Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Die folgenden Codebeispiele zeigen, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }
```

```
    }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Konsumieren eines SQS-Ereignisses mit Lambda unter Verwendung. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Konsumieren eines SQS-Ereignisses mit Lambda unter Verwendung. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";
```

```
export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
```

```
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
```

```
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion
 time.
        .without_time()
```

```
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Die folgenden Codebeispiele zeigen, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
```

```
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Go

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batch-Elementfehlern mit Lambda unter Verwendung von. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Melden von SQS-Batch-Elementfehlern mit Lambda unter Verwendung von. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord }
 from 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];
```

```
    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

PHP

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier":
 record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
```

```
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) ->
 Result<SqsBatchResponse, Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

Codebeispiele für Step Functions mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Step Functions mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Step Functions – Weitere Informationen zu Step Functions.
- API-Referenz für Step Functions – Details zu allen verfügbaren Step-Functions-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Step Functions mit AWS SDKs
  - Hello Step Functions
  - Erlernen Sie die Grundlagen von Step Functions mit einem AWS SDK
  - Aktionen für Step Functions mit AWS SDKs
    - Verwenden Sie CreateActivitymit einem AWS SDK
    - Verwenden Sie CreateStateMachinemit einem AWS SDK
    - Verwenden Sie DeleteActivitymit einem AWS SDK
    - Verwenden Sie DeleteStateMachinemit einem AWS SDK
    - Verwenden Sie DescribeExecutionmit einem AWS SDK
    - Verwenden Sie DescribeStateMachinemit einem AWS SDK
    - Verwenden Sie GetActivityTaskmit einem AWS SDK
    - Verwenden Sie ListActivitiesmit einem AWS SDK
    - Verwenden Sie ListExecutionsmit einem AWS SDK
    - Verwenden Sie ListStateMachinesmit einem AWS SDK
    - Verwenden Sie SendTaskSuccessmit einem AWS SDK
    - Verwenden Sie StartExecutionmit einem AWS SDK
- Szenarien für Step Functions mit AWS SDKs
  - Erstellen einer Messenger-Anwendung mit Step Functions
  - Erste Schritte mit der Workflow-Orchestrierung

- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Grundlegende Beispiele für Step Functions mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Step Functions mit AWS SDKs verwendet werden.

Beispiele

- Hello Step Functions
- Erlernen Sie die Grundlagen von Step Functions mit einem AWS SDK
- Aktionen für Step Functions mit AWS SDKs
  - Verwenden Sie CreateActivitymit einem AWS SDK
  - Verwenden Sie CreateStateMachinemit einem AWS SDK
  - Verwenden Sie DeleteActivitymit einem AWS SDK
  - Verwenden Sie DeleteStateMachinemit einem AWS SDK
  - Verwenden Sie DescribeExecutionmit einem AWS SDK
  - Verwenden Sie DescribeStateMachinemit einem AWS SDK
  - Verwenden Sie GetActivityTaskmit einem AWS SDK
  - Verwenden Sie ListActivitiesmit einem AWS SDK
  - Verwenden Sie ListExecutionsmit einem AWS SDK
  - Verwenden Sie ListStateMachinesmit einem AWS SDK
  - Verwenden Sie SendTaskSuccessmit einem AWS SDK
  - Verwenden Sie StartExecutionmit einem AWS SDK

Hello Step Functions

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Step Functions beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

public class HelloStepFunctions
{
    static async Task Main()
    {
        var stepFunctionsClient = new AmazonStepFunctionsClient();

        Console.Clear();
        Console.WriteLine("Welcome to AWS Step Functions");
        Console.WriteLine("Let's list up to 10 of your state machines:");
        var stateMachineListRequest = new ListStateMachinesRequest { MaxResults =
 10 };

        // Get information for up to 10 Step Functions state machines.
        var response = await
 stepFunctionsClient.ListStateMachinesAsync(stateMachineListRequest);

        if (response.StateMachines.Count > 0)
        {
            response.StateMachines.ForEach(stateMachine =>
            {
                Console.WriteLine($"State Machine
 Name: {stateMachine.Name}\tAmazon Resource Name (ARN):
 {stateMachine.StateMachineArn}");
            });
        }
        else
```

```
        {
            Console.WriteLine("\tNo state machines were found.");
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Java-Version von Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
```

```
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
```

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_stepfunctions(stepfunctions_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Step Functions client and
 list
    the state machines in your account. This list might be empty if you haven't
 created
    any state machines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param stepfunctions_client: A Boto3 Step Functions Client object.
    """
    print("Hello, Step Functions! Let's list up to 10 of your state machines:")
    state_machines = stepfunctions_client.list_state_machines(maxResults=10)
    for sm in state_machines["stateMachines"]:
        print(f"\t{sm['name']}: {sm['stateMachineArn']}")


if __name__ == "__main__":
    hello_stepfunctions(boto3.client("stepfunctions"))
```

- Einzelheiten zur API finden Sie ListStateMachinesin AWS SDK for Python (Boto3) API Reference.

Erlernen Sie die Grundlagen von Step Functions mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
global using System.Text.Json;
global using Amazon.StepFunctions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using StepFunctionsActions;
global using LogLevel = Microsoft.Extensions.Logging.LogLevel;



using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.StepFunctions.Model;

namespace StepFunctionsBasics;

public class StepFunctionsBasics
{
    private static ILogger _logger = null!;
    private static IConfigurationRoot _configuration = null!;
    private static IAmazonIdentityManagementService _iamService = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Step Functions.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonStepFunctions>()
                    .AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<StepFunctionsWrapper>()
            )
            .Build();

        _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<StepFunctionsBasics>();

        // Load configuration settings.
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var activityName = _configuration["ActivityName"];
        var stateMachineName = _configuration["StateMachineName"];

        var roleName = _configuration["RoleName"];
        var repoBaseDir = _configuration["RepoBaseDir"];
        var jsonFilePath = _configuration["JsonFilePath"];
        var jsonFileName = _configuration["JsonFileName"];

        var uiMethods = new UiMethods();
        var stepFunctionsWrapper =
 host.Services.GetRequiredService<StepFunctionsWrapper>();

        _iamService =
 host.Services.GetRequiredService<IAmazonIdentityManagementService>();

        // Load definition for the state machine from a JSON file.
        var stateDefinitionJson = File.ReadAllText($"{repoBaseDir}{jsonFilePath}
{jsonFileName}");

        Console.Clear();
        uiMethods.DisplayOverview();
        uiMethods.PressEnter();
```

```
        uiMethods.DisplayTitle("Create activity");
        Console.WriteLine("Let's start by creating an activity.");
        string activityArn;
        string stateMachineArn;

        // Check to see if the activity already exists.
        var activityList = await stepFunctionsWrapper.ListActivitiesAsync();
        var existingActivity = activityList.FirstOrDefault(activity =>
activity.Name == activityName);
        if (existingActivity is not null)
        {
            activityArn = existingActivity.ActivityArn;
            Console.WriteLine($"Activity, {activityName}, already exists.");
        }
        else
        {
            activityArn = await
stepFunctionsWrapper.CreateActivity(activityName);
        }

        // Swap the placeholder in the JSON file with the Amazon Resource Name
(ARN)
        // of the recently created activity.
        var stateDefinition =
stateDefinitionJson.Replace("{{DOC_EXAMPLE_ACTIVITY_ARN}}", activityArn);

        uiMethods.DisplayTitle("Create state machine");
        Console.WriteLine("Now we'll create a state machine.");

        // Find or create an IAM role that can be assumed by Step Functions.
        var role = await GetOrCreateStateMachineRole(roleName);

        // See if the state machine already exists.
        var stateMachineList = await
stepFunctionsWrapper.ListStateMachinesAsync();
        var existingStateMachine =
            stateMachineList.FirstOrDefault(stateMachine => stateMachine.Name ==
stateMachineName);
        if (existingStateMachine is not null)
        {
            Console.WriteLine($"State machine, {stateMachineName}, already
exists.");
            stateMachineArn = existingStateMachine.StateMachineArn;
```

```
        }
        else
        {
            // Create the state machine.
            stateMachineArn =
                await stepFunctionsWrapper.CreateStateMachine(stateMachineName,
stateDefinition, role.Arn);
            uiMethods.PressEnter();
        }

        Console.WriteLine("The state machine has been created.");
        var describeStateMachineResponse = await
stepFunctionsWrapper.DescribeStateMachineAsync(stateMachineArn);


Console.WriteLine($"{describeStateMachineResponse.Name}\t{describeStateMachineResponse.S
        Console.WriteLine($"Current status:
{describeStateMachineResponse.Status}");
        Console.WriteLine($"Amazon Resource Name (ARN) of the role assumed by the
state machine: {describeStateMachineResponse.RoleArn}");

        var userName = string.Empty;
        Console.Write("Before we start the state machine, tell me what should
ChatSFN call you? ");
        userName = Console.ReadLine();

        // Keep asking until the user enters a string value.
        while (string.IsNullOrEmpty(userName))
        {
            Console.Write("Enter your name: ");
            userName = Console.ReadLine();
        }

        var executionJson = @"{""name"": """ + userName + @"""}";

        // Start the state machine execution.
        Console.WriteLine("Now we'll start execution of the state machine.");
        var executionArn = await
stepFunctionsWrapper.StartExecutionAsync(executionJson, stateMachineArn);
        Console.WriteLine("State machine started.");

        Console.WriteLine($"Thank you, {userName}. Now let's get started...");
        uiMethods.PressEnter();
```

```
        uiMethods.DisplayTitle("ChatSFN");

        var isDone = false;
        var response = new GetActivityTaskResponse();
        var taskToken = string.Empty;
        var userChoice = string.Empty;

        while (!isDone)
        {
            response = await
stepFunctionsWrapper.GetActivityTaskAsync(activityArn, "MvpWorker");
            taskToken = response.TaskToken;

            // Parse the returned JSON string.
            var taskJsonResponse = JsonDocument.Parse(response.Input);
            var taskJsonObject = taskJsonResponse.RootElement;
            var message = taskJsonObject.GetProperty("message").GetString();
            var actions =
taskJsonObject.GetProperty("actions").EnumerateArray().Select(x =>
x.ToString()).ToList();
            Console.WriteLine($"\n{message}\n");

            // Prompt the user for another choice.
            Console.WriteLine("ChatSFN: What would you like me to do?");
            actions.ForEach(action => Console.WriteLine($"\t{action}"));
            Console.Write($"\n{userName}, tell me your choice: ");
            userChoice = Console.ReadLine();
            if (userChoice?.ToLower() == "done")
            {
                isDone = true;
            }

            Console.WriteLine($"You have selected: {userChoice}");
            var jsonResponse = @"{""action"": """ + userChoice + @"""}";

            await stepFunctionsWrapper.SendTaskSuccessAsync(taskToken,
jsonResponse);
        }

        await stepFunctionsWrapper.StopExecution(executionArn);
        Console.WriteLine("Now we will wait for the execution to stop.");
        DescribeExecutionResponse executionResponse;
        do
        {
```

```
            executionResponse = await
stepFunctionsWrapper.DescribeExecutionAsync(executionArn);
        } while (executionResponse.Status == ExecutionStatus.RUNNING);

        Console.WriteLine("State machine stopped.");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("State machine executions");
        Console.WriteLine("Now let's take a look at the execution values for the
state machine.");

        // List the executions.
        var executions = await
stepFunctionsWrapper.ListExecutionsAsync(stateMachineArn);

        uiMethods.DisplayTitle("Step function execution values");
        executions.ForEach(execution =>
        {
            Console.WriteLine($"{execution.Name}\t{execution.StartDate} to
{execution.StopDate}");
        });

        uiMethods.PressEnter();

        // Now delete the state machine and the activity.
        uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the state machine...");

        await stepFunctionsWrapper.DeleteStateMachine(stateMachineArn);
        Console.WriteLine("State machine deleted.");

        Console.WriteLine("Deleting the activity...");
        await stepFunctionsWrapper.DeleteActivity(activityArn);
        Console.WriteLine("Activity deleted.");

        Console.WriteLine("The Amazon Step Functions scenario is now complete.");
    }

    static async Task<Role> GetOrCreateStateMachineRole(string roleName)
    {
        // Define the policy document for the role.
        var stateMachineRolePolicy = @"{
         ""Version"": ""2012-10-17"",
        ""Statement"": [{
```

```
            ""Sid"": """",
            ""Effect"": ""Allow"",
            ""Principal"": {
                ""Service"": ""states.amazonaws.com""},
            ""Action"": ""sts:AssumeRole""}]}";

        var role = new Role();
        var roleExists = false;

        try
        {
            var getRoleResponse = await _iamService.GetRoleAsync(new
 GetRoleRequest { RoleName = roleName });
            roleExists = true;
            role = getRoleResponse.Role;
        }
        catch (NoSuchEntityException)
        {
            // The role doesn't exist. Create it.
            Console.WriteLine($"Role, {roleName} doesn't exist. Creating it...");
        }

        if (!roleExists)
        {
            var request = new CreateRoleRequest
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = stateMachineRolePolicy,
            };

            var createRoleResponse = await _iamService.CreateRoleAsync(request);
            role = createRoleResponse.Role;
        }

        return role;
    }
}


namespace StepFunctionsBasics;

/// <summary>
/// Some useful methods to make screen display easier.
/// </summary>
```

```
public class UiMethods
{
    private readonly string _sepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Welcome to the AWS Step Functions Demo");

        Console.WriteLine("This example application will do the following:");
        Console.WriteLine("\t 1. Create an activity.");
        Console.WriteLine("\t 2. Create a state machine.");
        Console.WriteLine("\t 3. Start an execution.");
        Console.WriteLine("\t 4. Run the worker, then stop it.");
        Console.WriteLine("\t 5. List executions.");
        Console.WriteLine("\t 6. Clean up the resources created for the
 example.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue.");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter"></param>
    /// <returns></returns>
    private string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
```

```
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(_sepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(_sepBar);
    }
}
```

Definieren Sie eine Klasse, die Zustandsautomaten und Aktivitätsaktionen umschließt.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

/// <summary>
/// Wrapper that performs AWS Step Functions actions.
/// </summary>
public class StepFunctionsWrapper
{
    private readonly IAmazonStepFunctions _amazonStepFunctions;

    /// <summary>
    /// The constructor for the StepFunctionsWrapper. Initializes the
    /// client object passed to it.
    /// </summary>
    /// <param name="amazonStepFunctions">An initialized Step Functions client
 object.</param>
    public StepFunctionsWrapper(IAmazonStepFunctions amazonStepFunctions)
    {
        _amazonStepFunctions = amazonStepFunctions;
    }

    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
```

```
    /// <param name="activityName">The name for the new Step Functions
activity.</param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }


    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }


    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
```

```
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
 DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
 DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
 executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
 DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
```

```
    public async Task<DescribeStateMachineResponse>
DescribeStateMachineAsync(string StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }


    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }


    /// <summary>
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
        {
            var response = await
_amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
```

```
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }


    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
 stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
 stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }


    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
```

```
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
 ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }


    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the
 task.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
 taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
 SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</
param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
```

```
    public async Task<string> StartExecutionAsync(string executionJson, string
 stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
 _amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }


    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new
 StopExecutionRequest { ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution

- DescribeStateMachine
- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/
sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 1. Creates an activity.
```

```
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
 machine.
                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.
                jsonFile - The location of the chat_sfn_state_machine.json file.
 You can located it in resources/sample_files.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "states.amazonaws.com"
```

```
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
            """;


        Scanner sc = new Scanner(System.in);
        boolean action = false;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);
```

```
        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN,
stateMachineName, stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
with it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " +
runArn);
        List<String> myList;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is
complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
```

```
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has
succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
        try {
            SendTaskSuccessRequest successRequest =
SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String
actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest =
DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();
```

```
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn)
{
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The
status is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }
```

```
   public static String startWorkflow(SfnClient sfnClient, String
stateMachineArn, String jsonEx) {
       UUID uuid = UUID.randomUUID();
       String uuidValue = uuid.toString();
       try {
           StartExecutionRequest executionRequest =
StartExecutionRequest.builder()
               .input(jsonEx)
               .stateMachineArn(stateMachineArn)
               .name(uuidValue)
               .build();

           StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
           return response.executionArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }

   public static String createMachine(SfnClient sfnClient, String roleARN,
String stateMachineName, String json) {
       try {
           CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
               .definition(json)
               .name(stateMachineName)
               .roleArn(roleARN)
               .type(StateMachineType.STANDARD)
               .build();

           CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
           return response.stateMachineArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

```
    public static String createActivity(SfnClient sfnClient, String activityName)
 {
        try {
            CreateActivityRequest activityRequest =
 CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:
```

```
 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
 located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
```

```
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)
```

```
    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
 activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName,
 stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
```

```
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
```

```
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
```

```
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
```

```
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is
 ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
```

```
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class StateMachineScenario:
    """Runs an interactive scenario that shows how to get started using Step
 Functions."""

    def __init__(self, activity, state_machine, iam_client):
        """
        :param activity: An object that wraps activity actions.
        :param state_machine: An object that wraps state machine actions.
```

```
        :param iam_client: A Boto3 AWS Identity and Access Management (IAM)
client.
        """
        self.activity = activity
        self.state_machine = state_machine
        self.iam_client = iam_client
        self.state_machine_role = None

    def prerequisites(self, state_machine_role_name):
        """
        Finds or creates an IAM role that can be assumed by Step Functions.
        A role of this kind is required to create a state machine.
        The state machine used in this example does not call any additional
services,
        so it needs no additional permissions.

        :param state_machine_role_name: The name of the role.
        :return: Data about the role.
        """
        trust_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {"Service": "states.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        try:
            role = self.iam_client.get_role(RoleName=state_machine_role_name)
            print(f"Prerequisite IAM role {state_machine_role_name} already
exists.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                role = None
            else:
                logger.error(
                    "Couldn't get prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        if role is None:
            try:
                role = self.iam_client.create_role(
                    RoleName=state_machine_role_name,
                    AssumeRolePolicyDocument=json.dumps(trust_policy),
                )
            except ClientError as err:
                logger.error(
                    "Couldn't create prerequisite IAM role %s. Here's why: %s:
%s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        self.state_machine_role = role["Role"]

    def find_or_create_activity(self, activity_name):
        """
        Finds or creates a Step Functions activity.

        :param activity_name: The name of the activity.
        :return: The Amazon Resource Name (ARN) of the activity.
        """
        print("First, let's set up an activity and state machine.")
        activity_arn = self.activity.find(activity_name)
        if activity_arn is None:
            activity_arn = self.activity.create(activity_name)
            print(
                f"Activity {activity_name} created. Its Amazon Resource Name
(ARN) is "
                f"{activity_arn}."
            )
        else:
            print(f"Activity {activity_name} already exists.")
        return activity_arn

    def find_or_create_state_machine(
        self, state_machine_name, activity_arn, state_machine_file
    ):
        """
        Finds or creates a Step Functions state machine.
```

```
        :param state_machine_name: The name of the state machine.
        :param activity_arn: The ARN of an activity that is used as a step in the
state
                                machine. This ARN is injected into the state machine
                                definition that's used to create the state machine.
        :param state_machine_file: The path to a file containing the state
machine
                                      definition.
        :return: The ARN of the state machine.
        """
        state_machine_arn = self.state_machine.find(state_machine_name)
        if state_machine_arn is None:
            with open(state_machine_file) as state_machine_file:
                state_machine_def = state_machine_file.read().replace(
                    "{{DOC_EXAMPLE_ACTIVITY_ARN}}", activity_arn
                )
                state_machine_arn = self.state_machine.create(
                    state_machine_name,
                    state_machine_def,
                    self.state_machine_role["Arn"],
                )
            print(f"State machine {state_machine_name} created.")
        else:
            print(f"State machine {state_machine_name} already exists.")
        print("-" * 88)
        print(f"Here's some information about state machine
{state_machine_name}:")
        state_machine_info = self.state_machine.describe(state_machine_arn)
        for field in ["name", "status", "stateMachineArn", "roleArn"]:
            print(f"\t{field}: {state_machine_info[field]}")
        return state_machine_arn

    def run_state_machine(self, state_machine_arn, activity_arn):
        """
        Run the state machine. The state machine used in this example is a simple
        chat simulation. It contains an activity step in a loop that is used for
user
        interaction. When the state machine gets to the activity step, it waits
for
        an external application to get task data and submit a response. This
function
        acts as the activity application by getting task input and responding
with
        user input.
```

```
        :param state_machine_arn: The ARN of the state machine.
        :param activity_arn: The ARN of the activity used as a step in the state
machine.
        :return: The ARN of the run.
        """
        print(
            f"Let's run the state machine. It's a simplistic, non-AI chat
simulator "
            f"we'll call ChatSFN."
        )
        user_name = q.ask("What should ChatSFN call you? ", q.non_empty)
        run_input = {"name": user_name}
        print("Starting state machine...")
        run_arn = self.state_machine.start(state_machine_arn,
json.dumps(run_input))
        action = None
        while action != "done":
            activity_task = self.activity.get_task(activity_arn)
            task_input = json.loads(activity_task["input"])
            print(f"ChatSFN: {task_input['message']}")
            action = task_input["actions"][
                q.choose("What now? ", task_input["actions"])
            ]
            task_response = {"action": action}
            self.activity.send_task_success(
                activity_task["taskToken"], json.dumps(task_response)
            )
        return run_arn

    def finish_state_machine_run(self, run_arn):
        """
        Wait for the state machine run to finish, then print final status and
output.

        :param run_arn: The ARN of the run to retrieve.
        """
        print(f"Let's get the final output from the state machine:")
        status = "RUNNING"
        while status == "RUNNING":
            run_output = self.state_machine.describe_run(run_arn)
            status = run_output["status"]
            if status == "RUNNING":
                print(
```

```
                    "The state machine is still running, let's wait for it to
finish."
                )
                wait(1)
            elif status == "SUCCEEDED":
                print(f"ChatSFN: {json.loads(run_output['output'])['message']}")
            else:
                print(f"Run status: {status}.")

    def cleanup(
        self,
        state_machine_name,
        state_machine_arn,
        activity_name,
        activity_arn,
        state_machine_role_name,
    ):
        """
        Clean up resources created by this example.

        :param state_machine_name: The name of the state machine.
        :param state_machine_arn: The ARN of the state machine.
        :param activity_name: The name of the activity.
        :param activity_arn: The ARN of the activity.
        :param state_machine_role_name: The name of the role used by the state
machine.
        """
        if q.ask(
            "Do you want to delete the state machine, activity, and role created
for this "
            "example? (y/n) ",
            q.is_yesno,
        ):
            self.state_machine.delete(state_machine_arn)
            print(f"Deleted state machine {state_machine_name}.")
            self.activity.delete(activity_arn)
            print(f"Deleted activity {activity_name}.")
            self.iam_client.delete_role(RoleName=state_machine_role_name)
            print(f"Deleted role {state_machine_role_name}.")

    def run_scenario(self, activity_name, state_machine_name):
        print("-" * 88)
        print("Welcome to the AWS Step Functions state machines demo.")
        print("-" * 88)
```

```
        activity_arn = self.find_or_create_activity(activity_name)
        state_machine_arn = self.find_or_create_state_machine(
            state_machine_name,
            activity_arn,
            "../../../resources/sample_files/chat_sfn_state_machine.json",
        )
        print("-" * 88)
        run_arn = self.run_state_machine(state_machine_arn, activity_arn)
        print("-" * 88)
        self.finish_state_machine_run(run_arn)
        print("-" * 88)
        self.cleanup(
            state_machine_name,
            state_machine_arn,
            activity_name,
            activity_arn,
            self.state_machine_role["RoleName"],
        )

        print("-" * 88)
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        stepfunctions_client = boto3.client("stepfunctions")
        iam_client = boto3.client("iam")
        scenario = StateMachineScenario(
            Activity(stepfunctions_client),
            StateMachine(stepfunctions_client),
            iam_client,
        )
        scenario.prerequisites("doc-example-state-machine-chat")
        scenario.run_scenario("doc-example-activity", "doc-example-state-
machine")
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Aktionen von Zustandsautomaten umschließt.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in
 the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is
 assumed by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]


    def find(self, name):
        """
```

```
        Find a state machine by name. This requires listing the state machines
until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also
known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about
the run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]


    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

Definieren Sie eine Klasse, die Aktivitätsaktionen umschließt.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
```

```
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state
machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with
an
        activity step waits for the activity to get task data and then respond
with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included
in the
                           response to the get_activity_task action and must be
sent
```

```
                            without modification.
        :param task_response: The response data from the activity. This data is
                                received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen für Step Functions mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Step Functions Functions-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Step-Functions-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Step Functions mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Step Functions -API-Referenz.

Beispiele

- Verwenden Sie CreateActivitymit einem AWS SDK
- Verwenden Sie CreateStateMachinemit einem AWS SDK
- Verwenden Sie DeleteActivitymit einem AWS SDK
- Verwenden Sie DeleteStateMachinemit einem AWS SDK
- Verwenden Sie DescribeExecutionmit einem AWS SDK

- Verwenden Sie DescribeStateMachinemit einem AWS SDK
- Verwenden Sie GetActivityTaskmit einem AWS SDK
- Verwenden Sie ListActivitiesmit einem AWS SDK
- Verwenden Sie ListExecutionsmit einem AWS SDK
- Verwenden Sie ListStateMachinesmit einem AWS SDK
- Verwenden Sie SendTaskSuccessmit einem AWS SDK
- Verwenden Sie StartExecutionmit einem AWS SDK

Verwenden Sie **`CreateActivity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateActivity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
    /// <param name="activityName">The name for the new Step Functions
activity.</param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
```

```
    }
```

- Einzelheiten zur API finden Sie CreateActivityin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createActivity(SfnClient sfnClient, String activityName)
{
       try {
           CreateActivityRequest activityRequest =
CreateActivityRequest.builder()
               .name(activityName)
               .build();

           CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
           return response.activityArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie CreateActivityin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Einzelheiten zur API finden Sie CreateActivityin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
```

```
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]
```

- Einzelheiten zur API finden Sie CreateActivityin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
      DATA(lo_result) = lo_sfn->createactivity(
        iv_name = iv_name
      ).
      ov_activity_arn = lo_result->get_activityarn( ).
      MESSAGE 'Activity created successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfnactivityalrdyex.
      MESSAGE 'Activity already exists.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidname.
      MESSAGE 'Invalid activity name.' TYPE 'E'.
    CATCH /aws1/cx_sfnactivitylimitexcd.
      MESSAGE 'Activity limit exceeded.' TYPE 'E'.
  ENDTRY.
```

- Einzelheiten zur API finden Sie CreateActivityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateStateMachine`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `CreateStateMachine` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Workflow-Orchestrierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Create a Step Functions state machine.
/// </summary>
```

```
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }
```

- Einzelheiten zur API finden Sie CreateStateMachinein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createMachine(SfnClient sfnClient, String roleARN,
String stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
```

```
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie CreateStateMachinein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
```

```
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- Einzelheiten zur API finden Sie CreateStateMachinein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
```

```
        :param definition: The Amazon States Language definition of the steps in
the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is
assumed by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]
```

- Einzelheiten zur API finden Sie CreateStateMachinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_sfn->createstatemachine(
          iv_name = iv_name
```

```
          iv_definition = iv_definition
          iv_rolearn = iv_role_arn
        ).
        ov_state_machine_arn = lo_result->get_statemachinearn( ).
        MESSAGE 'State machine created successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinealrex.
        MESSAGE 'State machine already exists.' TYPE 'E'.
      CATCH /aws1/cx_sfninvaliddefinition.
        MESSAGE 'Invalid state machine definition.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidname.
        MESSAGE 'Invalid state machine name.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid role ARN.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateStateMachinein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteActivity`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteActivity` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a Step Machine activity.
```

```
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteActivityin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest =
DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DeleteActivityin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- Einzelheiten zur API finden Sie DeleteActivityin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""
```

```
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DeleteActivityin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_sfn->deleteactivity(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteActivityin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteStateMachine`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteStateMachine` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Workflow-Orchestrierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteStateMachinein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn)
{
```

```
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The
status is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }
```

- Einzelheiten zur API finden Sie DeleteStateMachinein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
```

```
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- Einzelheiten zur API finden Sie DeleteStateMachinein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
```

```
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DeleteStateMachinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sfn->deletestatemachine(
          iv_statemachinearn = iv_state_machine_arn
        ).
        MESSAGE 'State machine deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
      CATCH /aws1/cx_sfnvalidationex.
        MESSAGE 'Validation error occurred.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteStateMachinein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeExecution`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeExecution` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Workflow-Orchestrierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }
```

- Einzelheiten zur API finden Sie DescribeExecutionin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has
succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
```

- Einzelheiten zur API finden Sie DescribeExecutionin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
```

```
}
```

- Einzelheiten zur API finden Sie DescribeExecutionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DescribeExecutionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_sfn->describeexecution(
      iv_executionarn = iv_execution_arn
    ).
    MESSAGE 'Execution described successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnexecdoesnotexist.
    MESSAGE 'Execution does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid execution ARN.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeExecutionin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeStateMachine`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeStateMachine` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse>
 DescribeStateMachineAsync(string StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
 DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }
```

- Einzelheiten zur API finden Sie DescribeStateMachinein der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeStateMachine(SfnClient sfnClient, String
 stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
 DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
 sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
 response.name());
            System.out.println("The status of the State machine is " +
 response.status());
            System.out.println("The ARN value of the State machine is " +
 response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }
```

- Einzelheiten zur API finden Sie DescribeStateMachinein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
```

```
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is
 ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeStateMachinein der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
```

```
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DescribeStateMachinein AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sfn->describestatemachine(
          iv_statemachinearn = iv_state_machine_arn
        ).
        MESSAGE 'State machine described successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinedoes00.
        MESSAGE 'State machine does not exist.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeStateMachinein der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetActivityTask`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `GetActivityTask` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }
```

- Einzelheiten zur API finden Sie GetActivityTaskin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static List<String> getActivityTask(SfnClient sfnClient, String
actArn) {
       List<String> myList = new ArrayList<>();
       GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
           .activityArn(actArn)
           .build();

       GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
       myList.add(response.taskToken());
       myList.add(response.input());
       return myList;
   }
```

- Einzelheiten zur API finden Sie GetActivityTaskin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}
```

- Einzelheiten zur API finden Sie GetActivityTaskin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
```

```
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state
machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie GetActivityTaskin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_sfn->getactivitytask(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity task retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivitydoesnotex.
    MESSAGE 'Activity does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivityworkerlm00.
    MESSAGE 'Activity worker limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetActivityTaskin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListActivities`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListActivities` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
        {
            var response = await
_amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }
```

- Einzelheiten zur API finden Sie ListActivitiesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
```

```
                    .maxResults(10)
                    .build();

            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListActivitiesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
```

```
        }
    }
}
```

- Einzelheiten zur API finden Sie ListActivitiesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
 found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator =
 self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
```

```
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListActivitiesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_sfn->listactivities( ).
        DATA(lt_activities) = lo_result->get_activities( ).
        LOOP AT lt_activities INTO DATA(lo_activity).
          IF lo_activity->get_name( ) = iv_name.
            ov_activity_arn = lo_activity->get_activityarn( ).
            EXIT.
          ENDIF.
        ENDLOOP.
        MESSAGE 'Activities listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidtoken.
        MESSAGE 'Invalid pagination token.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListActivitiesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListExecutions`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListExecutions` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }
```

- Einzelheiten zur API finden Sie ListExecutionsin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void getExeHistory(SfnClient sfnClient, String exeARN) {
        try {
            GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
                    .executionArn(exeARN)
                    .maxResults(10)
                    .build();

            GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
            List<HistoryEvent> events = historyResponse.events();
            for (HistoryEvent event : events) {
                System.out.println("The event type is " +
event.type().toString());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ListExecutionsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListExecutionsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`ListStateMachines`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListStateMachines` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }
```

- Einzelheiten zur API finden Sie ListStateMachinesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
```

```
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
```

```
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- Einzelheiten zur API finden Sie ListStateMachinesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Suchen Sie eine Zustandsmaschine anhand ihres Namens, indem Sie die Liste der Zustandsmaschinen für das Konto durchsuchen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
```

```
    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines
until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListStateMachinesin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_sfn->liststatemachines( ).
        DATA(lt_state_machines) = lo_result->get_statemachines( ).
        LOOP AT lt_state_machines INTO DATA(lo_state_machine).
          IF lo_state_machine->get_name( ) = iv_name.
            ov_state_machine_arn = lo_state_machine->get_statemachinearn( ).
            EXIT.
          ENDIF.
        ENDLOOP.
        MESSAGE 'State machines listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidtoken.
        MESSAGE 'Invalid pagination token.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListStateMachinesin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`SendTaskSuccess`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `SendTaskSuccess` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
```

```
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the
task.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie SendTaskSuccessin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
        try {
            SendTaskSuccessRequest successRequest =
SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie SendTaskSuccessin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- Einzelheiten zur API finden Sie SendTaskSuccessin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with
 an
        activity step waits for the activity to get task data and then respond
 with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included
 in the
                           response to the get_activity_task action and must be
 sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie SendTaskSuccessin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sfn->sendtasksuccess(
          iv_tasktoken = iv_task_token
          iv_output = iv_task_response
        ).
        MESSAGE 'Task success sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidtoken.
        MESSAGE 'Invalid task token.' TYPE 'E'.
      CATCH /aws1/cx_sfntaskdoesnotexist.
        MESSAGE 'Task does not exist.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidoutput.
        MESSAGE 'Invalid task output.' TYPE 'E'.
      CATCH /aws1/cx_sfntasktimedout.
        MESSAGE 'Task timed out.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendTaskSuccessin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartExecution`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartExecution` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit der Workflow-Orchestrierung

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</
param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
 stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
 _amazonStepFunctions.StartExecutionAsync(executionRequest);
```

```
        return response.ExecutionArn;
    }
```

- Einzelheiten zur API finden Sie StartExecutionin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String startWorkflow(SfnClient sfnClient, String
stateMachineArn, String jsonEx) {
       UUID uuid = UUID.randomUUID();
       String uuidValue = uuid.toString();
       try {
           StartExecutionRequest executionRequest =
StartExecutionRequest.builder()
               .input(jsonEx)
               .stateMachineArn(stateMachineArn)
               .name(uuidValue)
               .build();

           StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
           return response.executionArn();

       } catch (SfnException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie StartExecutionin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SFNClient, StartExecutionCommand } from "@aws-sdk/client-sfn";

/**
 * @param {{ sfnClient: SFNClient, stateMachineArn: string }} config
 */
export async function startExecution({ sfnClient, stateMachineArn }) {
  const response = await sfnClient.send(
    new StartExecutionCommand({
      stateMachineArn,
    }),
  );
  console.log(response);
  // Example response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '202a9309-c16a-454b-adeb-c4d19afe3bf2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   executionArn: 'arn:aws:states:us-
east-1:000000000000:execution:MyStateMachine:aaaaaaaa-f787-49fb-
a20c-1b61c64eafe6',
  //   startDate: 2024-01-04T15:54:08.362Z
  // }
  return response;
}
```

```
// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  startExecution({ sfnClient: new SFNClient({}), stateMachineArn: "ARN" });
}
```

- Einzelheiten zur API finden Sie StartExecutionin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- Einzelheiten zur API finden Sie StartExecutionin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also
 known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about
 the run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return response["executionArn"]
```

- Einzelheiten zur API finden Sie StartExecutionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_sfn->startexecution(
      iv_statemachinearn = iv_state_machine_arn
      iv_input = iv_input
    ).
    ov_execution_arn = lo_result->get_executionarn( ).
    MESSAGE 'Execution started successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnstatemachinedoes00.
    MESSAGE 'State machine does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidexecinput.
    MESSAGE 'Invalid execution input.' TYPE 'E'.
  CATCH /aws1/cx_sfnexeclimitexceeded.
    MESSAGE 'Execution limit exceeded.' TYPE 'E'.
ENDTRY.
```

- Einzelheiten zur API finden Sie StartExecutionin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für Step Functions mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Step Functions mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Step Functions oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen einer Messenger-Anwendung mit Step Functions
- Erste Schritte mit der Workflow-Orchestrierung
- Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions
- Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Erstellen einer Messenger-Anwendung mit Step Functions

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

Python

SDK für Python (Boto3)

Zeigt, wie AWS SDK für Python (Boto3) mit AWS Step Functions dem with eine Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. Die Zustandsmaschine ist mit einer AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht werden kann.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.

- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

Erste Schritte mit der Workflow-Orchestrierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Eine IAM-Rolle für Step Functions erstellen
- Erstellen Sie Ihre erste Zustandsmaschine
- Starten Sie die Ausführung Ihrer Zustandsmaschine
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS
 Step Functions Getting Started tutorial
```

```
set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
```

```
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be
 less reliable."
    echo "Consider installing jq for better error handling: brew install jq
 (macOS) or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
```

```
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"
```

```
    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] ||
 [ "$wait_time" -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
```

```
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
```

```
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/
null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed
 to delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
```

```
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
```

```
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
```

```
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
```

```
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi
```

```
# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs
 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
```

```
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
```

```
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
```

```
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
```

```
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"
```

```
    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
```

```
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
```

```
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment
 analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)
```

```
    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in
 $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
```

```
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

Entwickeln und Orchestrieren von Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

Python

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finden Sie das aktuelle Repository mit den meisten Trends GitHub und fassen Sie es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock

- Runtime der Agenten für Amazon Bedrock
- Step Functions

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie Sie eine AWS Step Functions Zustandsmaschine erstellen, die nacheinander AWS Lambda Funktionen aufruft.

Java

SDK für Java 2.x

Zeigt, wie Sie einen AWS serverlosen Workflow mithilfe von AWS Step Functions und erstellen. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

Codebeispiele für AWS STS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS -Security-Token-Service mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS STS Benutzerhandbuch — Weitere Informationen zu AWS STS.
- AWS STS API-Referenz — Details zu allen verfügbaren AWS STS Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS STS verwenden AWS SDKs
  - Aktionen für AWS STS verwenden AWS SDKs
    - Verwenden Sie mit AssumeRoleeinem AWS SDK oder CLI
    - Verwendung von AssumeRoleWithWebIdentity mit einer CLI
    - Verwendung von DecodeAuthorizationMessage mit einer CLI
    - Verwendung von GetFederationToken mit einer CLI
    - Verwenden Sie mit einem GetSessionToken AWS SDK oder CLI
- Szenarien für AWS STS verwenden AWS SDKs
  - Gehen Sie von einer IAM-Rolle aus, für die ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK
  - Konfigurieren Sie die Konnektivität des Containerdienstes
  - Konstruieren Sie eine URL mit AWS STS für Verbundbenutzer mit einem AWS SDK
  - Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
  - Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
  - Einen Container-Service für Instanzen virtueller Maschinen erstellen
  - Einen verwalteten Monitoring-Workspace erstellen
  - Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
  - Holen Sie sich ein Sitzungstoken, für das ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK
  - Erste Schritte mit der Konfigurationsverwaltung
  - Erste Schritte mit Container-Registern
  - Erste Schritte mit Internet of Things-Messaging
  - Erste Schritte mit Feature-Stores für maschinelles Lernen

- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Push-Benachrichtigungen
- Erste Schritte mit Such- und Analysemaschinen
- Erste Schritte mit Firewalls für Webanwendungen
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch
- Systemmanagement einrichten

Grundlegende Beispiele für AWS STS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS -Security-Token-Service mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS STS verwenden AWS SDKs
  - Verwenden Sie mit AssumeRoleeinem AWS SDK oder CLI
  - Verwendung von AssumeRoleWithWebIdentity mit einer CLI
  - Verwendung von DecodeAuthorizationMessage mit einer CLI
  - Verwendung von GetFederationToken mit einer CLI
  - Verwenden Sie mit einem GetSessionToken AWS SDK oder CLI

Aktionen für AWS STS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS STS Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS STS API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS STS verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS -Security-Token-Service -API-Referenz.

Beispiele

- Verwenden Sie mit AssumeRoleeinem AWS SDK oder CLI
- Verwendung von AssumeRoleWithWebIdentity mit einer CLI
- Verwendung von DecodeAuthorizationMessage mit einer CLI
- Verwendung von GetFederationToken mit einer CLI
- Verwenden Sie mit einem GetSessionToken AWS SDK oder CLI

Verwenden Sie mit **`AssumeRole`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AssumeRole` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Übernehmen Sie eine IAM-Rolle, die ein MFA-Token erfordert
- Erstellen einer URL für Verbundbenutzer

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
```

```
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do
 not specify a region,
        // or to use a specific region.
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };
```

```
            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
#############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS CLI Befehlsreferenz.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
```

```
                                const Aws::String &roleSessionName,
                                const Aws::String &externalId,
                                Aws::Auth::AWSCredentials &credentials,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So übernehmen Sie eine Rolle

Der folgende `assume-role`-Befehl ruft eine Reihe von kurzfristigen Anmeldeinformationen für die IAM-Rolle `s3-access-example` ab.

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

Ausgabe:

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

Die Ausgabe des Befehls enthält einen Zugriffsschlüssel, einen geheimen Schlüssel und ein Sitzungs-Token, die Sie zur Authentifizierung bei AWS verwenden können.

Für die Verwendung AWS über die CLI können Sie ein benanntes Profil einrichten, das einer Rolle zugeordnet ist. Wenn Sie das Profil verwenden, ruft die AWS CLI assume-role auf und verwaltet die Anmeldeinformationen für Sie. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle in der AWS CLI im AWS CLI-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter AssumeRole AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
```

```
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
 String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
```

```
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
 exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { STSClient } from "@aws-sdk/client-sts";
```

```
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

Übernehmen Sie die IAM-Rolle.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für JavaScript API-Referenz.

SDK für JavaScript (v2)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);
```

```
  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt einen Satz temporärer Anmeldeinformationen (Zugriffsschlüssel, geheimer Schlüssel und Sitzungstoken) zurück, die eine Stunde lang für den Zugriff auf AWS Ressourcen verwendet werden können, auf die der anfragende Benutzer normalerweise keinen Zugriff hat. Die zurückgegebenen Anmeldeinformationen verfügen über die Berechtigungen, die von der Zugriffsrichtlinie der übernommenen Rolle und der bereitgestellten Richtlinie zugelassen sind (Sie können die bereitgestellte Richtlinie nicht verwenden, um Berechtigungen zu gewähren, die über die in der Zugriffsrichtlinie der übernommenen Rolle definierten Berechtigungen hinausgehen).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

Beispiel 2: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine Stunde gültig sind und über dieselben Berechtigungen verfügen, die in der Zugriffsrichtlinie der übernommenen Rolle definiert sind.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

Beispiel 3: Gibt einen Satz temporärer Anmeldeinformationen zurück, die die Seriennummer und das generierte Token von einem MFA bereitstellen, das den Anmeldeinformationen des Benutzers zugeordnet ist, die zur Ausführung des Cmdlet verwendet werden.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Beispiel 4: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine in einem Kundenkonto definierte Rolle übernommen haben. Für jede Rolle, die der Drittanbieter übernehmen kann, muss das Kundenkonto eine Rolle mithilfe einer Kennung erstellen, die jedes Mal, wenn die Rolle übernommen wird, im ExternalId Parameter - übergeben werden muss.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- Einzelheiten zur API finden Sie unter AssumeRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt einen Satz temporärer Anmeldeinformationen (Zugriffsschlüssel, geheimer Schlüssel und Sitzungstoken) zurück, die eine Stunde lang für den Zugriff auf AWS Ressourcen verwendet werden können, auf die der anfragende Benutzer normalerweise keinen Zugriff hat. Die zurückgegebenen Anmeldeinformationen verfügen über die Berechtigungen, die von der Zugriffsrichtlinie der übernommenen Rolle und der bereitgestellten Richtlinie zugelassen sind (Sie können die bereitgestellte Richtlinie nicht verwenden, um Berechtigungen zu gewähren, die über die in der Zugriffsrichtlinie der übernommenen Rolle definierten Berechtigungen hinausgehen).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

Beispiel 2: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine Stunde gültig sind und über dieselben Berechtigungen verfügen, die in der Zugriffsrichtlinie der übernommenen Rolle definiert sind.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

Beispiel 3: Gibt einen Satz temporärer Anmeldeinformationen zurück, die die Seriennummer und das generierte Token von einem MFA bereitstellen, das den Anmeldeinformationen des Benutzers zugeordnet ist, die zur Ausführung des Cmdlet verwendet werden.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Beispiel 4: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine in einem Kundenkonto definierte Rolle übernommen haben. Für jede Rolle, die der Drittanbieter übernehmen kann, muss das Kundenkonto eine Rolle mithilfe einer Kennung erstellen, die jedes Mal, wenn die Rolle übernommen wird, im ExternalId Parameter - übergeben werden muss.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- Einzelheiten zur API finden Sie unter AssumeRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Übernehmen Sie eine IAM-Rolle, die ein MFA-Token erfordert, und verwenden Sie temporäre Anmeldeinformationen, um Amazon-S3-Buckets für das Konto aufzulisten.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
```

```
                              grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                                device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Einzelheiten zur API finden Sie AssumeRolein AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- Einzelheiten zur API finden Sie AssumeRolein der API-Referenz zum AWS SDK für Rust.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSTS
```

```
    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- Einzelheiten zur API finden Sie AssumeRolein der API-Referenz zum AWS SDK für Swift.

Verwendung von **`AssumeRoleWithWebIdentity`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AssumeRoleWithWebIdentity` verwendet wird.

CLI

AWS CLI

So erhalten Sie kurzfristige Anmeldeinformationen für eine mit Web Identity (OAuth 2.0) authentifizierte Rolle

Der folgende `assume-role-with-web-identity`-Befehl ruft eine Reihe von kurzfristigen Anmeldeinformationen für die IAM-Rolle `app1` ab. Die Anforderung wird mithilfe des Webidentitäts-Tokens authentifiziert, das vom angegebenen Web-Identitätsanbieter bereitgestellt wird. Um die Möglichkeiten des Benutzers weiter einzuschränken, werden auf die Sitzung zwei zusätzliche Richtlinien angewendet. Die zurückgegebenen Anmeldeinformationen verfallen eine Stunde nach ihrer Generierung.

```
aws sts assume-role-with-web-identity \
    --duration-seconds 3600 \
    --role-session-name "app1" \
    --provider-id "www.amazon.com" \
    --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
    --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
    --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1wl7WTI7jn-Pcb6M-
buCgHhFOzTQxod27L9CqnOLio7N3gZAGpsp6n1-
AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ"
```

Ausgabe:

```
{
    "SubjectFromWebIdentityToken": "amzn1.account.AF6RHO7KZU5XRVQJGXK6HB56KR2A",
    "Audience": "client.5498841531868486423.1548@apps.example.com",
    "AssumedRoleUser": {
        "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
        "AssumedRoleId": "AROACLKWSDQRAOEXAMPLE:app1"
    },
    "Credentials": {
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    },
    "Provider": "www.amazon.com"
}
```

Weitere Informationen finden Sie unter Anfordern von temporären Sicherheitsanmeldeinformationen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie AssumeRoleWithWebIdentityin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt einen temporären Satz Anmeldeinformationen mit einer Gültigkeit von einer Stunde für einen Benutzer zurück, der mit dem Identitätsanbieter „Anmeldung mit Amazon“ authentifiziert wurde. Die Anmeldeinformationen übernehmen die Zugriffsrichtlinie, die der durch die Rollen-ARN identifizierten Rolle zugeordnet ist. Optional können Sie eine JSON-Richtlinie an den Parameter -Richtlinie übergeben, die die Zugriffsberechtigungen weiter verfeinert (Sie können keine Berechtigungen mehr gewähren, als in den mit der Rolle verknüpften Berechtigungen verfügbar sind). Der an - übergebene Wert WebIdentityToken ist die eindeutige Benutzer-ID, die vom Identitätsanbieter zurückgegeben wurde.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Einzelheiten zur API finden Sie unter AssumeRoleWithWebIdentity AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt einen temporären Satz Anmeldeinformationen mit einer Gültigkeit von einer Stunde für einen Benutzer zurück, der mit dem Identitätsanbieter „Anmeldung mit Amazon“ authentifiziert wurde. Die Anmeldeinformationen übernehmen die Zugriffsrichtlinie, die der durch die Rollen-ARN identifizierten Rolle zugeordnet ist. Optional können Sie eine JSON-Richtlinie an den Parameter -Richtlinie übergeben, die die Zugriffsberechtigungen weiter verfeinert (Sie können keine Berechtigungen mehr gewähren, als in den mit der Rolle verknüpften Berechtigungen verfügbar sind). Der an - übergebene Wert WebIdentityToken ist die eindeutige Benutzerkennung, die vom Identitätsanbieter zurückgegeben wurde.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- Einzelheiten zur API finden Sie unter AssumeRoleWithWebIdentity AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DecodeAuthorizationMessage`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DecodeAuthorizationMessage` verwendet wird.

CLI

AWS CLI

So decodieren Sie eine codierte Autorisierungsnachricht, die als Antwort auf eine Anfrage zurückgegeben wird

Das folgende `decode-authorization-message`-Beispiel decodiert zusätzliche Informationen zum Autorisierungsstatus einer Anfrage aus einer codierten Nachricht, die als Antwort auf eine Anfrage von Amazon Web Services zurückgegeben wird.

```
aws sts decode-authorization-message \
    --encoded-message EXAMPLEWodyRNrtlQARDip-
eTA6i6DrlUhHhPQrLWB_lAbl5pAKxl9mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqeO7e2NMk5j_hmzAiChJN-8oy3
Ojau7BMjOTWwOtHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
pOu3FZjwYStfvTb3GHs3-6rLribGO9jZOktkfE6vqxlFzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioqgo
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrHlI-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYGtPp5NIeSpuf3fOHanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjjOtNPBOZCcuEKE
```

Ausgabe:

```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
\"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\",\"effect
\":\"DENY\",\"principals\":{\"items\":[{\"value\":\"AROA123456789EXAMPLE
\"}]},\"principalGroups\":{\"items\":[]},\"actions\":{\"items\":[{\"value
\":\"ec2:RunInstances\"}]},\"resources\":{\"items\":[{\"value\":\"*
\"}]},\"conditions\":{\"items\":[]}}]},\"failures\":{\"items\":[]},
\"context\":{\"principal\":{\"id\":\"AROA123456789EXAMPLE:Ana\",\"arn
\":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"},\"action\":
\"RunInstances\",\"resource\":\"arn:aws:ec2:us-east-1:111122223333:instance/*
\",\"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
\"values\":{\"items\":[{\"value\":\"2\"}]}},{\"key\":\"ec2:InstanceMarketType
\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource
\",\"values\":{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account
\",\"values\":{\"items\":[{\"value\":\"111122223333\"}]}},{\"key\":
\"ec2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}},
```

```
{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\":\"false\"}]}},
{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":
\"false\"}]}},{\"key\":\"ec2:InstanceType\",\"values\":{\"items\":[{\"value
\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":
[{\"value\":\"ebs\"}]}},{\"key\":\"aws:Region\",\"values\":{\"items\":[{\"value
\":\"us-east-1\"}]}},{\"key\":\"ec2:MetadataHttpEndpoint\",\"values\":{\"items
\":[{\"value\":\"enabled\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items
\":[{\"value\":\"ec2\"}]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":
[{\"value\":\"*\"}]}},{\"key\":\"ec2:MetadataHttpTokens\",\"values\":{\"items
\":[{\"value\":\"required\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items
\":[{\"value\":\"instance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items
\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items
\":[{\"value\":\"us-east-1\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":
[{\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}}"
}
```

Weitere Informationen finden Sie unter Logik der Richtlinienauswertung im Handbuch zu AWS -IAM-Benutzer.

- Einzelheiten zur API finden Sie DecodeAuthorizationMessagein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Decodiert die zusätzlichen Informationen, die im bereitgestellten codierten Nachrichteninhalt enthalten sind, der als Antwort auf eine Anfrage zurückgegeben wurde. Die Zusatzinformationen werden verschlüsselt, da es sich bei den Details des Autorisierungsstatus um vertrauliche Informationen handeln kann, die der Benutzer, der die Aktion angefordert hat, nicht sehen sollte.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Einzelheiten zur API finden Sie unter DecodeAuthorizationMessage AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Decodiert die zusätzlichen Informationen, die im bereitgestellten codierten Nachrichteninhalt enthalten sind, der als Antwort auf eine Anfrage zurückgegeben wurde. Die

Zusatzinformationen werden verschlüsselt, da es sich bei den Details des Autorisierungsstatus um vertrauliche Informationen handeln kann, die der Benutzer, der die Aktion angefordert hat, nicht sehen sollte.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- Einzelheiten zur API finden Sie unter DecodeAuthorizationMessage AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetFederationToken`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetFederationToken` verwendet wird.

CLI

AWS CLI

So geben Sie einen Satz temporärer Sicherheits-Anmeldeinformationen mit den Anmeldeinformationen des IAM-Benutzer-Zugriffsschlüssels zurück

Das folgende `get-federation-token`-Beispiel gibt einen Satz temporärer Sicherheits-Anmeldeinformationen (bestehend aus einer Zugriffsschlüssel-ID, einem geheimen Zugriffsschlüssel und einem Sicherheits-Token) für einen Benutzer zurück. Sie müssen den `GetFederationToken`-Vorgang mit den langfristigen Sicherheits-Anmeldeinformationen eines IAM-Benutzers aufrufen.

```
aws sts get-federation-token \
    --name Bob \
    --policy file://myfile.json \
    --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --duration-seconds 900
```

Inhalt von `myfile.json`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe*",
            "Resource": "*"
        }
    ]
}
```

Ausgabe:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQunWMTfKq0DCOP/////////
wEQAxoMNDUyOTI1MTcwNTA3Igxuy3AOpuuoLsk3MJwqgQPg8QOd9HuoClUxq26wnc/nm
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
        "Expiration": "2023-12-20T02:06:07+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "111122223333:Bob",
        "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
}
```

Weitere Informationen finden Sie unter Anfordern von temporären Sicherheitsanmeldeinformationen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetFederationTokenin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Fordert ein Verbund-Token an, das eine Stunde lang gültig ist, mit „Bob“ als Namen des Verbundbenutzers. Dieser Name kann verwendet werden, um in einer ressourcenbasierten Richtlinie (z. B. einer Amazon-S3-Bucket-Richtlinie) auf den Namen des Verbundbenutzers zu verweisen. Die bereitgestellte IAM-Richtlinie im JSON-Format wird verwendet, um die Berechtigungen, die dem IAM-Benutzer zur Verfügung stehen, einzuschränken. Die angegebene Richtlinie kann nicht mehr Berechtigungen gewähren, als dem anfordernden Benutzer gewährt wurden. Die endgültigen Berechtigungen für den Verbundbenutzer sind der restriktivste Satz, basierend auf der Schnittmenge der übergebenen Richtlinie und der IAM-Benutze-Richtlinie.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Einzelheiten zur API finden Sie unter GetFederationToken AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Fordert ein Verbund-Token an, das eine Stunde lang gültig ist, mit „Bob“ als Namen des Verbundbenutzers. Dieser Name kann verwendet werden, um in einer ressourcenbasierten Richtlinie (z. B. einer Amazon-S3-Bucket-Richtlinie) auf den Namen des Verbundbenutzers zu verweisen. Die bereitgestellte IAM-Richtlinie im JSON-Format wird verwendet, um die Berechtigungen, die dem IAM-Benutzer zur Verfügung stehen, einzuschränken. Die angegebene Richtlinie kann nicht mehr Berechtigungen gewähren, als dem anfordernden Benutzer gewährt wurden. Die endgültigen Berechtigungen für den Verbundbenutzer sind der restriktivste Satz, basierend auf der Schnittmenge der übergebenen Richtlinie und der IAM-Benutze-Richtlinie.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- Einzelheiten zur API finden Sie unter GetFederationToken AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetSessionToken`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSessionToken` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Abrufen eines Sitzungs-Tokens, das ein MFA-Token erfordert

CLI

AWS CLI

So erhalten Sie einen Satz kurzfristiger Anmeldeinformationen für eine IAM-Identität

Der folgende `get-session-token`-Befehl ruft einen Satz kurzfristiger Anmeldeinformationen für die IAM-Identität ab, die den Aufruf ausführt. Die resultierenden Anmeldeinformationen können für Anfragen verwendet werden, bei denen die Richtlinie eine Multi-Faktor-Authentifizierung (MFA) erfordert. Die Anmeldeinformationen verfallen 15 Minuten nach ihrer Generierung.

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

Ausgabe:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
```

```
    }
}
```

Weitere Informationen finden Sie unter Anfordern von temporären Sicherheitsanmeldeinformationen im AWS -IAM-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetSessionTokenin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die für einen festgelegten Zeitraum gültig sind. Die Anmeldeinformationen, die zum Anfordern temporärer Anmeldeinformationen verwendet werden, werden aus den aktuellen Shell-Standardwerten abgeleitet. Um andere Anmeldeinformationen anzugeben, verwenden Sie die SecretKey Parameter - ProfileName oder - AccessKey /-.

```
Get-STSSessionToken
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Beispiel 2: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Die zum Stellen der Anfrage verwendeten Anmeldeinformationen werden aus dem angegebenen Profil abgerufen.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
```

```
-----------                                 ----------
 ---------------                            ------------
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

Beispiel 3: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Dabei wird die Identifikationsnummer des MFA-Geräts verwendet, das dem Konto zugeordnet ist, dessen Anmeldeinformationen im Profil „myprofilename“ angegeben sind, und der vom Gerät bereitgestellte Wert.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Ausgabe:

```
AccessKeyId                                 Expiration
 SecretAccessKey                            SessionToken
-----------                                 ----------
 ---------------                            ------------
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

- Einzelheiten zur API finden Sie unter GetSessionToken AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die für einen festgelegten Zeitraum gültig sind. Die Anmeldeinformationen, die zum Anfordern temporärer Anmeldeinformationen verwendet werden, werden aus den aktuellen Shell-Standardwerten abgeleitet. Um andere Anmeldeinformationen anzugeben, verwenden Sie die Parameter - ProfileName oder -AccessKey /-SecretKey .

```
Get-STSSessionToken
```

Ausgabe:

```
AccessKeyId                                 Expiration
 SecretAccessKey                            SessionToken
```

```
-----------                                 ----------
 ---------------                            ------------
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

Beispiel 2: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Die zum Stellen der Anfrage verwendeten Anmeldeinformationen werden aus dem angegebenen Profil abgerufen.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Ausgabe:

```
AccessKeyId                                 Expiration
 SecretAccessKey                            SessionToken
-----------                                 ----------
 ---------------                            ------------
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

Beispiel 3: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Dabei wird die Identifikationsnummer des MFA-Geräts verwendet, das dem Konto zugeordnet ist, dessen Anmeldeinformationen im Profil „myprofilename“ angegeben sind, und der vom Gerät bereitgestellte Wert.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Ausgabe:

```
AccessKeyId                                 Expiration
 SecretAccessKey                            SessionToken
-----------                                 ----------
 ---------------                            ------------
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

- Einzelheiten zur API finden Sie unter GetSessionToken AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Sitzungs-Token ab, indem Sie ein MFA-Token übergeben, und verwenden Sie es, um Amazon-S3-Buckets für das Konto aufzulisten.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
```

```
    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- Einzelheiten zur API finden Sie GetSessionTokenin AWS SDK for Python (Boto3) API Reference.

Szenarien für AWS STS verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS STS mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS STS oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Gehen Sie von einer IAM-Rolle aus, für die ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK
- Konfigurieren Sie die Konnektivität des Containerdienstes
- Konstruieren Sie eine URL mit AWS STS für Verbundbenutzer mit einem AWS SDK
- Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp
- Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration
- Einen Container-Service für Instanzen virtueller Maschinen erstellen
- Einen verwalteten Monitoring-Workspace erstellen
- Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable
- Holen Sie sich ein Sitzungstoken, für das ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK
- Erste Schritte mit der Konfigurationsverwaltung
- Erste Schritte mit Container-Registern

- Erste Schritte mit Internet of Things-Messaging
- Erste Schritte mit Feature-Stores für maschinelles Lernen
- Erste Schritte mit verwalteten Kubernetes-Clustern
- Erste Schritte mit verwaltetem Streaming
- Erste Schritte mit Push-Benachrichtigungen
- Erste Schritte mit Such- und Analysemaschinen
- Erste Schritte mit Firewalls für Webanwendungen
- Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher
- Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch
- Systemmanagement einrichten

Gehen Sie von einer IAM-Rolle aus, für die ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK

Das folgende Code-Beispiel veranschaulicht, wie Sie eine Rolle anfügen, die ein MFA-Token erfordert.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie eine IAM-Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt.
- Erstellen Sie einen IAM-Benutzer, der nur dann berechtigt ist, die Rolle zu übernehmen, wenn MFA-Anmeldeinformationen bereitgestellt werden.
- Registrieren Sie ein MFA-Gerät für den Benutzer.
- Übernehmen der Rolle und Verwendung von temporären Anmeldeinformationen zum Auflisten von Amazon-S3-Buckets.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer, registrieren Sie ein MFA-Gerät, und erstellen Sie eine Rolle, die die Berechtigung zum Auflisten von S3-Buckets gewährt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
 MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
 role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
```

```
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
```

```
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
```

```
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

Zeigen Sie, dass die Annahme der Rolle ohne MFA-Token nicht zulässig ist.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
 results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

Übernehmen Sie die Rolle, die die Berechtigung zum Auflisten von S3-Buckets erteilt, indem Sie das erforderliche MFA-Token übergeben, und zeigen Sie, dass Buckets aufgelistet werden können.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
```

```
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- Einzelheiten zur API finden Sie AssumeRolein AWS SDK for Python (Boto3) API Reference.

Konfigurieren Sie die Konnektivität des Containerdienstes

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
```

```
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
```

```
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"
```

```
    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
```

```
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
```

```
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
```

```
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
```

```
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
```

```
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
```

```
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
```

```
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
```

```
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure
```

```
    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
dependency issues."
    fi
```

```
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices

- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Konstruieren Sie eine URL mit AWS STS für Verbundbenutzer mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle, die nur Lesezugriff auf die Amazon-S3-Ressourcen des aktuellen Kontos gewährt.
- Holen Sie sich ein Sicherheitstoken vom AWS Verbundendpunkt.
- Erstellen Sie eine URL, die für den Zugriff auf die Konsole mit Verbund-Anmeldeinformationen verwendet werden kann.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Rolle, die nur Lesezugriff auf die S3-Ressourcen des aktuellen Kontos gewährt.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 instance
                         that has the permission to create a role.
```

```
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return role
```

Rufen Sie ein Sicherheitstoken vom AWS Verbundendpunkt ab und erstellen Sie eine URL, die für den Zugriff auf die Konsole mit Verbundanmeldedaten verwendet werden kann.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
 Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
 that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
```

```
        AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS
federation
        endpoint, includes the sign-in token for authentication, and redirects to
        the AWS Management Console with permissions defined by the role that was
        specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
granted.
                            The current user must have permission to assume the
role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")
```

```
    # Make a federated URL that can be used to sign into the AWS Management
 Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
```

```
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- Einzelheiten zur API finden Sie AssumeRolein AWS SDK for Python (Boto3) API Reference.

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"
```

```
    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "=========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "=========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
```

```
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
```

```
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
```

```
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
 then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME'" "Delete ECS service"
            else
```

```
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
" ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
```

```
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}
```

```
# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
```

```
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
```

```
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
```

```
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
```

```
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"
```

```
# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
```

```
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices

- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Eine Lambda-Funktion erstellen und bereitstellen
- Erstellen Sie eine REST-API
- Lambda-Proxyintegration konfigurieren
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value
```

```
        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
```

```
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
```

```
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
```

```
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}
```

```
# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""
```

```
# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy

- CreateDeployment
- CreateFunction
- CreateResource
- CreateRestApi
- CreateRole
- DeleteFunction
- DeleteRestApi
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetResources
- GetRestApis
- PutIntegration
- PutMethod

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Einen Dienst erstellen und überwachen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"
```

```
    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
```

```
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
```

```
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
 2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
 before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
 $max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
```

```
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
```

```
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")
```

```
    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
```

```
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
```

```
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
```

```
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
```

```
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
```

```
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster

- CreateInstanceProfile
- CreateKeyPair
- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Einen verwalteten Monitoring-Workspace erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Ihren Workspace
- Erstellen Sie einen Grafana-Arbeitsbereich
- Konfigurieren Sie die Authentifizierung
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
```

```
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"
```

```
# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."
```

```
# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
```

```
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
```

```
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
```

```
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"
```

```
    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Erstellen eines Monitoring-Dashboards mit dem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard

- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."
```

```
    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")
```

```
if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
```

```
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null
```

```
    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
```

```
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
```

```
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi
```

```
# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
```

```
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
```

```
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards

- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

Holen Sie sich ein Sitzungstoken, für das ein MFA-Token erforderlich ist, mit AWS STS unter Verwendung eines AWS SDK

Das folgende Code-Beispiel veranschaulicht, wie Sie ein Sitzungstoken erhalten, das ein MFA-Token erfordert.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie eine IAM-Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt.
- Erstellen Sie einen IAM-Benutzer, der nur dann berechtigt ist, die Rolle zu übernehmen, wenn MFA-Anmeldeinformationen bereitgestellt werden.
- Registrieren Sie ein MFA-Gerät für den Benutzer.
- Stellen Sie MFA-Anmeldeinformationen bereit, um ein Sitzungstoken abzurufen, und verwenden Sie temporäre Anmeldeinformationen, um S3-Buckets aufzulisten.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer, registrieren Sie ein MFA-Gerät und erstellen Sie eine Rolle, die dem Benutzer die Berechtigung erteilt, S3-Buckets nur aufzulisten, wenn MFA-Anmeldeinformationen verwendet werden.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
```

```
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
```

```
    print(
        f"Created an inline policy for {user.name} that lets the user list
 buckets, "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

Rufen Sie temporäre Sitzungs-Anmeldeinformationen ab, indem Sie ein MFA-Token übergeben, und verwenden Sie die Anmeldeinformationen, um S3-Buckets für das Konto aufzulisten.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
```

```
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
```

```
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- Einzelheiten zur API finden Sie GetSessionTokenin AWS SDK for Python (Boto3) API Reference.

Erste Schritte mit der Konfigurationsverwaltung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein

- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}
```

```
# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
```

```
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}
```

```
# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
```

```
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
```

```
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
```

```
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi
```

```
# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
```

```
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Erste Schritte mit Container-Registern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Docker-Image
- Erstellen Sie ein Amazon ECR-Repository
- Löschen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="

# Check prerequisites
```

```
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"
```

```
    echo "==================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "==================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "==================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi
```

```
    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
```

```
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
```

```
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity
  - GetLoginPassword

Erste Schritte mit Internet of Things-Messaging

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- IoT-Ressourcen erstellen
- Konfigurieren Ihres Geräts
- Ausführen der Beispielanwendungen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
```

```
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID
 --new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
```

```
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a
 $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
```

```
    echo "AWS region not configured. Please run 'aws configure' to set your
 region." | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
```

```
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-
document file://iot-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn'
 --output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active
 --certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile
 $CERTS_DIR/public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
```

```
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." |
 tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
```

```
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-
python-v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
```

```
# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
```

```
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.

- AttachPolicy
- AttachThingPrincipal
- CreateKeysAndCertificate
- CreatePolicy
- CreateThing
- DeleteCertificate
- DeletePolicy
- DeleteThing
- DescribeEndpoint
- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
```

```
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
```

```
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=sagemaker-featurestore 2>&1
```

```
    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"
```

```
    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
```

```
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
```

```
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")
```

```
# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."
```

```
# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
```

```
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
```

```
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi
```

```
echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
```

```
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
```

```
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
```

```
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}
```

```
# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
```

```
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
```

```
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
```

```
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")
```

```
echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"
```

```
echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
```

```
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
```

```
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
```

```
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
```

```
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
```

```
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
 removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
```

```
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
```

```
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)
```

```
            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
```

```
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
```

```
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
```

```
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
```

```
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"
```

```
# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi
```

```
echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"
```

```
# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
```

```
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
```

```
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
```

```
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
```

```
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
```

```
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
```

```
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
```

```
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy

- AuthorizeSecurityGroupIngress
- CreateCluster
- CreateInstanceProfile
- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile

- RevokeSecurityGroupEgress

- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit Push-Benachrichtigungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Anwendung
- Aktivieren Sie Push-Benachrichtigungskanäle
- Push-Benachrichtigung senden
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail
```

```
# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM
```

```
# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi
```

```
    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."
```

```
    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters"
 >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
```

```
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi

check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi
```

```
echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "============================================"
echo "ENABLING FCM (GCM) CHANNEL"
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-
demo-only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server
 Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real
 certificates)
echo ""
```

```
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --
apns-channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-
channel command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
```

```
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF
```

```
echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens
 with actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder
 tokens)
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
```

```
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
```

```
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder
 credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps

- GetCallerIdentity
- SendMessages
- UpdateApnsChannel
- UpdateGcmChannel

Erste Schritte mit Such- und Analysemaschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Dienstdomäne
- Laden Sie Daten auf Ihre Domain hoch
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
```

```
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to
 delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/
null 2>&1; then
```

```
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
```

```
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
# This policy restricts access to specific actions and should be further
 restricted to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for
 completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack
 creation failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
```

```
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo
 "No"; fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
```

```
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel",
 "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
 "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
 "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
 Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955,
 "actor": ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal",
 "Backus, Jim", "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen,
 Corey", "Birch, Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh,
 Nelson", "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira",
 "McMahon, David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)
```

```
    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }

    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
```

```
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous
 (authentication not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential
 backoff..."

            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT
+1))/$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
```

```
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:"
| cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
(HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding
with data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
```

```
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
 '"result"'; then
            echo "# Single document uploaded successfully! (HTTP
 $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
 $UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
 '"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
 $BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
```

```
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
```

```
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-
name "$DOMAIN_NAME" --query 'DomainStatus.{AdvancedSecurityOptions:
 AdvancedSecurityOptions, AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user
 works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
```

```
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
```

```
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/
_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateDomain

- DeleteDomain
- DescribeDomain
- GetCallerIdentity

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleich von Zeichenketten hinzu
- Verwaltete Regeln hinzufügen
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
```

```
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}
```

```
# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
```

```
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
```

```
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "==================================================="
echo "STEP 2: Adding String Match Rule"
echo "==================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
```

```
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
```

```
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
```

```
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "==================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "==================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true
```

```
    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
```

```
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done
```

```
# Step 4: List CloudFront distributions
echo ""
echo "==================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "==================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "==================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "==================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
```

```
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
```

```
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Übertragung von fest codierten Geheimnissen in einen sicheren geheimen Speicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisiere das Geheimnis
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi
```

```
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
```

```
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)
```

```
# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
```

```
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
```

```
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)
```

```
check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
```

```
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue
  - PutResourcePolicy
  - UpdateSecret

Führen Sie CPU-Stresstests auf Instanzen virtueller Maschinen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen

- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Experimentvorlage
- Führen Sie das Experiment durch
- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
```

```
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
```

```
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
```

```
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")
```

```
# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
```

```
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"
```

```
# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
```

```
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
```

```
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""
```

```
while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
```

```
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
```

```
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
```

```
    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi
```

```
echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment

- TerminateInstances
- Wait

Systemmanagement einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- IAM-Berechtigungen für Systems Manager erstellen
- Erstellen einer IAM-Rolle für Systems Manager
- Systems Manager konfigurieren
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"
```

```
# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"
```

```
    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
```

```
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
```

```
          "ssm-quicksetup:*"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmReadOnly",
        "Effect": "Allow",
        "Action": [
          "ssm:DescribeAutomationExecutions",
          "ssm:GetAutomationExecution",
          "ssm:ListAssociations",
          "ssm:DescribeAssociation",
          "ssm:ListDocuments",
          "ssm:ListResourceDataSync",
          "ssm:DescribePatchBaselines",
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
```

```
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
```

```
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
```

```
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
```

```
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
```

```
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."
```

```
# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
```

```
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"
```

```
# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."
```

```
# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."
```

```
# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Codebeispiele für Support verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Support mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Support Benutzerhandbuch — Weitere Informationen zu Support.
- Support API-Referenz — Details zu allen verfügbaren Support Aktionen.

- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Support verwenden AWS SDKs
  - Hallo Support
  - Lernen Sie die Grundlagen von Support mit einem AWS SDK
  - Aktionen für Support verwenden AWS SDKs
    - Verwenden Sie mit AddAttachmentsToSeteinem AWS SDK oder CLI
    - Verwenden Sie mit AddCommunicationToCaseeinem AWS SDK oder CLI
    - Verwenden Sie mit CreateCaseeinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeAttachmenteinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeCaseseinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeCommunicationseinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeServiceseinem AWS SDK oder CLI
    - Verwenden Sie mit DescribeSeverityLevelseinem AWS SDK oder CLI
    - Verwendung von DescribeTrustedAdvisorCheckRefreshStatuses mit einer CLI
    - Verwendung von DescribeTrustedAdvisorCheckResult mit einer CLI
    - Verwendung von DescribeTrustedAdvisorCheckSummaries mit einer CLI
    - Verwendung von DescribeTrustedAdvisorChecks mit einer CLI
    - Verwendung von RefreshTrustedAdvisorCheck mit einer CLI
    - Verwenden Sie mit einem ResolveCase AWS SDK oder CLI
- Szenarien für Support verwenden AWS SDKs
  - Erste Schritte mit dem technischen Support

Grundlegende Beispiele für Support verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Support mit AWS SDKs verwendet werden.

Beispiele

- Hallo Support
- Lernen Sie die Grundlagen von Support mit einem AWS SDK
- Aktionen für Support verwenden AWS SDKs
  - Verwenden Sie mit AddAttachmentsToSeteinem AWS SDK oder CLI
  - Verwenden Sie mit AddCommunicationToCaseeinem AWS SDK oder CLI
  - Verwenden Sie mit CreateCaseeinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeAttachmenteinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeCaseseinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeCommunicationseinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeServiceseinem AWS SDK oder CLI
  - Verwenden Sie mit DescribeSeverityLevelseinem AWS SDK oder CLI
  - Verwendung von DescribeTrustedAdvisorCheckRefreshStatuses mit einer CLI
  - Verwendung von DescribeTrustedAdvisorCheckResult mit einer CLI
  - Verwendung von DescribeTrustedAdvisorCheckSummaries mit einer CLI
  - Verwendung von DescribeTrustedAdvisorChecks mit einer CLI
  - Verwendung von RefreshTrustedAdvisorCheck mit einer CLI
  - Verwenden Sie mit einem ResolveCase AWS SDK oder CLI

Hallo Support

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Support beginnen.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.AWSSupport;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        // You must have one of the following AWS Support plans: Business,
 Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            ).Build();

        // Now the client is available for injection.
        var supportClient =
 host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\tHello AWS Support! There are
 {response.Services.Count} services available.");
    }
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS SDK für .NET API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
```

```
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie „main()“ auf, um das Beispiel auszuführen.

```
import {
  DescribeServicesCommand,
  SupportClient,
} from "@aws-sdk/client-support";

// Change the value of 'region' to your preferred AWS Region.
const client = new SupportClient({ region: "us-east-1" });

const getServiceCount = async () => {
  try {
    const { services } = await client.send(new DescribeServicesCommand({}));
    return services.length;
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

export const main = async () => {
  try {
    const count = await getServiceCount();
    console.log(`Hello, AWS Support! There are ${count} services available.`);
  } catch (err) {
    console.error("Failed to get service count: ", err.message);
  }
};
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
```

```
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_support(support_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Support client and count
    the available services in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param support_client: A Boto3 Support Client object.
    """
    try:
        print("Hello, AWS Support! Let's count the available Support services:")
        response = support_client.describe_services()
        print(f"There are {len(response['services'])} services available.")
    except ClientError as err:
        if err.response["Error"]["Code"] == "SubscriptionRequiredException":
            logger.info(
                "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                "examples."
            )
        else:
            logger.error(
                "Couldn't count services. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


if __name__ == "__main__":
    hello_support(boto3.client("support"))
```

- Einzelheiten zur API finden Sie DescribeServicesin AWS SDK for Python (Boto3) API Reference.

Lernen Sie die Grundlagen von Support mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Hello AWS Support example.
/// </summary>
public static class SupportCaseScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    To use the AWS Support API, you must have one of the following AWS Support
 plans: Business, Enterprise On-Ramp, or Enterprise.

    This .NET example performs the following tasks:
    1.  Get and display services. Select a service from the list.
```

```
   2.  Select a category from the selected service.
   3.  Get and display severity levels and select a severity level from the
list.
   4.  Create a support case using the selected service, category, and severity
level.
   5.  Get and display a list of open support cases for the current day.
   6.  Create an attachment set with a sample text file to add to the case.
   7.  Add a communication with the attachment to the support case.
   8.  List the communications of the support case.
   9.  Describe the attachment set.
   10. Resolve the support case.
   11. Get a list of resolved cases for the current day.
  */

   private static SupportWrapper _supportWrapper = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the AWS Support service.
       // Use your AWS profile name, or leave it blank to use the default
profile.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonAWSSupport>(new AWSOptions()
{ Profile = "default" })
                   .AddTransient<SupportWrapper>()
           )
           .Build();

       var logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger(typeof(SupportCaseScenario));

       _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();

       Console.WriteLine(new string('-', 80));
       Console.WriteLine("Welcome to the AWS Support case example scenario.");
```

```
        Console.WriteLine(new string('-', 80));

        try
        {
            var apiSupported = await _supportWrapper.VerifySubscription();
            if (!apiSupported)
            {
                logger.LogError("You must have a Business, Enterprise On-Ramp, or
Enterprise Support " +
                                "plan to use the AWS Support API. \n\tPlease
upgrade your subscription to run these examples.");
                return;
            }

            var service = await DisplayAndSelectServices();

            var category = DisplayAndSelectCategories(service);

            var severityLevel = await DisplayAndSelectSeverity();

            var caseId = await CreateSupportCase(service, category,
severityLevel);

            await DescribeTodayOpenCases();

            var attachmentSetId = await CreateAttachmentSet();

            await AddCommunicationToCase(attachmentSetId, caseId);

            var attachmentId = await ListCommunicationsForCase(caseId);

            await DescribeCaseAttachment(attachmentId);

            await ResolveCase(caseId);

            await DescribeTodayResolvedCases();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("AWS Support case example scenario complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
```

```
        }
    }

    /// <summary>
    /// List some available services from AWS Support, and select a service for
the example.
    /// </summary>
    /// <returns>The selected service.</returns>
    private static async Task<Service> DisplayAndSelectServices()
    {
        Console.WriteLine(new string('-', 80));
        var services = await _supportWrapper.DescribeServices();
        Console.WriteLine($"AWS Support client returned {services.Count}
services.");

        Console.WriteLine($"1. Displaying first 10 services:");
        for (int i = 0; i < 10 && i < services.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {services[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > services.Count)
        {
            Console.WriteLine(
                "Select an example support service by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return services[choiceNumber - 1];
    }

    /// <summary>
    /// List the available categories for a service and select a category for the
example.
    /// </summary>
    /// <param name="service">Service to use for displaying categories.</param>
    /// <returns>The selected category.</returns>
    private static Category DisplayAndSelectCategories(Service service)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
        for (int i = 0; i < service.Categories.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {service.Categories[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
        {
            Console.WriteLine(
                "Select an example support category by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        Console.WriteLine(new string('-', 80));

        return service.Categories[choiceNumber - 1];
    }

    /// <summary>
    /// List available severity levels from AWS Support, and select a level for
the example.
    /// </summary>
    /// <returns>The selected severity level.</returns>
    private static async Task<SeverityLevel> DisplayAndSelectSeverity()
    {
        Console.WriteLine(new string('-', 80));
        var severityLevels = await _supportWrapper.DescribeSeverityLevels();

        Console.WriteLine($"3. Get and display available severity levels:");
        for (int i = 0; i < 10 && i < severityLevels.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {severityLevels[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
        {
            Console.WriteLine(
```

```
                "Select an example severity level by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return severityLevels[choiceNumber - 1];
    }

    /// <summary>
    /// Create an example support case.
    /// </summary>
    /// <param name="service">Service to use for the new case.</param>
    /// <param name="category">Category to use for the new case.</param>
    /// <param name="severity">Severity to use for the new case.</param>
    /// <returns>The caseId of the new support case.</returns>
    private static async Task<string> CreateSupportCase(Service service,
        Category category, SeverityLevel severity)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Create an example support case" +
                          $" with the following settings:" +
                          $" \n\tService: {service.Name}, Category:
{category.Name} " +
                          $"and Severity Level: {severity.Name}.");
        var caseId = await _supportWrapper.CreateCase(service.Code,
category.Code, severity.Code,
            "Example case for testing, ignore.", "This is my example support
case.");

        Console.WriteLine($"\tNew case created with ID {caseId}");

        Console.WriteLine(new string('-', 80));

        return caseId;
    }

    /// <summary>
    /// List open cases for the current day.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeTodayOpenCases()
    {
```

```
        Console.WriteLine($"5. List the open support cases for the current
day.");
        // Describe the cases. If it is empty, try again and allow time for the
new case to appear.
        List<CaseDetails> currentOpenCases = null!;
        while (currentOpenCases == null || currentOpenCases.Count == 0)
        {
            Thread.Sleep(1000);
            currentOpenCases = await _supportWrapper.DescribeCases(
                new List<string>(),
                null,
                false,
                false,
                DateTime.UtcNow.Date,
                DateTime.UtcNow);
        }

        foreach (var openCase in currentOpenCases)
        {
            Console.WriteLine($"\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an attachment set for a support case.
    /// </summary>
    /// <returns>The attachment set id.</returns>
    private static async Task<string> CreateAttachmentSet()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Create an attachment set for a support case.");
        var fileName = "example_attachment.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for attachment to a support case.");
        }
```

```
        await using var ms = new MemoryStream(await
 File.ReadAllBytesAsync(fileName));

        var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
            ms,
            fileName);

        Console.WriteLine($"\tNew attachment set created with id: \n
\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    /// <summary>
    /// Add an attachment set and communication to a case.
    /// </summary>
    /// <param name="attachmentSetId">Id of the attachment set.</param>
    /// <param name="caseId">Id of the case to receive the attachment set.</
param>
    /// <returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId,
 string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Add attachment set and communication to
 {caseId}.");

        await _supportWrapper.AddCommunicationToCase(
            caseId,
            "This is an example communication added to a support case.",
            attachmentSetId);

        Console.WriteLine($"\tNew attachment set and communication added to
 {caseId}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the communications for a case.
    /// </summary>
    /// <param name="caseId">Id of the case to describe.</param>
```

```
    /// <returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. List communications for case {caseId}.");

        var communications = await
_supportWrapper.DescribeCommunications(caseId);
        var attachmentId = "";
        foreach (var communication in communications)
        {
            Console.WriteLine(
                $"\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
            if (communication.AttachmentSet.Any())
            {
                attachmentId = communication.AttachmentSet.First().AttachmentId;
            }
        }

        Console.WriteLine(new string('-', 80));
        return attachmentId;
    }

    /// <summary>
    /// Describe an attachment by id.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment to describe.</param>
    /// <returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Describe the attachment set.");

        var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
        var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
        Console.WriteLine($"\tAttachment includes {attachment.FileName} with
data: \n\t{data}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Resolve the support case.
```

```
    /// </summary>
    /// <param name="caseId">Id of the case to resolve.</param>
    /// <returns>Async task.</returns>
    private static async Task ResolveCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Resolve case {caseId}.");

        var status = await _supportWrapper.ResolveCase(caseId);
        Console.WriteLine($"\tCase {caseId} has final status {status}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List resolved cases for the current day.
    /// </summary>
    /// <returns>Async Task.</returns>
    private static async Task DescribeTodayResolvedCases()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. List the resolved support cases for the current
 day.");
        var currentCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
            true,
            DateTime.UtcNow.Date,
            DateTime.UtcNow);

        foreach (var currentCase in currentCases)
        {
            if (currentCase.Status == "resolved")
            {
                Console.WriteLine(
                    $"\tCase: {currentCase.CaseId}: status
 {currentCase.Status}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
}
```

Wrapper-Methoden, die vom Szenario für Support Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
public class SupportWrapper
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }


    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }



    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
```

```
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language
= "en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }



    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the
new case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null,
string issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
```

```
                CommunicationBody = body
            });
        return response.CaseId;
    }



    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does
not exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates
a new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string
fileName, string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }



    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
```

```
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }



    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
 email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }



    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
```

```
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in
 paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }



    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication.
 Defaults to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases.
 Defaults to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
```

```
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
string? displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }



    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
            {
                CaseId = caseId
            });
        return response.FinalCaseStatus;
    }
```

```
    /// <summary>
    /// Verify the support level for AWS Support API access.
    /// </summary>
    /// <returns>True if the subscription level supports API access.</returns>
    public async Task<bool> VerifySubscription()
    {
        try
        {
            var response = await _amazonSupport.DescribeServicesAsync(
                new DescribeServicesRequest()
                {
                    Language = "en"
                });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Amazon.AWSSupport.AmazonAWSSupportException ex)
        {
            if (ex.ErrorCode == "SubscriptionRequiredException")
            {
                return false;
            }
            else throw;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie verschiedene Support Operationen aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import
 software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import
 software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import
 software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import
 software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import
 software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import
 software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
```

```
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) {
       final String usage = """

               Usage:
                   <fileAttachment>Where:
                   fileAttachment - The file can be a simple saved .txt file to
use as an email attachment.\s
               """;

     //  if (args.length != 1) {
     //      System.out.println(usage);
     //      System.exit(1);
     //  }

       String fileAttachment = "C:\\AWS\\test.txt" ; //args[0];
       Region region = Region.US_WEST_2;
       SupportClient supportClient = SupportClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("***** Welcome to the AWS Support case example
scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Get and display available services.");
       List<String> sevCatList = displayServices(supportClient);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("2. Get and display Support severity levels.");
       String sevLevel = displaySevLevels(supportClient);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("3. Create a support case using the selected service,
category, and severity level.");
       String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
       if (caseId.compareTo("") == 0) {
           System.out.println("A support case was not successfully created!");
```

```
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to
add to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the
support case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current
day.");
```

```
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .includeResolvedCases(true)
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();
```

```
            ResolveCaseResponse response =
supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
```

```
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication
to an AWS Support case");
            else
                System.out.println("There was an error adding the communication
to an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
```

```
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                    .fileName(myFile.getName())
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
```

```
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();
```

```
            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();
```

```
                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario im Terminal aus.

```
import {
  AddAttachmentsToSetCommand,
  AddCommunicationToCaseCommand,
  CreateCaseCommand,
  DescribeAttachmentCommand,
  DescribeCasesCommand,
  DescribeCommunicationsCommand,
  DescribeServicesCommand,
  DescribeSeverityLevelsCommand,
  ResolveCaseCommand,
  SupportClient,
} from "@aws-sdk/client-support";
import * as inquirer from "@inquirer/prompts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};

const client = new SupportClient({ region: "us-east-1" });

// Verify that the account has a Support plan.
export const verifyAccount = async () => {
  const command = new DescribeServicesCommand({});

  try {
    await client.send(command);
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
```

```
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

/**
 * Select a service from the list returned from DescribeServices.
 */
export const getService = async () => {
  const { services } = await client.send(new DescribeServicesCommand({}));
  const selectedService = await inquirer.select({
    message:
      "Select a service. Your support case will be created for this service. The
 list of services is truncated for readability.",
    choices: services.slice(0, 10).map((s) => ({ name: s.name, value: s })),
  });
  return selectedService;
};

/**
 * @param {{ categories: import('@aws-sdk/client-support').Category[]}} service
 */
export const getCategory = async (service) => {
  const selectedCategory = await inquirer.select({
    message: "Select a category.",
    choices: service.categories.map((c) => ({ name: c.name, value: c })),
  });
  return selectedCategory;
};

// Get the available severity levels for the account.
export const getSeverityLevel = async () => {
  const command = new DescribeSeverityLevelsCommand({});
  const { severityLevels } = await client.send(command);
  const selectedSeverityLevel = await inquirer.select({
    message: "Select a severity level.",
    choices: severityLevels.map((s) => ({ name: s.name, value: s })),
  });
  return selectedSeverityLevel;
};

/**
```

```
 * Create a new support case
 * @param {{
 *  selectedService: import('@aws-sdk/client-support').Service
 *  selectedCategory: import('@aws-sdk/client-support').Category
 *  selectedSeverityLevel: import('@aws-sdk/client-support').SeverityLevel
 * }} selections
 * @returns
 */
export const createCase = async ({
  selectedService,
  selectedCategory,
  selectedSeverityLevel,
}) => {
  const command = new CreateCaseCommand({
    subject: "IGNORE: Test case",
    communicationBody: "This is a test. Please ignore.",
    serviceCode: selectedService.code,
    categoryCode: selectedCategory.code,
    severityCode: selectedSeverityLevel.code,
  });
  const { caseId } = await client.send(command);
  return caseId;
};

// Get a list of open support cases created today.
export const getTodaysOpenCases = async () => {
  const d = new Date();
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
  });

  const { cases } = await client.send(command);

  if (cases.length === 0) {
    throw new Error(
      "Unexpected number of cases. Expected more than 0 open cases.",
    );
  }
  return cases;
};

// Create an attachment set.
```

```
export const createAttachmentSet = async () => {
  const command = new AddAttachmentsToSetCommand({
    attachments: [
      {
        fileName: "example.txt",
        data: new TextEncoder().encode("some example text"),
      },
    ],
  });
  const { attachmentSetId } = await client.send(command);
  return attachmentSetId;
};

export const linkAttachmentSetToCase = async (attachmentSetId, caseId) => {
  const command = new AddCommunicationToCaseCommand({
    attachmentSetId,
    caseId,
    communicationBody: "Adding attachment set to case.",
  });
  await client.send(command);
};

// Get all communications for a support case.
export const getCommunications = async (caseId) => {
  const command = new DescribeCommunicationsCommand({
    caseId,
  });
  const { communications } = await client.send(command);
  return communications;
};

/**
 * @param {import('@aws-sdk/client-support').Communication[]} communications
 */
export const getFirstAttachment = (communications) => {
  const firstCommWithAttachment = communications.find(
    (c) => c.attachmentSet.length > 0,
  );
  return firstCommWithAttachment?.attachmentSet[0].attachmentId;
};

// Get an attachment.
export const getAttachment = async (attachmentId) => {
  const command = new DescribeAttachmentCommand({
```

```
    attachmentId,
  });
  const { attachment } = await client.send(command);
  return attachment;
};

// Resolve the case matching the given case ID.
export const resolveCase = async (caseId) => {
  const shouldResolve = await inquirer.confirm({
    message: `Do you want to resolve ${caseId}?`,
  });

  if (shouldResolve) {
    const command = new ResolveCaseCommand({
      caseId: caseId,
    });

    await client.send(command);
    return true;
  }
  return false;
};

/**
 * Find a specific case in the list of provided cases by case ID.
 * If the case is not found, and the results are paginated, continue
 * paging through the results.
 * @param {{
 *   caseId: string,
 *   cases: import('@aws-sdk/client-support').CaseDetails[]
 *   nextToken: string
 * }} options
 * @returns
 */
export const findCase = async ({ caseId, cases, nextToken }) => {
  const foundCase = cases.find((c) => c.caseId === caseId);

  if (foundCase) {
    return foundCase;
  }

  if (nextToken) {
    const response = await client.send(
      new DescribeCasesCommand({
```

```
        nextToken,
        includeResolvedCases: true,
      }),
    );
    return findCase({
      caseId,
      cases: response.cases,
      nextToken: response.nextToken,
    });
  }

  throw new Error(`${caseId} not found.`);
};

// Get all cases created today.
export const getTodaysResolvedCases = async (caseIdToWaitFor) => {
  const d = new Date("2023-01-18");
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
    includeResolvedCases: true,
  });
  const { cases, nextToken } = await client.send(command);
  await findCase({ cases, caseId: caseIdToWaitFor, nextToken });
  return cases.filter((c) => c.status === "resolved");
};

const main = async () => {
  let caseId;
  try {
    console.log(wrapText("Welcome to the AWS Support basic usage scenario."));

    // Verify that the account is subscribed to support.
    await verifyAccount();

    // Provided a truncated list of services and prompt the user to select one.
    const selectedService = await getService();

    // Provided the categories for the selected service and prompt the user to
 select one.
    const selectedCategory = await getCategory(selectedService);
```

```
   // Provide the severity available severity levels for the account and prompt
the user to select one.
   const selectedSeverityLevel = await getSeverityLevel();

   // Create a support case.
   console.log("\nCreating a support case.");
   caseId = await createCase({
     selectedService,
     selectedCategory,
     selectedSeverityLevel,
   });
   console.log(`Support case created: ${caseId}`);

   // Display a list of open support cases created today.
   const todaysOpenCases = await retry(
     { intervalInMs: 1000, maxRetries: 15 },
     getTodaysOpenCases,
   );
   console.log(
     `\nOpen support cases created today: ${todaysOpenCases.length}`,
   );
   console.log(todaysOpenCases.map((c) => `${c.caseId}`).join("\n"));

   // Create an attachment set.
   console.log("\nCreating an attachment set.");
   const attachmentSetId = await createAttachmentSet();
   console.log(`Attachment set created: ${attachmentSetId}`);

   // Add the attachment set to the support case.
   console.log(`\nAdding attachment set to ${caseId}`);
   await linkAttachmentSetToCase(attachmentSetId, caseId);
   console.log(`Attachment set added to ${caseId}`);

   // List the communications for a support case.
   console.log(`\nListing communications for ${caseId}`);
   const communications = await getCommunications(caseId);
   console.log(
     communications
       .map(
         (c) =>
           `Communication created on ${c.timeCreated}. Has
${c.attachmentSet.length} attachments.`,
       )
       .join("\n"),
```

```
    );

    // Describe the first attachment.
    console.log(`\nDescribing attachment ${attachmentSetId}`);
    const attachmentId = getFirstAttachment(communications);
    const attachment = await getAttachment(attachmentId);
    console.log(
      `Attachment is the file '${
        attachment.fileName
      }' with data: \n${new TextDecoder().decode(attachment.data)}`,
    );

    // Confirm that the support case should be resolved.
    const isResolved = await resolveCase(caseId);
    if (isResolved) {
      // List the resolved cases and include the one previously created.
      // Resolved cases can take a while to appear.
      console.log(
        "\nWaiting for case status to be marked as resolved. This can take some
 time.",
      );
      const resolvedCases = await retry(
        { intervalInMs: 20000, maxRetries: 15 },
        () => getTodaysResolvedCases(caseId),
      );
      console.log("Resolved cases:");
      console.log(resolvedCases.map((c) => c.caseId).join("\n"));
    }
  } catch (err) {
    console.error(err);
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases

- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
In addition, you must have the AWS Business Support Plan to use the AWS Support
 Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following tasks:
1. Gets and displays available services.
2. Gets and displays severity levels.
3. Creates a support case by using the selected service, category, and severity
 level.
4. Gets a list of open cases for the current day.
5. Creates an attachment set with a generated file.
6. Adds a communication with the attachment to the support case.
7. Lists the communications of the support case.
8. Describes the attachment set included with the communication.
9. Resolves the support case.
10. Gets a list of resolved cases for the current day.
```

```
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()

    println("***** Step 3. Create a support case using the selected service,
 category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add
 to the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
 case.")
```

```
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
 communication.")
    describeAttachment(attachId)

    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
```

```
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
```

```
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS
 Support case")
        } else {
            println("There was an error adding the communication to an AWS
 Support case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
```

```
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
```

```
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
```

```
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SupportCasesScenario:
    """Runs an interactive scenario that shows how to get started using AWS
 Support."""

    def __init__(self, support_wrapper):
        """
        :param support_wrapper: An object that wraps AWS Support actions.
        """
        self.support_wrapper = support_wrapper

    def display_and_select_service(self):
        """
        Lists support services and prompts the user to select one.

        :return: The support service selected by the user.
        """
        print("-" * 88)
        services_list = self.support_wrapper.describe_services("en")
        print(f"AWS Support client returned {len(services_list)} services.")
        print("Displaying first 10 services:")

        service_choices = [svc["name"] for svc in services_list[:10]]
        selected_index = q.choose(
            "Select an example support service by entering a number from the
 preceding list:",
            service_choices,
        )
        selected_service = services_list[selected_index]
        print("-" * 88)
        return selected_service

    def display_and_select_category(self, service):
        """
        Lists categories for a support service and prompts the user to select
 one.

        :param service: The service of the categories.
        :return: The selected category.
        """
        print("-" * 88)
        print(
```

```
            f"Available support categories for Service {service['name']}
{len(service['categories'])}:"
        )
        categories_choices = [category["name"] for category in
service["categories"]]
        selected_index = q.choose(
            "Select an example support category by entering a number from the
preceding list:",
            categories_choices,
        )
        selected_category = service["categories"][selected_index]
        print("-" * 88)
        return selected_category

    def display_and_select_severity(self):
        """
        Lists available severity levels and prompts the user to select one.

        :return: The selected severity level.
        """
        print("-" * 88)
        severity_levels_list =
self.support_wrapper.describe_severity_levels("en")
        print(f"Available severity levels:")
        severity_choices = [level["name"] for level in severity_levels_list]
        selected_index = q.choose(
            "Select an example severity level by entering a number from the
preceding list:",
            severity_choices,
        )
        selected_severity = severity_levels_list[selected_index]
        print("-" * 88)
        return selected_severity

    def create_example_case(self, service, category, severity_level):
        """
        Creates an example support case with the user's selections.

        :param service: The service for the new case.
        :param category: The category for the new case.
        :param severity_level: The severity level for the new case.
        :return: The caseId of the new support case.
        """
        print("-" * 88)
```

```
        print(f"Creating new case for service {service['name']}.")
        case_id = self.support_wrapper.create_case(service, category,
severity_level)
        print(f"\tNew case created with ID {case_id}.")
        print("-" * 88)
        return case_id

    def list_open_cases(self):
        """
        List the open cases for the current day.
        """
        print("-" * 88)
        print("Let's list the open cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        open_cases = self.support_wrapper.describe_cases(start_time, end_time,
False)
        for case in open_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def create_attachment_set(self):
        """
        Create an attachment set with a sample file.

        :return: The attachment set ID of the new attachment set.
        """
        print("-" * 88)
        print("Creating attachment set with a sample file.")
        attachment_set_id = self.support_wrapper.add_attachment_to_set()
        print(f"\tNew attachment set created with ID {attachment_set_id}.")
        print("-" * 88)
        return attachment_set_id

    def add_communication(self, case_id, attachment_set_id):
        """
        Add a communication with an attachment set to the case.

        :param case_id: The ID of the case for the communication.
        :param attachment_set_id: The ID of the attachment set to
        add to the communication.
        """
        print("-" * 88)
        print(f"Adding a communication and attachment set to the case.")
```

```
        self.support_wrapper.add_communication_to_case(attachment_set_id,
case_id)
        print(
            f"Added a communication and attachment set {attachment_set_id} to the
case {case_id}."
        )
        print("-" * 88)

    def list_communications(self, case_id):
        """
        List the communications associated with a case.

        :param case_id: The ID of the case.
        :return: The attachment ID of an attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attachment_id = ""
        communications =
self.support_wrapper.describe_all_case_communications(case_id)
        for communication in communications:
            print(
                f"\tCommunication created on {communication['timeCreated']} "
                f"has {len(communication['attachmentSet'])} attachments."
            )
            if len(communication["attachmentSet"]) > 0:
                attachment_id = communication["attachmentSet"][0]["attachmentId"]
        print("-" * 88)
        return attachment_id

    def describe_case_attachment(self, attachment_id):
        """
        Describe an attachment associated with a case.

        :param attachment_id: The ID of the attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attached_file = self.support_wrapper.describe_attachment(attachment_id)
        print(f"\tAttachment includes file {attached_file}.")
        print("-" * 88)

    def resolve_case(self, case_id):
        """
```

```
        Shows how to resolve an AWS Support case by its ID.

        :param case_id: The ID of the case to resolve.
        """
        print("-" * 88)
        print(f"Resolving case with ID {case_id}.")
        case_status = self.support_wrapper.resolve_case(case_id)
        print(f"\tFinal case status is {case_status}.")
        print("-" * 88)

    def list_resolved_cases(self):
        """
        List the resolved cases for the current day.
        """
        print("-" * 88)
        print("Let's list the resolved cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        resolved_cases = self.support_wrapper.describe_cases(start_time,
end_time, True)
        for case in resolved_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print("Welcome to the AWS Support get started with support cases demo.")
        print("-" * 88)

        selected_service = self.display_and_select_service()
        selected_category = self.display_and_select_category(selected_service)
        selected_severity = self.display_and_select_severity()
        new_case_id = self.create_example_case(
            selected_service, selected_category, selected_severity
        )
        wait(10)
        self.list_open_cases()
        new_attachment_set_id = self.create_attachment_set()
        self.add_communication(new_case_id, new_attachment_set_id)
        new_attachment_id = self.list_communications(new_case_id)
        self.describe_case_attachment(new_attachment_id)
```

```
        self.resolve_case(new_case_id)
        wait(10)
        self.list_resolved_cases()

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = SupportCasesScenario(SupportWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die unterstützende Client-Aktionen umschließt.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a
 language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
```

```
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
```

```
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
```

```
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does
not exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a
support case.",
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
```

```
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.
```

```
        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator =
self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
```

```
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

```
        else:
            return final_status


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
```

```
            return cases
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen für Support verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Support Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Support API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Support verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Support -API-Referenz.

Beispiele

- Verwenden Sie mit AddAttachmentsToSeteinem AWS SDK oder CLI
- Verwenden Sie mit AddCommunicationToCaseeinem AWS SDK oder CLI
- Verwenden Sie mit CreateCaseeinem AWS SDK oder CLI
- Verwenden Sie mit DescribeAttachmenteinem AWS SDK oder CLI

- Verwenden Sie mit DescribeCaseseinem AWS SDK oder CLI
- Verwenden Sie mit DescribeCommunicationseinem AWS SDK oder CLI
- Verwenden Sie mit DescribeServiceseinem AWS SDK oder CLI
- Verwenden Sie mit DescribeSeverityLevelseinem AWS SDK oder CLI
- Verwendung von DescribeTrustedAdvisorCheckRefreshStatuses mit einer CLI
- Verwendung von DescribeTrustedAdvisorCheckResult mit einer CLI
- Verwendung von DescribeTrustedAdvisorCheckSummaries mit einer CLI
- Verwendung von DescribeTrustedAdvisorChecks mit einer CLI
- Verwendung von RefreshTrustedAdvisorCheck mit einer CLI
- Verwenden Sie mit einem ResolveCase AWS SDK oder CLI

Verwenden Sie mit **`AddAttachmentsToSet`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AddAttachmentsToSet` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does
not exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
```

```
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates
 a new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string
 fileName, string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie einen Anhang zu einem Satz hinzu

Im folgenden `add-attachments-to-set` Beispiel wird einem Set ein Bild hinzugefügt, das Sie dann für einen Support-Fall in Ihrem AWS Konto angeben können.

```
aws support add-attachments-to-set \
    --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE" \
    --attachments fileName=troubleshoot-screenshot.png,data=base64-encoded-string
```

Ausgabe:

```
{
    "attachmentSetId": "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE",
    "expiryTime": "2020-05-14T17:04:40.790+0000"
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
       try {
           File myFile = new File(fileAttachment);
           InputStream sourceStream = new FileInputStream(myFile);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           Attachment attachment = Attachment.builder()
                   .fileName(myFile.getName())
                   .data(sourceBytes)
                   .build();

           AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                   .attachments(attachment)
                   .build();

           AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
           return response.attachmentSetId();
```

```
        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AddAttachmentsToSetCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new attachment set or add attachments to an existing set.
    // Provide an 'attachmentSetId' value to add attachments to an existing set.
    // Use AddCommunicationToCase or CreateCase to associate an attachment set
 with a support case.
    const response = await client.send(
      new AddAttachmentsToSetCommand({
        // You can add up to three attachments per set. The size limit is 5 MB
 per attachment.
        attachments: [
          {
            fileName: "example.txt",
            data: new TextEncoder().encode("some example text"),
          },
```

```
        ],
      }),
    );
    // Use this ID in AddCommunicationToCase or CreateCase.
    console.log(response.attachmentSetId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
```

```
        return response.attachmentSetId
    }
}
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does
 not exist.
```

```
        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a
support case.",
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id
```

- Einzelheiten zur API finden Sie AddAttachmentsToSetin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`AddCommunicationToCase`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AddCommunicationToCase` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
 email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So fügen Sie Kommunikation zu einem Fall hinzu

Im folgenden `add-communication-to-case` Beispiel werden Mitteilungen zu einem Support-Fall in Ihrem AWS Konto hinzugefügt.

```
aws support add-communication-to-case \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
    --communication-body "I'm attaching a set of images to this case." \
    --cc-email-addresses "myemail@example.com" \
    --attachment-set-id "as-2f5a6faa2a4a1e600-mu-nk5xQlBr70-
G1cUos5LZkd38KOAHZa9BMDVzNEXAMPLE"
```

Ausgabe:

```
{
    "result": true
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie AddCommunicationToCasein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
       try {
           AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                   .caseId(caseId)
                   .attachmentSetId(attachmentSetId)
                   .communicationBody("Please refer to attachment for details.")
                   .build();

           AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
           if (response.result())
               System.out.println("You have successfully added a communication
to an AWS Support case");
           else
               System.out.println("There was an error adding the communication
to an AWS Support case");

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AddCommunicationToCaseCommand } from "@aws-sdk/client-support";
```

```
import { client } from "../libs/client.js";

export const main = async () => {
  let attachmentSetId;

  try {
    // Add a communication to a case.
    const response = await client.send(
      new AddCommunicationToCaseCommand({
        communicationBody: "Adding an attachment.",
        // Set value to an existing support case id.
        caseId: "CASE_ID",
        // Optional. Set value to an existing attachment set id to add
 attachments to the case.
        attachmentSetId,
      }),
    );
    console.log(response);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
```

```
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS
 Support case")
        } else {
            println("There was an error adding the communication to an AWS
 Support case")
        }
    }
}
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Fügt den Text einer E-Mail-Kommunikation zum angegebenen Fall hinzu.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Beispiel 2: Fügt den Text einer E-Mail-Nachricht zum angegebenen Fall sowie eine oder mehrere E-Mail-Adressen hinzu, die in der CC-Zeile der E-Mail enthalten sind.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Einzelheiten zur API finden Sie unter AddCommunicationToCase AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Fügt den Text einer E-Mail-Kommunikation zum angegebenen Fall hinzu.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Beispiel 2: Fügt den Text einer E-Mail-Nachricht zum angegebenen Fall sowie eine oder mehrere E-Mail-Adressen hinzu, die in der CC-Zeile der E-Mail enthalten sind.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- Einzelheiten zur API finden Sie unter AddCommunicationToCase AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
```

```
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- Einzelheiten zur API finden Sie AddCommunicationToCasein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`CreateCase`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateCase` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Create a new support case.
   /// </summary>
   /// <param name="serviceCode">Service code for the new case.</param>
   /// <param name="categoryCode">Category for the new case.</param>
   /// <param name="severityCode">Severity code for the new case.</param>
   /// <param name="subject">Subject of the new case.</param>
   /// <param name="body">Body text of the new case.</param>
   /// <param name="language">Optional language support for your case.
   /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
   /// <param name="attachmentSetId">Optional Id for an attachment set for the
new case.</param>
   /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
   /// <returns>The caseId of the new support case.</returns>
   public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
       string body, string language = "en", string? attachmentSetId = null,
string issueType = "customer-service")
```

```
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }
```

- Einzelheiten zur API finden Sie CreateCasein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie einen Fall

Im folgenden `create-case` Beispiel wird ein Support-Fall für Ihr AWS Konto erstellt.

```
aws support create-case \
    --category-code "using-aws" \
    --cc-email-addresses "myemail@example.com" \
    --communication-body "I want to learn more about an AWS service." \
    --issue-type "technical" \
    --language "en" \
    --service-code "general-info" \
    --severity-code "low" \
    --subject "Question about my account"
```

Ausgabe:

```
{
    "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47"
```

```
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie CreateCasein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                   .categoryCode(caseCat.toLowerCase())
                   .serviceCode(serviceCode.toLowerCase())
                   .severityCode(sevLevel.toLowerCase())
                   .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                   .subject("Test case, please ignore")
                   .language("en")
                   .issueType("technical")
                   .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie CreateCasein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new case and log the case id.
    // Important: This creates a real support case in your account.
    const response = await client.send(
      new CreateCaseCommand({
        // The subject line of the case.
        subject: "IGNORE: Test case",
        // Use DescribeServices to find available service codes for each service.
        serviceCode: "service-quicksight-end-user",
        // Use DescribeSecurityLevels to find available severity codes for your
 support plan.
        severityCode: "low",
        // Use DescribeServices to find available category codes for each
 service.
        categoryCode: "end-user-support",
        // The main description of the support case.
        communicationBody: "This is a test. Please ignore.",
      }),
    );
    console.log(response.caseId);
    return response;
  } catch (err) {
    console.error(err);
```

```
  }
};
```

- Einzelheiten zur API finden Sie CreateCasein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- Einzelheiten zur API finden Sie CreateCasein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Erstellt einen neuen Fall im AWS Support Center. Werte für die CategoryCode Parameter - ServiceCode und - können mit dem Get-ASAService Cmdlet abgerufen werden. Der Wert für den SeverityCode Parameter - kann mit dem Cmdlet abgerufen werden. Get-ASASeverityLevel Der Wert des IssueType Parameters - kann entweder „Kundenservice“ oder „technisch“ sein. Bei Erfolg wird die AWS Support-Fallnummer ausgegeben. Standardmäßig wird der Fall auf Englisch bearbeitet. Um Japanisch zu verwenden, fügen Sie den Parameter -Language „ja“ hinzu. Die CommunicationBody Parameter -ServiceCode, -CategoryCode, -Subject und - sind obligatorisch.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Einzelheiten zur API finden Sie unter CreateCase AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellt einen neuen Fall im AWS Support Center. Werte für die CategoryCode Parameter - ServiceCode und - können mit dem Get-ASAService Cmdlet abgerufen werden. Der Wert für den SeverityCode Parameter - kann mit dem Cmdlet abgerufen werden. Get-ASASeverityLevel Der Wert des IssueType Parameters - kann entweder „Kundenservice“ oder „technisch“ sein. Bei Erfolg wird die AWS Support-Fallnummer ausgegeben. Standardmäßig wird der Fall auf Englisch bearbeitet. Um Japanisch zu verwenden, fügen Sie den Parameter -Language „ja“ hinzu. Die CommunicationBody Parameter -ServiceCode, -CategoryCode, -Subject und - sind obligatorisch.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- Einzelheiten zur API finden Sie unter CreateCase AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
```

```
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id
```

- Einzelheiten zur API finden Sie CreateCasein AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeAttachment`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAttachment` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
 describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }
```

- Einzelheiten zur API finden Sie DescribeAttachmentin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie einen Anhang

Im folgenden Beispiel für `describe-attachment` werden Informationen zum Anhang mit der angegebenen ID zurückgegeben.

```
aws support describe-attachment \
```

```
    --attachment-id "attachment-KBnjRNrePd9D6Jx0-Mm00xZuDEaL2JAj_0-
gJv9qqDooTipsz3V1Nb19rCfkZneeQeDPgp8X1iVJyHH7UuhZDdNeqGoduZsPrAhyMakqlc60-
iJjL5HqyYGiT1FG8EXAMPLE"
```

Ausgabe:

```
{
    "attachment": {
        "fileName": "troubleshoot-screenshot.png",
        "data": "base64-blob"
    }
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeAttachmentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeAttachment(SupportClient supportClient, String
attachId) {
       try {
           DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                   .attachmentId(attachId)
                   .build();

           DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
           System.out.println("The name of the file is " +
response.attachment().fileName());
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeAttachmentin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeAttachmentCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the metadata and content of an attachment.
    const response = await client.send(
      new DescribeAttachmentCommand({
        // Set value to an existing attachment id.
        // Use DescribeCommunications or DescribeCases to find an attachment id.
        attachmentId: "ATTACHMENT_ID",
      }),
    );
    console.log(response.attachment?.fileName);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie DescribeAttachmentin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- Einzelheiten zur API finden Sie DescribeAttachmentin der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
```

```
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file
```

- Einzelheiten zur API finden Sie DescribeAttachmentin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeCases`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCases` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication.
Defaults to true.</param>
```

```
    /// <param name="includeResolvedCases">True to include resolved cases.
 Defaults to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
 (“ko”) are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
 string? displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie DescribeCasesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie einen Fall

Das folgende `describe-cases` Beispiel gibt Informationen über den angegebenen Supportfall in Ihrem AWS Konto zurück.

```
aws support describe-cases \
    --display-id "1234567890" \
    --after-time "2020-03-23T21:31:47.774Z" \
    --include-resolved-cases \
    --language "en" \
    --no-include-communications \
    --max-item 1
```

Ausgabe:

```
{
    "cases": [
        {
            "status": "resolved",
            "ccEmailAddresses": [],
            "timeCreated": "2020-03-23T21:31:47.774Z",
            "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
            "severityCode": "low",
            "language": "en",
            "categoryCode": "using-aws",
            "serviceCode": "general-info",
            "submittedBy": "myemail@example.com",
            "displayId": "1234567890",
            "subject": "Question about my account"
        }
    ]
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeCasesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie DescribeCasesin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeCasesCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all of the unresolved cases in your account.
    // Filter or expand results by providing parameters to the
 DescribeCasesCommand. Refer
    // to the TypeScript definition and the API doc for more information on
 possible parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecasescommandinput.html
    const response = await client.send(new DescribeCasesCommand({}));
    const caseIds = response.cases.map((supportCase) => supportCase.caseId);
    console.log(caseIds);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie DescribeCasesin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeCasesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt die Details aller Supportfälle zurück.

```
Get-ASACase
```

Beispiel 2: Gibt die Details aller Supportfälle seit dem angegebenen Datum und der angegebenen Uhrzeit zurück.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Beispiel 3: Gibt die Details der ersten 10 Supportfälle zurück, einschließlich derer, die gelöst wurden.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Beispiel 4: Gibt die Details des einzelnen angegebenen Supportfalls zurück.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 5: Gibt die Details der angegebenen Supportfälle zurück.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- Einzelheiten zur API finden Sie unter DescribeCases AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Details aller Supportfälle zurück.

```
Get-ASACase
```

Beispiel 2: Gibt die Details aller Supportfälle seit dem angegebenen Datum und der angegebenen Uhrzeit zurück.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Beispiel 3: Gibt die Details der ersten 10 Supportfälle zurück, einschließlich derer, die gelöst wurden.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Beispiel 4: Gibt die Details des einzelnen angegebenen Supportfalls zurück.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 5: Gibt die Details der angegebenen Supportfälle zurück.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- Einzelheiten zur API finden Sie unter DescribeCases AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_cases(self, after_time, before_time, resolved):
```

```
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- Einzelheiten zur API finden Sie DescribeCasesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeCommunications`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeCommunications` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
```

```
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in
 paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So beschreiben Sie die neueste Kommunikation für einen Fall

Das folgende `describe-communications` Beispiel gibt die neueste Kommunikation für den angegebenen Supportfall in Ihrem AWS Konto zurück.

```
aws support describe-communications \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47" \
    --after-time "2020-03-23T21:31:47.774Z" \
    --max-item 1
```

Ausgabe:

```
{
    "communications": [
        {
            "body": "I want to learn more about an AWS service.",
            "attachmentSet": [],
            "caseId": "case-12345678910-2013-c4c1d2bf33c5cf47",
            "timeCreated": "2020-05-12T23:12:35.000Z",
```

```
            "submittedBy": "Amazon Web Services"
        }
    ],
    "NextToken":
 "eyJuZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQEXAMPLE=="
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeCommunicationsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String listCommunications(SupportClient supportClient, String
caseId) {
       try {
           String attachId = null;
           DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                   .caseId(caseId)
                   .maxResults(10)
                   .build();

           DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
           List<Communication> communications = response.communications();
           for (Communication comm : communications) {
               System.out.println("the body is: " + comm.body());

               // Get the attachment id value.
               List<AttachmentDetails> attachments = comm.attachmentSet();
               for (AttachmentDetails detail : attachments) {
                   attachId = detail.attachmentId();
```

```
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeCommunicationsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all communications for the support case.
    // Filter results by providing parameters to the
 DescribeCommunicationsCommand. Refer
    // to the TypeScript definition and the API doc for more information on
 possible parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecommunicationscommandinput.html
    const response = await client.send(
      new DescribeCommunicationsCommand({
        // Set value to an existing case id.
```

```
        caseId: "CASE_ID",
      }),
    );
    const text = response.communications.map((item) => item.body).join("\n");
    console.log(text);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
```

```
    }
    return ""
}
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt alle Kommunikationen für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 2: Gibt alle Kommunikationen seit Mitternacht UTC am 1. Januar 2012 für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- Einzelheiten zur API finden Sie unter DescribeCommunications AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle Kommunikationen für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 2: Gibt alle Kommunikationen seit Mitternacht UTC am 1. Januar 2012 für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- Einzelheiten zur API finden Sie unter DescribeCommunications AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator =
 self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications
```

- Einzelheiten zur API finden Sie DescribeCommunicationsin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeServices`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeServices` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um AWS Dienste und Dienstkategorien aufzulisten

Im folgenden Beispiel für `describe-services` werden die verfügbaren Servicekategorien für die Anforderung allgemeiner Informationen aufgelistet.

```
aws support describe-services \
    --service-code-list "general-info"
```

Ausgabe:

```
{
    "services": [
        {
            "code": "general-info",
            "name": "General Info and Getting Started",
            "categories": [
                {
                    "code": "charges",
                    "name": "How Will I Be Charged?"
                },
                {
                    "code": "gdpr-queries",
                    "name": "Data Privacy Query"
                },
                {
                    "code": "reserved-instances",
                    "name": "Reserved Instances"
                },
                {
                    "code": "resource",
                    "name": "Where is my Resource?"
                },
                {
                    "code": "using-aws",
                    "name": "Using AWS & Services"
                },
                {
                    "code": "free-tier",
                    "name": "Free Tier"
                },
                {
                    "code": "security-and-compliance",
                    "name": "Security & Compliance"
                },
                {
                    "code": "account-structure",
                    "name": "Account Structure"
                }
```

```
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeServicesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
```

```
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
```

- Einzelheiten zur API finden Sie DescribeServicesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
```

```
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Einzelheiten zur API finden Sie DescribeServicesin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt alle verfügbaren Servicecodes, Namen und Kategorien zurück.

```
Get-ASAService
```

Beispiel 2: Gibt den Namen und die Kategorien für den Service mit dem angegebenen Code zurück.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Beispiel 3: Gibt den Namen und die Kategorien für die angegebenen Servicecodes zurück.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Beispiel 4: Gibt den Namen und die Kategorien (in Japanisch) für die angegebenen Servicecodes zurück. Derzeit werden die Sprachcodes Englisch („en“) und Japanisch („ja“) unterstützt.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt alle verfügbaren Servicecodes, Namen und Kategorien zurück.

```
Get-ASAService
```

Beispiel 2: Gibt den Namen und die Kategorien für den Service mit dem angegebenen Code zurück.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Beispiel 3: Gibt den Namen und die Kategorien für die angegebenen Servicecodes zurück.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Beispiel 4: Gibt den Namen und die Kategorien (in Japanisch) für die angegebenen Servicecodes zurück. Derzeit werden die Sprachcodes Englisch („en“) und Japanisch („ja“) unterstützt.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeServices AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)
```

```
    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a
language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services
```

- Einzelheiten zur API finden Sie DescribeServicesin AWS SDK for Python (Boto3) API Reference.

Verwenden Sie mit **`DescribeSeverityLevels`**einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeSeverityLevels` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean
(“ko”) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language
= "en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So listen Sie die verfügbaren Schweregrade auf

Im folgenden Beispiel für `describe-severity-levels` werden die verfügbaren Schweregrade für einen Supportfall aufgelistet.

```
aws support describe-severity-levels
```

Ausgabe:

```
{
    "severityLevels": [
        {
            "code": "low",
            "name": "Low"
        },
        {
            "code": "normal",
            "name": "Normal"
        },
        {
            "code": "high",
            "name": "High"
        },
        {
            "code": "urgent",
            "name": "Urgent"
        },
        {
            "code": "critical",
            "name": "Critical"
        }
    ]
}
```

Weitere Informationen finden Sie unter Auswählen eines Schweregrads im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String displaySevLevels(SupportClient supportClient) {
       try {
           DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                   .language("en")
                   .build();

           DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
           List<SeverityLevel> severityLevels = response.severityLevels();
           String levelName = null;
           for (SeverityLevel sevLevel : severityLevels) {
               System.out.println("The severity level name is: " +
sevLevel.name());
               if (sevLevel.name().compareTo("High") == 0)
                   levelName = sevLevel.name();
           }
           return levelName;

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSeverityLevelsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the list of severity levels.
    // The available values depend on the support plan for the account.
    const response = await client.send(new DescribeSeverityLevelsCommand({}));
    console.log(response.severityLevels);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response =
 supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können.

```
Get-ASASeverityLevel
```

Beispiel 2: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können. Die Namen der Grade werden auf Japanisch zurückgegeben.

```
Get-ASASeverityLevel -Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeSeverityLevels AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können.

```
Get-ASASeverityLevel
```

Beispiel 2: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können. Die Namen der Grade werden auf Japanisch zurückgegeben.

```
Get-ASASeverityLevel -Language "ja"
```

- Einzelheiten zur API finden Sie unter DescribeSeverityLevels AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
```

```
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why:
%s: %s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels
```

- Einzelheiten zur API finden Sie DescribeSeverityLevelsin AWS SDK for Python (Boto3) API Reference.

Verwendung von **`DescribeTrustedAdvisorCheckRefreshStatuses`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrustedAdvisorCheckRefreshStatuses` verwendet wird.

CLI

AWS CLI

Um den Aktualisierungsstatus von AWS Trusted Advisor Advisor-Prüfungen aufzulisten

Im folgenden Beispiel für `describe-trusted-advisor-check-refresh-statuses` wird der Aktualisierungsstatus für zwei Trusted-Advisor-Prüfungen aufgelistet: Bucket-Berechtigungen von Amazon S3 und Verwendung von IAM.

```
aws support describe-trusted-advisor-check-refresh-statuses \
    --check-id "Pfx0RwqBli" "zXCkfM1nI3"
```

Ausgabe:

```
{
    "statuses": [
        {
            "checkId": "Pfx0RwqBli",
            "status": "none",
            "millisUntilNextRefreshable": 0
        },
        {
            "checkId": "zXCkfM1nI3",
            "status": "none",
            "millisUntilNextRefreshable": 0
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS Trusted Advisor im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckRefreshStatuses AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt den aktuellen Status der Aktualisierungsanforderungen für die angegebenen Prüfungen zurück. Request-ASATrustedAdvisorCheckRefresh kann verwendet werden, um anzufordern, dass die Statusinformationen der Prüfungen aktualisiert werden.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckRefreshStatuses AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt den aktuellen Status der Aktualisierungsanforderungen für die angegebenen Prüfungen zurück. Request-ASATrustedAdvisorCheckRefresh kann verwendet werden, um anzufordern, dass die Statusinformationen der Prüfungen aktualisiert werden.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckRefreshStatuses AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTrustedAdvisorCheckResult`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrustedAdvisorCheckResult` verwendet wird.

CLI

AWS CLI

Um die Ergebnisse einer AWS Trusted Advisor-Prüfung aufzulisten

Im folgenden Beispiel für `describe-trusted-advisor-check-result` werden die Ergebnisse der IAM-Nutzungsprüfung aufgelistet.

```
aws support describe-trusted-advisor-check-result \
    --check-id "zXCkfM1nI3"
```

Ausgabe:

```
{
    "result": {
        "checkId": "zXCkfM1nI3",
        "timestamp": "2020-05-13T21:38:05Z",
        "status": "ok",
        "resourcesSummary": {
            "resourcesProcessed": 1,
            "resourcesFlagged": 0,
            "resourcesIgnored": 0,
            "resourcesSuppressed": 0
        },
        "categorySpecificSummary": {
            "costOptimizing": {
                "estimatedMonthlySavings": 0.0,
                "estimatedPercentMonthlySavings": 0.0
            }
        },
        "flaggedResources": [
            {
                "status": "ok",
                "resourceId": "47DEQpj8HBSa-_TImW-5JCeuQeRkm5NMpJWZEXAMPLE",
                "isSuppressed": false
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter AWS Trusted Advisor im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeTrustedAdvisorCheckResultunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Ergebnisse einer Prüfung von Trusted Advisor zurück. Die Liste der verfügbaren Trusted Advisor Advisor-Checks finden Sie unter Get-ASATrustedAdvisorChecks. Die Ausgabe umfasst den Gesamtstatus der Prüfung, den Zeitstempel, zu dem die Prüfung

zuletzt durchgeführt wurde, und die eindeutige Prüf-ID für die jeweilige Prüfung. Um die Ergebnisse auf Japanisch auszugeben, fügen Sie den Parameter -Language „ja“ hinzu.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckResult AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Ergebnisse einer Prüfung von Trusted Advisor zurück. Die Liste der verfügbaren Trusted Advisor Advisor-Checks finden Sie unter Get-ASATrustedAdvisorChecks. Die Ausgabe umfasst den Gesamtstatus der Prüfung, den Zeitstempel, zu dem die Prüfung zuletzt durchgeführt wurde, und die eindeutige Prüf-ID für die jeweilige Prüfung. Um die Ergebnisse auf Japanisch auszugeben, fügen Sie den Parameter -Language „ja“ hinzu.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckResult AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTrustedAdvisorCheckSummaries`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrustedAdvisorCheckSummaries` verwendet wird.

CLI

AWS CLI

Um die Zusammenfassungen der AWS Trusted Advisor Advisor-Prüfungen aufzulisten

Im folgenden Beispiel für `describe-trusted-advisor-check-summaries` werden die Ergebnisse von zwei Trusted-Advisor-Prüfungen aufgelistet: Bucket-Berechtigungen von Amazon S3 und Verwendung von IAM.

```
aws support describe-trusted-advisor-check-summaries \
    --check-ids "Pfx0RwqBli" "zXCkfM1nI3"
```

Ausgabe:

```
{
    "summaries": [
        {
            "checkId": "Pfx0RwqBli",
            "timestamp": "2020-05-13T21:38:12Z",
            "status": "ok",
            "hasFlaggedResources": true,
            "resourcesSummary": {
                "resourcesProcessed": 44,
                "resourcesFlagged": 0,
                "resourcesIgnored": 0,
                "resourcesSuppressed": 0
            },
            "categorySpecificSummary": {
                "costOptimizing": {
                    "estimatedMonthlySavings": 0.0,
                    "estimatedPercentMonthlySavings": 0.0
                }
            }
        },
        {
            "checkId": "zXCkfM1nI3",
            "timestamp": "2020-05-13T21:38:05Z",
            "status": "ok",
            "hasFlaggedResources": true,
            "resourcesSummary": {
                "resourcesProcessed": 1,
                "resourcesFlagged": 0,
                "resourcesIgnored": 0,
                "resourcesSuppressed": 0
            },
            "categorySpecificSummary": {
                "costOptimizing": {
                    "estimatedMonthlySavings": 0.0,
                    "estimatedPercentMonthlySavings": 0.0
                }
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS Trusted Advisor im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckSummaries AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die neueste Zusammenfassung für die angegebene Prüfung von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Beispiel 2: Gibt die neuesten Zusammenfassungen für die angegebenen Prüfungen von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckSummaries AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die neueste Zusammenfassung für die angegebene Prüfung von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Beispiel 2: Gibt die neuesten Zusammenfassungen für die angegebenen Prüfungen von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorCheckSummaries AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTrustedAdvisorChecks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTrustedAdvisorChecks` verwendet wird.

CLI

AWS CLI

Um die verfügbaren AWS Trusted Advisor Advisor-Checks aufzulisten

Das folgende `describe-trusted-advisor-checks` Beispiel listet die verfügbaren Trusted Advisor Advisor-Checks in Ihrem AWS Konto auf. Zu diesen Informationen gehören der Name, die ID, die Beschreibung, die Kategorie und die Metadaten der Prüfung. Beachten Sie, dass die Ausgabe aus Gründen der Lesbarkeit gekürzt ist.

```
aws support describe-trusted-advisor-checks \
    --language "en"
```

Ausgabe:

```
{
    "checks": [
        {
            "id": "zXCkfM1nI3",
            "name": "IAM Use",
            "description": "Checks for your use of AWS Identity and Access
 Management (IAM). You can use IAM to create users, groups, and roles in
 AWS, and you can use permissions to control access to AWS resources. \n<br>
\n<br>\n<b>Alert Criteria</b><br>\nYellow: No IAM users have been created
 for this account.\n<br>\n<br>\n<b>Recommended Action</b><br>\nCreate one or
 more IAM users and groups in your account. You can then create additional
 users whose permissions are limited to perform specific tasks in your AWS
 environment. For more information, see <a href=\"https://docs.aws.amazon.com/
IAM/latest/UserGuide/IAMGettingStarted.html\" target=\"_blank\">Getting
 Started</a>. \n<br><br>\n<b>Additional Resources</b><br>\n<a href=\"https://
docs.aws.amazon.com/IAM/latest/UserGuide/IAM_Introduction.html\" target=\"_blank
\">What Is IAM?</a>",
            "category": "security",
            "metadata": []
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS Trusted Advisor im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie DescribeTrustedAdvisorChecksin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Sammlung von Prüfungen von Trusted Advisor zurück. Sie müssen den Parameter „Language“ angeben, der entweder „en“ für englische Ausgabe oder „ja“ für japanische Ausgabe akzeptieren kann.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorChecks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Sammlung von Prüfungen von Trusted Advisor zurück. Sie müssen den Parameter „Language“ angeben, der entweder „en“ für englische Ausgabe oder „ja“ für japanische Ausgabe akzeptieren kann.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- Einzelheiten zur API finden Sie unter DescribeTrustedAdvisorChecks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RefreshTrustedAdvisorCheck`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RefreshTrustedAdvisorCheck` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie eine AWS Trusted Advisor Advisor-Prüfung

Im folgenden `refresh-trusted-advisor-check` Beispiel wird der Trusted Advisor Advisor-Check für Amazon S3 Bucket Permissions in Ihrem AWS Konto aktualisiert.

```
aws support refresh-trusted-advisor-check \
```

```
    --check-id "Pfx0RwqBli"
```

Ausgabe:

```
{
    "status": {
        "checkId": "Pfx0RwqBli",
        "status": "enqueued",
        "millisUntilNextRefreshable": 3599992
    }
}
```

Weitere Informationen finden Sie unter AWS Trusted Advisor im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie unter RefreshTrustedAdvisorCheck AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Fordert eine Aktualisierung für die angegebene Prüfung von Trusted Advisor an.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter RefreshTrustedAdvisorCheck AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Fordert eine Aktualisierung für die angegebene Prüfung von Trusted Advisor an.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- Einzelheiten zur API finden Sie unter RefreshTrustedAdvisorCheck AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ResolveCase`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ResolveCase` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Kennenlernen der Grundlagen
- Erste Schritte mit dem technischen Support

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Resolve a support case by caseId.
/// </summary>
/// <param name="caseId">Id for the support case.</param>
/// <returns>The final status of the case after resolving.</returns>
public async Task<string> ResolveCase(string caseId)
{
    var response = await _amazonSupport.ResolveCaseAsync(
        new ResolveCaseRequest()
        {
            CaseId = caseId
        });
    return response.FinalCaseStatus;
}
```

- Einzelheiten zur API finden Sie ResolveCasein der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Um einen Support-Fall zu lösen

Das folgende `resolve-case` Beispiel löst einen Supportfall in Ihrem AWS Konto.

```
aws support resolve-case \
    --case-id "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Ausgabe:

```
{
    "finalCaseStatus": "resolved",
    "initialCaseStatus": "work-in-progress"
}
```

Weitere Informationen finden Sie unter Fallmanagement im Benutzerhandbuch für AWS Support.

- Einzelheiten zur API finden Sie ResolveCasein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void resolveSupportCase(SupportClient supportClient, String
 caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();
```

```
            ResolveCaseResponse response =
 supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
 response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Einzelheiten zur API finden Sie ResolveCasein der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ResolveCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

const main = async () => {
  try {
    const response = await client.send(
      new ResolveCaseCommand({
        caseId: "CASE_ID",
      }),
    );

    console.log(response.finalCaseStatus);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- Einzelheiten zur API finden Sie ResolveCasein der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- Einzelheiten zur API finden Sie ResolveCasein der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Gibt den Anfangszustand des angegebenen Falls und den aktuellen Zustand nach Abschluss des Aufrufs zur Lösung zurück.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Einzelheiten zur API finden Sie unter ResolveCase AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt den Anfangszustand des angegebenen Falls und den aktuellen Zustand nach Abschluss des Aufrufs zur Lösung zurück.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- Einzelheiten zur API finden Sie unter ResolveCase AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.
```

```
        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status
```

- Einzelheiten zur API finden Sie ResolveCasein AWS SDK for Python (Boto3) API Reference.

Szenarien für Support verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien Support mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb Support oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit dem technischen Support

Erste Schritte mit dem technischen Support

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie die verfügbaren Dienste und Schweregrade
- Einen Support-Fall erstellen
- Fügen Sie einem Fall Mitteilungen hinzu

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Support CLI Tutorial Script
# This script demonstrates how to use AWS Support API through AWS CLI

set -euo pipefail

# Security: Validate script location and permissions
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
readonly SCRIPT_DIR

# Security: Use secure temporary directory
readonly TEMP_DIR="$(mktemp -d)" || { echo "Failed to create temp directory";
 exit 1; }
trap "rm -rf '$TEMP_DIR'" EXIT

# Set up logging with secure permissions
```

```
LOG_FILE="${TEMP_DIR}/aws-support-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH" >&2
    exit 1
fi

# Security: Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not properly configured" >&2
    exit 1
fi

{
    echo "Starting AWS Support Tutorial at $(date)"
    echo "Script: $0"
    echo "User: $(whoami)"
    echo "---"
} >> "$LOG_FILE"

# Function to log commands and their outputs securely
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command" >> "$LOG_FILE"
    # Don't echo the actual command to prevent credential leakage
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local is_fatal="${4:-true}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command returned status: $cmd_status" >> "$LOG_FILE"

        # Check for subscription error
```

```
        if [[ "$cmd_output" =~ "SubscriptionRequiredException" ]]; then
            {
                echo ""
                echo "====================================================="
                echo "IMPORTANT: This account does not have the required AWS
 Support plan."
                echo "You need a Business, Enterprise On-Ramp, or Enterprise
 Support plan"
                echo "to use the AWS Support API."
                echo ""
                echo "This script will now demonstrate the commands that would be
 run"
                echo "if you had the appropriate support plan, but will not
 execute them."
                echo "====================================================="
            } | tee -a "$LOG_FILE"

            DEMO_MODE=true
            return 0
        fi

        if [[ "$is_fatal" == "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    echo "No persistent resources were created that need cleanup." | tee -a
 "$LOG_FILE"
}

# Function to run a command in demo mode
demo_cmd() {
    local cmd="$1"
    local description="$2"

    {
        echo ""
        echo "DEMO: $description"
        echo "Command that would be executed:"
```

```
        echo "  [Command hidden for security]"
        echo ""
    } | tee -a "$LOG_FILE"
}

# Function to safely extract JSON values
extract_json_value() {
    local json_output="$1"
    local key="$2"

    echo "$json_output" | grep -o "\"$key\": \"[^\"]*\"" | head -1 | cut -d'"' -
f4 || echo ""
}

# Array to track created resources
declare -a CREATED_RESOURCES

# Initialize demo mode flag
DEMO_MODE=false

# Security: Validate input parameters
if [[ $# -gt 0 ]]; then
    echo "ERROR: This script does not accept parameters" >&2
    exit 1
fi

{
    echo "====================================================="
    echo "AWS Support CLI Tutorial"
    echo "====================================================="
    echo "This script demonstrates how to use AWS Support API"
    echo "Note: You must have a Business, Enterprise On-Ramp,"
    echo "or Enterprise Support plan to use the AWS Support API."
    echo "====================================================="
    echo ""
} | tee -a "$LOG_FILE"

# Step 1: Check available services
echo "Step 1: Checking available AWS Support services..." | tee -a "$LOG_FILE"
SERVICES_OUTPUT=$(log_cmd "aws support describe-services --language en" 2>&1) ||
 SERVICES_OUTPUT=""
check_error "$SERVICES_OUTPUT" $? "Failed to retrieve AWS Support services"

# If we're in demo mode, set default values
```

```
if [[ "$DEMO_MODE" == "true" ]]; then
    SERVICE_CODE="general-info"
    echo "Using demo service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
else
    # Extract a service code for demonstration using safer method
    SERVICE_CODE=$(extract_json_value "$SERVICES_OUTPUT" "code") ||
 SERVICE_CODE=""
    if [[ -z "$SERVICE_CODE" ]]; then
        SERVICE_CODE="general-info"
        echo "Using default service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 2: Check available severity levels
echo "Step 2: Checking available severity levels..." | tee -a "$LOG_FILE"
if [[ "$DEMO_MODE" == "true" ]]; then
    demo_cmd "aws support describe-severity-levels --language en" "Check
 available severity levels"
    SEVERITY_CODE="low"
    echo "Using demo severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
else
    SEVERITY_OUTPUT=$(log_cmd "aws support describe-severity-levels --language
 en" 2>&1) || SEVERITY_OUTPUT=""
    check_error "$SEVERITY_OUTPUT" $? "Failed to retrieve severity levels"

    SEVERITY_CODE=$(extract_json_value "$SEVERITY_OUTPUT" "code") ||
 SEVERITY_CODE=""
    if [[ -z "$SEVERITY_CODE" ]]; then
        SEVERITY_CODE="low"
        echo "Using default severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 3: Create a test support case
{
    echo ""
    echo "==================================================="
    echo "SUPPORT CASE CREATION"
    echo "==================================================="
} | tee -a "$LOG_FILE"
```

```
if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "DEMO MODE: The following steps would create and manage a support
 case"
        echo "if you had a Business, Enterprise On-Ramp, or Enterprise Support
 plan."
        echo ""
    } | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"

    demo_cmd "aws support create-case" "Create a support case"

    CASE_ID="case-12345678910-2013-c4c1d2bf33c5cf47"
    echo "Demo case ID: $CASE_ID" | tee -a "$LOG_FILE"

    demo_cmd "aws support describe-cases" "List support cases"
    demo_cmd "aws support add-communication-to-case" "Add communication to case"
    demo_cmd "aws support describe-communications" "View case communications"
    demo_cmd "aws support resolve-case" "Resolve the support case"

else
    echo "Creating a test support case..." | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"
    CC_EMAIL_PARAM="--cc-email-addresses $USER_EMAIL"

    # Create the case
    CASE_OUTPUT=$(log_cmd "aws support create-case --subject 'AWS CLI Tutorial
 Test Case' --service-code '$SERVICE_CODE' --category-code 'using-aws' --
communication-body 'This is a test case created as part of an AWS CLI tutorial.'
 --severity-code '$SEVERITY_CODE' --language 'en' $CC_EMAIL_PARAM" 2>&1) ||
 CASE_OUTPUT=""

    check_error "$CASE_OUTPUT" $? "Failed to create support case"

    # Extract the case ID safely
    CASE_ID=$(extract_json_value "$CASE_OUTPUT" "caseId") || CASE_ID=""

    if [[ -n "$CASE_ID" ]]; then
        echo "Successfully created support case with ID: $CASE_ID" | tee -a
 "$LOG_FILE"
        CREATED_RESOURCES+=("Support Case: $CASE_ID")
```

```
        # Step 4: List the case we just created
        echo "" | tee -a "$LOG_FILE"
        echo "Step 4: Listing the support case we just created..." | tee -a
 "$LOG_FILE"
        CASES_OUTPUT=$(log_cmd "aws support describe-cases --case-id-
list '$CASE_ID' --include-resolved-cases false --language 'en'" 2>&1) ||
 CASES_OUTPUT=""

        check_error "$CASES_OUTPUT" $? "Failed to retrieve case details"

        # Step 5: Add a communication to the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 5: Adding a communication to the support case..." | tee -a
 "$LOG_FILE"
        COMM_OUTPUT=$(log_cmd "aws support add-communication-to-case --case-id
 '$CASE_ID' --communication-body 'This is an additional communication for the
 test case.' $CC_EMAIL_PARAM" 2>&1) || COMM_OUTPUT=""

        check_error "$COMM_OUTPUT" $? "Failed to add communication to case"

        # Step 6: View communications for the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 6: Viewing communications for the support case..." | tee -a
 "$LOG_FILE"
        COMMS_OUTPUT=$(log_cmd "aws support describe-communications --case-id
 '$CASE_ID' --language 'en'" 2>&1) || COMMS_OUTPUT=""

        check_error "$COMMS_OUTPUT" $? "Failed to retrieve case communications"

        # Step 7: Resolve the case
        {
            echo ""
            echo "====================================================="
            echo "CASE RESOLUTION"
            echo "====================================================="
            echo "Resolving the support case..."
        } | tee -a "$LOG_FILE"

        RESOLVE_OUTPUT=$(log_cmd "aws support resolve-case --case-id '$CASE_ID'"
 2>&1) || RESOLVE_OUTPUT=""

        check_error "$RESOLVE_OUTPUT" $? "Failed to resolve case"
        echo "Successfully resolved support case: $CASE_ID" | tee -a "$LOG_FILE"
```

```
    else
        echo "Could not extract case ID from the response." | tee -a "$LOG_FILE"
    fi
fi

# Display summary of created resources
{
    echo ""
    echo "==================================================="
    echo "TUTORIAL SUMMARY"
    echo "==================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "This was a demonstration in DEMO MODE."
        echo "No actual AWS Support cases were created."
        echo "To use the AWS Support API, you need a Business, Enterprise On-
Ramp,"
        echo "or Enterprise Support plan."
    } | tee -a "$LOG_FILE"
else
    {
        echo "Resources created during this tutorial:"
        if [[ ${#CREATED_RESOURCES[@]} -eq 0 ]]; then
            echo "No resources were created."
        else
            for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}";
 do
                echo "- $resource"
            done
        fi
    } | tee -a "$LOG_FILE"
fi

{
    echo ""
    echo "Tutorial completed successfully!"
    echo "Log file: $LOG_FILE"
    echo "==================================================="
} | tee -a "$LOG_FILE"

# Display log file path to user
```

```
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddCommunicationToCase
  - CreateCase
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Codebeispiele für Systems Manager mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS Systems Manager mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Systems-Manager-Benutzerhandbuch – Weitere Informationen zu Systems Manager.
- API-Referenz für Systems Manager – Details zu allen verfügbaren Systems-Manager-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Systems Manager AWS SDKs
  - Hallo Systems Manager
  - Erlernen Sie die Grundlagen von Systems Manager mit einem AWS SDK
  - Aktionen für Systems Manager mit AWS SDKs
    - Verwendung von AddTagsToResource mit einer CLI
    - Verwendung von CancelCommand mit einer CLI
    - Verwendung von CreateActivation mit einer CLI
    - Verwendung von CreateAssociation mit einer CLI
    - Verwendung von CreateAssociationBatch mit einer CLI
    - Verwenden Sie mit einem CreateDocument AWS SDK oder CLI
    - Verwenden Sie CreateMaintenanceWindowmit einem AWS SDK oder CLI
    - Verwenden Sie CreateOpsItemmit einem AWS SDK oder CLI
    - Verwendung von CreatePatchBaseline mit einer CLI
    - Verwendung von DeleteActivation mit einer CLI
    - Verwendung von DeleteAssociation mit einer CLI
    - Verwenden Sie mit einem DeleteDocument AWS SDK oder CLI
    - Verwenden Sie DeleteMaintenanceWindowmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteOpsItemmit einem AWS SDK
    - Verwendung von DeleteParameter mit einer CLI
    - Verwendung von DeletePatchBaseline mit einer CLI
    - Verwendung von DeregisterManagedInstance mit einer CLI
    - Verwendung von DeregisterPatchBaselineForPatchGroup mit einer CLI
    - Verwendung von DeregisterTargetFromMaintenanceWindow mit einer CLI
    - Verwendung von DeregisterTaskFromMaintenanceWindow mit einer CLI
    - Verwendung von DescribeActivations mit einer CLI
    - Verwendung von DescribeAssociation mit einer CLI
    - Verwendung von DescribeAssociationExecutionTargets mit einer CLI
    - Verwendung von DescribeAssociationExecutions mit einer CLI
    - Verwendung von DescribeAutomationExecutions mit einer CLI

    - Verwendung von DescribeAutomationStepExecutions mit einer CLI

- Verwendung von DescribeAvailablePatches mit einer CLI
- Verwenden Sie mit einem DescribeDocument AWS SDK oder CLI
- Verwendung von DescribeDocumentPermission mit einer CLI
- Verwendung von DescribeEffectiveInstanceAssociations mit einer CLI
- Verwendung von DescribeEffectivePatchesForPatchBaseline mit einer CLI
- Verwendung von DescribeInstanceAssociationsStatus mit einer CLI
- Verwendung von DescribeInstanceInformation mit einer CLI
- Verwendung von DescribeInstancePatchStates mit einer CLI
- Verwendung von DescribeInstancePatchStatesForPatchGroup mit einer CLI
- Verwendung von DescribeInstancePatches mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutionTaskInvocations mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutionTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutions mit einer CLI
- Verwendung von DescribeMaintenanceWindowTargets mit einer CLI
- Verwendung von DescribeMaintenanceWindowTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindows mit einer CLI
- Verwenden Sie mit einem DescribeOpsItems AWS SDK oder CLI
- Verwenden Sie DescribeParametersmit einem AWS SDK oder CLI
- Verwendung von DescribePatchBaselines mit einer CLI
- Verwendung von DescribePatchGroupState mit einer CLI
- Verwendung von DescribePatchGroups mit einer CLI
- Verwendung von GetAutomationExecution mit einer CLI
- Verwendung von GetCommandInvocation mit einer CLI
- Verwendung von GetConnectionStatus mit einer CLI
- Verwendung von GetDefaultPatchBaseline mit einer CLI
- Verwendung von GetDeployablePatchSnapshotForInstance mit einer CLI
- Verwendung von GetDocument mit einer CLI
- Verwendung von GetInventory mit einer CLI
- Verwendung von GetInventorySchema mit einer CLI

- Verwendung von GetMaintenanceWindow mit einer CLI

- Verwendung von GetMaintenanceWindowExecution mit einer CLI
- Verwendung von GetMaintenanceWindowExecutionTask mit einer CLI
- Verwenden Sie mit einem GetParameter AWS SDK oder CLI
- Verwendung von GetParameterHistory mit einer CLI
- Verwendung von GetParameters mit einer CLI
- Verwendung von GetPatchBaseline mit einer CLI
- Verwendung von GetPatchBaselineForPatchGroup mit einer CLI
- Verwendung von ListAssociationVersions mit einer CLI
- Verwendung von ListAssociations mit einer CLI
- Verwenden Sie mit einem ListCommandInvocations AWS SDK oder CLI
- Verwendung von ListCommands mit einer CLI
- Verwendung von ListComplianceItems mit einer CLI
- Verwendung von ListComplianceSummaries mit einer CLI
- Verwendung von ListDocumentVersions mit einer CLI
- Verwendung von ListDocuments mit einer CLI
- Verwendung von ListInventoryEntries mit einer CLI
- Verwendung von ListResourceComplianceSummaries mit einer CLI
- Verwendung von ListTagsForResource mit einer CLI
- Verwendung von ModifyDocumentPermission mit einer CLI
- Verwendung von PutComplianceItems mit einer CLI
- Verwendung von PutInventory mit einer CLI
- Verwenden Sie mit einem PutParameter AWS SDK oder CLI
- Verwendung von RegisterDefaultPatchBaseline mit einer CLI
- Verwendung von RegisterPatchBaselineForPatchGroup mit einer CLI
- Verwendung von RegisterTargetWithMaintenanceWindow mit einer CLI
- Verwendung von RegisterTaskWithMaintenanceWindow mit einer CLI
- Verwendung von RemoveTagsFromResource mit einer CLI
- Verwenden Sie mit einem SendCommand AWS SDK oder CLI
- Verwendung von StartAutomationExecution mit einer CLI

- Verwendung von StartSession mit einer CLI

    - Verwendung von StopAutomationExecution mit einer CLI
    - Verwendung von UpdateAssociation mit einer CLI
    - Verwendung von UpdateAssociationStatus mit einer CLI
    - Verwendung von UpdateDocument mit einer CLI
    - Verwendung von UpdateDocumentDefaultVersion mit einer CLI
    - Verwenden Sie mit einem UpdateMaintenanceWindow AWS SDK oder CLI
    - Verwendung von UpdateManagedInstanceRole mit einer CLI
    - Verwenden Sie mit einem UpdateOpsItem AWS SDK oder CLI
    - Verwendung von UpdatePatchBaseline mit einer CLI
- Szenarien für die Verwendung von Systems Manager AWS SDKs
  - Einen Container-Service für VM-Instanzen erstellen
  - Erste Schritte mit virtuellen Maschinen
  - Systemmanagement einrichten

Grundlegende Beispiele für die Verwendung von Systems Manager AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS Systems Manager mit AWS SDKs verwendet werden.

Beispiele

- Hallo Systems Manager
- Erlernen Sie die Grundlagen von Systems Manager mit einem AWS SDK
- Aktionen für Systems Manager mit AWS SDKs
  - Verwendung von AddTagsToResource mit einer CLI
  - Verwendung von CancelCommand mit einer CLI
  - Verwendung von CreateActivation mit einer CLI
  - Verwendung von CreateAssociation mit einer CLI
  - Verwendung von CreateAssociationBatch mit einer CLI
  - Verwenden Sie mit einem CreateDocument AWS SDK oder CLI
  - Verwenden Sie CreateMaintenanceWindowmit einem AWS SDK oder CLI

- Verwenden Sie CreateOpsItemmit einem AWS SDK oder CLI
- Verwendung von CreatePatchBaseline mit einer CLI
- Verwendung von DeleteActivation mit einer CLI
- Verwendung von DeleteAssociation mit einer CLI
- Verwenden Sie mit einem DeleteDocument AWS SDK oder CLI
- Verwenden Sie DeleteMaintenanceWindowmit einem AWS SDK oder CLI
- Verwenden Sie DeleteOpsItemmit einem AWS SDK
- Verwendung von DeleteParameter mit einer CLI
- Verwendung von DeletePatchBaseline mit einer CLI
- Verwendung von DeregisterManagedInstance mit einer CLI
- Verwendung von DeregisterPatchBaselineForPatchGroup mit einer CLI
- Verwendung von DeregisterTargetFromMaintenanceWindow mit einer CLI
- Verwendung von DeregisterTaskFromMaintenanceWindow mit einer CLI
- Verwendung von DescribeActivations mit einer CLI
- Verwendung von DescribeAssociation mit einer CLI
- Verwendung von DescribeAssociationExecutionTargets mit einer CLI
- Verwendung von DescribeAssociationExecutions mit einer CLI
- Verwendung von DescribeAutomationExecutions mit einer CLI
- Verwendung von DescribeAutomationStepExecutions mit einer CLI
- Verwendung von DescribeAvailablePatches mit einer CLI
- Verwenden Sie mit einem DescribeDocument AWS SDK oder CLI
- Verwendung von DescribeDocumentPermission mit einer CLI
- Verwendung von DescribeEffectiveInstanceAssociations mit einer CLI
- Verwendung von DescribeEffectivePatchesForPatchBaseline mit einer CLI
- Verwendung von DescribeInstanceAssociationsStatus mit einer CLI
- Verwendung von DescribeInstanceInformation mit einer CLI
- Verwendung von DescribeInstancePatchStates mit einer CLI
- Verwendung von DescribeInstancePatchStatesForPatchGroup mit einer CLI
- Verwendung von DescribeInstancePatches mit einer CLI

- Verwendung von DescribeMaintenanceWindowExecutionTaskInvocations mit einer CLI

- Verwendung von DescribeMaintenanceWindowExecutionTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutions mit einer CLI
- Verwendung von DescribeMaintenanceWindowTargets mit einer CLI
- Verwendung von DescribeMaintenanceWindowTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindows mit einer CLI
- Verwenden Sie mit einem DescribeOpsItems AWS SDK oder CLI
- Verwenden Sie DescribeParametersmit einem AWS SDK oder CLI
- Verwendung von DescribePatchBaselines mit einer CLI
- Verwendung von DescribePatchGroupState mit einer CLI
- Verwendung von DescribePatchGroups mit einer CLI
- Verwendung von GetAutomationExecution mit einer CLI
- Verwendung von GetCommandInvocation mit einer CLI
- Verwendung von GetConnectionStatus mit einer CLI
- Verwendung von GetDefaultPatchBaseline mit einer CLI
- Verwendung von GetDeployablePatchSnapshotForInstance mit einer CLI
- Verwendung von GetDocument mit einer CLI
- Verwendung von GetInventory mit einer CLI
- Verwendung von GetInventorySchema mit einer CLI
- Verwendung von GetMaintenanceWindow mit einer CLI
- Verwendung von GetMaintenanceWindowExecution mit einer CLI
- Verwendung von GetMaintenanceWindowExecutionTask mit einer CLI
- Verwenden Sie mit einem GetParameter AWS SDK oder CLI
- Verwendung von GetParameterHistory mit einer CLI
- Verwendung von GetParameters mit einer CLI
- Verwendung von GetPatchBaseline mit einer CLI
- Verwendung von GetPatchBaselineForPatchGroup mit einer CLI
- Verwendung von ListAssociationVersions mit einer CLI
- Verwendung von ListAssociations mit einer CLI
- Verwenden Sie mit einem ListCommandInvocations AWS SDK oder CLI

- Verwendung von ListCommands mit einer CLI

- Verwendung von ListComplianceItems mit einer CLI
- Verwendung von ListComplianceSummaries mit einer CLI
- Verwendung von ListDocumentVersions mit einer CLI
- Verwendung von ListDocuments mit einer CLI
- Verwendung von ListInventoryEntries mit einer CLI
- Verwendung von ListResourceComplianceSummaries mit einer CLI
- Verwendung von ListTagsForResource mit einer CLI
- Verwendung von ModifyDocumentPermission mit einer CLI
- Verwendung von PutComplianceItems mit einer CLI
- Verwendung von PutInventory mit einer CLI
- Verwenden Sie mit einem PutParameter AWS SDK oder CLI
- Verwendung von RegisterDefaultPatchBaseline mit einer CLI
- Verwendung von RegisterPatchBaselineForPatchGroup mit einer CLI
- Verwendung von RegisterTargetWithMaintenanceWindow mit einer CLI
- Verwendung von RegisterTaskWithMaintenanceWindow mit einer CLI
- Verwendung von RemoveTagsFromResource mit einer CLI
- Verwenden Sie mit einem SendCommand AWS SDK oder CLI
- Verwendung von StartAutomationExecution mit einer CLI
- Verwendung von StartSession mit einer CLI
- Verwendung von StopAutomationExecution mit einer CLI
- Verwendung von UpdateAssociation mit einer CLI
- Verwendung von UpdateAssociationStatus mit einer CLI
- Verwendung von UpdateDocument mit einer CLI
- Verwendung von UpdateDocumentDefaultVersion mit einer CLI
- Verwenden Sie mit einem UpdateMaintenanceWindow AWS SDK oder CLI
- Verwendung von UpdateManagedInstanceRole mit einer CLI
- Verwenden Sie mit einem UpdateOpsItem AWS SDK oder CLI
- Verwendung von UpdatePatchBaseline mit einer CLI

Hallo Systems Manager

Die folgenden Codebeispiele veranschaulichen, wie Sie mit der Verwendung von Systems Manager beginnen.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
```

```
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- Einzelheiten zur API finden Sie ListDocumentsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- Einzelheiten zur API finden Sie ListDocumentsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object
 wraps
                               the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")


if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
```

```
        print(err.response["Error"]["Message"])
```

- Einzelheiten zur API finden Sie ListDocumentsin AWS SDK for Python (Boto3) API Reference.

Erlernen Sie die Grundlagen von Systems Manager mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Wartungsfenster.
- Ändern Sie den Zeitplan des Wartungsfensters.
- Erstellen Sie ein Dokument.
- Senden Sie einen Befehl an eine bestimmte EC2-Instance.
- Erstellen Sie ein OpsItem.
- Aktualisieren und lösen Sie das OpsItem.
- Löschen Sie das Wartungsfenster OpsItem, und das Dokument.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    private static final String ROLES_STACK = "SsmStack3`1";
```

```
    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
                category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
                severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;

        System.out.println("Use AWS CloudFormation to create the EC2 instance
that is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS
applications and resources and a secure end-to-end management solution.
                The program's primary functionalities include creating a
maintenance window, creating a document, sending a command to a document,
                listing documents, listing commands, creating an OpsItem,
modifying an OpsItem, and deleting AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
```

```
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions
that Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: "
+ commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println("5. Lets get the time when the specific command was
sent to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager
(SSM) service.
             It is a type of operational data item that allows you to manage and
track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as
they arise.
             For example, you could create an OpsItem whenever your application
detects a critical error
             or an anomaly in your infrastructure.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category,
severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
 "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
```

```
            }
        }
    }
}
```

Eine Wrapper-Klasse für Systems-Manager-SDK-Methoden.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
```

```
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
```

```
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
           throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
       }
   }

   /**
    * Resolves an AWS SSM OpsItem asynchronously.
    *
    * @param opsID The ID of the OpsItem to resolve.
    * <p>
```

```
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
```

```
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
                       }
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
```

```
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description)
{
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());

        CompletableFuture<Void> future =
getOpsItem(opsItemId).thenCompose(opsItem -> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response ->
{
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
```

```
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
GetOpsItemRequest.builder().opsItemId(opsItemId).build();
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
```

```
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
     * Displays the date and time when the specific command was invoked.
     *
     * @param commandId The ID of the command to describe.
     * <p>
     * This method initiates an asynchronous request to list command invocations
and prints the date and time of each command invocation.
     * If an exception occurs, it handles the error appropriately.
     */
    public void displayCommands(String commandId) {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
                .commandId(commandId)
                .build();

        CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
        future.thenAccept(response -> {
            List<CommandInvocation> commandList = response.commandInvocations();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
            for (CommandInvocation invocation : commandList) {
                System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
            }
```

```
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Sends a SSM command to a managed node asynchronously.
     *
     * @param documentName The name of the document to use.
     * @param instanceId The ID of the instance to send the command to.
     * @return The command ID.
     * <p>
     * This method initiates asynchronous requests to send a SSM command to a
managed node.
     * It waits until the document is active, sends the command, and checks the
command execution status.
     */
    public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
        // Before we use Document to send a command - make sure it is active.
        CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
            boolean isDocumentActive = false;
            DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                    .name(documentName)
                    .build();

            while (!isDocumentActive) {
                CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
                String documentStatus =
response.join().document().statusAsString();
                if (documentStatus.equals("Active")) {
                    System.out.println("The SSM document is active and ready to
use.");
                    isDocumentActive = true;
                } else {
```

```
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
```

```
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
```

```
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

```
    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
```

```
    * @param winName The name of the maintenance window.
    * @return The ID of the created or existing maintenance window.
    * <p>
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();

       CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
       final String[] windowId = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String maintenanceWindowId = response.windowId();
               System.out.println("The maintenance window id is " +
maintenanceWindowId);
               windowId[0] = maintenanceWindowId;
           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof DocumentAlreadyExistsException) {
                   throw new CompletionException(cause);
               } else if (cause instanceof SsmException) {
                   throw new CompletionException(cause);
               } else {
                   throw new RuntimeException(cause);
               }
           }
       }).join();

       if (windowId[0] == null) {
           MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
```

```
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
 DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
 = getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
 describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow

- ListCommandInvocations
- SendCommand
- UpdateOpsItem

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { fileURLToPath } from "node:url";
import {
  CreateDocumentCommand,
  CreateMaintenanceWindowCommand,
  CreateOpsItemCommand,
  DeleteDocumentCommand,
  DeleteMaintenanceWindowCommand,
  DeleteOpsItemCommand,
  DescribeOpsItemsCommand,
  DocumentAlreadyExists,
  OpsItemStatus,
  waitUntilCommandExecuted,
  CancelCommandCommand,
  paginateListCommandInvocations,
  SendCommandCommand,
  UpdateMaintenanceWindowCommand,
  UpdateOpsItemCommand,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";
```

```
/**
 * @typedef {{
 *   ssmClient: import('@aws-sdk/client-ssm').SSMClient,
 *   documentName?: string
 *   maintenanceWindow?: string
 *   winId?: int
 *   ec2InstanceId?: string
 *   requestedDateTime?: Date
 *   opsItemId?: string
 *   askToDeleteResources?: boolean
 * }} State
 */

const defaultMaintenanceWindow = "ssm-maintenance-window";
const defaultDocumentName = "ssmdocument";
// The timeout duration is highly dependent on the specific setup and environment
 necessary. This example handles only the most common error cases, and uses a
 much shorter duration than most productions systems would use.
const COMMAND_TIMEOUT_DURATION_SECONDS = 30; // 30 seconds

const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the AWS
 SDK for JavaScript V3.
    Systems Manager is the operations hub for your AWS applications and resources
 and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window,
 creating a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an
 OpsItem, and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...`,
  { header: true },
);

const createMaintenanceWindow = new ScenarioOutput(
  "createMaintenanceWindow",
  "Step 1: Create a Systems Manager maintenance window.",
);
```

```
const getMaintenanceWindow = new ScenarioInput(
  "maintenanceWindow",
  "Please enter the maintenance window name:",
  { type: "input", default: defaultMaintenanceWindow },
);

export const sdkCreateMaintenanceWindow = new ScenarioAction(
  "sdkCreateMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          Schedule: "cron(0 10 ? * MON-FRI *)", //The schedule of the maintenance
 window in the form of a cron or rate expression.
          Duration: 2, //The duration of the maintenance window in hours.
          Cutoff: 1, //The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
          AllowUnassociatedTargets: true, //Allow the maintenance window to run
 on managed nodes, even if you haven't registered those nodes as targets.
        }),
      );
      state.winId = response.WindowId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const modifyMaintenanceWindow = new ScenarioOutput(
  "modifyMaintenanceWindow",
  "Modify the maintenance window by changing the schedule.",
);

const sdkModifyMaintenanceWindow = new ScenarioAction(
  "sdkModifyMaintenanceWindow",
  async (/** @type {State} */ state) => {
```

```
    try {
      await state.ssmClient.send(
        new UpdateMaintenanceWindowCommand({
          WindowId: state.winId,
          Schedule: "cron(0 0 ? * MON *)",
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while modifying the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const createSystemsManagerActions = new ScenarioOutput(
  "createSystemsManagerActions",
  "Create a document that defines the actions that Systems Manager performs on
 your EC2 instance.",
);

const getDocumentName = new ScenarioInput(
  "documentName",
  "Please enter the document: ",
  { type: "input", default: defaultDocumentName },
);

const sdkCreateSSMDoc = new ScenarioAction(
  "sdkCreateSSMDoc",
  async (/** @type {State} */ state) => {
    const contentData = `{
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
```

```
                        }
                    }
                ]
            }`;
    try {
      await state.ssmClient.send(
        new CreateDocumentCommand({
          Content: contentData,
          Name: state.documentName,
          DocumentType: "Command",
        }),
      );
    } catch (caught) {
      console.log(`Exception type: (${typeof caught})`);
      if (caught instanceof DocumentAlreadyExists) {
        console.log("Document already exists. Continuing...\n");
      } else {
        console.error(caught.message);
        console.log(
          `An error occurred while creating the document. Please fix the error
 and try again. Error message: ${caught.message}`,
        );
        throw caught;
      }
    }
  },
);

const ec2HelloWorld = new ScenarioOutput(
  "ec2HelloWorld",
  `Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'. In order to run this command, you must provide the instance ID
 of a Linux EC2 instance. If you do not already have a running Linux EC2 instance
 in your account, you can create one using the AWS console. For information
 about creating an EC2 instance, see https://docs.aws.amazon.com/AWSEC2/latest/
UserGuide/ec2-launch-instance-wizard.html.`,
);

const enterIdOrSkipEC2HelloWorld = new ScenarioInput(
  "enterIdOrSkipEC2HelloWorld",
  "Enter your EC2 InstanceId or press enter to skip this step: ",
  { type: "input", default: "" },
);
```

```
const sdkEC2HelloWorld = new ScenarioAction(
  "sdkEC2HelloWorld",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new SendCommandCommand({
          DocumentName: state.documentName,
          InstanceIds: [state.ec2InstanceId],
          TimeoutSeconds: COMMAND_TIMEOUT_DURATION_SECONDS,
        }),
      );
      state.CommandId = response.Command.CommandId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while sending the command. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const sdkGetCommandTime = new ScenarioAction(
  "sdkGetCommandTime",
  async (/** @type {State} */ state) => {
    const listInvocationsPaginated = [];
    console.log(
      "Let's get the time when the specific command was sent to the specific
 managed node.",
    );

    console.log(
      `First, we'll wait for the command to finish executing. This may take up to
 ${COMMAND_TIMEOUT_DURATION_SECONDS} seconds.`,
    );
    const commandExecutedResult = waitUntilCommandExecuted(
      { client: state.ssmClient },
      {
        CommandId: state.CommandId,
```

```
        InstanceId: state.ec2InstanceId,
      },
    );
    // This is necessary because the TimeoutSeconds of SendCommandCommand is only
for the delivery, not execution.
    try {
      await new Promise((_, reject) =>
        setTimeout(
          reject,
          COMMAND_TIMEOUT_DURATION_SECONDS * 1000,
          new Error("Command Timed Out"),
        ),
      );
    } catch (caught) {
      if (caught.message === "Command Timed Out") {
        commandExecutedResult.state = "TIMED_OUT";
      } else {
        throw caught;
      }
    }

    if (commandExecutedResult.state !== "SUCCESS") {
      console.log(
        `The command with id: ${state.CommandId} did not execute in the allotted
time. Canceling command.`,
      );
      state.ssmClient.send(
        new CancelCommandCommand({
          CommandId: state.CommandId,
        }),
      );
      state.enterIdOrSkipEC2HelloWorld === "";
      return;
    }

    for await (const page of paginateListCommandInvocations(
      { client: state.ssmClient },
      { CommandId: state.CommandId },
    )) {
      listInvocationsPaginated.push(...page.CommandInvocations);
    }
    /**
     * @type {import('@aws-sdk/client-ssm').CommandInvocation}
     */
```

```
    const commandInvocation = listInvocationsPaginated.shift(); // Because the
 call was made with CommandId, there's only one result, so shift it off.
    state.requestedDateTime = commandInvocation.RequestedDateTime;

    console.log(
      `The command invocation happened at: ${state.requestedDateTime}.`,
    );
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const createSSMOpsItem = new ScenarioOutput(
  "createSSMOpsItem",
  `Now we will create a Systems Manager OpsItem. An OpsItem is a feature provided
 by the Systems Manager service. It is a type of operational data item that
 allows you to manage and track various operational issues, events, or tasks
 within your AWS environment.
You can create OpsItems to track and manage operational issues as they arise.
 For example, you could create an OpsItem whenever your application detects a
 critical error or an anomaly in your infrastructure.`,
);

const sdkCreateSSMOpsItem = new ScenarioAction(
  "sdkCreateSSMOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateOpsItemCommand({
          Description: "Created by the System Manager Javascript API",
          Title: "Disk Space Alert",
          Source: "EC2",
          Category: "Performance",
          Severity: "2",
        }),
      );
      state.opsItemId = response.OpsItemId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
```

```
      );
      throw caught;
    }
  },
);

const updateOpsItem = new ScenarioOutput(
  "updateOpsItem",
  (/** @type {State} */ state) =>
    `Now we will update the OpsItem: ${state.opsItemId}`,
);

const sdkUpdateOpsItem = new ScenarioAction(
  "sdkUpdateOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Description: `An update to ${state.opsItemId}`,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const getOpsItemStatus = new ScenarioOutput(
  "getOpsItemStatus",
  (/** @type {State} */ state) =>
    `Now we will get the status of the OpsItem: ${state.opsItemId}`,
);

const sdkOpsItemStatus = new ScenarioAction(
  "sdkGetOpsItemStatus",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
```

```
        new DescribeOpsItemsCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      state.opsItemStatus = response.OpsItemStatus;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while describing the ops item. Please fix the error
 and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const resolveOpsItem = new ScenarioOutput(
  "resolveOpsItem",
  (/** @type {State} */ state) =>
    `Now we will resolve the OpsItem: ${state.opsItemId}`,
);

const sdkResolveOpsItem = new ScenarioAction(
  "sdkResolveOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Status: OpsItemStatus.RESOLVED,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
```

```
  "askToDeleteResources",
  "Would you like to delete the Systems Manager resources created during this
 example run?",
  { type: "confirm" },
);

const confirmDeleteChoice = new ScenarioOutput(
  "confirmDeleteChoice",
  (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You chose to delete the resources.";
    }
    return "The Systems Manager resources will not be deleted. Please delete them
 manually to avoid charges.";
  },
);

export const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new DeleteOpsItemCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      console.log(`The ops item: ${state.opsItemId} was successfully deleted.`);
    } catch (caught) {
      console.log(
        `There was a problem deleting the ops item: ${state.opsItemId}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          WindowId: state.winId,
        }),
      );
      console.log(
        `The maintenance window: ${state.maintenanceWindow} was successfully
 deleted.`,
```

```
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the maintenance window: ${state.opsItemId}.
 Please delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteDocumentCommand({
          Name: state.documentName,
        }),
      );
      console.log(
        `The document: ${state.documentName} was successfully deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the document: ${state.documentName}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the Systems Manager Basics scenario for the AWS Javascript SDK
 v3. Thank you!",
);

const myScenario = new Scenario(
  "SSM Basics",
  [
    greet,
    pressEnter,
    createMaintenanceWindow,
    getMaintenanceWindow,
    sdkCreateMaintenanceWindow,
    modifyMaintenanceWindow,
    pressEnter,
    sdkModifyMaintenanceWindow,
```

```
    createSystemsManagerActions,
    getDocumentName,
    sdkCreateSSMDoc,
    ec2HelloWorld,
    enterIdOrSkipEC2HelloWorld,
    sdkEC2HelloWorld,
    sdkGetCommandTime,
    pressEnter,
    createSSMOpsItem,
    pressEnter,
    sdkCreateSSMOpsItem,
    updateOpsItem,
    pressEnter,
    sdkUpdateOpsItem,
    getOpsItemStatus,
    pressEnter,
    sdkOpsItemStatus,
    resolveOpsItem,
    pressEnter,
    sdkResolveOpsItem,
    askToDeleteResources,
    confirmDeleteChoice,
    sdkDeleteResources,
    goodbye,
  ],
  { ssmClient: new SSMClient({}) },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SystemsManagerScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Systems Manager."""

    def __init__(self, document_wrapper, maintenance_window_wrapper,
 ops_item_wrapper):
        """
        :param document_wrapper: An object that wraps Systems Manager document
 functions.
        :param maintenance_window_wrapper: An object that wraps Systems Manager
 maintenance window functions.
        :param ops_item_wrapper: An object that wraps Systems Manager OpsItem
 functions.
        """
```

```
        self.document_wrapper = document_wrapper
        self.maintenance_window_wrapper = maintenance_window_wrapper
        self.ops_item_wrapper = ops_item_wrapper

    def run(self):
        """Demonstrates how to use the AWS SDK for Python (Boto3) to get started
 with Systems Manager."""
        try:
            print("-" * 88)
            print(
                """
Welcome to the AWS Systems Manager SDK Getting Started scenario.
This program demonstrates how to interact with Systems Manager using the AWS SDK
 for Python (Boto3).
Systems Manager is the operations hub for your AWS applications and resources and
 a secure end-to-end management
solution. The program's primary functions include creating a maintenance window,
 creating a document, sending a
command to a document, listing documents, listing commands, creating an OpsItem,
 modifying an OpsItem, and deleting
Systems Manager resources. Upon completion of the program, all AWS resources are
 cleaned up.
Let's get started..."""
            )
            q.ask("Please hit Enter")

            print("-" * 88)
            print("Create a Systems Manager maintenance window.")
            maintenance_window_name = q.ask(
                "Please enter the maintenance window name (default is ssm-
maintenance-window):",
            )
            if not maintenance_window_name:
                maintenance_window_name = "ssm-maintenance-window"

            self.maintenance_window_wrapper.create(
                name=maintenance_window_name,
                schedule="cron(0 10 ? * MON-FRI *)",
                duration=2,
                cutoff=1,
                allow_unassociated_targets=True,
            )

            print("-" * 88)
```

```
            print("Modify the maintenance window by changing the schedule")
            q.ask("Please hit Enter")

            self.maintenance_window_wrapper.update(
                name=maintenance_window_name,
                schedule="cron(0 0 ? * MON *)",
                duration=24,
                cutoff=1,
                allow_unassociated_targets=True,
                enabled=True,
            )

            print("-" * 88)
            print(
                "Create a document that defines the actions that Systems Manager
 performs on your EC2 instance."
            )
            document_name = q.ask(
                "Please enter the document name (default is ssmdocument):"
            )

            if not document_name:
                document_name = "ssmdocument"

            self.document_wrapper.create(
                name=document_name,
                content="""
{
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
        }
    ]
}
            """,
            )
```

```
            self.document_wrapper.wait_until_active()

            print(
                """
Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'.
In order to run this command, you must provide the instance ID of a Linux EC2
 instance. If you do
not already have a running Linux EC2 instance in your account, you can create one
 using the AWS console.
For information about creating an EC2 instance, see
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-
wizard.html.
            """
            )

            if q.ask(
                "Would you like to run a command on an EC2 instance? (y/n)",
                q.is_yesno,
            ):
                instance_id = q.ask(
                    "Please enter the instance ID of the EC2 instance:",
 q.non_empty
                )
                command_id = self.document_wrapper.send_command(
                    instance_ids=[instance_id]
                )

                self.document_wrapper.wait_command_executed(
                    command_id=command_id, instance_id=instance_id
                )

                print("-" * 88)
                print(
                    "Lets get the time when the specific command was sent to the
 specific managed node"
                )
                q.ask("Please hit Enter")


 self.document_wrapper.list_command_invocations(instance_id=instance_id)

            print("-" * 88)
```

```
            print("-" * 88)
            print(
                """
Now we will create a  Systems Manager OpsItem.
An OpsItem is a feature provided by the Systems Manager service.
It is a type of operational data item that allows you to manage and track various
 operational issues,
events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.
For example, you could create an OpsItem whenever your application detects a
 critical error
or an anomaly in your infrastructure.
            """
            )
            q.ask("Please hit Enter")

            self.ops_item_wrapper.create(
                title="Disk Space Alert",
                description="Created by the Systems Manager Python (Boto3) API",
                source="EC2",
                category="Performance",
                severity="2",
            )

            print("-" * 88)
            print("-" * 88)
            print(f"Now we will update  the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(
                title="Disk Space Alert",
                description=f"An update to {self.ops_item_wrapper.id}",
            )

            print(
                f"Now we will get the status of the OpsItem
 {self.ops_item_wrapper.id}"
            )
            q.ask("Please hit Enter")

            # It may take a second for the ops item to be available
            counter = 0
            while not self.ops_item_wrapper.describe() and counter < 5:
```

```
                counter += 1
                time.sleep(1)

            print(f"Now we will resolve the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(status="Resolved")

            print("-" * 88)
            print("-" * 88)
            if q.ask(
                "Would you like to delete the Systems Manager resources? (y/n)",
                q.is_yesno,
            ):
                print("You selected to delete the resources.")
                self.cleanup()
            else:
                print("The Systems Manager resources will not be deleted")

            print("-" * 88)
            print("This concludes the Systems Manager SDK Getting Started
 scenario.")
            print("-" * 88)

        except Exception:
            self.cleanup()
            raise

    def cleanup(self):
        self.maintenance_window_wrapper.delete()
        self.ops_item_wrapper.delete()
        self.document_wrapper.delete()


if __name__ == "__main__":
    try:
        scenario = SystemsManagerScenario(
            DocumentWrapper.from_client(),
            MaintenanceWindowWrapper.from_client(),
            OpsItemWrapper.from_client(),
        )
        scenario.run()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Dokument- und Befehlsaktionen umschließt.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def describe(self):
        """
        Describes the document.

        :return: Document status.
        """
        try:
            response = self.ssm_client.describe_document(Name=self.name)
            return response["Document"]["Status"]
        except ClientError as err:
            logger.error(
                "Couldn't get %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def wait_until_active(self, max_attempts=20, delay=5):
        """
        Waits until the document is active.

        :param max_attempts: The maximum number of attempts for checking the
status.
        :param delay: The delay in seconds between each check.
        """
        attempt = 0
        status = ""
        while attempt <= max_attempts:
            status = self.describe()
            if status == "Active":
                break
            attempt += 1
            time.sleep(delay)

        if status != "Active":
            logger.error("Document is not active.")
        else:
            logger.info("Document is active.")

    def wait_command_executed(self, command_id, instance_id):
```

```
        """
        Waits until the command is executed on the instance.

        :param command_id: The ID of the command.
        :param instance_id: The ID of the instance.
        """

        waiter = self.ssm_client.get_waiter("command_executed")
        waiter.wait(CommandId=command_id, InstanceId=instance_id)

    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Definieren Sie eine Klasse, die Ops-Objekt-Aktionen umschließt.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
```

```
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
 OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
```

```
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
"Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
```

```
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Definieren Sie eine Klasse, die Wartungsfenster-Aktionen umschließt.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
 allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
```

```
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
```

```
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(
        self, name, enabled, schedule, duration, cutoff,
allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on
managed nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
```

```
            logger.error(
                "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Aktionen für Systems Manager mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Systems Manager Manager-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Systems Manager Manager-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Systems Manager AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS Systems Manager -API-Referenz.

Beispiele

- Verwendung von AddTagsToResource mit einer CLI
- Verwendung von CancelCommand mit einer CLI
- Verwendung von CreateActivation mit einer CLI
- Verwendung von CreateAssociation mit einer CLI
- Verwendung von CreateAssociationBatch mit einer CLI
- Verwenden Sie mit einem CreateDocument AWS SDK oder CLI
- Verwenden Sie CreateMaintenanceWindowmit einem AWS SDK oder CLI
- Verwenden Sie CreateOpsItemmit einem AWS SDK oder CLI
- Verwendung von CreatePatchBaseline mit einer CLI
- Verwendung von DeleteActivation mit einer CLI
- Verwendung von DeleteAssociation mit einer CLI
- Verwenden Sie mit einem DeleteDocument AWS SDK oder CLI
- Verwenden Sie DeleteMaintenanceWindowmit einem AWS SDK oder CLI
- Verwenden Sie DeleteOpsItemmit einem AWS SDK
- Verwendung von DeleteParameter mit einer CLI
- Verwendung von DeletePatchBaseline mit einer CLI
- Verwendung von DeregisterManagedInstance mit einer CLI
- Verwendung von DeregisterPatchBaselineForPatchGroup mit einer CLI
- Verwendung von DeregisterTargetFromMaintenanceWindow mit einer CLI
- Verwendung von DeregisterTaskFromMaintenanceWindow mit einer CLI
- Verwendung von DescribeActivations mit einer CLI
- Verwendung von DescribeAssociation mit einer CLI
- Verwendung von DescribeAssociationExecutionTargets mit einer CLI
- Verwendung von DescribeAssociationExecutions mit einer CLI
- Verwendung von DescribeAutomationExecutions mit einer CLI
- Verwendung von DescribeAutomationStepExecutions mit einer CLI
- Verwendung von DescribeAvailablePatches mit einer CLI

- Verwenden Sie mit einem DescribeDocument AWS SDK oder CLI
- Verwendung von DescribeDocumentPermission mit einer CLI
- Verwendung von DescribeEffectiveInstanceAssociations mit einer CLI
- Verwendung von DescribeEffectivePatchesForPatchBaseline mit einer CLI
- Verwendung von DescribeInstanceAssociationsStatus mit einer CLI
- Verwendung von DescribeInstanceInformation mit einer CLI
- Verwendung von DescribeInstancePatchStates mit einer CLI
- Verwendung von DescribeInstancePatchStatesForPatchGroup mit einer CLI
- Verwendung von DescribeInstancePatches mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutionTaskInvocations mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutionTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindowExecutions mit einer CLI
- Verwendung von DescribeMaintenanceWindowTargets mit einer CLI
- Verwendung von DescribeMaintenanceWindowTasks mit einer CLI
- Verwendung von DescribeMaintenanceWindows mit einer CLI
- Verwenden Sie mit einem DescribeOpsItems AWS SDK oder CLI
- Verwenden Sie DescribeParametersmit einem AWS SDK oder CLI
- Verwendung von DescribePatchBaselines mit einer CLI
- Verwendung von DescribePatchGroupState mit einer CLI
- Verwendung von DescribePatchGroups mit einer CLI
- Verwendung von GetAutomationExecution mit einer CLI
- Verwendung von GetCommandInvocation mit einer CLI
- Verwendung von GetConnectionStatus mit einer CLI
- Verwendung von GetDefaultPatchBaseline mit einer CLI
- Verwendung von GetDeployablePatchSnapshotForInstance mit einer CLI
- Verwendung von GetDocument mit einer CLI
- Verwendung von GetInventory mit einer CLI
- Verwendung von GetInventorySchema mit einer CLI
- Verwendung von GetMaintenanceWindow mit einer CLI
- Verwendung von GetMaintenanceWindowExecution mit einer CLI

- Verwendung von GetMaintenanceWindowExecutionTask mit einer CLI
- Verwenden Sie mit einem GetParameter AWS SDK oder CLI
- Verwendung von GetParameterHistory mit einer CLI
- Verwendung von GetParameters mit einer CLI
- Verwendung von GetPatchBaseline mit einer CLI
- Verwendung von GetPatchBaselineForPatchGroup mit einer CLI
- Verwendung von ListAssociationVersions mit einer CLI
- Verwendung von ListAssociations mit einer CLI
- Verwenden Sie mit einem ListCommandInvocations AWS SDK oder CLI
- Verwendung von ListCommands mit einer CLI
- Verwendung von ListComplianceItems mit einer CLI
- Verwendung von ListComplianceSummaries mit einer CLI
- Verwendung von ListDocumentVersions mit einer CLI
- Verwendung von ListDocuments mit einer CLI
- Verwendung von ListInventoryEntries mit einer CLI
- Verwendung von ListResourceComplianceSummaries mit einer CLI
- Verwendung von ListTagsForResource mit einer CLI
- Verwendung von ModifyDocumentPermission mit einer CLI
- Verwendung von PutComplianceItems mit einer CLI
- Verwendung von PutInventory mit einer CLI
- Verwenden Sie mit einem PutParameter AWS SDK oder CLI
- Verwendung von RegisterDefaultPatchBaseline mit einer CLI
- Verwendung von RegisterPatchBaselineForPatchGroup mit einer CLI
- Verwendung von RegisterTargetWithMaintenanceWindow mit einer CLI
- Verwendung von RegisterTaskWithMaintenanceWindow mit einer CLI
- Verwendung von RemoveTagsFromResource mit einer CLI
- Verwenden Sie mit einem SendCommand AWS SDK oder CLI
- Verwendung von StartAutomationExecution mit einer CLI
- Verwendung von StartSession mit einer CLI
- Verwendung von StopAutomationExecution mit einer CLI

- Verwendung von UpdateAssociation mit einer CLI
- Verwendung von UpdateAssociationStatus mit einer CLI
- Verwendung von UpdateDocument mit einer CLI
- Verwendung von UpdateDocumentDefaultVersion mit einer CLI
- Verwenden Sie mit einem UpdateMaintenanceWindow AWS SDK oder CLI
- Verwendung von UpdateManagedInstanceRole mit einer CLI
- Verwenden Sie mit einem UpdateOpsItem AWS SDK oder CLI
- Verwendung von UpdatePatchBaseline mit einer CLI

Verwendung von **`AddTagsToResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `AddTagsToResource` verwendet wird.

CLI

AWS CLI

Beispiel 1: So fügen Sie einem Wartungsfenster Tags hinzu

Im folgenden Beispiel `add-tags-to-resource` wird dem angegebenen Wartungsfenster ein Tag hinzugefügt.

```
aws ssm add-tags-to-resource \
    --resource-type "MaintenanceWindow" \
    --resource-id "mw-03eb9db428EXAMPLE" \
    --tags "Key=Stack,Value=Production"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So fügen Sie einem Parameter Tags hinzu

Im folgenden Beispiel `add-tags-to-resource` werden dem angegebenen Parameter zwei Tags hinzugefügt.

```
aws ssm add-tags-to-resource \
    --resource-type "Parameter" \
    --resource-id "My-Parameter" \
    --tags '[{"Key":"Region","Value":"East"},{"Key":"Environment",
 "Value":"Production"}]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: So fügen Sie Tags zu SSM-Dokumenten hinzu

Im folgenden `add-tags-to-resource`-Beispiel wird dem angegebenen Dokument ein Tag hinzugefügt.

```
aws ssm add-tags-to-resource \
    --resource-type "Document" \
    --resource-id "My-Document" \
    --tags "Key=Quarter,Value=Q322"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Markieren von Systems-Manager-Ressourcen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie AddTagsToResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster mit neuen Tags aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Einzelheiten zur API finden Sie unter AddTagsToResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster mit neuen Tags aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"

Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- Einzelheiten zur API finden Sie unter AddTagsToResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CancelCommand`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CancelCommand` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um einen Befehl für alle Instances abzubrechen

Im folgenden `cancel-command`-Beispiel wird versucht, den angegebenen Befehl abzubrechen, der bereits für alle Instances ausgeführt wird.

```
aws ssm cancel-command \
```

```
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: So brechen Sie einen Befehl für bestimmte Instances ab

Im folgenden `cancel-command`-Beispiel wird versucht, einen Befehl nur für die angegebene Instance abzubrechen.

```
aws ssm cancel-command \
    --command-id "662add3d-5831-4a10-b64a-f2ff3EXAMPLE"
    --instance-ids "i-02573cafcfEXAMPLE"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Markieren von Systems-Manager-Parametern im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CancelCommandin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird versucht, einen Befehl abzubrechen. Wenn der Vorgang erfolgreich ist, erfolgt keine Ausgabe.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Einzelheiten zur API finden Sie unter CancelCommand AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird versucht, einen Befehl abzubrechen. Wenn der Vorgang erfolgreich ist, erfolgt keine Ausgabe.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- Einzelheiten zur API finden Sie unter CancelCommand AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateActivation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateActivation` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Aktivierung für eine verwaltete Instance

Das folgende `create-activation`-Beispiel erstellt eine verwaltete Instance-Aktivierung.

```
aws ssm create-activation \
    --default-instance-name "HybridWebServers" \
    --iam-role "HybridWebServersRole" \
    --registration-limit 5
```

Ausgabe:

```
{
    "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
    "ActivationCode": "dRmgnYaFv567vEXAMPLE"
}
```

Weitere Informationen finden Sie unter Schritt 4: Managed-Instance Aktivierung für eine Hybridumgebung erstellen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateActivationunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine verwaltete Instance erstellt.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Ausgabe:

```
ActivationCode       ActivationId
--------------       ------------
```

```
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Einzelheiten zur API finden Sie unter CreateActivation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine verwaltete Instance erstellt.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole
 "SSMAutomationRole" -RegistrationLimit 10
```

Ausgabe:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- Einzelheiten zur API finden Sie unter CreateActivation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAssociation` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um ein Dokument mithilfe von Instance-IDs zuzuordnen

In diesem Beispiel wird mithilfe von Instance-IDs ein Konfigurationsdokument einer Instance zugeordnet.

```
aws ssm create-association \
    --instance-id "i-0cb2b964d3e14fd9f" \
    --name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
{
    "AssociationDescription": {
```

```
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Weitere Informationen finden Sie CreateAssociationin der AWS Systems Manager API-Referenz.

Beispiel 2: So verknüpfen Sie ein Dokument mithilfe von Zielen

In diesem Beispiel wird mithilfe von Zielen ein Konfigurationsdokument einer Instance zugeordnet.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Status": {
```

```
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Weitere Informationen finden Sie CreateAssociationin der AWS Systems Manager API-Referenz.

Beispiel 3: Um eine Zuordnung zu erstellen, die nur einmal ausgeführt wird

In diesem Beispiel wird eine neue Zuordnung erstellt, die nur einmal am angegebenen Datum und zu der angegebenen Uhrzeit ausgeführt wird. Verknüpfungen, die mit einem Datum in der Vergangenheit oder Gegenwart erstellt wurden (zum Zeitpunkt der Verarbeitung liegt das Datum in der Vergangenheit), werden sofort ausgeführt.

```
aws ssm create-association \
    --name "AWS-UpdateSSMAgent" \
    --targets "Key=instanceids,Values=i-0cb2b964d3e14fd9f" \
    --schedule-expression "at(2020-05-14T15:55:00)"  \
    --apply-only-at-cron-interval
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Status": {
            "Date": 1487875500.33,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-0cb2b964d3e14fd9f",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "b7c3266e-a544-44db-877e-b20d3a108189",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487875500.33,
        "Date": 1487875500.33,
        "Targets": [
            {
                "Values": [
                    "i-0cb2b964d3e14fd9f"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Weitere Informationen finden Sie CreateAssociationin der AWS Systems Manager API-Referenz oder Referenz: Cron- und Rate-Ausdrücke für Systems Manager im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mithilfe von Instance-IDs einer Instance zugeordnet.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Beispiel 2: In diesem Beispiel wird mithilfe von Zielen ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Beispiel 3: In diesem Beispiel wird mithilfe von Zielen und Parametern ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Ausgabe:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Beispiel 4: In diesem Beispiel wird eine Zuordnung mit allen Instances in der Region erstellt, mit**`AWS-GatherSoftwareInventory`**. Außerdem werden benutzerdefinierte Dateien und Registrierungsverzeichnisse in den zu erfassenden Parametern bereitgestellt

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Ausgabe:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter CreateAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mithilfe von Instance-IDs einer Instance zugeordnet.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Beispiel 2: In diesem Beispiel wird mithilfe von Zielen ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Beispiel 3: In diesem Beispiel wird mithilfe von Zielen und Parametern ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
```

```
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName
 "CWConfiguration" -Target $target -Parameter $params
```

Ausgabe:

```
Name                  : Configure-CloudWatch
InstanceId            :
Date                  : 5/17/2018 3:17:44 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

Beispiel 4: In diesem Beispiel wird eine Zuordnung mit allen Instances in der Region erstellt, mit**`AWS-GatherSoftwareInventory`**. Außerdem werden benutzerdefinierte Dateien und Registrierungsverzeichnisse in den zu erfassenden Parametern bereitgestellt

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Ausgabe:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter CreateAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateAssociationBatch`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateAssociationBatch` verwendet wird.

CLI

AWS CLI

So erstellen Sie mehrere Zuordnungen

In diesem Beispiel wird ein Konfigurationsdokument mehreren Instances zugeordnet. Die Ausgabe gibt gegebenenfalls eine Liste mit erfolgreichen und fehlgeschlagenen Vorgängen zurück.

Befehl:

```
aws ssm create-association-batch --entries "Name=AWS-UpdateSSMAgent,InstanceId=i-1234567890abcdef0" "Name=AWS-UpdateSSMAgent,InstanceId=i-9876543210abcdef0"
```

Ausgabe:

```
{
  "Successful": [
      {
          "Name": "AWS-UpdateSSMAgent",
          "InstanceId": "i-1234567890abcdef0",
          "AssociationVersion": "1",
          "Date": 1550504725.007,
          "LastUpdateAssociationDate": 1550504725.007,
          "Status": {
              "Date": 1550504725.007,
              "Name": "Associated",
              "Message": "Associated with AWS-UpdateSSMAgent"
          },
          "Overview": {
              "Status": "Pending",
              "DetailedStatus": "Creating"
          },
          "DocumentVersion": "$DEFAULT",
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Targets": [
```

```
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ]
        },
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationVersion": "1",
            "Date": 1550504725.057,
            "LastUpdateAssociationDate": 1550504725.057,
            "Status": {
                "Date": 1550504725.057,
                "Name": "Associated",
                "Message": "Associated with AWS-UpdateSSMAgent"
            },
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Creating"
            },
            "DocumentVersion": "$DEFAULT",
            "AssociationId": "9c9f7f20-5154-4fed-a83e-0123456789ab",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ]
        }
    ],
    "Failed": []
}
```

- Einzelheiten zur API finden Sie CreateAssociationBatchin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mehreren Instances zugeordnet. Die Ausgabe gibt gegebenenfalls eine Liste mit erfolgreichen und fehlgeschlagenen Vorgängen zurück.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Ausgabe:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Beispiel 2: In diesem Beispiel werden alle Details eines erfolgreichen Vorgangs angezeigt.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Einzelheiten zur API finden Sie unter CreateAssociationBatch AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mehreren Instances zugeordnet. Die Ausgabe gibt gegebenenfalls eine Liste mit erfolgreichen und fehlgeschlagenen Vorgängen zurück.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Ausgabe:

```
Failed  Successful
```

```
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Beispiel 2: In diesem Beispiel werden alle Details eines erfolgreichen Vorgangs angezeigt.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- Einzelheiten zur API finden Sie unter CreateAssociationBatch AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`CreateDocument`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateDocument` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So erstellen Sie ein Dokument

Das folgende `create-document`-Beispiel erstellt ein Systems-Manager-Dokument.

```
aws ssm create-document \
    --content file://exampleDocument.yml \
    --name "Example" \
    --document-type "Automation" \
    --document-format YAML
```

Ausgabe:

```
{
```

```
    "DocumentDescription": {
        "Hash":
"fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583256349.452,
        "Status": "Creating",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
Automation to perform the actions on your behalf. If no role is specified,
Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

Weitere Informationen finden Sie unter Erstellen von Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateDocumentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates an AWS SSM document asynchronously.
 *
 * @param docName The name of the document to create.
 * <p>
 * This method initiates an asynchronous request to create an SSM document.
 * If the request is successful, it prints the document status.
 * If an exception occurs, it handles the error appropriately.
 */
public void createSSMDoc(String docName) throws SsmException {
    String jsonData = """
    {
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
          }
        ]
    }
    """;

    CreateDocumentRequest request = CreateDocumentRequest.builder()
            .content(jsonData)
            .name(docName)
            .documentType(DocumentType.COMMAND)
```

```
                .build();

        CompletableFuture<CreateDocumentResponse> future =
 getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
 response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- Einzelheiten zur API finden Sie CreateDocumentin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ content, name, documentType }) => {
  const client = new SSMClient({});
```

```
  try {
    const { documentDescription } = await client.send(
      new CreateDocumentCommand({
        Content: content, // The content for the new SSM document. The content
 must not exceed 64KB.
        Name: name,
        DocumentType: documentType, // Document format type can be JSON, YAML, or
 TEXT. The default format is JSON.
      }),
    );
    console.log("Document created successfully.");
    return { DocumentDescription: documentDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DocumentAlreadyExists") {
      console.warn(`${caught.message}. Did you provide a new document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateDocumentin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Dokument in Ihrem Konto erstellt. Das Dokument muss im JSON-Format vorliegen. Weitere Informationen zum Schreiben eines Konfigurationsdokuments finden Sie unter Konfigurationsdokument in der SSM-API-Referenz.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Ausgabe:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
```

```
DocumentType    : Command
DocumentVersion : 1
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- Einzelheiten zur API finden Sie unter CreateDocument AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument in Ihrem Konto erstellt. Das Dokument muss im JSON-Format vorliegen. Weitere Informationen zum Schreiben eines Konfigurationsdokuments finden Sie unter Konfigurationsdokument in der SSM-API-Referenz.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Ausgabe:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
```

```
Status          : Creating
```

- Einzelheiten zur API finden Sie unter CreateDocument AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
```

```
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateDocumentin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ssm->createdocument(
            iv_name = iv_name
            iv_content = iv_content
            iv_documenttype = 'Command' ).
        MESSAGE 'Document created.' TYPE 'I'.
      CATCH /aws1/cx_ssmdocalreadyexists.
        MESSAGE 'Document already exists.' TYPE 'I'.
      CATCH /aws1/cx_ssminvaliddoccontent.
        MESSAGE 'Invalid document content.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateDocumentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateMaintenanceWindow`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateMaintenanceWindow` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So erstellen Sie ein Wartungsfenster

Im folgenden `create-maintenance-window`-Beispiel wird ein neues Wartungsfenster erstellt, das alle fünf Minuten für bis zu zwei Stunden (je nach Bedarf) alle fünf Minuten erstellt, verhindert, dass neue Aufgaben innerhalb einer Stunde nach Ende der Ausführung des Wartungsfensters gestartet werden, nicht zugeordnete Ziele (Instances, die Sie nicht für das Wartungsfenster registriert haben) zulässt und durch die Verwendung benutzerdefinierter Tags darauf hinweist, dass der Ersteller beabsichtigt, es in einem Tutorial zu verwenden.

```
aws ssm create-maintenance-window \
    --name "My-Tutorial-Maintenance-Window" \
    --schedule "rate(5 minutes)" \
    --duration 2 --cutoff 1 \
    --allow-unassociated-targets \
    --tags "Key=Purpose,Value=Tutorial"
```

Ausgabe:

```
{
    "WindowId": "mw-0c50858d01EXAMPLE"
}
```

Beispiel 2: Erstellen eines Wartungsfensters, das nur einmal ausgeführt wird

Im folgenden `create-maintenance-window`-Beispiel wird ein neues Wartungsfenster erstellt, das nur einmal am angegebenen Datum und zur angegebenen Uhrzeit ausgeführt wird.

```
aws ssm create-maintenance-window \
    --name My-One-Time-Maintenance-Window \
    --schedule "at(2020-05-14T15:55:00)" \
    --duration 5 \
    --cutoff 2 \
    --allow-unassociated-targets \
    --tags "Key=Environment,Value=Production"
```

Ausgabe:

```
{
    "WindowId": "mw-01234567890abcdef"
}
```

Weitere Informationen finden Sie unter Wartungsfenster im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateMaintenanceWindowin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
 window.
```

```
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
                .name(winName)
                .description("This is my maintenance window")
                .allowUnassociatedTargets(true)
                .duration(2)
                .cutoff(1)
                .schedule("cron(0 10 ? * MON-FRI *)")
                .build();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();
```

```
            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture
= getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- Einzelheiten zur API finden Sie CreateMaintenanceWindowin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM maintenance window.
 * @param {{ name: string, allowUnassociatedTargets: boolean, duration: number,
 cutoff: number, schedule: string, description?: string }}
 */
export const main = async ({
  name,
  allowUnassociatedTargets, // Allow the maintenance window to run on managed
 nodes, even if you haven't registered those nodes as targets.
  duration, // The duration of the maintenance window in hours.
  cutoff, // The number of hours before the end of the maintenance window that
 Amazon Web Services Systems Manager stops scheduling new tasks for execution.
  schedule, // The schedule of the maintenance window in the form of a cron or
 rate expression.
  description = undefined,
}) => {
  const client = new SSMClient({});

  try {
    const { windowId } = await client.send(
      new CreateMaintenanceWindowCommand({
        Name: name,
        Description: description,
        AllowUnassociatedTargets: allowUnassociatedTargets, // Allow the
 maintenance window to run on managed nodes, even if you haven't registered those
 nodes as targets.
        Duration: duration, // The duration of the maintenance window in hours.
```

```
        Cutoff: cutoff, // The number of hours before the end of the maintenance
 window that Amazon Web Services Systems Manager stops scheduling new tasks for
 execution.
        Schedule: schedule, // The schedule of the maintenance window in the form
 of a cron or rate expression.
      }),
    );
    console.log(`Maintenance window created with Id: ${windowId}`);
    return { WindowId: windowId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateMaintenanceWindowin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein neues Wartungsfenster mit dem angegebenen Namen erstellt, das an jedem Dienstag um 16 Uhr für 4 Stunden läuft, mit einer Frist von 1 Stunde, und das Ziele ohne Zuordnung zulässt.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Ausgabe:

```
mw-03eb53e1ea7383998
```

- Einzelheiten zur API finden Sie unter CreateMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein neues Wartungsfenster mit dem angegebenen Namen erstellt, das an jedem Dienstag um 16 Uhr für 4 Stunden läuft, mit einer Frist von 1 Stunde, und das Ziele ohne Zuordnung zulässt.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Ausgabe:

```
mw-03eb53e1ea7383998
```

- Einzelheiten zur API finden Sie unter CreateMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
```

```
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff,
allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes
as targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateMaintenanceWindowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ssm->createmaintenancewindow(
            iv_name = iv_name
            iv_schedule = iv_schedule
            iv_duration = iv_duration
            iv_cutoff = iv_cutoff
            iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
        MESSAGE 'Maintenance window created.' TYPE 'I'.
      CATCH /aws1/cx_ssmresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateMaintenanceWindowin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`CreateOpsItem`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateOpsItem` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Um ein zu erstellen OpsItems

Im folgenden `create-ops-item` Beispiel wird der aws/resources Schlüssel/verwendet, um eine OperationalData Ressource zu erstellen, die OpsItem mit einer Amazon DynamoDB zusammenhängt.

```
aws ssm create-ops-item \
    --title "EC2 instance disk full" \
    --description "Log clean up may have failed which caused the disk to be full"
 \
    --priority 2 \
    --source ec2 \
    --operational-data '{"/aws/resources":{"Value":"[{\"arn
\": \"arn:aws:dynamodb:us-west-2:12345678:table/OpsItems
\"}]","Type":"SearchableString"}}' \
    --notifications Arn="arn:aws:sns:us-west-2:12345678:TestUser"
```

Ausgabe:

```
{
    "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

Weitere Informationen finden Sie unter Creating OpsItems im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreateOpsItemunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- Einzelheiten zur API finden Sie CreateOpsItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM OpsItem.
 * @param {{ title: string, source: string, category?: string, severity?:
 string }}
 */
export const main = async ({
  title,
  source,
  category = undefined,
  severity = undefined,
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new CreateOpsItemCommand({
        Title: title,
        Source: source, // The origin of the OpsItem, such as Amazon EC2 or
 Systems Manager.
        Category: category,
        Severity: severity,
      }),
    );
    console.log(`Ops item created with id: ${opsItemId}`);
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
```

```
    }
  }
};
```

- Einzelheiten zur API finden Sie CreateOpsItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem
```

```
        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
 OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie CreateOpsItemin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ssm->createopsitem(
        iv_title = iv_title
        iv_source = iv_source
        iv_category = iv_category
        iv_severity = iv_severity
        iv_description = iv_description ).
    MESSAGE 'OpsItem created.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsitemlimitexcdex.
    MESSAGE 'You have exceeded your open OpsItem limit.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsitemalrdyexex.
    MESSAGE 'OpsItem already exists.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie CreateOpsItemin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`CreatePatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreatePatchBaseline` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Patch-Baseline mit automatischer Genehmigung

Das folgende `create-patch-baseline`-Beispiel erstellt eine Patch-Baseline für Windows Server, die Patches sieben Tage nach ihrer Veröffentlichung durch Microsoft für eine Produktionsumgebung genehmigt.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Ausgabe:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
}
```

Beispiel 2: So erstellen Sie eine Patch-Baseline mit einem Stichtag für die Genehmigung

Im folgenden `create-patch-baseline`-Beispiel wird eine Patch-Baseline für Windows Server erstellt, die alle Patches für eine Produktionsumgebung genehmigt, die am oder vor dem 7. Juli 2020 veröffentlicht wurden.

```
aws ssm create-patch-baseline \
    --name "Windows-Production-Baseline-AutoApproval" \
    --operating-system "WINDOWS" \
    --approval-
rules "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,1
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},App
 \
    --description "Baseline containing all updates approved for Windows Server
 production systems"
```

Ausgabe:

```
{
    "BaselineId": "pb-045f10b4f3EXAMPLE"
}
```

Beispiel 3: So erstellen Sie eine Patch-Baseline mit Genehmigungsregeln, die in einer JSON-Datei gespeichert sind

Im folgenden `create-patch-baseline`-Beispiel wird eine Patch-Baseline für Amazon Linux 2017.09 erstellt, die Patches für eine Produktionsumgebung sieben Tage nach ihrer Veröffentlichung genehmigt, Genehmigungsregeln für die Patch-Baseline festlegt und ein benutzerdefiniertes Repository für Patches festlegt.

```
aws ssm create-patch-baseline \
    --cli-input-json file://my-amazon-linux-approval-rules-and-repo.json
```

Inhalt von `my-amazon-linux-approval-rules-and-repo.json`:

```
{
    "Name": "Amazon-Linux-2017.09-Production-Baseline",
    "Description": "My approval rules patch baseline for Amazon Linux 2017.09
 instances",
    "OperatingSystem": "AMAZON_LINUX",
    "Tags": [
        {
            "Key": "Environment",
            "Value": "Production"
        }
    ],
    "ApprovalRules": {
        "PatchRules": [
            {
                "ApproveAfterDays": 7,
                "EnableNonSecurity": true,
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "SEVERITY",
                            "Values": [
                                "Important",
                                "Critical"
                            ]
                        },
                        {
                            "Key": "CLASSIFICATION",
                            "Values": [
                                "Security",
                                "Bugfix"
                            ]
                        },
```

```
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "AmazonLinux2017.09"
                            ]
                        }
                    ]
                }
            }
        ]
    },
    "Sources": [
        {
            "Name": "My-AL2017.09",
            "Products": [
                "AmazonLinux2017.09"
            ],
            "Configuration": "[amzn-main] \nname=amzn-main-Base
\nmirrorlist=http://repo./$awsregion./$awsdomain//$releasever/main/
mirror.list //nmirrorlist_expire=300//nmetadata_expire=300 \npriority=10
 \nfailovermethod=priority \nfastestmirror_enabled=0 \ngpgcheck=1
 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1 \nretries=3
 \ntimeout=5\nreport_instanceid=yes"
        }
    ]
}
```

Beispiel 4: So erstellen Sie eine Patch-Baseline, die genehmigte und abgelehnte Patches angibt

Im folgenden `create-patch-baseline`-Beispiel werden Patches, die genehmigt und abgelehnt werden sollen, ausdrücklich als Ausnahme von den Standard-Genehmigungsregeln angegeben.

```
aws ssm create-patch-baseline \
    --name "Amazon-Linux-2017.09-Alpha-Baseline" \
    --description "My custom approve/reject patch baseline for Amazon Linux
 2017.09 instances" \
    --operating-system "AMAZON_LINUX" \
    --approved-patches "CVE-2018-1234567,example-pkg-EE-2018*.amzn1.noarch" \
    --approved-patches-compliance-level "HIGH" \
    --approved-patches-enable-non-security \
```

```
    --tags "Key=Environment,Value=Alpha"
```

Weitere Informationen finden Sie unter Benutzerdefineirte Patch-Baseline erstellen im AWS - Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie CreatePatchBaselinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline erstellt, die Patches sieben Tage nach der Veröffentlichung durch Microsoft für verwaltete Instances genehmigt, auf denen Windows Server 2019 in einer Produktionsumgebung ausgeführt wird.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters
```

```
New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Ausgabe:

```
pb-0z4z6221c4296b23z
```

- Einzelheiten zur API finden Sie unter CreatePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline erstellt, die Patches sieben Tage nach der Veröffentlichung durch Microsoft für verwaltete Instances genehmigt, auf denen Windows Server 2019 in einer Produktionsumgebung ausgeführt wird.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
```

```
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description
 "Baseline containing all updates approved for production systems" -
ApprovalRules_PatchRule $rule
```

Ausgabe:

```
pb-0z4z6221c4296b23z
```

- Einzelheiten zur API finden Sie unter CreatePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteActivation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteActivation` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Aktivierung für eine verwaltete Instance

Im folgenden Beispiel für `delete-activation` wird die Aktivierung einer verwalteten Instance gelöscht.

```
aws ssm delete-activation \
    --activation-id "aa673477-d926-42c1-8757-1358cEXAMPLE"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Setting Up AWS Systems Manager for Hybrid Environments im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteActivationunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Aktivierung gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Einzelheiten zur API finden Sie unter DeleteActivation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Aktivierung gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- Einzelheiten zur API finden Sie unter DeleteActivation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteAssociation` verwendet wird.

CLI

AWS CLI

Beispiel 1: So löschen Sie eine Zuordnung mithilfe der Zuordnungs-ID

Im folgenden Beispiel `delete-association` wird die Zuordnung für die angegebene Zuordnungs-ID gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
aws ssm delete-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Bearbeiten und Erstellen einer neuen Version einer Zuordnung im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So löschen Sie eine Zuordnung

Im folgenden Beispiel für `delete-association` wird die Verknüpfung zwischen einer Instance und einem Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
aws ssm delete-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Arbeiten mit Zuordnungen in Systems Manager im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Verknüpfung zwischen einer Instance und einem Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- Einzelheiten zur API finden Sie unter DeleteAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Verknüpfung zwischen einer Instance und einem Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-
UpdateSSMAgent"
```

- Einzelheiten zur API finden Sie unter DeleteAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DeleteDocument`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteDocument` verwendet wird.

CLI

AWS CLI

Löschen eines Dokuments

Das folgende `delete-document`-Beispiel löscht ein Systems-Manager-Dokument.

```
aws ssm delete-document \
    --name "Example"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Erstellen von Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteDocumentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
```

```
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteDocumentin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM document.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(new DeleteDocumentCommand({ Name: documentName }));
    console.log(`Document '${documentName}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteDocumentin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Einzelheiten zur API finden Sie unter DeleteDocument AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMDocument -Name "RunShellScript"
```

- Einzelheiten zur API finden Sie unter DeleteDocument AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
```

```
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteDocumentin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ssm->deletedocument( iv_name = iv_name ).
        MESSAGE 'Document deleted.' TYPE 'I'.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
      CATCH /aws1/cx_ssmassocdinstances.
        MESSAGE 'Document has associated instances.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteDocumentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteMaintenanceWindow`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteMaintenanceWindow` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Löschen eines Wartungsfensters

In diesem `delete-maintenance-window`-Beispiel wird das angegebene Wartungsfenster entfernt.

```
aws ssm delete-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9"
```

Ausgabe:

```
{
    "WindowId":"mw-1a2b3c4d5e6f7g8h9"
}
```

Weitere Informationen finden Sie unter Löschen eines Wartungsfensters (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteMaintenanceWindowunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an AWS SSM Maintenance Window asynchronously.
    *
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was
successfully deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteMaintenanceWindowin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM maintenance window.
 * @param {{ windowId: string }}
 */
export const main = async ({ windowId }) => {
  const client = new SSMClient({});
  try {
```

```
    await client.send(
      new DeleteMaintenanceWindowCommand({ WindowId: windowId }),
    );
    console.log(`Maintenance window '${windowId}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie DeleteMaintenanceWindowin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster entfernt.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Ausgabe:

```
mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DeleteMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster entfernt.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Ausgabe:

```
mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DeleteMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
```

```
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteMaintenanceWindowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->deletemaintenancewindow( iv_windowid = iv_window_id ).
    MESSAGE 'Maintenance window deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssminternalservererr.
    MESSAGE 'Internal server error occurred.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteMaintenanceWindowin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteOpsItem`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DeleteOpsItem` verwendet wird.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DeleteOpsItemin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->deleteopsitem( iv_opsitemid = iv_ops_item_id ).
    MESSAGE 'OpsItem deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsiteminvparamex.
    MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteOpsItemin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DeleteParameter`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteParameter` verwendet wird.

CLI

AWS CLI

So löschen Sie einen Parameter

Das folgende `delete-parameter`Beispiel löscht die angegebene einzelne Parameter.

```
aws ssm delete-parameter \
    --name "MyParameter"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Arbeiten mit Parameter Store im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeleteParameterin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Parameter gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMParameter -Name "helloWorld"
```

- Einzelheiten zur API finden Sie unter DeleteParameter AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Parameter gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMParameter -Name "helloWorld"
```

- Einzelheiten zur API finden Sie unter DeleteParameter AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeletePatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePatchBaseline` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Patch-Baseline

Das folgende `delete-patch-baseline`-Beispiel löscht die angegebene Patch-Baseline.

```
aws ssm delete-patch-baseline \
    --baseline-id "pb-045f10b4f382baeda"
```

Ausgabe:

```
{
    "BaselineId": "pb-045f10b4f382baeda"
}
```

Weitere Informationen finden Sie unter Aktualisieren und Löschen einer Patch-Baseline (Konsole) im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeletePatchBaselinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline gelöscht.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Ausgabe:

```
pb-045f10b4f382baeda
```

- Einzelheiten zur API finden Sie unter DeletePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline gelöscht.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Ausgabe:

```
pb-045f10b4f382baeda
```

- Einzelheiten zur API finden Sie unter DeletePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterManagedInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterManagedInstance` verwendet wird.

CLI

AWS CLI

So heben Sie die Registrierung einer verwalteten Instance auf

Im folgenden `deregister-managed-instance`-Beispiel wird die Registrierung der angegebenen verwalteten Instance aufgehoben.

```
aws ssm deregister-managed-instance \
    --instance-id 'mi-08ab247cdfEXAMPLE'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Abmelden verwalteter Knoten in einer Hybrid- und Multi-Cloud-Umgebung im Benutzerhandbuch zuAWS Systems Manager.

- Einzelheiten zur API finden Sie DeregisterManagedInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Registrierung einer verwalteten Instance aufgehoben. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Einzelheiten zur API finden Sie unter DeregisterManagedInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung einer verwalteten Instance aufgehoben. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- Einzelheiten zur API finden Sie unter DeregisterManagedInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterPatchBaselineForPatchGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterPatchBaselineForPatchGroup` verwendet wird.

CLI

AWS CLI

So heben Sie die Registrierung einer Patch-Gruppe für eine Patch-Baseline auf

Im folgenden `deregister-patch-baseline-for-patch-group`-Beispiel wird die Registrierung der angegebenen Patchgruppe von der angegebenen Patch-Baseline aufgehoben.

```
aws ssm deregister-patch-baseline-for-patch-group \
    --patch-group "Production" \
    --baseline-id "pb-0ca44a362fEXAMPLE"
```

Ausgabe:

```
{
  "PatchGroup":"Production",
  "BaselineId":"pb-0ca44a362fEXAMPLE"
}
```

Weitere Informationen finden Sie unter Hinzufügen einer Patch-Gruppe zu einer Patch-Baseline im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeregisterPatchBaselineForPatchGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Registrierung einer Patchgruppe von einer Patch-Baseline aufgehoben.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter DeregisterPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung einer Patchgruppe von einer Patch-Baseline aufgehoben.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter DeregisterPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterTargetFromMaintenanceWindow`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterTargetFromMaintenanceWindow` verwendet wird.

CLI

AWS CLI

So entfernen Sie ein Ziel aus einem Wartungsfenster

Im folgenden Beispiel `deregister-target-from-maintenance-window` wird das angegebene Ziel aus dem angegebenen Wartungsfenster entfernt.

```
aws ssm deregister-target-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-target-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
```

Ausgabe:

```
{
    "WindowId":"mw-ab12cd34ef56gh78",
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Weitere Informationen finden Sie unter Aktualisieren eines Wartungsfensters (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeregisterTargetFromMaintenanceWindowunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Ziel aus einem Wartungsfenster entfernt.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Einzelheiten zur API finden Sie unter DeregisterTargetFromMaintenanceWindow AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Ziel aus einem Wartungsfenster entfernt.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId
 "6ab5c208-9fc4-4697-84b7-b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- Einzelheiten zur API finden Sie unter DeregisterTargetFromMaintenanceWindow AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeregisterTaskFromMaintenanceWindow`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeregisterTaskFromMaintenanceWindow` verwendet wird.

CLI

AWS CLI

So entfernen Sie eine Aufgabe aus einem Wartungsfenster

Im folgenden Beispiel `deregister-task-from-maintenance-window` wird die angegebene Aufgabe aus dem angegebenen Wartungsfenster entfernt.

```
aws ssm deregister-task-from-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --window-task-id "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c"
```

Ausgabe:

```
{
    "WindowTaskId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",
    "WindowId":"mw-ab12cd34ef56gh78"
}
```

Weitere Informationen finden Sie unter Systems Manager Maintenance Windows Tutorials (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DeregisterTaskFromMaintenanceWindowunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Aufgabe aus einem Wartungsfenster entfernt.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Ausgabe:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Einzelheiten zur API finden Sie unter DeregisterTaskFromMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Aufgabe aus einem Wartungsfenster entfernt.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Ausgabe:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- Einzelheiten zur API finden Sie unter DeregisterTaskFromMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeActivations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeActivations` verwendet wird.

CLI

AWS CLI

Um Aktivierungen zu beschreiben

Das folgende `describe-activations` Beispiel listet Details zu den Aktivierungen in Ihrem AWS Konto auf.

```
aws ssm describe-activations
```

Ausgabe:

```
{
    "ActivationList": [
        {
            "ActivationId": "5743558d-563b-4457-8682-d16c3EXAMPLE",
            "Description": "Example1",
            "IamRole": "HybridWebServersRole,
```

```
            "RegistrationLimit": 5,
            "RegistrationsCount": 5,
            "ExpirationDate": 1584316800.0,
            "Expired": false,
            "CreatedDate": 1581954699.792
        },
        {
            "ActivationId": "3ee0322b-f62d-40eb-b672-13ebfEXAMPLE",
            "Description": "Example2",
            "IamRole": "HybridDatabaseServersRole",
            "RegistrationLimit": 5,
            "RegistrationsCount": 5,
            "ExpirationDate": 1580515200.0,
            "Expired": true,
            "CreatedDate": 1578064132.002
        },
    ]
}
```

Weitere Informationen finden Sie unter Schritt 4: Managed-Instance Aktivierung für eine Hybridumgebung erstellen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeActivationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel enthält Details zu den Aktivierungen in Ihrem Konto.

```
Get-SSMActivation
```

Ausgabe:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
```

```
RegistrationsCount  : 0
```

- Einzelheiten zur API finden Sie unter DescribeActivations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel enthält Details zu den Aktivierungen in Ihrem Konto.

```
Get-SSMActivation
```

Ausgabe:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- Einzelheiten zur API finden Sie unter DescribeActivations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAssociation` verwendet wird.

CLI

AWS CLI

Beispiel 1: So rufen Sie Details zu einer Zuordnung ab

Das folgende `describe-association`-Beispiel beschreibt die Zuordnung für die angegebene Zuordnungs-ID.

```
aws ssm describe-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Name": "AWS-GatherSoftwareInventory",
        "AssociationVersion": "1",
        "Date": 1534864780.995,
        "LastUpdateAssociationDate": 1543235759.81,
        "Overview": {
            "Status": "Success",
            "AssociationStatusAggregatedCount": {
                "Success": 2
            }
        },
        "DocumentVersion": "$DEFAULT",
        "Parameters": {
            "applications": [
                "Enabled"
            ],
            "awsComponents": [
                "Enabled"
            ],
            "customInventory": [
                "Enabled"
            ],
            "files": [
                ""
            ],
            "instanceDetailedInformation": [
                "Enabled"
            ],
            "networkConfig": [
                "Enabled"
            ],
            "services": [
                "Enabled"
            ],
            "windowsRegistry": [
                ""
            ],
            "windowsRoles": [
                "Enabled"
            ],
            "windowsUpdates": [
```

```
                "Enabled"
            ]
        },
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "*"
                ]
            }
        ],
        "ScheduleExpression": "rate(24 hours)",
        "LastExecutionDate": 1550501886.0,
        "LastSuccessfulExecutionDate": 1550501886.0,
        "AssociationName": "Inventory-Association"
    }
}
```

Weitere Informationen finden Sie unter Bearbeiten und Erstellen einer neuen Version einer Zuordnung im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So rufen Sie Details zu einer Zuordnung für eine bestimmte Instance und ein bestimmtes Dokument ab

Das folgende `describe-association`-Beispiel beschreibt die Zuordnung zwischen einer Instance und einem Dokument.

```
aws ssm describe-association \
    --instance-id "i-1234567890abcdef0" \
    --name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Status": {
            "Date": 1487876122.564,
            "Message": "Associated with AWS-UpdateSSMAgent",
            "Name": "Associated"
        },
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-1234567890abcdef0",
```

```
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Associated",
            "AssociationStatusAggregatedCount": {
                "Pending": 1
            }
        },
        "AssociationId": "d8617c07-2079-4c18-9847-1234567890ab",
        "DocumentVersion": "$DEFAULT",
        "LastUpdateAssociationDate": 1487876122.564,
        "Date": 1487876122.564,
        "Targets": [
            {
                "Values": [
                    "i-1234567890abcdef0"
                ],
                "Key": "InstanceIds"
            }
        ]
    }
}
```

Weitere Informationen finden Sie unter Bearbeiten und Erstellen einer neuen Version einer Zuordnung im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die Zuordnung zwischen einer Instance und einem Dokument.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
```

```
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Einzelheiten zur API finden Sie unter DescribeAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt die Zuordnung zwischen einer Instance und einem Dokument.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Einzelheiten zur API finden Sie unter DescribeAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAssociationExecutionTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAssociationExecutionTargets` verwendet wird.

CLI

AWS CLI

So rufen Sie Details zu einer Zuordnung ab

Das folgende `describe-association-execution-targets`-Beispiel beschreibt die angegebene Zuordnungsausführung.

```
aws ssm describe-association-execution-targets \
```

```
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --execution-id "7abb6378-a4a5-4f10-8312-0123456789ab"
```

Ausgabe:

```
{
    "AssociationExecutionTargets": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "ResourceId": "i-1234567890abcdef0",
            "ResourceType": "ManagedInstance",
            "Status": "Success",
            "DetailedStatus": "Success",
            "LastExecutionDate": 1550505538.497,
            "OutputSource": {
                "OutputSourceId": "97fff367-fc5a-4299-aed8-0123456789ab",
                "OutputSourceType": "RunCommand"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Anzeigen von Zuordnungsverläufen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAssociationExecutionTargetsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Ressourcen-ID und ihr Ausführungsstatus angezeigt, die Teil der Ausführungsziele der Zuordnung sind

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Ausgabe:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Beispiel 2: Dieser Befehl überprüft die jeweilige Ausführung einer bestimmten Automatisierung seit gestern, der ein Befehlsdokument zugeordnet ist. Außerdem wird geprüft, ob die Ausführung der Zuordnung fehlgeschlagen ist, und wenn ja, werden die Details zum Befehlsaufruf für die Ausführung zusammen mit der Instance-ID angezeigt

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Ausgabe:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
```

```
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutionTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Ressourcen-ID und ihr Ausführungsstatus angezeigt, die Teil der Ausführungsziele der Zuordnung sind

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Ausgabe:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Beispiel 2: Dieser Befehl überprüft die jeweilige Ausführung einer bestimmten Automatisierung seit gestern, der ein Befehlsdokument zugeordnet ist. Außerdem wird geprüft, ob die Ausführung der Zuordnung fehlgeschlagen ist, und wenn ja, werden die Details zum Befehlsaufruf für die Ausführung zusammen mit der Instance-ID angezeigt

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
```

```
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId
 $execution.OutputSource.OutputSourceId -Detail:$true | Select-Object -
ExpandProperty CommandPlugins
    }
}
```

Ausgabe:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0
 on i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
StatusDetails          : Failed
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutionTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAssociationExecutions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAssociationExecutions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erhalten Sie Details zu allen Ausführungen für eine Zuordnung

Das folgende `describe-association-executions`-Beispiel beschreibt alle Ausführungen der angegebenen Zuordnung.

```
aws ssm describe-association-executions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Ausgabe:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

Weitere Informationen finden Sie unter Anzeigen von Zuordnungsverläufen im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So erhalten Sie Details zu allen Ausführungen für eine Zuordnung nach einem bestimmten Datum und einer bestimmten Uhrzeit

Im folgenden `describe-association-executions`-Beispiel werden alle Ausführungen einer Zuordnung nach dem angegebenen Datum und der angegebenen Uhrzeit beschrieben.

```
aws ssm describe-association-executions \
```

```
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --filters "Key=CreatedTime,Value=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

Ausgabe:

```
{
    "AssociationExecutions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "474925ef-1249-45a2-b93d-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505827.119,
            "ResourceCountByStatus": "{Success=1}"
        },
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
            "Status": "Success",
            "DetailedStatus": "Success",
            "CreatedTime": 1550505536.843,
            "ResourceCountByStatus": "{Success=1}"
        },
        ...
    ]
}
```

Weitere Informationen finden Sie unter Anzeigen von Zuordnungsverläufen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAssociationExecutionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Ausführungen für die angegebene Zuordnungs-ID zurückgegeben

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Ausführungen für die angegebene Zuordnungs-ID zurückgegeben

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- Einzelheiten zur API finden Sie unter DescribeAssociationExecutions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAutomationExecutions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutomationExecutions` verwendet wird.

CLI

AWS CLI

So beschreiben Sie die Ausführung einer Automatisierung

Im folgenden `describe-automation-executions`-Beispiel werden Details einer Automation-Ausführung gezeigt.

```
aws ssm describe-automation-executions \
    --filters Key=ExecutionId,Values=73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Ausgabe:

```
{
    "AutomationExecutionMetadataList": [
        {
            "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
            "DocumentName": "AWS-StartEC2Instance",
            "DocumentVersion": "1",
            "AutomationExecutionStatus": "Success",
            "ExecutionStartTime": 1583737233.748,
            "ExecutionEndTime": 1583737234.719,
            "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/
mw_service_role/OrchestrationService",
            "LogFile": "",
            "Outputs": {},
            "Mode": "Auto",
            "Targets": [],
            "ResolvedTargets": {
                "ParameterValues": [],
                "Truncated": false
            },
            "AutomationType": "Local"
        }
    ]
}
```

Weitere Informationen finden Sie unter Ausführen eines einfachen Automation-Workflows im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAutomationExecutionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle aktiven und beendeten Automation-Ausführungen beschrieben, die mit Ihrem Konto verknüpft sind.

```
Get-SSMAutomationExecutionList
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Beispiel 2: In diesem Beispiel werden die Ausführungs-ID, das Dokument und der start/end Ausführungszeitstempel für Ausführungen angezeigt, bei denen es sich nicht um „Success“ AutomationExecutionStatus handelt

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Ausgabe:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
```

```
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeAutomationExecutionsV4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle aktiven und beendeten Automation-Ausführungen beschrieben, die mit Ihrem Konto verknüpft sind.

```
Get-SSMAutomationExecutionList
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Beispiel 2: In diesem Beispiel werden die Ausführungs-ID, das Dokument und der start/end Ausführungszeitstempel für Ausführungen angezeigt, bei denen es sich nicht um „Success“ handelt AutomationExecutionStatus

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Ausgabe:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
```

```
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi
 Failed                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (DescribeAutomationExecutionsV5).

Verwendung von **`DescribeAutomationStepExecutions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAutomationStepExecutions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie alle Schritte für eine Automatisierungsausführung beschreiben

Im folgenden `describe-automation-step-executions`-Beispiel werden Details zu den Schritten einer Automation-Ausführung gezeigt.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Ausgabe:

```
{
    "StepExecutions": [
        {
            "StepName": "startInstances",
            "Action": "aws:changeInstanceState",
            "ExecutionStartTime": 1583737234.134,
            "ExecutionEndTime": 1583737234.672,
            "StepStatus": "Success",
            "Inputs": {
                "DesiredState": "\"running\"",
                "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
            },
            "Outputs": {
```

```
                "InstanceStates": [
                    "running"
                ]
            },
            "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
            "OverriddenParameters": {}
        }
    ]
}
```

Beispiel 2: So können Sie einen bestimmten Schritt für eine Automatisierungsausführung beschreiben

Im folgenden `describe-automation-step-executions`-Beispiel werden Details zu einem bestimmten Schritt einer Automation-Ausführung gezeigt.

```
aws ssm describe-automation-step-executions \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE \
    --filters Key=StepExecutionId,Values=95e70479-cf20-4d80-8018-7e4e2EXAMPLE
```

Weitere Informationen finden Sie unter Schrittweises Ausführen eines Automation-Workflows (Befehlszeile) im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAutomationStepExecutionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen über alle aktiven und beendeten Schrittausführungen in einem Automation-Workflow angezeigt.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Ausgabe:

```
StepName                  Action                  StepStatus
--------                  ------                  ----------
LaunchInstance            aws:runInstances        Success
```

```
OSCompatibilityCheck      aws:runCommand          Success
RunPreUpdateScript        aws:runCommand          Success
UpdateEC2Config           aws:runCommand          Cancelled
UpdateSSMAgent            aws:runCommand          Pending
UpdateAWSPVDriver         aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe             aws:runCommand          Pending
InstallWindowsUpdates     aws:runCommand          Pending
RunPostUpdateScript       aws:runCommand          Pending
RunSysprepGeneralize      aws:runCommand          Pending
StopInstance              aws:changeInstanceState Pending
CreateImage               aws:createImage         Pending
TerminateInstance         aws:changeInstanceState Pending
```

- Einzelheiten zur API finden Sie unter DescribeAutomationStepExecutions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen über alle aktiven und beendeten Schrittausführungen in einem Automation-Workflow angezeigt.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Ausgabe:

```
StepName                  Action                  StepStatus
--------                  ------                  ----------
LaunchInstance            aws:runInstances        Success
OSCompatibilityCheck      aws:runCommand          Success
RunPreUpdateScript        aws:runCommand          Success
UpdateEC2Config           aws:runCommand          Cancelled
UpdateSSMAgent            aws:runCommand          Pending
UpdateAWSPVDriver         aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe             aws:runCommand          Pending
InstallWindowsUpdates     aws:runCommand          Pending
RunPostUpdateScript       aws:runCommand          Pending
RunSysprepGeneralize      aws:runCommand          Pending
StopInstance              aws:changeInstanceState Pending
CreateImage               aws:createImage         Pending
TerminateInstance         aws:changeInstanceState Pending
```

- Einzelheiten zur API finden Sie unter DescribeAutomationStepExecutions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeAvailablePatches`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeAvailablePatches` verwendet wird.

CLI

AWS CLI

So erhalten Sie verfügbare Patches

Im folgenden Beispiel `describe-available-patches` werden Details zu allen verfügbaren Patches für Windows Server 2019 abgerufen, die den MSRC-Schweregrad Kritisch haben.

```
aws ssm describe-available-patches \
    --
filters "Key=PRODUCT,Values=WindowsServer2019" "Key=MSRC_SEVERITY,Values=Critical"
```

Ausgabe:

```
{
    "Patches": [
        {
            "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
            "ReleaseDate": 1544047205.0,
            "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
            "Description": "Install this update to resolve issues in Windows.
 For a complete listing of the issues that are included in this update, see the
 associated Microsoft Knowledge Base article for more information. After you
 install this item, you may have to restart your computer.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4470788",
            "MsrcNumber": "",
            "Language": "All"
```

```
        },
        {
            "Id": "c96115e1-5587-4115-b851-22baa46a3f11",
            "ReleaseDate": 1549994410.0,
            "Title": "2019-02 Security Update for Adobe Flash Player for Windows
 Server 2019 for x64-based Systems (KB4487038)",
            "Description": "A security issue has been identified in a Microsoft
 software product that could affect your system. You can help protect your system
 by installing this update from Microsoft. For a complete listing of the issues
 that are included in this update, see the associated Microsoft Knowledge Base
 article. After you install this update, you may have to restart your system.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4487038",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2019",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4487038",
            "MsrcNumber": "",
            "Language": "All"
        },
        ...
    ]
}
```

So erhalten Sie Details zu einem bestimmten Patch

Im folgenden `describe-available-patches`-Beispiel werden Details zum angegebenen Patch abgerufen.

```
aws ssm describe-available-patches \
    --filters "Key=PATCH_ID,Values=KB4480979"
```

Ausgabe:

```
{
    "Patches": [
        {
            "Id": "680861e3-fb75-432e-818e-d72e5f2be719",
            "ReleaseDate": 1546970408.0,
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
```

```
            "Description": "A security issue has been identified in a Microsoft
 software product that could affect your system. You can help protect your system
 by installing this update from Microsoft. For a complete listing of the issues
 that are included in this update, see the associated Microsoft Knowledge Base
 article. After you install this update, you may have to restart your system.",
            "ContentUrl": "https://support.microsoft.com/en-us/kb/4480979",
            "Vendor": "Microsoft",
            "ProductFamily": "Windows",
            "Product": "WindowsServer2016",
            "Classification": "SecurityUpdates",
            "MsrcSeverity": "Critical",
            "KbNumber": "KB4480979",
            "MsrcNumber": "",
            "Language": "All"
        }
    ]
}
```

Weitere Informationen finden Sie unter So funktionieren Patch-Manager-Operationen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeAvailablePatchesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel ruft alle verfügbaren Patches für Windows Server 2012 ab, die den MSRC-Schweregrad „Kritisch“ aufweisen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Ausgabe:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
```

```
                 over it. You can help protect your system by installing this
 update from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Beispiel 3: In diesem Beispiel werden alle Updates abgerufen, die in den letzten 20 Tagen veröffentlicht wurden und für Produkte gelten, die 2019 entsprechen WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Ausgabe:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
```

```
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Einzelheiten zur API finden Sie unter DescribeAvailablePatches AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel ruft alle verfügbaren Patches für Windows Server 2012 ab, die den MSRC-Schweregrad „Kritisch“ aufweisen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Ausgabe:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                 over it. You can help protect your system by installing this
 update from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Beispiel 3: In diesem Beispiel werden alle Updates abgerufen, die in den letzten 20 Tagen veröffentlicht wurden und für Produkte gelten, die 2019 entsprechen WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20)
 | Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate,
 Product, Title
```

Ausgabe:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash
 Player for Windows Server 2019 for x64-based Systems (KB4493478)
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows
 Server 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- Einzelheiten zur API finden Sie unter DescribeAvailablePatches AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeDocument`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDocument` verwendet wird.

CLI

AWS CLI

So können Sie Details eines Dokuments anzeigen

Im folgenden `describe-document` Beispiel werden Details zu einem Systems Manager Manager-Dokument in Ihrem AWS Konto angezeigt.

```
aws ssm describe-document \
    --name "Example"
```

Ausgabe:

```
{
    "Document": {
        "Hash":
 "fc2410281f40779e694a8b95975d0f9f316da8a153daa94e3d9921102EXAMPLE",
        "HashType": "Sha256",
        "Name": "Example",
        "Owner": "29884EXAMPLE",
        "CreatedDate": 1583257938.266,
        "Status": "Active",
        "DocumentVersion": "1",
        "Description": "Document Example",
        "Parameters": [
            {
                "Name": "AutomationAssumeRole",
                "Type": "String",
                "Description": "(Required) The ARN of the role that allows
 Automation to perform the actions on your behalf. If no role is specified,
 Systems Manager Automation uses your IAM permissions to execute this document.",
                "DefaultValue": ""
            },
            {
                "Name": "InstanceId",
                "Type": "String",
                "Description": "(Required) The ID of the Amazon EC2 instance.",
                "DefaultValue": ""
            }
        ],
        "PlatformTypes": [
            "Windows",
            "Linux"
        ],
        "DocumentType": "Automation",
        "SchemaVersion": "0.3",
        "LatestVersion": "1",
        "DefaultVersion": "1",
```

```
        "DocumentFormat": "YAML",
        "Tags": []
    }
}
```

Weitere Informationen finden Sie unter Erstellen von Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeDocumentunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu einem Dokument zurückgegeben.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Einzelheiten zur API finden Sie unter DescribeDocument AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu einem Dokument zurückgegeben.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            :
 f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- Einzelheiten zur API finden Sie unter DescribeDocument AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ssm->describedocument( iv_name = iv_name ).
        DATA(lo_document) = lo_result->get_document( ).
        IF lo_document IS BOUND.
          rv_status = lo_document->get_status( ).
          MESSAGE |Document status: { rv_status }| TYPE 'I'.
```

```
        ENDIF.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeDocumentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`DescribeDocumentPermission`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeDocumentPermission` verwendet wird.

CLI

AWS CLI

So können Sie die Berechtigungen für Dokumente beschreiben

Im folgenden `describe-document-permission`-Beispiel werden Berechtigungsdetails zu einem Systems-Manager-Dokument angezeigt, das öffentlich geteilt wird.

```
aws ssm describe-document-permission \
    --name "Example" \
    --permission-type "Share"
```

Ausgabe:

```
{
    "AccountIds": [
        "all"
    ],
    "AccountSharingInfoList": [
        {
            "AccountId": "all",
            "SharedDocumentVersion": "$DEFAULT"
        }
    ]
}
```

Weitere Informationen finden Sie unter Systems-Manager-Dokument teilen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeDocumentPermissionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Einzelheiten zur API finden Sie unter DescribeDocumentPermission AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- Einzelheiten zur API finden Sie unter DescribeDocumentPermission AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeEffectiveInstanceAssociations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEffectiveInstanceAssociations` verwendet wird.

CLI

AWS CLI

Um Details zu den effektiven Verknüpfungen für eine Instance abzurufen

Im folgenden `describe-effective-instance-associations`-Beispiel werden Details zu den effektiven Verknüpfungen für eine Instance abgerufen.

Befehl:

```
aws ssm describe-effective-instance-associations --instance-
id "i-1234567890abcdef0"
```

Ausgabe:

```
{
    "Associations": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "InstanceId": "i-1234567890abcdef0",
            "Content": "{\n    \"schemaVersion\": \"1.2\",\n    \"description\":
 \"Update the Amazon SSM Agent to the latest version or specified version.\",\n
  \"parameters\": {\n        \"version\": {\n            \"default\": \"\",\n
        \"description\": \"(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agent will be updated to the latest version.
\",\n            \"type\": \"String\"\n        },\n        \"allowDowngrade
\": {\n            \"default\": \"false\",\n            \"description\":
 \"(Optional) Allow the Amazon SSM Agent service to be downgraded to an earlier
 version. If set to false, the service can be upgraded to newer versions only
 (default). If set to true, specify the earlier version.\",\n            \"type
\": \"String\",\n            \"allowedValues\": [\n                \"true\",\n
                \"false\"\n            ]\n        }\n    },\n    \"runtimeConfig
\": {\n        \"aws:updateSsmAgent\": {\n            \"properties\": [\n
         {\n                \"agentName\": \"amazon-ssm-agent\",\n
  \"source\": \"https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-
manifest.json\",\n                \"allowDowngrade\": \"{{ allowDowngrade }}\",\n
                \"targetVersion\": \"{{ version }}\"\n                }\n
    ]\n        }\n    }\n}\n",
            "AssociationVersion": "1"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeEffectiveInstanceAssociationsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel beschreibt die effektiven Zuordnungen für eine Instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Ausgabe:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Beispiel 2: In diesem Beispiel wird der Inhalt der effektiven Verknüpfungen für eine Instance angezeigt.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Ausgabe:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
```

```
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
 to false, the service can be upgraded to newer versions only (default). If set
 to true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- Einzelheiten zur API finden Sie unter DescribeEffectiveInstanceAssociations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel beschreibt die effektiven Zuordnungen für eine Instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5
```

Ausgabe:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Beispiel 2: In diesem Beispiel wird der Inhalt der effektiven Verknüpfungen für eine Instance angezeigt.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Ausgabe:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or
 specified version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent
 to install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
 to false, the service can be upgraded to newer versions only (default). If set
 to true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
```

```
            ]
        }
    }
}
```

- Einzelheiten zur API finden Sie unter DescribeEffectiveInstanceAssociations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeEffectivePatchesForPatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeEffectivePatchesForPatchBaseline` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erhalten Sie alle Patches, die durch eine benutzerdefinierte Patch-Baseline definiert sind

Im folgenden `describe-effective-patches-for-patch-baseline` Beispiel werden die Patches zurückgegeben, die durch eine benutzerdefinierte Patch-Baseline im aktuellen AWS Konto definiert sind. Beachten Sie, dass für eine benutzerdefinierte Baseline nur die ID für `--baseline-id` erforderlich ist.

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "pb-08b654cf9b9681f04"
```

Ausgabe:

```
{
    "EffectivePatches": [
        {
            "Patch": {
                "Id": "fe6bd8c2-3752-4c8b-ab3e-1a7ed08767ba",
                "ReleaseDate": 1544047205.0,
                "Title": "2018-11 Update for Windows Server 2019 for x64-based
 Systems (KB4470788)",
                "Description": "Install this update to resolve issues in Windows.
 For a complete listing of the issues that are included in this update, see the
```

```
associated Microsoft Knowledge Base article for more information. After you
install this item, you may have to restart your computer.",
               "ContentUrl": "https://support.microsoft.com/en-us/kb/4470788",
               "Vendor": "Microsoft",
               "ProductFamily": "Windows",
               "Product": "WindowsServer2019",
               "Classification": "SecurityUpdates",
               "MsrcSeverity": "Critical",
               "KbNumber": "KB4470788",
               "MsrcNumber": "",
               "Language": "All"
           },
           "PatchStatus": {
               "DeploymentStatus": "APPROVED",
               "ComplianceLevel": "CRITICAL",
               "ApprovalDate": 1544047205.0
           }
       },
       {
           "Patch": {
               "Id": "915a6b1a-f556-4d83-8f50-b2e75a9a7e58",
               "ReleaseDate": 1549994400.0,
               "Title": "2019-02 Cumulative Update for .NET Framework 3.5 and
4.7.2 for Windows Server 2019 for x64 (KB4483452)",
               "Description": "A security issue has been identified in a
Microsoft software product that could affect your system. You can help protect
your system by installing this update from Microsoft. For a complete listing
of the issues that are included in this update, see the associated Microsoft
Knowledge Base article. After you install this update, you may have to restart
your system.",
               "ContentUrl": "https://support.microsoft.com/en-us/kb/4483452",
               "Vendor": "Microsoft",
               "ProductFamily": "Windows",
               "Product": "WindowsServer2019",
               "Classification": "SecurityUpdates",
               "MsrcSeverity": "Important",
               "KbNumber": "KB4483452",
               "MsrcNumber": "",
               "Language": "All"
           },
           "PatchStatus": {
               "DeploymentStatus": "APPROVED",
               "ComplianceLevel": "CRITICAL",
               "ApprovalDate": 1549994400.0
```

```
            }
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Beispiel 2: Um alle Patches abzurufen, die durch eine AWS verwaltete Patch-Baseline definiert sind

Im folgenden `describe-effective-patches-for-patch-baseline` Beispiel werden die durch eine AWS verwaltete Patch-Baseline definierten Patches zurückgegeben. Beachten Sie, dass für eine AWS verwaltete Baseline der vollständige Baseline-ARN erforderlich ist für `--baseline-id`

```
aws ssm describe-effective-patches-for-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter So werden Sicherheitspatches ausgewählt im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeEffectivePatchesForPatchBaselinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 aufgeführt.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Ausgabe:

```
Patch                                       PatchStatus
-----                                       -----------
```

```
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Beispiel 2: In diesem Beispiel wird der Patchstatus für alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 angezeigt.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Ausgabe:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Einzelheiten zur API finden Sie unter DescribeEffectivePatchesForPatchBaseline AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 aufgeführt.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Ausgabe:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Beispiel 2: In diesem Beispiel wird der Patchstatus für alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 angezeigt.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Ausgabe:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- Einzelheiten zur API finden Sie unter DescribeEffectivePatchesForPatchBaseline AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstanceAssociationsStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceAssociationsStatus` verwendet wird.

CLI

AWS CLI

So beschreiben Sie den Status der Zuordnungen einer Instance

Dieses Beispiel zeigt Details zu den Zuordnungen für eine Instance.

Befehl:

```
aws ssm describe-instance-associations-status --instance-id "i-1234567890abcdef0"
```

Ausgabe:

```
{
  "InstanceAssociationStatusInfos": [
      {
          "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
          "Name": "AWS-GatherSoftwareInventory",
          "DocumentVersion": "1",
          "AssociationVersion": "1",
          "InstanceId": "i-1234567890abcdef0",
          "ExecutionDate": 1550501886.0,
          "Status": "Success",
          "ExecutionSummary": "1 out of 1 plugin processed, 1 success, 0 failed,
 0 timedout, 0 skipped. ",
          "AssociationName": "Inventory-Association"
      },
      {
          "AssociationId": "5c5a31f6-6dae-46f9-944c-0123456789ab",
```

```
            "Name": "AWS-UpdateSSMAgent",
            "DocumentVersion": "1",
            "AssociationVersion": "1",
            "InstanceId": "i-1234567890abcdef0",
            "ExecutionDate": 1550505828.548,
            "Status": "Success",
            "DetailedStatus": "Success",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

- Einzelheiten zur API finden Sie DescribeInstanceAssociationsStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt Details zu den Zuordnungen für eine Instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Ausgabe:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Beispiel 2: In diesem Beispiel wird der Status der Instance-Zuordnung für die angegebene Instance-ID überprüft und außerdem der Ausführungsstatus dieser Zuordnungen angezeigt

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Ausgabe:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAssociationsStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt Details zu den Zuordnungen für eine Instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Ausgabe:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Beispiel 2: In diesem Beispiel wird der Status der Instance-Zuordnung für die angegebene Instance-ID überprüft und außerdem der Ausführungsstatus dieser Zuordnungen angezeigt

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-
Object {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Ausgabe:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
```

```
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- Einzelheiten zur API finden Sie unter DescribeInstanceAssociationsStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstanceInformation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstanceInformation` verwendet wird.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Informationen zu verwalteten Instances

Im folgenden `describe-instance-information`-Beispiel werden Details zu jeder Ihrer verwalteten Instances abgerufen.

```
aws ssm describe-instance-information
```

Beispiel 2: Um Informationen über eine bestimmte verwaltete Instance zu beschreiben

Das folgende `describe-instance-information`-Beispiel zeigt Details der verwalteten Instance `i-028ea792daEXAMPLE`.

```
aws ssm describe-instance-information \
    --filters "Key=InstanceIds,Values=i-028ea792daEXAMPLE"
```

Beispiel 3: So beschreiben Sie Informationen zu verwalteten Instances mit einem bestimmten Tag-Schlüssel

Das folgende `describe-instance-information`-Beispiel zeigt Details für verwaltete Instances, die über den Tag-Schlüssel `DEV` verfügen.

```
aws ssm describe-instance-information \
```

```
    --filters "Key=tag-key,Values=DEV"
```

Ausgabe:

```
{
    "InstanceInformationList": [
        {
            "InstanceId": "i-028ea792daEXAMPLE",
            "PingStatus": "Online",
            "LastPingDateTime": 1582221233.421,
            "AgentVersion": "2.3.842.0",
            "IsLatestVersion": true,
            "PlatformType": "Linux",
            "PlatformName": "SLES",
            "PlatformVersion": "15.1",
            "ResourceType": "EC2Instance",
            "IPAddress": "192.0.2.0",
            "ComputerName": "ip-198.51.100.0.us-east-2.compute.internal",
            "AssociationStatus": "Success",
            "LastAssociationExecutionDate": 1582220806.0,
            "LastSuccessfulAssociationExecutionDate": 1582220806.0,
            "AssociationOverview": {
                "DetailedStatus": "Success",
                "InstanceAssociationStatusAggregatedCount": {
                    "Success": 2
                }
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwaltete Instances im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeInstanceInformationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel zeigt Details zu jeder Ihrer Instances.

```
Get-SSMInstanceInformation
```

Ausgabe:

```
ActivationId                          :
AgentVersion                          : 2.0.672.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                               :
InstanceId                            : i-0cb2b964d3e14fd9f
IPAddress                             : 172.31.44.222
IsLatestVersion                       : True
LastAssociationExecutionDate          : 2/24/2017 3:18:09 AM
LastPingDateTime                      : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                  :
PingStatus                            : ConnectionLost
PlatformName                          : Amazon Linux AMI
PlatformType                          : Linux
PlatformVersion                       : 2016.09
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance
```

Beispiel 2: Dieses Beispiel zeigt, wie der Parameter -Filter verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in der Region **`us-east-1`** mit dem Wert **`AgentVersion`** von **`2.2.800.0`** zu filtern. Eine Liste der gültigen -Filter-Schlüsselwerte finden Sie im InstanceInformation API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Ausgabe:

```
ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEb0792d98ce
IPAddress                              : 10.0.0.01
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.02
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Beispiel 3: Dieses Beispiel zeigt, wie der InstanceInformationFilterList Parameter - verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in **`PlatformTypes`** der Region **`us-east-1`** mit **`Windows`** oder zu filtern**`Linux`**. Eine Liste der gültigen InstanceInformationFilterList Schlüsselwerte finden Sie im InstanceInformationFilter API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Ausgabe:

```
ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                               :
InstanceId                            : i-EXAMPLEb0792d98ce
IPAddress                             : 10.0.0.27
IsLatestVersion                       : False
LastAssociationExecutionDate          : 8/16/2018 12:02:50 AM
LastPingDateTime                      : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                  :
PingStatus                            : Online
PlatformName                          : Ubuntu Server 18.04 LTS
PlatformType                          : Linux
PlatformVersion                       : 18.04
RegistrationDate                      : 1/1/0001 12:00:00 AM
ResourceType                          : EC2Instance

ActivationId                          :
AgentVersion                          : 2.2.800.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
```

```
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.100
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 4: In diesem Beispiel werden von SSM verwaltete Instanzen und Exporte InstanceId LastPingDateTime sowie PlatformName in eine CSV-Datei aufgeführt. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- Einzelheiten zur API finden Sie unter DescribeInstanceInformation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel zeigt Details zu jeder Ihrer Instances.

```
Get-SSMInstanceInformation
```

Ausgabe:

```
ActivationId                            :
AgentVersion                            : 2.0.672.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : ip-172-31-44-222.us-
west-2.compute.internal
IamRole                                 :
```

```
InstanceId                              : i-0cb2b964d3e14fd9f
IPAddress                               : 172.31.44.222
IsLatestVersion                         : True
LastAssociationExecutionDate            : 2/24/2017 3:18:09 AM
LastPingDateTime                        : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                    :
PingStatus                              : ConnectionLost
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 2: Dieses Beispiel zeigt, wie der Parameter -Filter verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in der Region **`us-east-1`** mit dem Wert **`AgentVersion`** von **`2.2.800.0`** zu filtern. Eine Liste der gültigen -Filter-Schlüsselwerte finden Sie im InstanceInformation API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Ausgabe:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.01
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
```

```
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.02
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 3: Dieses Beispiel zeigt, wie der InstanceInformationFilterList Parameter - verwendet wird, um Ergebnisse nur nach den AWS Systems Manager Manager-Instanzen in **`PlatformTypes`** der Region **`us-east-1`** mit **`Windows`** oder zu filtern**`Linux`**. Eine Liste der gültigen InstanceInformationFilterList Schlüsselwerte finden Sie im InstanceInformationFilter API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList
 $Filters
```

Ausgabe:

```
ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEb0792d98ce
IPAddress                              : 10.0.0.27
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:02:50 AM
LastPingDateTime                       : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Ubuntu Server 18.04 LTS
PlatformType                           : Linux
PlatformVersion                        : 18.04
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance

ActivationId                           :
AgentVersion                           : 2.2.800.0
AssociationOverview                    :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                      : Success
ComputerName                           : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                :
InstanceId                             : i-EXAMPLEac7501d023
IPAddress                              : 10.0.0.100
IsLatestVersion                        : False
LastAssociationExecutionDate           : 8/16/2018 12:00:20 AM
LastPingDateTime                       : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                   :
PingStatus                             : Online
PlatformName                           : Microsoft Windows Server 2016 Datacenter
PlatformType                           : Windows
PlatformVersion                        : 10.0.14393
RegistrationDate                       : 1/1/0001 12:00:00 AM
ResourceType                           : EC2Instance
```

Beispiel 4: In diesem Beispiel werden von SSM verwaltete Instanzen und Exporte InstanceId LastPingDateTime sowie PlatformName in eine CSV-Datei aufgeführt. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus,
 LastPingDateTime, PlatformName | Export-Csv Instance-details.csv -
NoTypeInformation
```

- Einzelheiten zur API finden Sie unter DescribeInstanceInformation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstancePatchStates`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstancePatchStates` verwendet wird.

CLI

AWS CLI

Um die Status der Patch-Zusammenfassung für Instances abzurufen

In diesem `describe-instance-patch-states`-Beispiel werden die Status der Patch-Zusammenfassung für eine Instance abgerufen.

```
aws ssm describe-instance-patch-states \
    --instance-ids "i-1234567890abcdef0"
```

Ausgabe:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PatchGroup": "my-patch-group",
            "BaselineId": "pb-0713accee01234567",
            "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
            "CriticalNonCompliantCount": 2,
            "SecurityNonCompliantCount": 2,
            "OtherNonCompliantCount": 1,
            "InstalledCount": 123,
            "InstalledOtherCount": 334,
            "InstalledPendingRebootCount": 0,
```

```
            "InstalledRejectedCount": 0,
            "MissingCount": 1,
            "FailedCount": 2,
            "UnreportedNotApplicableCount": 11,
            "NotApplicableCount": 2063,
            "OperationStartTime": "2021-05-03T11:00:56-07:00",
            "OperationEndTime": "2021-05-03T11:01:09-07:00",
            "Operation": "Scan",
            "LastNoRebootInstallOperationTime": "2020-06-14T12:17:41-07:00",
            "RebootOption": "RebootIfNeeded"
        }
    ]
}
```

Weitere Informationen finden Sie unter Über Patch-Compliance im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeInstancePatchStatesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Status der Patch-Zusammenfassung für eine Instance abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Beispiel 2: In diesem Beispiel werden die Status der Patch-Zusammenfassung für zwei Instances abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStates AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Status der Patch-Zusammenfassung für eine Instance abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Beispiel 2: In diesem Beispiel werden die Status der Patch-Zusammenfassung für zwei Instances abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStates AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstancePatchStatesForPatchGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstancePatchStatesForPatchGroup` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um die Instance-Status für eine Patchgruppe abzurufen

Im folgenden `describe-instance-patch-states-for-patch-group`-Beispiel werden Details zu den Status der Patchzusammenfassung pro Instance für die angegebene Patchgruppe abgerufen.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --patch-group "Production"
```

Ausgabe:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
```

```
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        },
        {
            "InstanceId": "i-0471e04240EXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-09ca3fb51fEXAMPLE",
            "SnapshotId": "05d8ffb0-1bbe-4812-ba2d-d9b7bEXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 0,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 2671,
            "NotApplicableCount": 400,
            "OperationStartTime": "2021-08-04T22:06:20.340000-07:00",
            "OperationEndTime": "2021-08-04T22:07:11.220000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```

Beispiel 2: Um den Instance-Status für eine Patch-Gruppe mit mehr als fünf fehlenden Patches abzurufen

Das folgende `describe-instance-patch-states-for-patch-group`-Beispiel ruft Details zu den Patch-Zusammenfassungszuständen für die angegebene Patch-Gruppe für Instances mit mehr als fünf fehlenden Patches ab.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=MissingCount,Type=GreaterThan,Values=5 \
    --patch-group "Production"
```

Ausgabe:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "PatchGroup": "Production",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "OwnerInformation": "",
            "InstalledCount": 46,
            "InstalledOtherCount": 4,
            "InstalledPendingRebootCount": 1,
            "InstalledRejectedCount": 1,
            "MissingCount": 7,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 232,
            "NotApplicableCount": 654,
            "OperationStartTime": "2021-08-04T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-04T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 1
        }
    ]
}
```

Beispiel 3: Um den Instance-Status für eine Patchgruppe mit weniger als zehn Instances abzurufen, für die ein Neustart erforderlich ist

Im folgenden `describe-instance-patch-states-for-patch-group`-Beispiel werden Details zum Status der Patch-Zusammenfassung für die angegebene Patchgruppe für Instances mit weniger als zehn Instances abgerufen, die einen Neustart erfordern.

```
aws ssm describe-instance-patch-states-for-patch-group \
    --filters Key=InstalledPendingRebootCount,Type=LessThan,Values=10 \
    --patch-group "Production"
```

Ausgabe:

```
{
    "InstancePatchStates": [
        {
            "InstanceId": "i-02573cafcfEXAMPLE",
            "BaselineId": "pb-0c10e65780EXAMPLE",
            "SnapshotId": "a3f5ff34-9bc4-4d2c-a665-4d1c1EXAMPLE",
            "PatchGroup": "Production",
            "OwnerInformation": "",
            "InstalledCount": 32,
            "InstalledOtherCount": 1,
            "InstalledPendingRebootCount": 4,
            "InstalledRejectedCount": 0,
            "MissingCount": 2,
            "FailedCount": 0,
            "UnreportedNotApplicableCount": 846,
            "NotApplicableCount": 212,
            "OperationStartTime": "2021-08-046T11:03:50.590000-07:00",
            "OperationEndTime": "2021-08-06T11:04:21.555000-07:00",
            "Operation": "Scan",
            "RebootOption": "NoReboot",
            "CriticalNonCompliantCount": 0,
            "SecurityNonCompliantCount": 1,
            "OtherNonCompliantCount": 0
        }
    ]
}
```

Weitere Informationen finden Sie unter Grundlegendes zu den Werten für den Patch-Kompatibilitätsstatus im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeInstancePatchStatesForPatchGroupin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden für eine Patch-Gruppe eine Übersicht über den Patch-Zustand auf der Ebene einzelner Instances abgerufen.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStatesForPatchGroup AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden für eine Patch-Gruppe eine Übersicht über den Patch-Zustand auf der Ebene einzelner Instances abgerufen.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatchStatesForPatchGroup AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeInstancePatches`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeInstancePatches` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um die Details zum Patch-Status für eine Instance abzurufen

Das folgende `describe-instance-patches`-Beispiel ruft Details zu den Patches für die angegebene Instance ab.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0"
```

Ausgabe:

```
{
```

```
    "Patches": [
        {
            "Title": "2019-01 Security Update for Adobe Flash Player for Windows
 Server 2016 for x64-based Systems (KB4480979)",
            "KBId": "KB4480979",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2019-01-09T00:00:00+00:00"
        },
        {
            "Title": "",
            "KBId": "KB4481031",
            "Classification": "",
            "Severity": "",
            "State": "InstalledOther",
            "InstalledTime": "2019-02-08T00:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Beispiel 2: Um eine Liste mit Patches mit dem Status Fehlend für eine Instance abzurufen

Im folgenden `describe-instance-patches`-Beispiel werden Informationen über Patches abgerufen, die sich für die angegebene Instance im Status Missing befinden.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Missing
```

Ausgabe:

```
{
    "Patches": [
        {
            "Title": "Windows Malicious Software Removal Tool x64 - February 2019
 (KB890830)",
            "KBId": "KB890830",
            "Classification": "UpdateRollups",
            "Severity": "Unspecified",
            "State": "Missing",
```

```
            "InstalledTime": "1970-01-01T00:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

Weitere Informationen finden Sie unter Über Patch-Compliance-Status im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 3: Um eine Liste der Patches abzurufen, die seit einem bestimmten Zeitpunkt InstalledTime für eine Instanz installiert wurden

Im folgenden Beispiel für `describe-instance-patches` werden Informationen über Patches abgerufen, die seit einem bestimmten Zeitpunkt für die angegebene Instance installiert wurden, indem die Verwendung von `--filters` und `--query` kombiniert wird.

```
aws ssm describe-instance-patches \
    --instance-id "i-1234567890abcdef0" \
    --filters Key=State,Values=Installed \
    --query "Patches[?InstalledTime >= `2023-01-01T16:00:00`]"
```

Ausgabe:

```
{
    "Patches": [
        {
            "Title": "2023-03 Cumulative Update for Windows Server 2019 (1809)
 for x64-based Systems (KB5023702)",
            "KBId": "KB5023702",
            "Classification": "SecurityUpdates",
            "Severity": "Critical",
            "State": "Installed",
            "InstalledTime": "2023-03-16T11:00:00+00:00"
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

- Einzelheiten zur API finden Sie DescribeInstancePatchesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details zur Patch-Konformität für eine Instance abgerufen.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatches AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details zur Patch-Konformität für eine Instance abgerufen.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- Einzelheiten zur API finden Sie unter DescribeInstancePatches AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindowExecutionTaskInvocations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindowExecutionTaskInvocations` verwendet wird.

CLI

AWS CLI

So können Sie die spezifischen Aufgabenaufrufen für die Ausführung einer Aufgabe in einem Wartungsfenster ausführen

Im folgenden Beispiel `describe-maintenance-window-execution-task-invocations` werden die Aufrufe für die angegebene Aufgabe aufgeführt, die im Rahmen der Ausführung des angegebenen Wartungsfensters ausgeführt wurden.

```
aws ssm describe-maintenance-window-execution-task-invocations \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2a638355" \
    --task-id "ac0c6ae1-daa3-4a89-832e-d384503b6586"
```

Ausgabe:

```
{
    "WindowExecutionTaskInvocationIdentities": [
        {
            "Status": "SUCCESS",
            "Parameters": "{\"documentName\":\"AWS-RunShellScript\",
\"instanceIds\":[\"i-0000293ffd8c57862\"],\"parameters\":{\"commands\":[\"df\"]},
\"maxConcurrency\":\"1\",\"maxErrors\":\"1\"}",
            "InvocationId": "e274b6e1-fe56-4e32-bd2a-8073c6381d8b",
            "StartTime": 1487692834.723,
            "EndTime": 1487692834.871,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2a638355",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d384503b6586"
        }
    ]
}
```

Weitere Informationen finden Sie unter Informationen zu Aufgaben und Aufgabenausführungen (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTaskInvocations AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet die Aufrufe einer Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
```

```
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTaskInvocations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet die Aufrufe einer Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTaskInvocations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindowExecutionTasks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindowExecutionTasks` verwendet wird.

CLI

AWS CLI

So können Sie alle Aufgaben auflisten, die mit der Ausführung eines Wartungsfensters verbunden sind

Im folgenden `ssm describe-maintenance-window-execution-tasks`-Beispiel werden die Aufgaben aufgeführt, die mit der Ausführung des angegebenen Wartungsfensters verknüpft sind.

```
aws ssm describe-maintenance-window-execution-tasks \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Ausgabe:

```
{
    "WindowExecutionTaskIdentities": [
        {
            "Status": "SUCCESS",
            "TaskArn": "AWS-RunShellScript",
            "StartTime": 1487692834.684,
            "TaskType": "RUN_COMMAND",
            "EndTime": 1487692835.005,
            "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
            "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
        }
    ]
}
```

Weitere Informationen finden Sie unter Informationen zu Aufgaben und Aufgabenausführungen (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTasks AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet die Aufgaben auf, die mit der Ausführung eines Wartungsfensters verbunden sind.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTasks AWS - Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet die Aufgaben auf, die mit der Ausführung eines Wartungsfensters verbunden sind.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutionTasks AWS - Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindowExecutions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindowExecutions` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Ausführungen für ein Wartungsfenster auf

Das folgende `describe-maintenance-window-executions`-Beispiel listet alle Ausführungen für das angegebene Wartungsfenster auf.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE"
```

Ausgabe:

```
{
    "WindowExecutions": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
            "Status": "IN_PROGRESS",
            "StartTime": "2021-08-04T11:00:00.000000-07:00"

        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "ff75b750-4834-4377-8f61-b3cadEXAMPLE",
            "Status": "SUCCESS",
            "StartTime": "2021-08-03T11:00:00.000000-07:00",
            "EndTime": "2021-08-03T11:37:21.450000-07:00"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
            "Status": "FAILED",
```

```
            "StatusDetails": "One or more tasks in the orchestration failed.",
            "StartTime": "2021-08-02T11:00:00.000000-07:00",
            "EndTime": "2021-08-02T11:22:36.190000-07:00"
        }
    ]
}
```

Beispiel 2: So listen Sie alle Ausführungen für ein Wartungsfenster vor einem bestimmten Datum auf

Im folgenden `describe-maintenance-window-executions`-Beispiel werden alle Ausführungen für das angegebene Wartungsfenster vor dem angegebenen Datum aufgeführt.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=ExecutedBefore,Values=2021-08-03T00:00:00Z"
```

Ausgabe:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "9fac7dd9-ff21-42a5-96ad-bbc4bEXAMPLE",
        "Status": "FAILED",
        "StatusDetails": "One or more tasks in the orchestration failed.",
        "StartTime": "2021-08-02T11:00:00.000000-07:00",
        "EndTime": "2021-08-02T11:22:36.190000-07:00"
    }
    ]
}
```

Beispiel 3: So listen Sie alle Ausführungen für ein Wartungsfenster nach einem bestimmten Datum auf

Im folgenden `describe-maintenance-window-executions`-Beispiel werden alle Ausführungen für das angegebene Wartungsfenster nach dem angegebenen Datum aufgeführt.

```
aws ssm describe-maintenance-window-executions \
    --window-id "mw-ab12cd34eEXAMPLE" \
```

```
    --filters "Key=ExecutedAfter,Values=2021-08-04T00:00:00Z"
```

Ausgabe:

```
{
    "WindowExecutions": [
        {
        "WindowId": "mw-ab12cd34eEXAMPLE",
        "WindowExecutionId": "6027b513-64fe-4cf0-be7d-1191aEXAMPLE",
        "Status": "IN_PROGRESS",
        "StartTime": "2021-08-04T11:00:00.000000-07:00"
        }
    ]
}
```

Weitere Informationen finden Sie unter Informationen zu Aufgaben und Aufgabenausführungen (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutions AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster vor einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Beispiel 3: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster nach einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster vor einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Beispiel 3: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster nach einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
```

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowExecutions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindowTargets`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindowTargets` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Ziele für ein Wartungsfenster auf

Das folgende `describe-maintenance-window-targets`-Beispiel listet alle Ziele für das angegebene Wartungsfenster auf.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Ausgabe:

```
{
    "Targets": [
        {
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Single instance",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "350d44e6-28cc-44e2-951f-4b2c9EXAMPLE"
        },
        {
```

```
            "ResourceType": "INSTANCE",
            "OwnerInformation": "Two instances in a list",
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-0000293ffdEXAMPLE",
                        "i-0cb2b964d3EXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "e078a987-2866-47be-bedd-d9cf4EXAMPLE"
        }
    ]
}
```

Beispiel 2: So listen Sie alle Ziele für ein Wartungsfenster auf, die einem bestimmten Eigentümerinformationswert entsprechen

In diesem `describe-maintenance-window-targets`-Beispiel werden alle Ziele für ein Wartungsfenster aufgeführt.

```
aws ssm describe-maintenance-window-targets \
    --window-id "mw-0ecb1226ddEXAMPLE" \
    --filters "Key=OwnerInformation,Values=CostCenter1"
```

Ausgabe:

```
{
    "Targets": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "WindowTargetId": "da89dcc3-7f9c-481d-ba2b-edcb7d0057f9",
            "ResourceType": "INSTANCE",
            "Targets": [
                {
                    "Key": "tag:Environment",
                    "Values": [
                        "Prod"
                    ]
                }
            ],
```

```
            "OwnerInformation": "CostCenter1",
            "Name": "ProdTarget1"
        }
    ]
}
```

Weitere Informationen finden Sie unter Informationen über Maintenance Windows (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeMaintenanceWindowTargetsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Ziele für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTargets AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Ziele für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTargets AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindowTasks`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindowTasks` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Aufgaben für ein Wartungsfenster auf

Das folgende `describe-maintenance-window-tasks`-Beispiel listet alle Aufgaben für das angegebene Wartungsfenster auf.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-06cf17cbefEXAMPLE"
```

Ausgabe:

```
{
    "Tasks": [
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "018b31c3-2d77-4b9e-bd48-c91edEXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "TaskParameters": {},
```

```
            "Type": "AUTOMATION",
            "Description": "Restarting EC2 Instance for maintenance",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "My-Automation-Example-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        },
        {
            "WindowId": "mw-06cf17cbefEXAMPLE",
            "WindowTaskId": "1943dee0-0a17-4978-9bf4-3cc2fEXAMPLE",
            "TaskArn": "AWS-DisableS3BucketPublicReadWrite",
            "TaskParameters": {},
            "Type": "AUTOMATION",
            "Description": "Automation task to disable read/write access on
 public S3 buckets",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Disable-S3-Public-Read-Write-Access-Automation-Task",
            "Priority": 0,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ]
        }
    ]
}
```

Beispiel 2: Um alle Aufgaben für ein Wartungsfenster aufzulisten, das das RunPowerShellScript Befehlsdokument AWS- aufruft

Das folgende `describe-maintenance-window-tasks`-Beispiel listet alle Aufgaben für das angegebene `AWS-RunPowerShellScript`-Wartungsfenster auf.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

Ausgabe:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 1,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyTask"
        }
    ]
}
```

Beispiel 3: So listen Sie alle Aufgaben für ein Wartungsfenster auf, die eine Priorität von 3 haben

Das folgende `describe-maintenance-window-tasks`-Beispiel listet alle Aufgaben für das angegebene Wartungsfenster auf, die eine `Priority` von 3 haben.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=3"
```

Ausgabe:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        },
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "ee45feff-ad65-4a6c-b478-5cab8EXAMPLE",
            "TaskArn": "AWS-RestartEC2Instance",
            "Type": "AUTOMATION",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
```

```
                }
            ],
            "TaskParameters": {},
            "Priority": 3,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "10",
            "MaxErrors": "5",
            "Name": "My-Automation-Task",
            "Description": "A description for my Automation task"
        }
    ]
}
```

Beispiel 4: So listen Sie alle Aufgaben für ein Wartungsfenster auf, die eine Priorität von 1 haben, und verwenden Sie Run Command

In diesem `describe-maintenance-window-tasks`-Beispiel werden alle Aufgaben für das angegebene Wartungsfenster aufgeführt, die einen `Priority` von 1 und `Run Command` nutzen.

```
aws ssm describe-maintenance-window-tasks \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

Ausgabe:

```
{
    "Tasks": [
        {
            "WindowId": "mw-ab12cd34eEXAMPLE",
            "WindowTaskId": "0d36e6b4-3a4f-411e-adcb-3558eEXAMPLE",
            "TaskArn": "AWS-RunPowerShellScript",
            "Type": "RUN_COMMAND",
            "Targets": [
                {
                    "Key": "WindowTargetIds",
                    "Values": [
                        "da89dcc3-7f9c-481d-ba2b-edcb7EXAMPLE"
                    ]
                }
            ],
```

```
            "TaskParameters": {},
            "Priority": 1,
            "ServiceRoleArn": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "MaxConcurrency": "1",
            "MaxErrors": "1",
            "Name": "MyRunCommandTask"
        }
    ]
}
```

Weitere Informationen finden Sie unter Informationen zu Wartungsfenstern (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeMaintenanceWindowTasksunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Aufgaben für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTasks AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Aufgaben für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindowTasks AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeMaintenanceWindows`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeMaintenanceWindows` verwendet wird.

CLI

AWS CLI

Beispiel 1: So listen Sie alle Wartungsfenster auf

Das folgende `describe-maintenance-windows` Beispiel listet alle Wartungsfenster in Ihrem AWS Konto in der aktuellen Region auf.

```
aws ssm describe-maintenance-windows
```

Ausgabe:

```
{
```

```
    "WindowIdentities": [
        {
            "WindowId": "mw-0ecb1226ddEXAMPLE",
            "Name": "MyMaintenanceWindow-1",
            "Enabled": true,
            "Duration": 2,
            "Cutoff": 1,
            "Schedule": "rate(180 minutes)",
            "NextExecutionTime": "2020-02-12T23:19:20.596Z"
        },
        {
            "WindowId": "mw-03eb9db428EXAMPLE",
            "Name": "MyMaintenanceWindow-2",
            "Enabled": true,
            "Duration": 3,
            "Cutoff": 1,
            "Schedule": "rate(7 days)",
            "NextExecutionTime": "2020-02-17T23:22:00.956Z"
        },
    ]
}
```

Beispiel 2: So listen Sie alle aktivierten Wartungsfenster auf

Das folgende `describe-maintenance-windows`-Beispiel listet alle aktivierten Wartungsfenster auf.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Enabled,Values=true"
```

Beispiel 3: So listen Sie Wartungsfenster auf, die einem bestimmten Namen entsprechen

In diesem `describe-maintenance-windows`-Beispiel werden alle Wartungsfenster mit dem angegebenen Namen aufgeführt.

```
aws ssm describe-maintenance-windows \
    --filters "Key=Name,Values=MyMaintenanceWindow"
```

Weitere Informationen finden Sie unter Informationen über Maintenance Windows (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeMaintenanceWindowsunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Wartungsfenster in Ihrem Konto aufgeführt.

```
Get-SSMMaintenanceWindowList
```

Ausgabe:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindows AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Wartungsfenster in Ihrem Konto aufgeführt.

```
Get-SSMMaintenanceWindowList
```

Ausgabe:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- Einzelheiten zur API finden Sie unter DescribeMaintenanceWindows AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`DescribeOpsItems`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeOpsItems` verwendet wird.

CLI

AWS CLI

Um eine Reihe von aufzulisten OpsItems

Im folgenden `describe-ops-items` Beispiel wird eine Liste aller offenen Konten OpsItems in Ihrem AWS Konto angezeigt.

```
aws ssm describe-ops-items \
    --ops-item-filters "Key=Status,Values=Open,Operator=Equal"
```

Ausgabe:

```
{
    "OpsItemSummaries": [
        {
            "CreatedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-03-14T17:02:46.375000-07:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-03-14T17:02:46.375000-07:00",
            "Source": "SSM",
            "Status": "Open",
            "OpsItemId": "oi-7cfc5EXAMPLE",
            "Title": "SSM Maintenance Window execution failed",
            "OperationalData": {
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-SSM-maintenance-
window-execution-failed\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
                    "Value": "[{\"arn\":\"arn:aws:ssm:us-
east-2:111222333444:maintenancewindow/mw-034093d322EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        },
        {
```

```
            "CreatedBy": "arn:aws:sts::1112223233444:assumed-role/OpsItem-CWE-
Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "CreatedTime": "2020-02-26T11:43:15.426000-08:00",
            "LastModifiedBy": "arn:aws:sts::111222333444:assumed-role/OpsItem-
CWE-Role/fbf77cbe264a33509569f23e4EXAMPLE",
            "LastModifiedTime": "2020-02-26T11:43:15.426000-08:00",
            "Source": "EC2",
            "Status": "Open",
            "OpsItemId": "oi-6f966EXAMPLE",
            "Title": "EC2 instance stopped",
            "OperationalData": {
                "/aws/automations": {
                    "Value": "[ { \"automationType\": \"AWS:SSM:Automation\",
 \"automationId\": \"AWS-RestartEC2Instance\" } ]",
                    "Type": "SearchableString"
                },
                "/aws/dedup": {
                    "Value": "{\"dedupString\":\"SSMOpsItems-EC2-instance-stopped
\"}",
                    "Type": "SearchableString"
                },
                "/aws/resources": {
                    "Value": "[{\"arn\":\"arn:aws:ec2:us-
east-2:111222333444:instance/i-0beccfbc02EXAMPLE\"}]",
                    "Type": "SearchableString"
                }
            },
            "Category": "Availability",
            "Severity": "3"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit OpsItems im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribeOpsItemsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Describes AWS SSM OpsItems asynchronously.
    *
    * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
    *
    * This method initiates an asynchronous request to describe SSM OpsItems.
    * If the request is successful, it prints the title and status of each
OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void describeOpsItems(String key) {
       OpsItemFilter filter = OpsItemFilter.builder()
               .key(OpsItemFilterKey.OPS_ITEM_ID)
               .values(key)
               .operator(OpsItemFilterOperator.EQUAL)
               .build();

       DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
               .maxResults(10)
               .opsItemFilters(filter)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().describeOpsItems(itemsRequest)
                   .thenAccept(itemsResponse -> {
                       List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                       for (OpsItemSummary item : items) {
                           System.out.println("The item title is " +
item.title() + " and the status is " + item.status().toString());
                       }
                   })
```

```
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Einzelheiten zur API finden Sie DescribeOpsItemsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  OpsItemFilterOperator,
  OpsItemFilterKey,
  paginateDescribeOpsItems,
```

```
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Describe SSM OpsItems.
 * @param {{ opsItemId: string }}
 */
export const main = async ({ opsItemId }) => {
  const client = new SSMClient({});
  try {
    const describeOpsItemsPaginated = [];
    for await (const page of paginateDescribeOpsItems(
      { client },
      {
        OpsItemFilters: {
          Key: OpsItemFilterKey.OPSITEM_ID,
          Operator: OpsItemFilterOperator.EQUAL,
          Values: opsItemId,
        },
      },
    )) {
      describeOpsItemsPaginated.push(...page.OpsItemSummaries);
    }
    console.log("Here are the ops items:");
    console.log(describeOpsItemsPaginated);
    return { OpsItemSummaries: describeOpsItemsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie DescribeOpsItemsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator":
 "Equal"}
                ]
            ):
```

```
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
 {item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie DescribeOpsItemsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create filter for OpsItem ID
        DATA(lt_filters) = VALUE /aws1/cl_ssmopsitemfilter=>tt_opsitemfilters(
          ( NEW /aws1/cl_ssmopsitemfilter(
              iv_key = 'OpsItemId'
              it_values = VALUE /aws1/
cl_ssmopsitemfiltvals_w=>tt_opsitemfiltervalues(
                ( NEW /aws1/cl_ssmopsitemfiltvals_w( iv_value =
 iv_ops_item_id ) )
```

```
              )
              iv_operator = 'Equal'
            ) )
        ).

        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->describeopsitems(
          it_opsitemfilters = lt_filters ).

        rv_found = abap_false.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmdescropsitemsrsp( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_opsitemsummaries( ) INTO DATA(lo_item).
            DATA(lv_title) = lo_item->get_title( ).
            DATA(lv_status) = lo_item->get_status( ).
            MESSAGE |The OpsItem title is { lv_title } and the status is
 { lv_status }| TYPE 'I'.
            rv_found = abap_true.
          ENDLOOP.
        ENDWHILE.
      CATCH /aws1/cx_ssminternalservererr.
        MESSAGE 'Internal server error occurred.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeOpsItemsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DescribeParameters`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeParameters` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie alle Parameter auflisten

Das folgende `describe-parameters` Beispiel listet alle Parameter im AWS Girokonto und in der Region auf.

```
aws ssm describe-parameters
```

Ausgabe:

```
{
    "Parameters": [
        {
            "Name": "MySecureStringParameter",
            "Type": "SecureString",
            "KeyId": "alias/aws/ssm",
            "LastModifiedDate": 1582155479.205,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/Admin/
Richard-Roe-Managed",
            "Description": "This is a SecureString parameter",
            "Version": 2,
            "Tier": "Advanced",
            "Policies": [
                {
                    "PolicyText": "{\"Type\":\"Expiration\",\"Version\":\"1.0\",
\"Attributes\":{\"Timestamp\":\"2020-07-07T22:30:00Z\"}}",
                    "PolicyType": "Expiration",
                    "PolicyStatus": "Pending"
                },
                {
                    "PolicyText": "{\"Type\":\"ExpirationNotification\",\"Version
\":\"1.0\",\"Attributes\":{\"Before\":\"12\",\"Unit\":\"Hours\"}}",
                    "PolicyType": "ExpirationNotification",
                    "PolicyStatus": "Pending"
                }
            ]
        },
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
```

```
            "Type": "String",
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Alejandro-
Rosalez",
            "Description": "This is a String parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "latestAmi",
            "Type": "String",
            "LastModifiedDate": 1580862415.521,
            "LastModifiedUser": "arn:aws:sts::111222333444:assumed-role/lambda-
ssm-role/Automation-UpdateSSM-Param",
            "Version": 3,
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

Beispiel 2: Um alle Parameter aufzulisten, die bestimmten Metadaten entsprechen

In diesem `describe-parameters`-Beispiel werden alle Parameter aufgeführt, die einem Filter entsprechen.

aws ssm describe-parameters --filters „Key=Type, Values=“ StringList

Ausgabe:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "LastModifiedDate": 1582154764.222,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is a StringList parameter",
            "Version": 1,
            "Tier": "Standard",
            "Policies": []
        }
    ]
```

```
}
```

Weitere Informationen finden Sie unter Suchen nach Systems-Manager-Parametern im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie in der Befehlsreferenz. DescribeParametersAWS CLI

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DescribeParametersin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet alle Parameter auf.

```
Get-SSMParameterList
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Einzelheiten zur API finden Sie unter DescribeParameters AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet alle Parameter auf.

```
Get-SSMParameterList
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- Einzelheiten zur API finden Sie unter DescribeParameters AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;

    for param in resp.parameters() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- Einzelheiten zur API finden Sie DescribeParametersin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`DescribePatchBaselines`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePatchBaselines` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie alle Patch-Baselines auflisten

Im folgenden `describe-patch-baselines` Beispiel werden Details für alle Patch-Baselines in Ihrem Konto in der aktuellen Region abgerufen.

```
aws ssm describe-patch-baselines
```

Ausgabe:

```
{
    "BaselineIdentities": [
        {
            "BaselineName": "AWS-SuseDefaultPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "Default Patch Baseline for Suse Provided by
 AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0123fdb36e334a3b2",
            "OperatingSystem": "SUSE"
```

```
        },
        {
            "BaselineName": "AWS-DefaultPatchBaseline",
            "DefaultBaseline": false,
            "BaselineDescription": "Default Patch Baseline Provided by AWS.",
            "BaselineId": "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-020d361a05defe4ed",
            "OperatingSystem": "WINDOWS"
        },
        ...
        {
            "BaselineName": "MyWindowsPatchBaseline",
            "DefaultBaseline": true,
            "BaselineDescription": "My patch baseline for EC2 instances for
 Windows Server",
            "BaselineId": "pb-0ad00e0dd7EXAMPLE",
            "OperatingSystem": "WINDOWS"
        }
    ]
}
```

Beispiel 2: Um alle Patch-Baselines aufzulisten, die bereitgestellt werden von AWS

Das folgende `describe-patch-baselines` Beispiel listet alle Patch-Baselines auf, die von bereitgestellt werden. AWS

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[AWS]"
```

Beispiel 3: So können Sie alle Patch-Baselines auflisten, die Ihnen gehören

Im folgenden `describe-patch-baselines`-Beispiel werden alle benutzerdefinierten Patch-Baselines aufgeführt, die in Ihrem Konto in der aktuellen Region erstellt wurden.

```
aws ssm describe-patch-baselines \
    --filters "Key=OWNER,Values=[Self]"
```

Weitere Informationen finden Sie unter Über vordefinierte und benutzerdefinierte Patch-Baselines im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DescribePatchBaselines AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet alle Patch-Baselines auf.

```
Get-SSMPatchBaseline
```

Ausgabe:

```
BaselineDescription                                                BaselineId
                                                        BaselineName
-------------------                                                ----------
                                                        ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Beispiel 2: Dieses Beispiel listet alle Patch-Baselines auf, die von bereitgestellt werden. AWS Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 3: In diesem Beispiel werden alle Patch-Baselines mit Ihnen als Eigentümer aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Ausgabe:

```
BaselineDescription                     BaselineId
                            BaselineName             DefaultBaselin


                                                     e
-------------------                     ----------
                            ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- Einzelheiten zur API finden Sie unter DescribePatchBaselines AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet alle Patch-Baselines auf.

```
Get-SSMPatchBaseline
```

Ausgabe:

```
BaselineDescription                                                BaselineId
                                                        BaselineName
-------------------                                                ----------
                                                        ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
```

```
Baseline containing all updates approved for production systems
 pb-045f10b4f382baeda
 Production-B...
Baseline containing all updates approved for production systems
 pb-0a2f1059b670ebd31
 Production-B...
```

Beispiel 2: Dieses Beispiel listet alle Patch-Baselines auf, die von bereitgestellt werden. AWS Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 3: In diesem Beispiel werden alle Patch-Baselines mit Ihnen als Eigentümer aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen.

```
$filter1 = New-Object
 Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Ausgabe:

```
BaselineDescription                     BaselineId
                              BaselineName             DefaultBaselin
```

```
                                                        e
-------------------                     ----------
                              ------------            --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline
 True
```

- Einzelheiten zur API finden Sie unter DescribePatchBaselines AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribePatchGroupState`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePatchGroupState` verwendet wird.

CLI

AWS CLI

So rufen Sie den Status einer Patchgruppe ab

Im folgenden Beispiel `describe-patch-group-state` wird die allgemeine Zusammenfassung der Patch-Konformität für eine Patchgruppe abgerufen.

```
aws ssm describe-patch-group-state \
    --patch-group "Production"
```

Ausgabe:

```
{
    "Instances": 21,
    "InstancesWithCriticalNonCompliantPatches": 1,
    "InstancesWithFailedPatches": 2,
    "InstancesWithInstalledOtherPatches": 3,
    "InstancesWithInstalledPatches": 21,
    "InstancesWithInstalledPendingRebootPatches": 2,
    "InstancesWithInstalledRejectedPatches": 1,
    "InstancesWithMissingPatches": 3,
    "InstancesWithNotApplicablePatches": 4,
    "InstancesWithOtherNonCompliantPatches": 1,
    "InstancesWithSecurityNonCompliantPatches": 1,
    "InstancesWithUnreportedNotApplicablePatches": 2
```

```
}
```

Weitere Informationen finden Sie unter About Patch Groups < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-patchgroups.html >__ und Understanding Patch Compliance State Values im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie DescribePatchGroupStatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die allgemeine Zusammenfassung der Patch-Konformität für eine Patch-Gruppe abgerufen.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Ausgabe:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- Einzelheiten zur API finden Sie unter DescribePatchGroupState AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die allgemeine Zusammenfassung der Patch-Konformität für eine Patch-Gruppe abgerufen.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Ausgabe:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
```

```
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- Einzelheiten zur API finden Sie unter DescribePatchGroupState AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribePatchGroups`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribePatchGroups` verwendet wird.

CLI

AWS CLI

So zeigen Sie Registrierungen für Patch-Gruppen an

Das folgende `describe-patch-groups`-Beispiel listet die Patch-Gruppenregistrierungen auf.

```
aws ssm describe-patch-groups
```

Ausgabe:

```
{
    "Mappings": [
        {
            "PatchGroup": "Production",
            "BaselineIdentity": {
                "BaselineId": "pb-0123456789abcdef0",
                "BaselineName": "ProdPatching",
                "OperatingSystem": "WINDOWS",
                "BaselineDescription": "Patches for Production",
                "DefaultBaseline": false
            }
        },
        {
            "PatchGroup": "Development",
            "BaselineIdentity": {
                "BaselineId": "pb-0713accee01234567",
                "BaselineName": "DevPatching",
                "OperatingSystem": "WINDOWS",
```

```
                "BaselineDescription": "Patches for Development",
                "DefaultBaseline": true
            }
        },
        ...
    ]
}
```

Weitere Informationen finden Sie unter Erstellen einer Patchgruppe < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ und Hinzufügen einer Patchgruppe zu einer Patch-Baseline im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter DescribePatchGroups AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Registrierungen der Patchgruppen aufgeführt.

```
Get-SSMPatchGroup
```

Ausgabe:

```
BaselineIdentity                                            PatchGroup
----------------                                            ----------
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Einzelheiten zur API finden Sie unter DescribePatchGroups AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Registrierungen der Patchgruppen aufgeführt.

```
Get-SSMPatchGroup
```

Ausgabe:

```
BaselineIdentity                                            PatchGroup
----------------                                            ----------
```

```
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- Einzelheiten zur API finden Sie unter DescribePatchGroups AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetAutomationExecution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetAutomationExecution` verwendet wird.

CLI

AWS CLI

So zeigen Sie Details zu einer Automatisierungsausführung an

Im folgenden `get-automation-execution`-Beispiel werden detaillierte Informationen zu einer Automation-Ausführung angezeigt.

```
aws ssm get-automation-execution \
    --automation-execution-id 73c8eef8-f4ee-4a05-820c-e354fEXAMPLE
```

Ausgabe:

```
{
    "AutomationExecution": {
        "AutomationExecutionId": "73c8eef8-f4ee-4a05-820c-e354fEXAMPLE",
        "DocumentName": "AWS-StartEC2Instance",
        "DocumentVersion": "1",
        "ExecutionStartTime": 1583737233.748,
        "ExecutionEndTime": 1583737234.719,
        "AutomationExecutionStatus": "Success",
        "StepExecutions": [
            {
                "StepName": "startInstances",
                "Action": "aws:changeInstanceState",
                "ExecutionStartTime": 1583737234.134,
                "ExecutionEndTime": 1583737234.672,
                "StepStatus": "Success",
                "Inputs": {
                    "DesiredState": "\"running\"",
                    "InstanceIds": "[\"i-0cb99161f6EXAMPLE\"]"
                },
```

```
                "Outputs": {
                    "InstanceStates": [
                        "running"
                    ]
                },
                "StepExecutionId": "95e70479-cf20-4d80-8018-7e4e2EXAMPLE",
                "OverriddenParameters": {}
            }
        ],
        "StepExecutionsTruncated": false,
        "Parameters": {
            "AutomationAssumeRole": [
                ""
            ],
            "InstanceId": [
                "i-0cb99161f6EXAMPLE"
            ]
        },
        "Outputs": {},
        "Mode": "Auto",
        "ExecutedBy": "arn:aws:sts::29884EXAMPLE:assumed-role/mw_service_role/
OrchestrationService",
        "Targets": [],
        "ResolvedTargets": {
            "ParameterValues": [],
            "Truncated": false
        }
    }
}
```

Weitere Informationen finden Sie unter Exemplarische Vorgehensweise: Patchen eines Linux-AMI (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetAutomationExecution AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details einer Automation-Ausführung angezeigt.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Beispiel 2: In diesem Beispiel werden die Schrittdetails für die angegebene Automatisierungsausführungs-ID aufgeführt

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Ausgabe:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- Einzelheiten zur API finden Sie unter GetAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details einer Automation-Ausführung angezeigt.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
```

```
                              Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                              pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU

 gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-q0CK0ezS3xfh_y0ilaUGOAZG-
xjQFuvU_JZedWpla3xi-MZsmblAifBI
                              (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                              6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                     : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                  : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions              : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Beispiel 2: In diesem Beispiel werden die Schrittdetails für die angegebene Automatisierungsausführungs-ID aufgeführt

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Ausgabe:

```
StepName                  Action                  StepStatus ValidNextSteps
--------                  ------                  ---------- --------------
LaunchInstance            aws:runInstances        Success
 {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand          Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand          Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand          Cancelled  {}
UpdateSSMAgent            aws:runCommand          Pending    {}
UpdateAWSPVDriver         aws:runCommand          Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand          Pending    {}
UpdateAWSNVMe             aws:runCommand          Pending    {}
```

```
InstallWindowsUpdates     aws:runCommand          Pending    {}
RunPostUpdateScript       aws:runCommand          Pending    {}
RunSysprepGeneralize      aws:runCommand          Pending    {}
StopInstance              aws:changeInstanceState Pending    {}
CreateImage               aws:createImage         Pending    {}
TerminateInstance         aws:changeInstanceState Pending    {}
```

- Einzelheiten zur API finden Sie unter GetAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetCommandInvocation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetCommandInvocation` verwendet wird.

CLI

AWS CLI

So zeigen Sie die Details eines Befehlsaufrufs an

Das folgende `get-command-invocation`-Beispiel listet alle Aufrufe des angegebenen Befehls auf der angegebenen Instance auf.

```
aws ssm get-command-invocation \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --instance-id "i-1234567890abcdef0"
```

Ausgabe:

```
{
    "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
    "InstanceId": "i-1234567890abcdef0",
    "Comment": "b48291dd-ba76-43e0-b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-
d6ce8EXAMPLE",
    "DocumentName": "AWS-UpdateSSMAgent",
    "DocumentVersion": "",
    "PluginName": "aws:updateSsmAgent",
    "ResponseCode": 0,
    "ExecutionStartDateTime": "2020-02-19T18:18:03.419Z",
    "ExecutionElapsedTime": "PT0.091S",
    "ExecutionEndDateTime": "2020-02-19T18:18:03.419Z",
```

```
    "Status": "Success",
    "StatusDetails": "Success",
    "StandardOutputContent": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
    "StandardOutputUrl": "",
    "StandardErrorContent": "",
    "StandardErrorUrl": "",
    "CloudWatchOutputConfig": {
        "CloudWatchLogGroupName": "",
        "CloudWatchOutputEnabled": false
    }
}
```

Weitere Informationen finden Sie unter Befehlsstatus verstehen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetCommandInvocationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details eines Befehls angezeigt, der auf einer Instance ausgeführt wurde.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Ausgabe:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
```

```
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Einzelheiten zur API finden Sie unter GetCommandInvocation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details eines Befehls angezeigt, der auf einer Instance ausgeführt wurde.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Ausgabe:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Einzelheiten zur API finden Sie unter GetCommandInvocation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetConnectionStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetConnectionStatus` verwendet wird.

CLI

AWS CLI

Um den Verbindungsstatus einer verwalteten Instance anzuzeigen

In diesem `get-connection-status`-Beispiel wird der Verbindungsstatus der angegebenen verwalteten Instance zurückgegeben.

```
aws ssm get-connection-status \
    --target i-1234567890abcdef0
```

Ausgabe:

```
{
    "Target": "i-1234567890abcdef0",
    "Status": "connected"
}
```

- Einzelheiten zur API finden Sie GetConnectionStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Session-Manager-Verbindungsstatus für eine Instance abgerufen, um festzustellen, ob sie verbunden und bereit ist, Session-Manager-Verbindungen zu empfangen.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Ausgabe:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Einzelheiten zur API finden Sie unter GetConnectionStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Session-Manager-Verbindungsstatus für eine Instance abgerufen, um festzustellen, ob sie verbunden und bereit ist, Session-Manager-Verbindungen zu empfangen.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Ausgabe:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- Einzelheiten zur API finden Sie unter GetConnectionStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDefaultPatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDefaultPatchBaseline` verwendet wird.

CLI

AWS CLI

Beispiel 1: So zeigen Sie die Standard-Windows-Patch-Baseline an

Im folgenden Beispiel `get-default-patch-baseline` werden Details für die Standard-Patch-Baseline für Windows Server abgerufen.

```
aws ssm get-default-patch-baseline
```

Ausgabe:

```
{
  "BaselineId": "pb-0713accee01612345",
  "OperatingSystem": "WINDOWS"
}
```

Beispiel 2: So zeigen Sie die Standard-Patch-Baseline für Amazon Linux an

Im folgenden Beispiel `get-default-patch-baseline` werden Details für die Standard-Patch-Baseline für Amazon Linux abgerufen.

```
aws ssm get-default-patch-baseline \
    --operating-system AMAZON_LINUX
```

Ausgabe:

```
{
    "BaselineId": "pb-047c6eb9c8fc12345",
    "OperatingSystem": "AMAZON_LINUX"
}
```

Weitere Informationen finden Sie unter Über vordefinierte und benutzerdefinierte Patch-Baselines < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-baselines.html >__ und Eine bestehende Patch-Baseline als Standard festlegen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetDefaultPatchBaselinein AWS CLI der Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Standard-Patch-Baseline angezeigt.

```
Get-SSMDefaultPatchBaseline
```

Ausgabe:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Einzelheiten zur API finden Sie unter GetDefaultPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Standard-Patch-Baseline angezeigt.

```
Get-SSMDefaultPatchBaseline
```

Ausgabe:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- Einzelheiten zur API finden Sie unter GetDefaultPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDeployablePatchSnapshotForInstance`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDeployablePatchSnapshotForInstance` verwendet wird.

CLI

AWS CLI

Um den aktuellen Snapshot für die Patch-Baseline abzurufen, verwendet eine Instance

Im folgenden `get-deployable-patch-snapshot-for-instance`-Beispiel werden Details für den aktuellen Snapshot für die angegebene Patch-Baseline abgerufen, die von einer Instance verwendet wird. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance ausgeführt werden. Um sicherzustellen, dass er die Instance-Anmeldeinformationen verwendet, führen Sie `aws configure` aus und geben Sie nur die Region Ihrer Instance an. Lassen Sie die Felder `Access Key` und `Secret Key` leer.

Tipp: Verwenden Sie `uuidgen`, um eine `snapshot-id` zu generieren.

```
aws ssm get-deployable-patch-snapshot-for-instance \
    --instance-id "i-1234567890abcdef0" \
    --snapshot-id "521c3536-930c-4aa9-950e-01234567abcd"
```

Ausgabe:

```
{
    "InstanceId": "i-1234567890abcdef0",
    "SnapshotId": "521c3536-930c-4aa9-950e-01234567abcd",
    "Product": "AmazonLinux2018.03",
```

```
    "SnapshotDownloadUrl": "https://patch-baseline-snapshot-us-
east-1.s3.amazonaws.com/
ed85194ef27214f5984f28b4d664d14f7313568fea7d4b6ac6c10ad1f729d7e7-773304212436/
AMAZON_LINUX-521c3536-930c-4aa9-950e-01234567abcd?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190215T164031Z&X-Amz-
SignedHeaders=host&X-Amz-Expires=86400&X-Amz-Credential=AKIAJ5C56P35AEBRX2QQ
%2F20190215%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-
Signature=efaaaf6e3878e77f48a6697e015efdbda9c426b09c5822055075c062f6ad2149"
}
```

Weitere Informationen finden Sie unter Parametername: Snapshot-ID im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetDeployablePatchSnapshotForInstancein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der aktuelle Snapshot für die von einer Instance verwendete Patch-Baseline angezeigt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance ausgeführt werden. Um sicherzustellen, dass die Instance-Anmeldedaten verwendet werden, übergibt das Beispiel ein **`Amazon.Runtime.InstanceProfileAWSCredentials`**-Objekt an den Anmeldeinformationen-Parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Ausgabe:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Beispiel 2: Dieses Beispiel zeigt, wie man die volle Version bekommt SnapshotDownloadUrl. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance

ausgeführt werden. Um sicherzustellen, dass die Instanzanmeldedaten verwendet werden, konfiguriert das Beispiel die PowerShell Sitzung für die Verwendung eines **`Amazon.Runtime.InstanceProfileAWSCredentials`** Objekts.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Ausgabe:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Einzelheiten zur API finden Sie unter GetDeployablePatchSnapshotForInstance AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der aktuelle Snapshot für die von einer Instance verwendete Patch-Baseline angezeigt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance ausgeführt werden. Um sicherzustellen, dass die Instance-Anmeldedaten verwendet werden, übergibt das Beispiel ein **`Amazon.Runtime.InstanceProfileAWSCredentials`**-Objekt an den Anmeldeinformationen-Parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Ausgabe:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Beispiel 2: Dieses Beispiel zeigt, wie man das volle SnapshotDownloadUrl bekommt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance ausgeführt werden. Um sicherzustellen, dass die Instanzanmeldedaten verwendet

werden, konfiguriert das Beispiel die PowerShell Sitzung für die Verwendung eines **`Amazon.Runtime.InstanceProfileAWSCredentials`** Objekts.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Ausgabe:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- Einzelheiten zur API finden Sie unter GetDeployablePatchSnapshotForInstance AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetDocument`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetDocument` verwendet wird.

CLI

AWS CLI

So rufen Sie den Inhalt des Dokuments ab

Im folgenden `get-document`-Beispiel wird der Inhalt eines Systems-Manager-Dokuments angezeigt.

```
aws ssm get-document \
    --name "AWS-RunShellScript"
```

Ausgabe:

```
{
    "Name": "AWS-RunShellScript",
    "DocumentVersion": "1",
    "Status": "Active",
    "Content": "{\n    \"schemaVersion\":\"1.2\",\n    \"description\":\"Run
 a shell script or specify the commands to run.\",\n    \"parameters\":{\n
```

```
        \"commands\":{\n            \"type\":\"StringList\",\n
 \"description\":\"(Required) Specify a shell script or a command to run.\",
\n            \"minItems\":1,\n            \"displayType\":\"textarea\"\n
      },\n        \"workingDirectory\":{\n            \"type\":\"String\",\n
            \"default\":\"\",\n            \"description\":\"(Optional) The
 path to the working directory on your instance.\",\n            \"maxChars
\":4096\n        },\n        \"executionTimeout\":{\n            \"type\":
\"String\",\n            \"default\":\"3600\",\n            \"description
\":\"(Optional) The time in seconds for a command to complete before it is
 considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48
 hours).\",\n            \"allowedPattern\":\"([1-9][0-9]{0,4})|(1[0-6][0-9]
{4})|(17[0-1][0-9]{3})|(172[0-7][0-9]{2})|(172800)\"\n        }\n    },\n
 \"runtimeConfig\":{\n        \"aws:runShellScript\":{\n            \"properties
\":[\n                {\n                    \"id\":\"0.aws:runShellScript
\",\n                    \"runCommand\":\"{{ commands }}\",\n
      \"workingDirectory\":\"{{ workingDirectory }}\",\n
 \"timeoutSeconds\":\"{{ executionTimeout }}\"\n                }\n
  ]\n        }\n    }\n}\n",
    "DocumentType": "Command",
    "DocumentFormat": "JSON"
}
```

Weitere Informationen finden Sie unter AWS -Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetDocumentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Inhalt eines Dokuments zurückgegeben.

```
Get-SSMDocument -Name "RunShellScript"
```

Ausgabe:

```
Content
-------
{...
```

Beispiel 2: In diesem Beispiel wird der vollständige Inhalt eines Dokuments angezeigt.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- Einzelheiten zur API finden Sie unter GetDocument AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Inhalt eines Dokuments zurückgegeben.

```
Get-SSMDocument -Name "RunShellScript"
```

Ausgabe:

```
Content
```

```
-------
{...
```

Beispiel 2: In diesem Beispiel wird der vollständige Inhalt eines Dokuments angezeigt.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
   "parameters":{
      "commands":{
         "type":"StringList",
         "description":"(Required) Specify a shell script or a command to run.",
         "minItems":1,
         "displayType":"textarea"
      }
   },
   "mainSteps":[
      {
         "action":"aws:runShellScript",
         "name":"runShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      },
      {
         "action":"aws:runPowerShellScript",
         "name":"runPowerShellScript",
         "inputs":{
            "commands":"{{ commands }}"
         }
      }
   ]
}
```

- Einzelheiten zur API finden Sie unter GetDocument AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetInventory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetInventory` verwendet wird.

CLI

AWS CLI

Um Ihr Inventar einzusehen

In diesem Beispiel werden die benutzerdefinierten Metadaten für Ihr Inventar abgerufen.

Befehl:

```
aws ssm get-inventory
```

Ausgabe:

```
{
  "Entities": [
      {
          "Data": {
              "AWS:InstanceInformation": {
                  "Content": [
                      {
                          "ComputerName": "ip-172-31-44-222.us-
west-2.compute.internal",
                          "InstanceId": "i-0cb2b964d3e14fd9f",
                          "IpAddress": "172.31.44.222",
                          "AgentType": "amazon-ssm-agent",
                          "ResourceType": "EC2Instance",
                          "AgentVersion": "2.0.672.0",
                          "PlatformVersion": "2016.09",
                          "PlatformName": "Amazon Linux AMI",
                          "PlatformType": "Linux"
                      }
                  ],
                  "TypeName": "AWS:InstanceInformation",
                  "SchemaVersion": "1.0",
                  "CaptureTime": "2017-02-20T18:03:58Z"
              }
          },
          "Id": "i-0cb2b964d3e14fd9f"
      }
  ]
}
```

- Einzelheiten zur API finden Sie GetInventoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die benutzerdefinierten Metadaten für Ihr Inventar abgerufen.

```
Get-SSMInventory
```

Ausgabe:

```
Data
     Id
----
     --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Einzelheiten zur API finden Sie unter GetInventory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die benutzerdefinierten Metadaten für Ihr Inventar abgerufen.

```
Get-SSMInventory
```

Ausgabe:

```
Data
     Id
----
     --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- Einzelheiten zur API finden Sie unter GetInventory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetInventorySchema`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetInventorySchema` verwendet wird.

CLI

AWS CLI

So zeigen Sie Ihr Inventarschema an

Dieses Beispiel gibt eine Liste mit Inventartypnamen für das Konto zurück.

Befehl:

```
aws ssm get-inventory-schema
```

Ausgabe:

```
{
  "Schemas": [
      {
          "TypeName": "AWS:AWSComponent",
          "Version": "1.0",
          "Attributes": [
              {
                  "Name": "Name",
                  "DataType": "STRING"
              },
              {
                  "Name": "ApplicationType",
                  "DataType": "STRING"
              },
              {
                  "Name": "Publisher",
                  "DataType": "STRING"
              },
              {
                  "Name": "Version",
                  "DataType": "STRING"
              },
              {
                  "Name": "InstalledTime",
                  "DataType": "STRING"
              },
```

```
                {
                    "Name": "Architecture",
                    "DataType": "STRING"
                },
                {
                    "Name": "URL",
                    "DataType": "STRING"
                }
            ]
        },
        ...
    ],
    "NextToken": "--token string truncated--"
}
```

So zeigen Sie das Inventarschema für einen bestimmten Inventartyp an

In diesem Beispiel wird das Inventarschema für den Inventartyp AWS„AWS Component“ zurückgegeben.

Befehl:

```
aws ssm get-inventory-schema --type-name "AWS:AWSComponent"
```

- Einzelheiten zur API finden Sie GetInventorySchemain der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Liste von Inventartypnamen für das Konto zurückgegeben.

```
Get-SSMInventorySchema
```

- Einzelheiten zur API finden Sie unter GetInventorySchema AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Liste von Inventartypnamen für das Konto zurückgegeben.

```
Get-SSMInventorySchema
```

- Einzelheiten zur API finden Sie unter GetInventorySchema AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetMaintenanceWindow`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetMaintenanceWindow` verwendet wird.

CLI

AWS CLI

Anzeigen von Informationen zu Wartungsfenstern

Im folgenden `get-maintenance-window`-Beispiel werden Details zum angegebenen Wartungsfenster abgerufen.

```
aws ssm get-maintenance-window \
    --window-id "mw-03eb9db428EXAMPLE"
```

Ausgabe:

```
{
    "AllowUnassociatedTargets": true,
    "CreatedDate": 1515006912.957,
    "Cutoff": 1,
    "Duration": 6,
    "Enabled": true,
    "ModifiedDate": 2020-01-01T10:04:04.099Z,
    "Name": "My-Maintenance-Window",
    "Schedule": "rate(3 days)",
    "WindowId": "mw-03eb9db428EXAMPLE",
    "NextExecutionTime": "2020-02-25T00:08:15.099Z"
}
```

Weitere Informationen finden Sie unter Informationen zu Wartungsfenstern (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetMaintenanceWindowunter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zu einem Wartungsfenster abgerufen.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zu einem Wartungsfenster abgerufen.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetMaintenanceWindowExecution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetMaintenanceWindowExecution` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zur Ausführung einer Wartungsfensteraufgabe

Das folgende `get-maintenance-window-execution`-Beispiel listet Informationen zu einer Aufgabe auf, die im Rahmen der Ausführung des angegebenen Wartungsfensters ausgeführt wurde.

```
aws ssm get-maintenance-window-execution \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE"
```

Ausgabe:

```
{
    "Status": "SUCCESS",
    "TaskIds": [
        "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
    ],
    "StartTime": 1487692834.595,
    "EndTime": 1487692835.051,
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
}
```

Weitere Informationen finden Sie unter Informationen zu Aufgaben und Aufgabenausführungen (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecution AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet Informationen über eine Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet Informationen über eine Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetMaintenanceWindowExecutionTask`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetMaintenanceWindowExecutionTask` verwendet wird.

CLI

AWS CLI

So erhalten Sie Informationen zur Ausführung einer Wartungsfensteraufgabe

Das folgende `get-maintenance-window-execution-task`-Beispiel listet Informationen zu einer Aufgabe auf, die Teil der Ausführung des angegebenen Wartungsfensters ist.

```
aws ssm get-maintenance-window-execution-task \
    --window-execution-id "518d5565-5969-4cca-8f0e-da3b2EXAMPLE" \
    --task-id "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE"
```

Ausgabe:

```
{
    "WindowExecutionId": "518d5565-5969-4cca-8f0e-da3b2EXAMPLE",
    "TaskExecutionId": "ac0c6ae1-daa3-4a89-832e-d3845EXAMPLE",
    "TaskArn": "AWS-RunPatchBaseline",
    "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
    "Type": "RUN_COMMAND",
    "TaskParameters": [
        {
            "BaselineOverride": {
                "Values": [
                    ""
                ]
            },
            "InstallOverrideList": {
                "Values": [
                    ""
                ]
            },
            "Operation": {
                "Values": [
                    "Scan"
```

```
                ]
            },
            "RebootOption": {
                "Values": [
                    "RebootIfNeeded"
                ]
            },
            "SnapshotId": {
                "Values": [
                    "{{ aws:ORCHESTRATION_ID }}"
                ]
            },
            "aws:InstanceId": {
                "Values": [
                    "i-02573cafcfEXAMPLE",
                    "i-0471e04240EXAMPLE",
                    "i-07782c72faEXAMPLE"
                ]
            }
        }
    ],
    "Priority": 1,
    "MaxConcurrency": "1",
    "MaxErrors": "3",
    "Status": "SUCCESS",
    "StartTime": "2021-08-04T11:45:35.088000-07:00",
    "EndTime": "2021-08-04T11:53:09.079000-07:00"
}
```

Weitere Informationen finden Sie unter Informationen zu Aufgaben und Aufgabenausführungen (AWS CLI) anzeigen im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecutionTask AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Informationen zu einer Aufgabe aufgeführt, die Teil einer Ausführung im Rahmen eines Wartungsfensters war.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecutionTask AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Informationen zu einer Aufgabe aufgeführt, die Teil einer Ausführung im Rahmen eines Wartungsfensters war.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-
d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
```

```
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsM
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- Einzelheiten zur API finden Sie unter GetMaintenanceWindowExecutionTask AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`GetParameter`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetParameter` verwendet wird.

CLI

AWS CLI

Beispiel 1: So zeigen Sie den Wert eines Parameters an

Das folgende `get-parameter`-Beispiel listet den Wert für den angegebenen Einzelparameter auf.

```
aws ssm get-parameter \
    --name "MyStringParameter"
```

Ausgabe:

```
{
    "Parameter": {
        "Name": "MyStringParameter",
        "Type": "String",
        "Value": "Veni",
        "Version": 1,
        "LastModifiedDate": 1530018761.888,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/MyStringParameter"
        "DataType": "text"
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameter Store im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: Um den Wert eines SecureString Parameters zu entschlüsseln

Im folgenden Beispiel `get-parameter` wird der Wert des angegebenen `SecureString`-Parameters entschlüsselt.

```
aws ssm get-parameter \
    --name "MySecureStringParameter" \
    --with-decryption
```

Ausgabe:

```
{
    "Parameter": {
        "Name": "MySecureStringParameter",
        "Type": "SecureString",
        "Value": "16679b88-310b-4895-a943-e0764EXAMPLE",
        "Version": 2,
        "LastModifiedDate": 1582155479.205,
        "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MySecureStringParameter"
        "DataType": "text"
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameter Store im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 3: So können den Wert eines Parameters mithilfe von Beschriftungen anzeigen

Das folgende `get-parameter`-Beispiel listet den Wert für den angegebenen Einzelparameter mit einer angegebenen Bezeichnung auf.

```
aws ssm get-parameter \
    --name "MyParameter:label"
```

Ausgabe:

```
{
```

```
    "Parameter": {
        "Name": "MyParameter",
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":label",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameterbeschriftungen im AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 4: So können den Wert eines Parameters mithilfe von Versionen anzeigen

Das folgende `get-parameter`-Beispiel listet den Wert für die angegebene Einzelparameter-Version auf.

```
aws ssm get-parameter \
    --name "MyParameter:2"
```

Ausgabe:

```
{
    "Parameter": {
        "Name": "MyParameter",
        "Type": "String",
        "Value": "parameter version 2",
        "Version": 2,
        "Selector": ":2",
        "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
        "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
        "DataType": "text"
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameterbeschriftungen im AWS - Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetParameterin der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
{
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
```

- Einzelheiten zur API finden Sie GetParameterin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`GetParameterHistory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetParameterHistory` verwendet wird.

CLI

AWS CLI

Um einen Werteverlauf für einen Parameter abzurufen

Das folgende `get-parameter-history`-Beispiel listet den Verlauf der Änderungen für den angegebenen Parameter auf, einschließlich seines Werts.

```
aws ssm get-parameter-history \
    --name "MyStringParameter"
```

Ausgabe:

```
{
    "Parameters": [
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582154711.976,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the first version of my String parameter",
            "Value": "Veni",
            "Version": 1,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156093.471,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the second version of my String parameter",
            "Value": "Vidi",
            "Version": 2,
            "Labels": [],
            "Tier": "Standard",
            "Policies": []
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "LastModifiedDate": 1582156117.545,
            "LastModifiedUser": "arn:aws:iam::111222333444:user/Mary-Major",
            "Description": "This is the third version of my String parameter",
            "Value": "Vici",
            "Version": 3,
            "Labels": [],
```

```
            "Tier": "Standard",
            "Policies": []
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameterversionen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetParameterHistoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Werteverlauf für einen Parameter aufgeführt.

```
Get-SSMParameterHistory -Name "Welcome"
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Einzelheiten zur API finden Sie unter GetParameterHistory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Werteverlauf für einen Parameter aufgeführt.

```
Get-SSMParameterHistory -Name "Welcome"
```

Ausgabe:

```
Description      :
```

```
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- Einzelheiten zur API finden Sie unter GetParameterHistory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetParameters`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetParameters` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Einen Containerdienst für VM-Instanzen erstellen

CLI

AWS CLI

Beispiel 1: So können Sie die Werte für einen Parameter auflisten

Das folgende `get-parameters`-Beispiel listet die Werte für die drei angegebenen Parameter auf.

```
aws ssm get-parameters \
    --names "MyStringParameter" "MyStringListParameter" "MyInvalidParameterName"
```

Ausgabe:

```
{
    "Parameters": [
        {
            "Name": "MyStringListParameter",
            "Type": "StringList",
            "Value": "alpha,beta,gamma",
            "Version": 1,
            "LastModifiedDate": 1582154764.222,
```

```
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringListParameter"
            "DataType": "text"
        },
        {
            "Name": "MyStringParameter",
            "Type": "String",
            "Value": "Vici",
            "Version": 3,
            "LastModifiedDate": 1582156117.545,
            "ARN": "arn:aws:ssm:us-east-2:111222333444:parameter/
MyStringParameter"
            "DataType": "text"
        }
    ],
    "InvalidParameters": [
        "MyInvalidParameterName"
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameter Store im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So können Sie Namen und Werte mehrerer Parameter mit der Option ``--query`` auflisten

Im folgenden Beispiel `get-parameters` werden die Namen und Werte für die angegebenen Parameter angezeigt.

```
aws ssm get-parameters \
    --names MyStringParameter MyStringListParameter \
    --query "Parameters[*].{Name:Name,Value:Value}"
```

Ausgabe:

```
[
    {
        "Name": "MyStringListParameter",
        "Value": "alpha,beta,gamma"
    },
    {
        "Name": "MyStringParameter",
```

```
        "Value": "Vidi"
    }
]
```

Weitere Informationen finden Sie unter Arbeiten mit Parameter Store im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 3: So können den Wert eines Parameters mithilfe von Beschriftungen anzeigen

Das folgende `get-parameter`-Beispiel listet den Wert für den angegebenen Einzelparameter mit einer angegebenen Bezeichnung auf.

```
aws ssm get-parameter \
    --name "MyParameter:label"
```

Ausgabe:

```
{
    "Parameters": [
        {
            "Name": "MyLabelParameter",
            "Type": "String",
            "Value": "parameter by label",
            "Version": 1,
            "Selector": ":label",
            "LastModifiedDate": "2021-07-12T09:49:15.865000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/MyParameter",
            "DataType": "text"
        },
        {
            "Name": "MyVersionParameter",
            "Type": "String",
            "Value": "parameter by version",
            "Version": 2,
            "Selector": ":2",
            "LastModifiedDate": "2021-03-24T16:20:28.236000-07:00",
            "ARN": "arn:aws:ssm:us-west-2:786973925828:parameter/unlabel-param",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

Weitere Informationen finden Sie unter Arbeiten mit Parameterbeschriftungen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetParametersin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Werte für einen Parameter aufgeführt.

```
Get-SSMParameterValue -Name "Welcome"
```

Ausgabe:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Beispiel 2: In diesem Beispiel werden die Details des Werts aufgeführt.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Ausgabe:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Einzelheiten zur API finden Sie unter GetParameters AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Werte für einen Parameter aufgeführt.

```
Get-SSMParameterValue -Name "Welcome"
```

Ausgabe:

```
InvalidParameters Parameters
----------------- ----------
```

```
{}                  {Welcome}
```

Beispiel 2: In diesem Beispiel werden die Details des Werts aufgeführt.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Ausgabe:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- Einzelheiten zur API finden Sie unter GetParameters AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPatchBaseline` verwendet wird.

CLI

AWS CLI

So zeigen Sie eine Patch-Baseline an

Das folgende `get-patch-baseline`-Beispiel ruft die Details für die angegebene Patch-Baseline ab.

```
aws ssm get-patch-baseline \
    --baseline-id "pb-0123456789abcdef0"
```

Ausgabe:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
        "PatchFilters": []
    },
    "ApprovalRules": {
```

```
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
    "RejectedPatches": [],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
    "PatchGroups": [
        "QA",
        "DEV"
    ],
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

Weitere Informationen finden Sie unter Über Patch-Baselines im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie GetPatchBaselinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Details für eine Patch-Baseline angezeigt.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Einzelheiten zur API finden Sie unter GetPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Details für eine Patch-Baseline angezeigt.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- Einzelheiten zur API finden Sie unter GetPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`GetPatchBaselineForPatchGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPatchBaselineForPatchGroup` verwendet wird.

CLI

AWS CLI

So zeigen Sie die Patch-Baseline für eine Patch-Gruppe an

Das folgende `get-patch-baseline-for-patch-group`-Beispiel ruft die Details über die Patch-Baseline für die angegebene Patchgruppe ab.

```
aws ssm get-patch-baseline-for-patch-group \
    --patch-group "DEV"
```

Ausgabe:

```
{
    "PatchGroup": "DEV",
    "BaselineId": "pb-0123456789abcdef0",
    "OperatingSystem": "WINDOWS"
}
```

Weitere Informationen finden Sie unter Erstellen einer Patchgruppe < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ und Hinzufügen einer Patchgruppe zu einer Patch-Baseline im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter GetPatchBaselineForPatchGroup AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Patch-Baseline für eine Patchgruppe angezeigt.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter GetPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Patch-Baseline für eine Patchgruppe angezeigt.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- Einzelheiten zur API finden Sie unter GetPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListAssociationVersions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListAssociationVersions` verwendet wird.

CLI

AWS CLI

So listen Sie alle Versionen einer Zuordnung für eine bestimmte Zuordnungs-ID auf

Das folgende `list-association-versions`-Beispiel listet alle Versionen der angegebenen Zuordnungen auf.

```
aws ssm list-association-versions \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab"
```

Ausgabe:

```
{
```

```
"AssociationVersions": [
        {
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "CreatedDate": 1550505536.726,
            "Name": "AWS-UpdateSSMAgent",
            "Parameters": {
                "allowDowngrade": [
                    "false"
                ],
                "version": [
                    ""
                ]
            },
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Zuordnungen in Systems Manager im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListAssociationVersionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Versionen der angegebenen Zuordnung abgerufen.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Einzelheiten zur API finden Sie unter ListAssociationVersions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Versionen der angegebenen Zuordnung abgerufen.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
```

```
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- Einzelheiten zur API finden Sie unter ListAssociationVersions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListAssociations`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListAssociations` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie Ihre Zuordnungen für eine bestimmte Instance auflisten

Das folgende Beispiel für list-associations listet alle Assoziationen mit dem UpdateSSMAgent auf AssociationName.

```
aws ssm list-associations /
    --association-filter-list "key=AssociationName,value=UpdateSSMAgent"
```

Ausgabe:

```
{
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-016648b75dd622dab"
                    ]
                }
            ],
            "Overview": {
                "Status": "Pending",
                "DetailedStatus": "Associated",
                "AssociationStatusAggregatedCount": {
                    "Pending": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Zuordnungen in Systems Manager im -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So können Sie Ihre Verknüpfungen für ein bestimmtes Dokument auflisten

Das folgende Beispiel listet alle Verknüpfungen für das angegebene Dokument auf.

```
aws ssm list-associations /
    --association-filter-list "key=Name,value=AWS-UpdateSSMAgent"
```

Ausgabe:

```
{
```

```
    "Associations": [
        {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-1234567890abcdef0",
            "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-1234567890abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550505828.548,
            "Overview": {
                "Status": "Success",
                "DetailedStatus": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
            },
            "ScheduleExpression": "cron(0 00 12 ? * SUN *)",
            "AssociationName": "UpdateSSMAgent"
        },
    {
            "Name": "AWS-UpdateSSMAgent",
            "InstanceId": "i-9876543210abcdef0",
            "AssociationId": "fbc07ef7-b985-4684-b82b-0123456789ab",
            "AssociationVersion": "1",
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-9876543210abcdef0"
                    ]
                }
            ],
            "LastExecutionDate": 1550507531.0,
            "Overview": {
                "Status": "Success",
                "AssociationStatusAggregatedCount": {
                    "Success": 1
                }
```

```
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Arbeiten mit Zuordnungen in Systems Manager im - Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter Befehlsreferenz. ListAssociationsAWS CLI

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet Details zu den Zuordnungen für eine Instance auf. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 2: In diesem Beispiel werden alle Verknüpfungen für ein Konfigurationsdokument aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
```

```
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Einzelheiten zur API finden Sie unter ListAssociations AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet Details zu den Zuordnungen für eine Instance auf. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
```

```
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 2: In diesem Beispiel werden alle Verknüpfungen für ein Konfigurationsdokument aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
```

```
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- Einzelheiten zur API finden Sie unter ListAssociations AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`ListCommandInvocations`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListCommandInvocations` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

So listen Sie die Aufrufe eines bestimmten Befehls auf

Das folgende `list-command-invocations`-Beispiel listet alle Aufrufe eines Befehls auf.

```
aws ssm list-command-invocations \
    --command-id "ef7fdfd8-9b57-4151-a15c-db9a12345678" \
    --details
```

Ausgabe:

```
{
    "CommandInvocations": [
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-02573cafcfEXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.089,
```

```
            "Status": "Success",
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.419,
                    "ResponseFinishDateTime": 1582136283.51,
                    "Output": "Updating amazon-ssm-agent from 2.3.842.0 to latest
\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-ssm-us-
east-2/ssm-agent-manifest.json\namazon-ssm-agent 2.3.842.0 has already been
 installed, update skipped\n",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": ""
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        },
        {
            "CommandId": "ef7fdfd8-9b57-4151-a15c-db9a12345678",
            "InstanceId": "i-0471e04240EXAMPLE",
            "InstanceName": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "RequestedDateTime": 1582136283.02,
            "Status": "Success",
```

```
            "StatusDetails": "Success",
            "StandardOutputUrl": "",
            "StandardErrorUrl": "",
            "CommandPlugins": [
                {
                    "Name": "aws:updateSsmAgent",
                    "Status": "Success",
                    "StatusDetails": "Success",
                    "ResponseCode": 0,
                    "ResponseStartDateTime": 1582136283.812,
                    "ResponseFinishDateTime": 1582136295.031,
                    "Output": "Updating amazon-ssm-agent from 2.3.672.0 to
 latest\nSuccessfully downloaded https://s3.us-east-2.amazonaws.com/amazon-
ssm-us-east-2/ssm-agent-manifest.json\nSuccessfully downloaded https://s3.us-
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent-updater/2.3.842.0/
amazon-ssm-agent-updater-snap-amd64.tar.gz\nSuccessfully downloaded https://
s3.us-east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.672.0/
amazon-ssm-agent-snap-amd64.tar.gz\nSuccessfully downloaded https://s3.us-
east-2.amazonaws.com/amazon-ssm-us-east-2/amazon-ssm-agent/2.3.842.0/amazon-ssm-
agent-snap-amd64.tar.gz\nInitiating amazon-ssm-agent update to 2.3.842.0\namazon-
ssm-agent updated successfully to 2.3.842.0",
                    "StandardOutputUrl": "",
                    "StandardErrorUrl": "",
                    "OutputS3Region": "us-east-2",
                    "OutputS3BucketName": "",
                    "OutputS3KeyPrefix": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE/
i-0471e04240EXAMPLE/awsupdateSsmAgent"
                }
            ],
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Befehlsstatus verstehen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListCommandInvocationsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { paginateListCommandInvocations, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * List SSM command invocations on an instance.
 * @param {{ instanceId: string }}
 */
export const main = async ({ instanceId }) => {
  const client = new SSMClient({});
  try {
    const listCommandInvocationsPaginated = [];
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListCommandInvocations(
      { client },
      {
        InstanceId: instanceId,
      },
    );
    for await (const page of paginator) {
      listCommandInvocationsPaginated.push(...page.CommandInvocations);
    }
    console.log("Here is the list of command invocations:");
    console.log(listCommandInvocationsPaginated);
    return { CommandInvocations: listCommandInvocationsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid instance ID?`);
```

```
    }
    throw caught;
  }
};
```

- Einzelheiten zur API finden Sie ListCommandInvocationsin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Aufrufe eines Befehls aufgeführt.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Ausgabe:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Beispiel 2: In diesem Beispiel wird der Aufruf der Befehls-ID CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f aufgeführt

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Ausgabe:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Einzelheiten AWS -Tools für PowerShell zur API ListCommandInvocationsfinden Sie unter Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Aufrufe eines Befehls aufgeführt.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -
Detail $true
```

Ausgabe:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
```

```
TraceOutput        :
```

Beispiel 2: In diesem Beispiel wird der Aufruf der Befehls-ID CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f aufgeführt

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Ausgabe:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- Einzelheiten AWS -Tools für PowerShell zur API ListCommandInvocationsfinden Sie unter Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
```

```
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie ListCommandInvocationsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->listcommandinvocations(
          iv_instanceid = iv_instance_id ).

        DATA lv_count TYPE i VALUE 0.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmlistcmdinvcsresult( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_commandinvocations( ) INTO DATA(lo_invocation).
            lv_count = lv_count + 1.
            DATA(lv_requested_datetime) = lo_invocation-
>get_requesteddatetime( ).
            MESSAGE |Command invocation requested at: { lv_requested_datetime }|
 TYPE 'I'.
          ENDLOOP.
        ENDWHILE.
```

```
        MESSAGE |{ lv_count } command invocation(s) found for instance
 { iv_instance_id }.| TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
      CATCH /aws1/cx_ssminvalidcommandid.
        MESSAGE 'Invalid command ID.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListCommandInvocationsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`ListCommands`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListCommands` verwendet wird.

CLI

AWS CLI

Beispiel 1: So erhalten Sie den Status eines bestimmten Befehls

Im folgenden Beispiel `list-commands` wird der Status des angegebenen Befehls abgerufen und angezeigt.

```
aws ssm list-commands \
    --command-id "0831e1a8-a1ac-4257-a1fd-c831bEXAMPLE"
```

Beispiel 2: So erhalten Sie den Status von Befehlen, die nach einem bestimmten Datum angefordert wurden

Im folgenden Beispiel `list-commands` werden die Details von Befehlen abgerufen, die nach dem angegebenen Datum angefordert wurden.

```
aws ssm list-commands \
    --filter "key=InvokedAfter,value=2020-02-01T00:00:00Z"
```

Beispiel 3: Um alle in einem AWS Konto angeforderten Befehle aufzulisten

Das folgende `list-commands` Beispiel listet alle Befehle auf, die von Benutzern im aktuellen AWS Konto und in der Region angefordert wurden.

```
aws ssm list-commands
```

Ausgabe:

```
{
    "Commands": [
        {
            "CommandId": "8bee3135-398c-4d31-99b6-e42d2EXAMPLE",
            "DocumentName": "AWS-UpdateSSMAgent",
            "DocumentVersion": "",
            "Comment": "b48291dd-ba76-43e0-
b9df-13e11ddaac26:6960febb-2907-4b59-8e1a-d6ce8EXAMPLE",
            "ExpiresAfter": "2020-02-19T11:28:02.500000-08:00",
            "Parameters": {},
            "InstanceIds": [
                "i-028ea792daEXAMPLE",
                "i-02feef8c46EXAMPLE",
                "i-038613f3f0EXAMPLE",
                "i-03a530a2d4EXAMPLE",
                "i-083b678d37EXAMPLE",
                "i-0dee81debaEXAMPLE"
            ],
            "Targets": [],
            "RequestedDateTime": "2020-02-19T10:18:02.500000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "",
            "OutputS3KeyPrefix": "",
            "MaxConcurrency": "50",
            "MaxErrors": "100%",
            "TargetCount": 6,
            "CompletedCount": 6,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "",
            "NotificationConfig": {
                "NotificationArn": "",
                "NotificationEvents": [],
                "NotificationType": ""
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
```

```
            }
        }
        {
            "CommandId": "e9ade581-c03d-476b-9b07-26667EXAMPLE",
            "DocumentName": "AWS-FindWindowsUpdates",
            "DocumentVersion": "1",
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:37:31.874000-08:00",
            "Parameters": {
                "KbArticleIds": [
                    ""
                ],
                "UpdateLevel": [
                    "All"
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:27:31.874000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
```

```
                ],
                "NotificationType": "Invocation"
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
        {
            "CommandId": "d539b6c3-70e8-4853-80e5-0ce4fEXAMPLE",
            "DocumentName": "AWS-RunPatchBaseline",
            "DocumentVersion": "1",
            "Comment": "",
            "ExpiresAfter": "2020-01-24T12:21:04.350000-08:00",
            "Parameters": {
                "InstallOverrideList": [
                    ""
                ],
                "Operation": [
                    "Install"
                ],
                "RebootOption": [
                    "RebootIfNeeded"
                ],
                "SnapshotId": [
                    ""
                ]
            },
            "InstanceIds": [],
            "Targets": [
                {
                    "Key": "InstanceIds",
                    "Values": [
                        "i-00ec29b21eEXAMPLE",
                        "i-09911ddd90EXAMPLE"
                    ]
                }
            ],
            "RequestedDateTime": "2020-01-24T11:11:04.350000-08:00",
            "Status": "Success",
            "StatusDetails": "Success",
            "OutputS3BucketName": "my-us-east-2-bucket",
            "OutputS3KeyPrefix": "my-rc-output",
            "MaxConcurrency": "50",
```

```
            "MaxErrors": "0",
            "TargetCount": 2,
            "CompletedCount": 2,
            "ErrorCount": 0,
            "DeliveryTimedOutCount": 0,
            "ServiceRole": "arn:aws:iam::111222333444:role/aws-service-role/
ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
            "NotificationConfig": {
                "NotificationArn": "arn:aws:sns:us-east-2:111222333444:my-us-
east-2-notification-arn",
                "NotificationEvents": [
                    "All"
                ],
                "NotificationType": "Invocation"
            },
            "CloudWatchOutputConfig": {
                "CloudWatchLogGroupName": "",
                "CloudWatchOutputEnabled": false
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListCommandsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle angeforderten Befehle aufgeführt.

```
Get-SSMCommand
```

Ausgabe:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
```

```
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Beispiel 2: In diesem Beispiel wird der Status eines bestimmten Befehls abgerufen.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Beispiel 3: In diesem Beispiel werden alle SSM-Befehle abgerufen, die nach dem 01.04.2019 aufgerufen wurden T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Ausgabe:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
```

```
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (ListCommandsV4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle angeforderten Befehle aufgeführt.

```
Get-SSMCommand
```

Ausgabe:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Beispiel 2: In diesem Beispiel wird der Status eines bestimmten Befehls abgerufen.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Beispiel 3: In diesem Beispiel werden alle SSM-Befehle abgerufen, die nach dem 01.04.2019 aufgerufen wurden T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} |
 Select-Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -
Property RequestedDateTime -Descending
```

Ausgabe:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled
 4/16/2019 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success
 4/6/2019 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed
 4/2/2019 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed
 4/1/2019 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success
 4/1/2019 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed
 4/1/2019 11:13:09 AM
```

- Einzelheiten zur API finden Sie unter AWS -Tools für PowerShell Cmdlet-Referenz (ListCommandsV5).

Verwendung von **`ListComplianceItems`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListComplianceItems` verwendet wird.

CLI

AWS CLI

So listen Sie Complianceelemente für eine bestimmte Instance auf

In diesem Beispiel werden alle Konformitätselemente für die angegebene Instance aufgeführt.

Befehl:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance"
```

Ausgabe:

```
{
  "ComplianceItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "8dfe3659-4309-493a-8755-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550408470.0
          },
          "Details": {
              "DocumentName": "AWS-GatherSoftwareInventory",
              "DocumentVersion": "1"
          }
      },
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Id": "e4c2ed6d-516f-41aa-aa2a-0123456789ab",
          "Title": "",
          "Status": "COMPLIANT",
          "Severity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550508475.0
          },
          "Details": {
              "DocumentName": "AWS-UpdateSSMAgent",
              "DocumentVersion": "1"
          }
      },
```

```
            ...
  ],
  "NextToken": "--token string truncated--"
}
```

So listen Sie Complianceelemente für eine bestimmte Instance und Zuordnungs-ID auf

In diesem Beispiel werden alle Konformitätselemente für die angegebene Instance und Zuordnungs-ID aufgeführt.

Befehl:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ComplianceType,Values=Association,Type=EQUAL" "Key=Id,Values=e4c2ed6d-516f-4
aa2a-0123456789ab,Type=EQUAL"
```

So listen Sie Compliance-Elemente für eine Instance nach einem bestimmten Datum und einer bestimmten Uhrzeit auf

In diesem Beispiel werden alle Compliance-Elemente für eine Instance nach dem angegebenen Datum und der angegebenen Uhrzeit aufgeführt.

Befehl:

```
aws ssm list-compliance-items --resource-ids "i-1234567890abcdef0" --resource-
types "ManagedInstance" --
filters "Key=ExecutionTime,Values=2019-02-18T16:00:00Z,Type=GREATER_THAN"
```

- Einzelheiten zur API finden Sie ListComplianceItemsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Liste der Compliance-Elemente für die angegebene Ressourcen-ID und den angegebenen Ressourcentyp aufgeführt, wobei nach dem Compliance-Typ „Association“ gefiltert wird

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Ausgabe:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- Einzelheiten zur API finden Sie unter ListComplianceItems AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Liste der Compliance-Elemente für die angegebene Ressourcen-ID und den angegebenen Ressourcentyp aufgeführt, wobei nach dem Compliance-Typ „Association“ gefiltert wird

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Ausgabe:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory],
 [DocumentVersion, 1]}
ExecutionSummary :
 Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- Einzelheiten zur API finden Sie unter ListComplianceItems AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListComplianceSummaries`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListComplianceSummaries` verwendet wird.

CLI

AWS CLI

So können Sie Konformitätszusammenfassungen für alle Compliance-Typen auflisten

In diesem Beispiel werden Konformitätszusammenfassungen für alle Compliance-Typen in Ihrem Konto aufgeführt.

Befehl:

```
aws ssm list-compliance-summaries
```

Ausgabe:

```
{
  "ComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
```

```
                }
            }
        },
        {
            "ComplianceType": "Patch",
            "CompliantSummary": {
                "CompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 1
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 1,
                "SeveritySummary": {
                    "CriticalCount": 1,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        },
                ...
    ],
    "NextToken": "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAyfQ=="
}
```

So können Sie Compliance-Zusammenfassungen für einen bestimmten Konformitätstyp auflisten

In diesem Beispiel wird die Konformitätszusammenfassung für den Konformitätstyp Patch aufgelistet.

Befehl:

```
aws ssm list-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- Einzelheiten zur API finden Sie ListComplianceSummariesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel gibt eine Sammelzahl konformer und nicht konformer Ressourcen für alle Konformitätstypen zurück.

```
Get-SSMComplianceSummaryList
```

Ausgabe:

```
ComplianceType CompliantSummary
 NonCompliantSummary
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Einzelheiten zur API finden Sie unter ListComplianceSummaries AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel gibt eine Sammelzahl konformer und nicht konformer Ressourcen für alle Konformitätstypen zurück.

```
Get-SSMComplianceSummaryList
```

Ausgabe:

```
ComplianceType CompliantSummary
 NonCompliantSummary
```

```
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- Einzelheiten zur API finden Sie unter ListComplianceSummaries AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDocumentVersions`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDocumentVersions` verwendet wird.

CLI

AWS CLI

Um Dokumentversionen aufzulisten

Das folgende `list-document-versions`-Beispiel listet alle Versionen eines Systems-Manager-Dokuments auf.

```
aws ssm list-document-versions \
    --name "Example"
```

Ausgabe:

```
{
    "DocumentVersions": [
        {
            "Name": "Example",
            "DocumentVersion": "1",
            "CreatedDate": 1583257938.266,
            "IsDefaultVersion": true,
            "DocumentFormat": "YAML",
            "Status": "Active"
```

```
        }
    ]
}
```

Weitere Informationen finden Sie unter Senden von Befehlen, die den Dokumentversionsparameter verwenden im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListDocumentVersionsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- Einzelheiten zur API finden Sie unter ListDocumentVersions AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
```

```
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- Einzelheiten zur API finden Sie unter ListDocumentVersions AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListDocuments`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListDocuments` verwendet wird.

CLI

AWS CLI

Beispiel 1: Dokumente auflisten

Das folgende `list-documents`-Beispiel listet Dokumente auf, die dem anfragenden Konto gehören und mit dem benutzerdefinierten Tag versehen sind.

```
aws ssm list-documents \
    --filters Key=Owner,Values=Self Key=tag:DocUse,Values=Testing
```

Ausgabe:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "29884EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Automation",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": [
                {
                    "Key": "DocUse",
```

```
                    "Value": "Testing"
                }
            ]
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS -Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: Geteilte Dokumente auflisten

Das folgende `list-documents` Beispiel listet gemeinsam genutzte Dokumente auf, einschließlich privater geteilter Dokumente, die nicht Eigentum von sind AWS.

```
aws ssm list-documents \
    --filters Key=Name,Values=sharedDocNamePrefix  Key=Owner,Values=Private
```

Ausgabe:

```
{
    "DocumentIdentifiers": [
        {
            "Name": "Example",
            "Owner": "12345EXAMPLE",
            "PlatformTypes": [
                "Windows",
                "Linux"
            ],
            "DocumentVersion": "1",
            "DocumentType": "Command",
            "SchemaVersion": "0.3",
            "DocumentFormat": "YAML",
            "Tags": []
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS -Systems-Manager-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ListDocumentsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Listet alle Konfigurationsdokumente in Ihrem Konto auf.

```
Get-SSMDocumentList
```

Ausgabe:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Beispiel 2: In diesem Beispiel werden alle Automatisierungsdokumente abgerufen, deren Name mit „Platform“ übereinstimmt

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Ausgabe:

```
DocumentFormat  : JSON
```

```
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Einzelheiten zur API finden Sie unter ListDocuments AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Listet alle Konfigurationsdokumente in Ihrem Konto auf.

```
Get-SSMDocumentList
```

Ausgabe:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Beispiel 2: In diesem Beispiel werden alle Automatisierungsdokumente abgerufen, deren Name mit „Platform“ übereinstimmt

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"}
 | Where-Object Name -Match "Platform"
```

Ausgabe:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- Einzelheiten zur API finden Sie unter ListDocuments AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListInventoryEntries`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListInventoryEntries` verwendet wird.

CLI

AWS CLI

Beispiel 1: So zeigen Sie bestimmte Inventartypeinträge für eine Instance an

Das folgende `list-inventory-entries` Beispiel listet die Inventareinträge für den AWS Inventartyp:Application auf einer bestimmten Instanz auf.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "AWS:Application"
```

Ausgabe:

```
{
  "TypeName": "AWS:Application",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.1",
  "CaptureTime": "2019-02-15T12:17:55Z",
  "Entries": [
    {
      "Architecture": "i386",
      "Name": "Amazon SSM Agent",
      "PackageId": "{88a60be2-89a1-4df8-812a-80863c2a2b68}",
      "Publisher": "Amazon Web Services",
      "Version": "2.3.274.0"
    },
    {
      "Architecture": "x86_64",
      "InstalledTime": "2018-05-03T13:42:34Z",
      "Name": "AmazonCloudWatchAgent",
      "Publisher": "",
      "Version": "1.200442.0"
    }
  ]
}
```

Beispiel 2: So zeigen Sie benutzerdefinierte Inventareinträge an, die einer Instance zugewiesen sind

Das folgende `list-inventory-entries`-Beispiel listet einen benutzerdefinierten Inventareintrag auf, der einer Instance zugewiesen ist.

```
aws ssm list-inventory-entries \
    --instance-id "i-1234567890abcdef0" \
    --type-name "Custom:RackInfo"
```

Ausgabe:

```
{
  "TypeName": "Custom:RackInfo",
  "InstanceId": "i-1234567890abcdef0",
  "SchemaVersion": "1.0",
  "CaptureTime": "2021-05-22T10:01:01Z",
  "Entries": [
    {
```

```
      "RackLocation": "Bay B/Row C/Rack D/Shelf E"
    }
  ]
}
```

- Einzelheiten zur API finden Sie unter ListInventoryEntries AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden alle benutzerdefinierten Inventareinträge für eine Instance aufgeführt.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Ausgabe:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Beispiel 2: In diesem Beispiel werden die Details aufgeführt.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Ausgabe:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Einzelheiten zur API finden Sie unter ListInventoryEntries AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden alle benutzerdefinierten Inventareinträge für eine Instance aufgeführt.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Ausgabe:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Beispiel 2: In diesem Beispiel werden die Details aufgeführt.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Ausgabe:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- Einzelheiten zur API finden Sie unter ListInventoryEntries AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListResourceComplianceSummaries`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListResourceComplianceSummaries` verwendet wird.

CLI

AWS CLI

So listen Sie die Anzahl der Compliance-Anforderungen auf Ressourcenebene auf

In diesem Beispiel werden die Compliance-Zusammenfassungszahlen auf Ressourcenebene aufgelistet.

Befehl:

```
aws ssm list-resource-compliance-summaries
```

Ausgabe:

```
{
  "ResourceComplianceSummaryItems": [
      {
          "ComplianceType": "Association",
          "ResourceType": "ManagedInstance",
          "ResourceId": "i-1234567890abcdef0",
          "Status": "COMPLIANT",
          "OverallSeverity": "UNSPECIFIED",
          "ExecutionSummary": {
              "ExecutionTime": 1550509273.0
          },
          "CompliantSummary": {
              "CompliantCount": 2,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 2
              }
          },
          "NonCompliantSummary": {
              "NonCompliantCount": 0,
              "SeveritySummary": {
                  "CriticalCount": 0,
                  "HighCount": 0,
                  "MediumCount": 0,
                  "LowCount": 0,
                  "InformationalCount": 0,
                  "UnspecifiedCount": 0
              }
          }
      },
```

```
        {
            "ComplianceType": "Patch",
            "ResourceType": "ManagedInstance",
            "ResourceId": "i-9876543210abcdef0",
            "Status": "COMPLIANT",
            "OverallSeverity": "UNSPECIFIED",
            "ExecutionSummary": {
                "ExecutionTime": 1550248550.0,
                "ExecutionId": "7abb6378-a4a5-4f10-8312-0123456789ab",
                "ExecutionType": "Command"
            },
            "CompliantSummary": {
                "CompliantCount": 397,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 397
                }
            },
            "NonCompliantSummary": {
                "NonCompliantCount": 0,
                "SeveritySummary": {
                    "CriticalCount": 0,
                    "HighCount": 0,
                    "MediumCount": 0,
                    "LowCount": 0,
                    "InformationalCount": 0,
                    "UnspecifiedCount": 0
                }
            }
        }
    ],
    "NextToken": "--token string truncated--"
}
```

So listen Sie die Compliance-Zusammenfassungen auf Ressourcenebene für einen bestimmten Konformitätstyp auf

In diesem Beispiel werden Konformitätszusammenfassungen auf Ressourcenebene für den Kompatibilitätstyp Patch aufgeführt.

Befehl:

```
aws ssm list-resource-compliance-summaries --
filters "Key=ComplianceType,Values=Patch,Type=EQUAL"
```

- Einzelheiten zur API finden Sie ListResourceComplianceSummariesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Zusammenfassung der Anzahl auf Ressourcenebene abgerufen. Die Zusammenfassung enthält Informationen über den Status „konform“ und „nicht konform“ sowie detaillierte Angaben zum Schweregrad von Produkten, die „Windows10“ entsprechen. Da der MaxResult Standardwert 100 ist, wenn der Parameter nicht angegeben ist und dieser Wert nicht gültig ist, wird der MaxResult Parameter hinzugefügt und der Wert auf 50 gesetzt.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Einzelheiten zur API finden Sie unter ListResourceComplianceSummaries AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Zusammenfassung der Anzahl auf Ressourcenebene abgerufen. Die Zusammenfassung enthält Informationen über den Status „konform“ und „nicht konform“ sowie detaillierte Angaben zum Schweregrad von Produkten, die „Windows10“ entsprechen. Da der MaxResult Standardwert 100 ist, wenn der Parameter nicht angegeben ist und dieser Wert nicht gültig ist, wird der MaxResult Parameter hinzugefügt und der Wert auf 50 gesetzt.

```
$FilterValues = @{
  "Key"="Product"
```

```
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- Einzelheiten zur API finden Sie unter ListResourceComplianceSummaries AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ListTagsForResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ListTagsForResource` verwendet wird.

CLI

AWS CLI

So listen Sie die auf eine Patch-Baseline angewendeten Tags auf

Das folgende `list-tags-for-resource`-Beispiel listet die Tags für eine Patch-Baseline auf.

```
aws ssm list-tags-for-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0"
```

Ausgabe:

```
{
    "TagList": [
        {
            "Key": "Environment",
            "Value": "Production"
        },
        {
            "Key": "Region",
            "Value": "EMEA"
        }
    ]
}
```

Weitere Informationen finden Sie unter AWS Ressourcen taggen in der AWS allgemeinen Referenz.

- Einzelheiten zur API finden Sie ListTagsForResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel listet die Tags für ein Wartungsfenster auf.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Ausgabe:

```
Key   Value
---   -----
Stack Production
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel listet die Tags für ein Wartungsfenster auf.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Ausgabe:

```
Key   Value
---   -----
Stack Production
```

- Einzelheiten zur API finden Sie unter ListTagsForResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyDocumentPermission`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyDocumentPermission` verwendet wird.

CLI

AWS CLI

So können Sie Dokumentberechtigungen ändern

Im folgenden `modify-document-permission`-Beispiel wird ein Systems-Manager-Dokument öffentlich freigegeben.

```
aws ssm modify-document-permission \
    --name "Example" \
    --permission-type "Share" \
    --account-ids-to-add "All"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Systems-Manager-Dokument teilen im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie ModifyDocumentPermissionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden allen Konten für ein Dokument „Teilen“-Berechtigungen hinzugefügt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Beispiel 2: Dieses Beispiel fügt einem bestimmten Konto die Berechtigung zum „Freigeben“ eines Dokuments hinzu. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Einzelheiten zur API finden Sie unter ModifyDocumentPermission AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden allen Konten für ein Dokument „Teilen“-Berechtigungen hinzugefügt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Beispiel 2: Dieses Beispiel fügt einem bestimmten Konto die Berechtigung zum „Freigeben“ eines Dokuments hinzu. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- Einzelheiten zur API finden Sie unter ModifyDocumentPermission AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutComplianceItems`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutComplianceItems` verwendet wird.

CLI

AWS CLI

Um einen Compliance-Typ und Compliance-Details zu einer bestimmten Instance zu registrieren

In diesem Beispiel wird der Konformitätstyp `Custom:AVCheck` für die angegebene verwaltete Instance registriert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

Befehl:

```
aws ssm put-compliance-items --resource-id "i-1234567890abcdef0" --
resource-type "ManagedInstance" --compliance-type "Custom:AVCheck"
 --execution-summary "ExecutionTime=2019-02-18T16:00:00Z" --
items "Id=Version2.0,Title=ScanHost,Severity=CRITICAL,Status=COMPLIANT"
```

- Einzelheiten zur API finden Sie PutComplianceItemsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein benutzerdefiniertes Compliance-Element für die angegebene verwaltete Instance geschrieben

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Einzelheiten zur API finden Sie unter PutComplianceItems AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein benutzerdefiniertes Compliance-Element für die angegebene verwaltete Instance geschrieben

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- Einzelheiten zur API finden Sie unter PutComplianceItems AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`PutInventory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutInventory` verwendet wird.

CLI

AWS CLI

So weisen Sie einer Instance Kundenmetadaten zu

In diesem Beispiel werden einer Instance Informationen zum Rack-Standort zugewiesen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

Befehl (Linux):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --items
 '[{"TypeName": "Custom:RackInfo","SchemaVersion": "1.0","CaptureTime":
 "2019-01-22T10:01:01Z","Content":[{"RackLocation": "Bay B/Row C/Rack D/Shelf
 E"}]}]'
```

Befehl (Windows):

```
aws ssm put-inventory --instance-id "i-016648b75dd622dab" --
items "TypeName=Custom:RackInfo,SchemaVersion=1.0,CaptureTime=2019-01-22T10:01:01Z,Conten
 B/Row C/Rack D/Shelf F'}]"
```

- Einzelheiten zur API finden Sie PutInventoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden einer Instance Informationen zum Rack-Standort zugewiesen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
```

```
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Einzelheiten zur API finden Sie unter PutInventory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden einer Instance Informationen zum Rack-Standort zugewiesen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")

$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object
 Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- Einzelheiten zur API finden Sie unter PutInventory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`PutParameter`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `PutParameter` verwendet wird.

CLI

AWS CLI

Beispiel 1: So ändern Sie einen Parameterwert

Das folgende Beispiel `put-parameter` ändert den Wert des angegebenen Parameters.

```
aws ssm put-parameter \
    --name "MyStringParameter" \
    --type "String" \
    --value "Vici" \
    --overwrite
```

Ausgabe:

```
{
    "Version": 2,
    "Tier": "Standard"
}
```

Weitere Informationen finden Sie unter Erstellen eines Systems Manager Manager-Parameters (AWS CLI), Verwalten von Parameterschichten und Arbeiten mit Parameterrichtlinien im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 2: So erstellen Sie einen erweiterten Parameter

Das folgende Beispiel `put-parameter` erstellt einen erweiterten Parameter.

```
aws ssm put-parameter \
    --name "MyAdvancedParameter" \
    --description "This is an advanced parameter" \
    --value "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
 veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat [truncated]" \
    --type "String" \
    --tier Advanced
```

Ausgabe:

```
{
```

```
    "Version": 1,
    "Tier": "Advanced"
}
```

Weitere Informationen finden Sie unter Erstellen eines Systems Manager Manager-Parameters (AWS CLI), Verwalten von Parameterschichten und Arbeiten mit Parameterrichtlinien im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 3: So konvertieren Sie einen Standardparameter in einen erweiterten Parameter

Im folgenden Beispiel für `put-parameter` wird ein vorhandener Standardparameter in einen erweiterten Parameter konvertiert.

```
aws ssm put-parameter \
    --name "MyConvertedParameter" \
    --value "abc123" \
    --type "String" \
    --tier Advanced \
    --overwrite
```

Ausgabe:

```
{
    "Version": 2,
    "Tier": "Advanced"
}
```

Weitere Informationen finden Sie unter Erstellen eines Systems Manager Manager-Parameters (AWS CLI), Verwalten von Parameterschichten und Arbeiten mit Parameterrichtlinien im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 4: So erstellen Sie einen Parameter mit angehängter Richtlinie

Im folgenden `put-parameter`-Beispiel wird ein erweiterter Parameter mit einer angehängten Parameterrichtlinie erstellt.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "P@sSwW)rd" \
    --type "SecureString" \
```

```
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
```

Ausgabe:

```
{
    "Version": 1,
    "Tier": "Advanced"
}
```

Weitere Informationen finden Sie unter Erstellen eines Systems Manager Manager-Parameters (AWS CLI), Verwalten von Parameterschichten und Arbeiten mit Parameterrichtlinien im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 5: So fügen Sie einem vorhandenen Parameter eine Richtlinie hinzu

Im folgenden `put-parameter`-Beispiel wird eine Richtlinie einem vorhandenen erweiterten Parameter angefügt.

```
aws ssm put-parameter \
    --name "/Finance/Payroll/q2accesskey" \
    --value "N3wP@sSwW)rd" \
    --type "SecureString" \
    --tier Advanced \
    --policies "[{\"Type\":\"Expiration\",\"Version\":\"1.0\",\"Attributes\":
{\"Timestamp\":\"2020-06-30T00:00:00.000Z\"}},{\"Type\":\"ExpirationNotification
\",\"Version\":\"1.0\",\"Attributes\":{\"Before\":\"5\",\"Unit\":\"Days\"}},
{\"Type\":\"NoChangeNotification\",\"Version\":\"1.0\",\"Attributes\":{\"After\":
\"60\",\"Unit\":\"Days\"}}]"
    --overwrite
```

Ausgabe:

```
{
    "Version": 2,
    "Tier": "Advanced"
}
```

Weitere Informationen finden Sie unter Erstellen eines Systems Manager Manager-Parameters (AWS CLI), Verwalten von Parameterschichten und Arbeiten mit Parameterrichtlinien im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie PutParameterin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        String paraValue = args[1];
```

```
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }

    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie PutParameterin der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Parameter erstellt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Beispiel 2: In diesem Beispiel wird ein Parameter geändert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Einzelheiten zur API finden Sie unter PutParameter AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Parameter erstellt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Beispiel 2: In diesem Beispiel wird ein Parameter geändert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- Einzelheiten zur API finden Sie unter PutParameter AWS -Tools für PowerShellCmdlet-Referenz (V5).

Rust

SDK für Rust

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
```

```
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());

    Ok(())
}
```

- Einzelheiten zur API finden Sie PutParameterin der API-Referenz zum AWS SDK für Rust.

Verwendung von **`RegisterDefaultPatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterDefaultPatchBaseline` verwendet wird.

CLI

AWS CLI

Festlegen der Standard-Patch-Baseline

Im folgenden Beispiel `register-default-patch-baseline` wird die angegebene benutzerdefinierte Patch-Baseline als Standard-Patch-Baseline für den unterstützten Betriebssystemtyp registriert.

```
aws ssm register-default-patch-baseline \
    --baseline-id "pb-abc123cf9bEXAMPLE"
```

Ausgabe:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

Im folgenden `register-default-patch-baseline` Beispiel wird die von AWS für CentOS bereitgestellte Standard-Patch-Baseline als Standard-Patch-Baseline registriert.

```
aws ssm register-default-patch-baseline \
    --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/
pb-0574b43a65ea646ed"
```

Ausgabe:

```
{
    "BaselineId":"pb-abc123cf9bEXAMPLE"
}
```

Weitere Informationen finden Sie unter Über vordefinierte und benutzerdefinierte Patch-Baselines im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter RegisterDefaultPatchBaseline AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline als Standard-Patch-Baseline registriert.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
pb-03da896ca3b68b639
```

- Einzelheiten zur API finden Sie unter RegisterDefaultPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline als Standard-Patch-Baseline registriert.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
pb-03da896ca3b68b639
```

- Einzelheiten zur API finden Sie unter RegisterDefaultPatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterPatchBaselineForPatchGroup`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterPatchBaselineForPatchGroup` verwendet wird.

CLI

AWS CLI

So registrieren Sie eine Patch-Baseline für eine Patch-Gruppe

Im folgenden `register-patch-baseline-for-patch-group`-Beispiel wird eine Patch-Baseline für eine Patch-Gruppe registriert.

```
aws ssm register-patch-baseline-for-patch-group \
    --baseline-id "pb-045f10b4f382baeda" \
    --patch-group "Production"
```

Ausgabe:

```
{
    "BaselineId": "pb-045f10b4f382baeda",
    "PatchGroup": "Production"
}
```

Weitere Informationen finden Sie unter Erstellen einer Patchgruppe < https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-group-tagging.html >__ und Hinzufügen einer Patchgruppe zu einer Patch-Baseline im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie unter RegisterPatchBaselineForPatchGroup AWS CLIBefehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel registriert eine Patch-Baseline für eine Patch-Gruppe.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Einzelheiten zur API finden Sie unter RegisterPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel registriert eine Patch-Baseline für eine Patch-Gruppe.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- Einzelheiten zur API finden Sie unter RegisterPatchBaselineForPatchGroup AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterTargetWithMaintenanceWindow`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterTargetWithMaintenanceWindow` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie ein einzelnes Ziel mit einem Wartungsfenster registrieren

Im folgenden Beispiel `register-target-with-maintenance-window` wird eine Instance mit einem Wartungsfenster registriert.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862" \
    --owner-information "Single instance" \
    --resource-type "INSTANCE"
```

Ausgabe:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Beispiel 2: So können Sie mehrere Ziele mithilfe von Instance-IDs für ein Wartungsfenster registrieren

Im folgenden Beispiel `register-target-with-maintenance-window` werden zwei Instances mit einem Wartungsfenster registriert, indem ihre Instance-IDs angegeben werden.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-ab12cd34ef56gh78" \
    --target "Key=InstanceIds,Values=i-0000293ffd8c57862,i-0cb2b964d3e14fd9f" \
    --owner-information "Two instances in a list" \
    --resource-type "INSTANCE"
```

Ausgabe:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Beispiel 3: So registrieren Sie Ziele mit einem Wartungsfenster mithilfe von Ressourcentags

Im folgenden Beispiel `register-target-with-maintenance-window` werden Instances mit einem Wartungsfenster registriert, indem Ressourcen-Tags angegeben werden, die auf die Instances angewendet wurden.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-06cf17cbefcb4bf4f" \
    --targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" \
```

```
    --owner-information "Production Web Servers" \
    --resource-type "INSTANCE"
```

Ausgabe:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Beispiel 4: So registrieren Sie Ziele mithilfe einer Gruppe von Tag-Schlüsseln

Im folgenden `register-target-with-maintenance-window`-Beispiel werden Instances registriert, denen unabhängig von ihren Schlüsselwerten jeweils ein oder mehrere Tag-Schlüssel zugewiesen sind.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

Ausgabe:

```
{
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Beispiel 5: Registrieren von Zielen unter Verwendung eines Ressourcengruppennamens

Das folgende `register-target-with-maintenance-window`-Beispiel registriert eine angegebene Ressourcengruppe, unabhängig vom Typ der darin enthaltenen Ressourcen.

```
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "RESOURCE_GROUP" \
    --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

Ausgabe:

```
{
```

```
    "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

Weitere Informationen finden Sie unter Registrieren einer Zielinstanz mit dem Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie RegisterTargetWithMaintenanceWindowin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Beispiel 2: In diesem Beispiel werden mehrere Instances mit einem Wartungsfenster registriert.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Beispiel 3: In diesem Beispiel wird mithilfe von EC2-Tags eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
```

```
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Ausgabe:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Einzelheiten zur API finden Sie unter RegisterTargetWithMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Beispiel 2: In diesem Beispiel werden mehrere Instances mit einem Wartungsfenster registriert.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Beispiel 3: In diesem Beispiel wird mithilfe von EC2-Tags eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
```

```
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Ausgabe:

```
2994977e-aefb-4a71-beac-df620352f184
```

- Einzelheiten zur API finden Sie unter RegisterTargetWithMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterTaskWithMaintenanceWindow`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterTaskWithMaintenanceWindow` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um eine Automation-Aufgabe mit einem Wartungsfenster zu registrieren

Im folgenden Beispiel `register-task-with-maintenance-window` wird eine Automation-Aufgabe mit einem Wartungsfenster registriert, das auf eine Instance ausgerichtet ist.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649EXAMPLE" \
    --targets Key=InstanceIds,Values=i-1234520122EXAMPLE \
    --task-arn AWS-RestartEC2Instance \
    --service-role-arn arn:aws:iam::111222333444:role/SSM --task-type AUTOMATION
 \
    --task-invocation-parameters "{\"Automation\":{\"DocumentVersion\":\"\$LATEST
\",\"Parameters\":{\"InstanceId\":[\"{{RESOURCE_ID}}\"]}}}" \
    --priority 0 \
    --max-concurrency 1 \
    --max-errors 1 \
    --name "AutomationExample" \
    --description "Restarting EC2 Instance for maintenance"
```

Ausgabe:

```
{
```

```
    "WindowTaskId":"11144444-5555-6666-7777-88888888"
}
```

Weitere Informationen finden Sie unter Registrieren einer Aufgabe im Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 2: So registrieren Sie eine Lambda-Aufgabe mit einem Wartungsfenster

Im folgenden Beispiel `register-task-with-maintenance-window` wird eine Lambda-Aufgabe mit einem Wartungsfenster registriert, das auf eine Instance ausgerichtet ist.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649dee04e4" \
    --targets Key=InstanceIds,Values=i-12344d305eEXAMPLE \
    --task-arn arn:aws:lambda:us-east-1:111222333444:function:SSMTestLAMBDA \
    --service-role-arn arn:aws:iam::111222333444:role/SSM \
    --task-type LAMBDA \
    --task-invocation-parameters '{"Lambda":{"Payload":"{\"InstanceId\":
\"{{RESOURCE_ID}}\",\"targetType\":\"{{TARGET_TYPE}}\"}","Qualifier":"$LATEST"}}'
 \
    --priority 0 \
    --max-concurrency 10 \
    --max-errors 5 \
    --name "Lambda_Example" \
    --description "My Lambda Example"
```

Ausgabe:

```
{
    "WindowTaskId":"22244444-5555-6666-7777-88888888"
}
```

Weitere Informationen finden Sie unter Registrieren einer Aufgabe im Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 3: So registrieren Sie eine Run-Command-Aufgabe mit einem Wartungsfenster

Im folgenden Beispiel `register-task-with-maintenance-window` wird eine Run-Command-Aufgabe mit einem Wartungsfenster registriert, das auf eine Instance ausgerichtet ist.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-082dcd7649dee04e4" \
    --targets "Key=InstanceIds,Values=i-12344d305eEXAMPLE" \
    --service-role-arn "arn:aws:iam::111222333444:role/SSM" \
    --task-type "RUN_COMMAND" \
    --name "SSMInstallPowerShellModule" \
    --task-arn "AWS-InstallPowerShellModule" \
    --task-invocation-parameters "{\"RunCommand\":{\"Comment\":\"\",
\"OutputS3BucketName\":\"runcommandlogs\",\"Parameters\":{\"commands\":[\"Get-
Module -ListAvailable\"],\"executionTimeout\":[\"3600\"],\"source\":[\"https:\/
\/gallery.technet.microsoft.com\/EZOut-33ae0fb7\/file\/110351\/1\/EZOut.zip\"],
\"workingDirectory\":[\"\\\\\"]},\"TimeoutSeconds\":600}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Ausgabe:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
}
```

Weitere Informationen finden Sie unter Registrieren einer Aufgabe im Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 4: So registrieren Sie eine Step-Functions-Aufgabe bei einem Wartungsfenster

Im folgenden Beispiel `register-task-with-maintenance-window` wird eine Step-Functions-Aufgabe mit einem Wartungsfenster registriert, das auf eine Instance ausgerichtet ist.

```
aws ssm register-task-with-maintenance-window \
    --window-id "mw-1234d787d6EXAMPLE" \
    --targets Key=WindowTargetIds,Values=12347414-69c3-49f8-95b8-ed2dcEXAMPLE \
    --task-arn arn:aws:states:us-
east-1:111222333444:stateMachine:SSMTestStateMachine \
    --service-role-arn arn:aws:iam::111222333444:role/MaintenanceWindows \
    --task-type STEP_FUNCTIONS \
    --task-invocation-parameters '{"StepFunctions":{"Input":"{\"InstanceId\":
\"{{RESOURCE_ID}}\"}"}}' \
    --priority 0 \
    --max-concurrency 10 \
```

```
    --max-errors 5 \
    --name "Step_Functions_Example" \
    --description "My Step Functions Example"
```

Ausgabe:

```
{
    "WindowTaskId":"44444444-5555-6666-7777-88888888"
}
```

Weitere Informationen finden Sie unter Registrieren einer Aufgabe im Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

Beispiel 5: So registrieren Sie eine Aufgabe mithilfe einer Windows-Wartungsziel-ID

Das folgende `register-task-with-maintenance-window`-Beispiel registriert eine Aufgabe mithilfe einer Wartungsfenster-Ziel-ID. Die Ziel-ID des Wartungsfensters war in der Ausgabe des `aws ssm register-target-with-maintenance-window`-Befehls enthalten. Sie können sie auch aus der Ausgabe des `aws ssm describe-maintenance-window-targets`-Befehls abrufen.

```
aws ssm register-task-with-maintenance-window \
    --targets "Key=WindowTargetIds,Values=350d44e6-28cc-44e2-951f-4b2c9EXAMPLE" \
    --task-arn "AWS-RunShellScript" \
    --service-role-arn "arn:aws:iam::111222333444:role/MaintenanceWindowsRole" \
    --window-id "mw-ab12cd34eEXAMPLE" \
    --task-type "RUN_COMMAND" \
    --task-parameters  "{\"commands\":{\"Values\":[\"df\"]}}" \
    --max-concurrency 1 \
    --max-errors 1 \
    --priority 10
```

Ausgabe:

```
{
    "WindowTaskId":"33344444-5555-6666-7777-88888888"
}
```

Weitere Informationen finden Sie unter Registrieren einer Aufgabe im Wartungsfenster (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie RegisterTaskWithMaintenanceWindowin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Instance-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 2: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Ziel-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 3: In diesem Beispiel wird ein Parameterobjekt für das Run-Befehlsdokument **`AWS-RunPowerShellScript`** und eine Aufgabe mit einem bestimmten Wartungsfenster unter Verwendung der Ziel-ID erstellt. Die Rückgabeausgabe ist die Aufgaben-ID.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Ausgabe:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Beispiel 4: In diesem Beispiel wird eine AWS Systems Manager Automation-Aufgabe mithilfe eines Dokuments mit dem Namen registriert**`Create-Snapshots`**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
```

```
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Einzelheiten zur API finden Sie unter RegisterTaskWithMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Instance-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 2: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Ziel-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)
```

```
register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -
TaskType "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 3: In diesem Beispiel wird ein Parameterobjekt für das Run-Befehlsdokument **`AWS-RunPowerShellScript`** und eine Aufgabe mit einem bestimmten Wartungsfenster unter Verwendung der Ziel-ID erstellt. Die Rückgabeausgabe ist die Aufgaben-ID.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target =
 @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Ausgabe:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Beispiel 4: In diesem Beispiel wird eine AWS Systems Manager Automation-Aufgabe mithilfe eines Dokuments mit dem Namen registriert**`Create-Snapshots`**.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- Einzelheiten zur API finden Sie unter RegisterTaskWithMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RemoveTagsFromResource`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RemoveTagsFromResource` verwendet wird.

CLI

AWS CLI

Entfernen eines Tags aus einer Patch-Baseline

Das folgende `remove-tags-from-resource`-Beispiel entfernt Tags aus einer Patch-Baseline.

```
aws ssm remove-tags-from-resource \
    --resource-type "PatchBaseline" \
    --resource-id "pb-0123456789abcdef0" \
    --tag-keys "Region"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter AWS Ressourcen taggen in der AWS allgemeinen Referenz.

- Einzelheiten zur API finden Sie RemoveTagsFromResourcein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Tag aus einem Wartungsfenster entfernt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromResource AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Tag aus einem Wartungsfenster entfernt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- Einzelheiten zur API finden Sie unter RemoveTagsFromResource AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`SendCommand`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `SendCommand` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Beispiel 1: So führen Sie einen Befehl auf einer oder mehreren Remote-Instances aus

Im folgenden Beispiel `send-command` wird ein `echo`-Befehl auf einer Ziel-Instance ausgeführt.

```
aws ssm send-command \
    --document-name "AWS-RunShellScript" \
    --parameters 'commands=["echo HelloWorld"]' \
    --targets "Key=instanceids,Values=i-1234567890abcdef0" \
    --comment "echo HelloWorld"
```

Ausgabe:

```
{
    "Command": {
        "CommandId": "92853adf-ba41-4cd6-9a88-142d1EXAMPLE",
        "DocumentName": "AWS-RunShellScript",
        "DocumentVersion": "",
        "Comment": "echo HelloWorld",
        "ExpiresAfter": 1550181014.717,
        "Parameters": {
            "commands": [
                "echo HelloWorld"
            ]
        },
        "InstanceIds": [
            "i-0f00f008a2dcbefe2"
        ],
        "Targets": [],
        "RequestedDateTime": 1550173814.717,
        "Status": "Pending",
        "StatusDetails": "Pending",
        "OutputS3BucketName": "",
        "OutputS3KeyPrefix": "",
        "MaxConcurrency": "50",
        "MaxErrors": "0",
        "TargetCount": 1,
        "CompletedCount": 0,
        "ErrorCount": 0,
        "DeliveryTimedOutCount": 0,
        "ServiceRole": "",
        "NotificationConfig": {
            "NotificationArn": "",
            "NotificationEvents": [],
            "NotificationType": ""
        },
        "CloudWatchOutputConfig": {
            "CloudWatchLogGroupName": "",
```

```
            "CloudWatchOutputEnabled": false
        }
    }
}
```

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 2: Um IP-Informationen über eine Instanz abzurufen

Im folgenden `send-command`-Beispiel werden Informationen über die Instances zurückgegeben.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 3: So führen Sie einen Befehl für Instances mit bestimmten Tags aus

Im folgenden Beispiel `send-command` wird ein Befehl auf Instances ausgeführt, die den Tag-Schlüssel „ENV“ und den Wert „Dev“ haben.

```
aws ssm send-command \
    --targets "Key=tag:ENV,Values=Dev" \
    --document-name "AWS-RunShellScript" \
    --parameters "commands=ifconfig"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 4: So führen Sie einen Befehl aus, der SNS-Benachrichtigungen sendet

Im folgenden Beispiel `send-command` wird ein Befehl ausgeführt, der SNS-Benachrichtigungen für alle Benachrichtigungsereignisse und den `Command`-Benachrichtigungstyp sendet.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --service-role-arn "arn:aws:iam::123456789012:role/SNS_Role" \
    --notification-config "NotificationArn=arn:aws:sns:us-
east-1:123456789012:SNSTopicName,NotificationEvents=All,NotificationType=Command"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 5: Um einen Befehl auszuführen, der an S3 ausgegeben wird und CloudWatch

Im folgenden `send-command` Beispiel wird ein Befehl ausgeführt, der Befehlsdetails an einen S3-Bucket und eine CloudWatch Logs-Protokollgruppe ausgibt.

```
aws ssm send-command \
    --instance-ids "i-1234567890abcdef0" \
    --document-name "AWS-RunShellScript" \
    --comment "IP config" \
    --parameters "commands=ifconfig" \
    --output-s3-bucket-name "s3-bucket-name" \
    --output-s3-key-prefix "runcommand" \
    --cloud-watch-output-
config "CloudWatchOutputEnabled=true,CloudWatchLogGroupName=CWLGroupName"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 6: So führen Sie Befehle auf mehreren Instances mit unterschiedlichen Tags aus

Im folgenden Beispiel `send-command` wird ein Befehl für Instances mit zwei verschiedenen Tag-Schlüsseln und -Werten ausgeführt.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev Key=tag:Role,Values=WebServers
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 7: So zielen Sie mehrere Instances mit demselben Tag-Schlüssel als Ziel ab

Im folgenden Beispiel `send-command` wird ein Befehl für Instances ausgeführt, die denselben Tag-Schlüssel, aber unterschiedliche Werte haben.

```
aws ssm send-command \
    --document-name "AWS-RunPowerShellScript" \
    --parameters commands=["echo helloWorld"] \
    --targets Key=tag:Env,Values=Dev,Test
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Run-Befehl mithilfe von Systems Manager in der AWS - Systems-Manager-Benutzerhandbuch.

Beispiel 8: So führen Sie einen Befehl aus, der ein freigegebenes Dokument verwendet

Im folgenden Beispiel `send-command` wird ein gemeinsam verwendetes Dokument auf einer Ziel-Instance ausgeführt.

```
aws ssm send-command \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
 \
    --targets "Key=instanceids,Values=i-1234567890abcdef0"
```

Eine Beispielausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter Verwenden von geteilten SSM-Dokumenten im AWS - Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie SendCommandunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture =
CompletableFuture.runAsync(() -> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus =
response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
```

```
                    System.out.println("The SSM document is not active. Status: "
+ documentStatus);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        });

        documentActiveFuture.join();

        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse>
invocationFuture = getAsyncClient().getCommandInvocation(invocationRequest);
```

```
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution
successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- Einzelheiten zur API finden Sie SendCommandin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendCommandCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Send an SSM command to a managed node.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new SendCommandCommand({
        DocumentName: documentName,
      }),
    );
    console.log("Command sent successfully.");
    return { Success: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid document name?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie SendCommandin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Echo-Befehl auf einer Ziel-Instance ausgeführt.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Ausgabe:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Beispiel 2: Dieses Beispiel zeigt, wie ein Befehl ausgeführt wird, der verschachtelte Parameter akzeptiert.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Einzelheiten zur API finden Sie unter SendCommand AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Echo-Befehl auf einer Ziel-Instance ausgeführt.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Ausgabe:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Beispiel 2: Dieses Beispiel zeigt, wie ein Befehl ausgeführt wird, der verschachtelte Parameter akzeptiert.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- Einzelheiten zur API finden Sie unter SendCommand AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
 TimeoutSeconds=3600
            )
```

```
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie SendCommandin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ssm->sendcommand(
            it_instanceids = it_instance_ids
            iv_documentname = iv_document_name
            iv_timeoutseconds = 3600 ).
        DATA(lo_command) = lo_result->get_command( ).
        IF lo_command IS BOUND.
          rv_command_id = lo_command->get_commandid( ).
          MESSAGE 'Command sent successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie SendCommandin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`StartAutomationExecution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StartAutomationExecution` verwendet wird.

CLI

AWS CLI

Beispiel 1: So führen Sie ein Automation-Dokument aus

Im folgenden Beispiel `start-automation-execution` wird ein Automation-Dokument ausgeführt.

```
aws ssm start-automation-execution \
    --document-name "AWS-UpdateLinuxAmi" \
    --parameters "AutomationAssumeRole=arn:aws:iam::123456789012:role/
SSMAutomationRole,SourceAmiId=ami-EXAMPLE,IamInstanceProfileName=EC2InstanceRole"
```

Ausgabe:

```
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

Weitere Informationen finden Sie unter Manuelles Ausführen eines Automation-Workflows im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So führen Sie ein gemeinsam genutztes Automatisierungsdokument aus

Im folgenden Beispiel `start-automation-execution` wird ein gemeinsam genutztes Automation-Dokument ausgeführt.

```
aws ssm start-automation-execution \
    --document-name "arn:aws:ssm:us-east-1:123456789012:document/ExampleDocument"
```

Ausgabe:

```
{
```

```
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
}
```

Weitere Informationen finden Sie unter Verwenden von geteilten SSM-Dokumenten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie StartAutomationExecutionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird ein Dokument ausgeführt, das eine Automation-Rolle, eine AMI-Quell-ID und eine Amazon-EC2-Instance-Rolle angibt.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Ausgabe:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Einzelheiten zur API finden Sie unter StartAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird ein Dokument ausgeführt, das eine Automation-Rolle, eine AMI-Quell-ID und eine Amazon-EC2-Instance-Rolle angibt.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-
f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Ausgabe:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- Einzelheiten zur API finden Sie unter StartAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StartSession`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StartSession` verwendet wird.

CLI

AWS CLI

Beispiel 1: So starten Sie eine Session-Manager-Sitzung

Dieses `start-session`-Beispiel initiiert eine Verbindung zu einer Instance für eine Session-Manager-Sitzung. Beachten Sie, dass für diesen interaktiven Befehl das Session-Manager-Plugin auf dem Client-Computer installiert sein muss, der den Anruf durchführt.

```
aws ssm start-session \
    --target "i-1234567890abcdef0"
```

Ausgabe:

```
Starting session with SessionId: Jane-Roe-07a16060613c408b5
```

Beispiel 2: So starten Sie eine Session-Manager-Sitzung mit SSH

Dieses `start-session`-Beispiel initiiert eine Verbindung zu einer Instance für eine Session-Manager-Sitzung mit SSH. Beachten Sie, dass für diesen interaktiven Befehl das Session-Manager-Plugin auf dem Client-Computer installiert sein muss, der den Anruf durchführt, und dass der Befehl den Standardbenutzer auf der Instance verwendet, z. B. `ec2-user` für EC2-Instances für Linux.

```
ssh -i /path/my-key-pair.pem ec2-user@i-02573cafcfEXAMPLE
```

Ausgabe:

```
Starting session with SessionId: ec2-user-07a16060613c408b5
```

Weitere Informationen finden Sie unter Starten einer Sitzung und Installieren des Session Manager-Plug-ins für die AWS CLI im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie StartSessionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Verbindung zu einem Ziel für eine Session-Manager-Sitzung initiiert, wodurch die Portweiterleitung aktiviert wird.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' }
```

Ausgabe:

```
SessionId    StreamUrl
----------    ----------
random-id0    wss://ssmmessages.amazonaws.com/v1/data-channel/random-id
```

- Einzelheiten zur API finden Sie unter StartSession AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Verbindung zu einem Ziel für eine Session-Manager-Sitzung initiiert, wodurch die Portweiterleitung aktiviert wird. Hinweis: Sie müssen den Regionsparameter hinzufügen, falls er nicht bereits mit AWSCredentials festgelegt wurde.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' } -Region 'us-west-1'
```

Ausgabe:

```
Starting session with SessionId: testUser-xi4glew849asyeryde34u4dfsdfy
Port 8080 opened for sessionId testUser-xi4glew849asyeryde34u4dfsdfy.
Waiting for connections...
```

Beispiel 2: Hier wird eine interaktive Sitzung mit einer angegebenen Instance für eine Session-Manager-Sitzung erstellt.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1'
```

Ausgabe:

```
Starting session with SessionId : testUser-xi4glew849asyeryde34u4dfsdfy
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.



Install the latest PowerShell for new features and improvements!


PS C:\Windows\system32> whoami
ec2amaz-fnsdrwv\ec2-test-user
PS C:\Windows\system32>
```

Beispiel 3: Dieses Beispiel erstellt eine Sitzung, ohne eine Verbindung zu ihr herzustellen, und gibt die TokenValue Eigenschaften SessionId, und zurück StreamUrl, die für die Verbindung mit der Sitzung erforderlich sind.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1' -
DisablePluginInvocation
```

Ausgabe:

```
SessionId        : testUser-xi4glew849asyeryde34u4dfsdfy
StreamUrl        : {StreamUrl value redacted}
TokenValue       : {Token value redacted}
ContentLength    : 1207
HttpStatusCode   : OK
```

- Einzelheiten zur API finden Sie unter StartSession AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StopAutomationExecution`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StopAutomationExecution` verwendet wird.

CLI

AWS CLI

So können Sie eine Automatisierungsausführung beenden

Im folgenden Beispiel `stop-automation-execution` wird ein Automation-Dokument gestoppt.

```
aws ssm stop-automation-execution
    --automation-execution-id "4105a4fc-f944-11e6-9d32-0a1b2EXAMPLE"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Manuelles Ausführen eines Automation-Workflows im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie StopAutomationExecutionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Automation-Ausführung gestoppt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Einzelheiten zur API finden Sie unter StopAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Automation-Ausführung gestoppt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- Einzelheiten zur API finden Sie unter StopAutomationExecution AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateAssociation`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAssociation` verwendet wird.

CLI

AWS CLI

Beispiel 1: Um eine Dokumentverknüpfung zu aktualisieren

Das folgende `update-association`-Beispiel aktualisiert eine Zuordnung mit einer neuen Dokumentversion.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --document-version "\$LATEST"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
        "AssociationVersion": "2",
        "Date": 1550508093.293,
        "LastUpdateAssociationDate": 1550508106.596,
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$LATEST",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "tag:Name",
                "Values": [
                    "Linux"
                ]
            }
        ],
        "LastExecutionDate": 1550508094.879,
        "LastSuccessfulExecutionDate": 1550508094.879
    }
}
```

Weitere Informationen finden Sie unter Bearbeiten und Erstellen einer neuen Version einer Zuordnung im AWS -Systems-Manager-Benutzerhandbuch.

Beispiel 2: So aktualisieren Sie den Zeitplanausdruck einer Zuordnung

Im folgenden `update-association`-Beispiel wird der Zeitplanausdruck für die angegebene Zuordnung aktualisiert.

```
aws ssm update-association \
    --association-id "8dfe3659-4309-493a-8755-0123456789ab" \
    --schedule-expression "cron(0 0 0/4 1/1 * ? *)"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Name": "AWS-HelloWorld",
        "AssociationVersion": "2",
        "Date": "2021-02-08T13:54:19.203000-08:00",
        "LastUpdateAssociationDate": "2021-06-29T11:51:07.933000-07:00",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "aws:NoOpAutomationTag",
                "Values": [
                    "AWS-NoOpAutomationTarget-Value"
                ]
            }
        ],
        "ScheduleExpression": "cron(0 0 0/4 1/1 * ? *)",
        "LastExecutionDate": "2021-06-26T19:00:48.110000-07:00",
        "ApplyOnlyAtCronInterval": false
    }
}
```

Weitere Informationen finden Sie unter Bearbeiten und Erstellen einer neuen Version einer Zuordnung im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateAssociationin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird eine Verknüpfung mit einer neuen Dokumentversion aktualisiert.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter UpdateAssociation AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird eine Verknüpfung mit einer neuen Dokumentversion aktualisiert.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
```

```
Status.AdditionalInfo :
```

- Einzelheiten zur API finden Sie unter UpdateAssociation AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **UpdateAssociationStatus** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateAssociationStatus` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie den Zuordnungsstatus

Im folgenden Beispiel `update-association-status` wird der Zuordnungsstatus der Verknüpfung zwischen einer Instance und einem Dokument aktualisiert.

```
aws ssm update-association-status \
    --name "AWS-UpdateSSMAgent" \
    --instance-id "i-1234567890abcdef0" \
    --association-
status "Date=1424421071.939,Name=Pending,Message=temp_status_change,AdditionalInfo=Additi
Config-Needed"
```

Ausgabe:

```
{
    "AssociationDescription": {
        "Name": "AWS-UpdateSSMAgent",
        "InstanceId": "i-1234567890abcdef0",
        "AssociationVersion": "1",
        "Date": 1550507529.604,
        "LastUpdateAssociationDate": 1550507806.974,
        "Status": {
            "Date": 1424421071.0,
            "Name": "Pending",
            "Message": "temp_status_change",
            "AdditionalInfo": "Additional-Config-Needed"
        },
        "Overview": {
            "Status": "Success",
```

```
            "AssociationStatusAggregatedCount": {
                "Success": 1
            }
        },
        "DocumentVersion": "$DEFAULT",
        "AssociationId": "8dfe3659-4309-493a-8755-0123456789ab",
        "Targets": [
            {
                "Key": "InstanceIds",
                "Values": [
                    "i-1234567890abcdef0"
                ]
            }
        ],
        "LastExecutionDate": 1550507808.0,
        "LastSuccessfulExecutionDate": 1550507808.0
    }
}
```

Weitere Informationen finden Sie unter Arbeiten mit Zuordnungen in Systems Manager im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateAssociationStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Zuordnungsstatus der Zuordnung zwischen einer Instance und einem Konfigurationsdokument aktualisiert.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
```

```
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Einzelheiten zur API finden Sie unter UpdateAssociationStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Zuordnungsstatus der Zuordnung zwischen einer Instance und einem Konfigurationsdokument aktualisiert.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- Einzelheiten zur API finden Sie unter UpdateAssociationStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateDocument`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateDocument` verwendet wird.

CLI

AWS CLI

Um eine neue Version eines Dokuments zu erstellen

Das folgende `update-document`-Beispiel erstellt eine neue Version eines Dokuments, wenn es auf einem Windows-Computer ausgeführt wird. Das von `--document` angegebene Dokument muss im JSON-Format vorliegen. Beachten Sie, dass `file://` darauf verwiesen werden muss, gefolgt vom Pfad der Inhaltsdatei. Wegen des $ am Anfang des `--document-version`-Parameters müssen Sie den Wert unter Windows in Anführungszeichen setzen. Unter Linux, macOS oder an einer PowerShell Eingabeaufforderung müssen Sie den Wert in einfache Anführungszeichen setzen.

Windows-Version:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version "$LATEST"
```

Linux/Mac Version:

```
aws ssm update-document \
    --name "RunShellScript" \
    --content "file://RunShellScript.json" \
    --document-version '$LATEST'
```

Ausgabe:

```
{
  "DocumentDescription": {
      "Status": "Updating",
      "Hash": "f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b",
      "Name": "RunShellScript",
      "Parameters": [
          {
              "Type": "StringList",
              "Name": "commands",
              "Description": "(Required) Specify a shell script or a command to
 run."
          }
      ],
      "DocumentType": "Command",
      "PlatformTypes": [
          "Linux"
      ],
      "DocumentVersion": "2",
```

```
        "HashType": "Sha256",
        "CreatedDate": 1487899655.152,
        "Owner": "809632081692",
        "SchemaVersion": "2.0",
        "DefaultVersion": "1",
        "LatestVersion": "2",
        "Description": "Run an updated script"
    }
}
```

- Einzelheiten zur API finden Sie UpdateDocumentin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dadurch wird eine neue Version eines Dokuments mit dem aktualisierten Inhalt der von Ihnen angegebenen JSON-Datei erstellt. Das Dokument muss im JSON-Format vorliegen. Sie können die Dokumentversion mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Ausgabe:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Einzelheiten zur API finden Sie unter UpdateDocument AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dadurch wird eine neue Version eines Dokuments mit dem aktualisierten Inhalt der von Ihnen angegebenen JSON-Datei erstellt. Das Dokument muss im JSON-Format vorliegen. Sie können die Dokumentversion mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-
Content -Raw "c:\temp\RunShellScript.json")
```

Ausgabe:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            :
 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- Einzelheiten zur API finden Sie unter UpdateDocument AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`UpdateDocumentDefaultVersion`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateDocumentDefaultVersion` verwendet wird.

CLI

AWS CLI

So aktualisieren Sie die Standardversion eines Dokuments

Im folgenden `update-document-default-version`-Beispiel wird die Standardversion eines Systems-Manager-Dokuments aktualisiert.

```
aws ssm update-document-default-version \
    --name "Example" \
    --document-version "2"
```

Ausgabe:

```
{
    "Description": {
        "Name": "Example",
        "DefaultVersion": "2"
    }
}
```

Weitere Informationen finden Sie unter Schreiben von SSM-Dokumentinhalten im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateDocumentDefaultVersionin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dadurch wird die Standardversion eines Dokuments aktualisiert. Sie können die verfügbaren Dokumentversionen mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Ausgabe:

```
DefaultVersion Name
-------------- ----
```

```
2               RunShellScript
```

- Einzelheiten zur API finden Sie unter UpdateDocumentDefaultVersion AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dadurch wird die Standardversion eines Dokuments aktualisiert. Sie können die verfügbaren Dokumentversionen mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Ausgabe:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- Einzelheiten zur API finden Sie unter UpdateDocumentDefaultVersion AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateMaintenanceWindow`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateMaintenanceWindow` verwendet wird.

CLI

AWS CLI

Beispiel 1: So aktualisieren Sie ein Wartungsfenster

Im folgenden Beispiel `update-maintenance-window` wird der Name eines Wartungsfensters aktualisiert.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --name "My-Renamed-MW"
```

Ausgabe:

```
{
```

```
    "Cutoff": 1,
    "Name": "My-Renamed-MW",
    "Schedule": "cron(0 16 ? * TUE *)",
    "Enabled": true,
    "AllowUnassociatedTargets": true,
    "WindowId": "mw-1a2b3c4d5e6f7g8h9",
    "Duration": 4
}
```

Beispiel 2: So deaktivieren Sie ein Wartungsfenster

Das folgende `update-maintenance-window`-Beispiel deaktiviert ein Wartungsfenster.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --no-enabled
```

Beispiel 3: So aktivieren Sie ein Wartungsfenster

Das folgende `update-maintenance-window`-Beispiel aktiviert ein Wartungsfenster.

```
aws ssm update-maintenance-window \
    --window-id "mw-1a2b3c4d5e6f7g8h9" \
    --enabled
```

Weitere Informationen finden Sie unter Aktualisieren eines Wartungsfensters (AWS CLI) im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateMaintenanceWindowunter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- Einzelheiten zur API finden Sie UpdateMaintenanceWindowin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM maintenance window.
 * @param {{ windowId: string, allowUnassociatedTargets?: boolean, duration?:
 number, enabled?: boolean, name?: string, schedule?: string }}
 */
export const main = async ({
  windowId,
  allowUnassociatedTargets = undefined, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
  duration = undefined, //The duration of the maintenance window in hours.
  enabled = undefined,
  name = undefined,
  schedule = undefined, //The schedule of the maintenance window in the form of a
 cron or rate expression.
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new UpdateMaintenanceWindowCommand({
        WindowId: windowId,
        AllowUnassociatedTargets: allowUnassociatedTargets,
        Duration: duration,
        Enabled: enabled,
        Name: name,
        Schedule: schedule,
      }),
    );
    console.log("Maintenance window updated.");
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Are these values correct?`);
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie UpdateMaintenanceWindowin der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Name eines Wartungsfensters aktualisiert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel wird ein Wartungsfenster aktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
```

```
Enabled                   : True
Name                      : My-Renamed-MW
Schedule                  : cron(0 */30 * * * ? *)
WindowId                  : mw-03eb9db42890fb82d
```

Beispiel 3: In diesem Beispiel wird ein Wartungsfenster deaktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Einzelheiten zur API finden Sie unter UpdateMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Name eines Wartungsfensters aktualisiert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-
MW"
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel wird ein Wartungsfenster aktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 3: In diesem Beispiel wird ein Wartungsfenster deaktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- Einzelheiten zur API finden Sie unter UpdateMaintenanceWindow AWS -Tools für PowerShellCmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""
```

```
    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(
        self, name, enabled, schedule, duration, cutoff,
allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on
managed nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                            if you haven't registered those nodes
as targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
```

```
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to update maintenance
window %s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie UpdateMaintenanceWindowin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ssm->updatemaintenancewindow(
            iv_windowid = iv_window_id
            iv_name = iv_name
            iv_enabled = iv_enabled
            iv_schedule = iv_schedule
            iv_duration = iv_duration
            iv_cutoff = iv_cutoff
            iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
```

```
        MESSAGE 'Maintenance window updated.' TYPE 'I'.
      CATCH /aws1/cx_ssmdoesnotexistex.
        MESSAGE 'Maintenance window does not exist.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateMaintenanceWindowin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`UpdateManagedInstanceRole`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateManagedInstanceRole` verwendet wird.

CLI

AWS CLI

Um die IAM-Rolle einer verwalteten Instance zu aktualisieren

Im folgenden Beispiel `update-managed-instance-role` wird das IAM-Instance-Profil einer verwalteten Instance aktualisiert.

```
aws ssm update-managed-instance-role \
    --instance-id "mi-08ab247cdfEXAMPLE" \
    --iam-role "ExampleRole"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Schritt 4: Erstellen eines Instance-Profils für Systems Manager im AWS -Systems-Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateManagedInstanceRolein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Rolle einer verwalteten Instance aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Einzelheiten zur API finden Sie unter UpdateManagedInstanceRole AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Rolle einer verwalteten Instance aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- Einzelheiten zur API finden Sie unter UpdateManagedInstanceRole AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwenden Sie mit einem **`UpdateOpsItem`** AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateOpsItem` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Kennenlernen der Grundlagen

CLI

AWS CLI

Um ein zu aktualisieren OpsItem

Im folgenden `update-ops-item` Beispiel werden die Beschreibung, Priorität und Kategorie für ein aktualisiert OpsItem. Darüber hinaus gibt der Befehl ein SNS-Thema an, an das die Benachrichtigungen gesendet werden, wenn dieses bearbeitet oder geändert OpsItem wird.

```
aws ssm update-ops-item \
    --ops-item-id "oi-287b5EXAMPLE" \
    --description "Primary OpsItem for failover event 2020-01-01-fh398yf" \
```

```
    --priority 2 \
    --category "Security" \
    --notifications "Arn=arn:aws:sns:us-east-2:111222333444:my-us-east-2-topic"
```

Ausgabe:

```
This command produces no output.
```

Weitere Informationen finden Sie unter Arbeiten mit OpsItems im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdateOpsItemin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Resolves an AWS SSM OpsItem asynchronously.
 *
 * @param opsID The ID of the OpsItem to resolve.
 * <p>
 * This method initiates an asynchronous request to resolve an SSM OpsItem.
 * If an exception occurs, it handles the error appropriately.
 */
public void resolveOpsItem(String opsID) {
    UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
            .opsItemId(opsID)
            .status(OpsItemStatus.RESOLVED)
            .build();

    CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
        getAsyncClient().updateOpsItem(opsItemRequest)
                .thenAccept(response -> {
                    System.out.println("OpsItem resolved successfully.");
```

```
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }
```

- Einzelheiten zur API finden Sie UpdateOpsItemin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM OpsItem.
```

```
 * @param {{ opsItemId: string, status?: OpsItemStatus }}
 */
export const main = async ({
  opsItemId,
  status = undefined, // The OpsItem status. Status can be Open, In Progress, or
 Resolved
}) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new UpdateOpsItemCommand({
        OpsItemId: opsItemId,
        Status: status,
      }),
    );
    console.log("Ops item updated.");
    return { Success: true };
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "OpsItemLimitExceededException"
    ) {
      console.warn(
        `Couldn't create ops item because you have exceeded your open OpsItem
 limit. ${caught.message}.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Einzelheiten zur API finden Sie UpdateOpsItemin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
```

```
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Einzelheiten zur API finden Sie UpdateOpsItemin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ssm->updateopsitem(
            iv_opsitemid = iv_ops_item_id
            iv_title = iv_title
            iv_description = iv_description
            iv_status = iv_status ).
        MESSAGE 'OpsItem updated.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemnotfoundex.
        MESSAGE 'OpsItem not found.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsiteminvparamex.
        MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateOpsItemin der API-Referenz zum AWS SDK für SAP ABAP.

Verwendung von **`UpdatePatchBaseline`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `UpdatePatchBaseline` verwendet wird.

CLI

AWS CLI

Beispiel 1: So können Sie eine Patch-Baseline aktualisieren

Im folgenden `update-patch-baseline`-Beispiel werden der angegebenen Patch-Baseline die beiden angegebenen Patches als abgelehnt und ein Patch als genehmigt hinzugefügt.

```
aws ssm update-patch-baseline \
        --baseline-id "pb-0123456789abcdef0" \
        --rejected-patches "KB2032276" "MS10-048" \
        --approved-patches "KB2124261"
```

Ausgabe:

```
{
    "BaselineId": "pb-0123456789abcdef0",
    "Name": "WindowsPatching",
    "OperatingSystem": "WINDOWS",
    "GlobalFilters": {
        "PatchFilters": []
    },
    "ApprovalRules": {
        "PatchRules": [
            {
                "PatchFilterGroup": {
                    "PatchFilters": [
                        {
                            "Key": "PRODUCT",
                            "Values": [
                                "WindowsServer2016"
```

```
                            ]
                        }
                    ]
                },
                "ComplianceLevel": "CRITICAL",
                "ApproveAfterDays": 0,
                "EnableNonSecurity": false
            }
        ]
    },
    "ApprovedPatches": [
        "KB2124261"
    ],
    "ApprovedPatchesComplianceLevel": "UNSPECIFIED",
    "ApprovedPatchesEnableNonSecurity": false,
    "RejectedPatches": [
        "KB2032276",
        "MS10-048"
    ],
    "RejectedPatchesAction": "ALLOW_AS_DEPENDENCY",
    "CreatedDate": 1550244180.465,
    "ModifiedDate": 1550244180.465,
    "Description": "Patches for Windows Servers",
    "Sources": []
}
```

Beispiel 2: So benennen Sie eine Patch-Baseline um

Im folgenden Beispiel `update-patch-baseline` wird die angegebene Patch-Baseline umbenannt.

```
aws ssm update-patch-baseline \
    --baseline-id "pb-0713accee01234567" \
    --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

Weitere Informationen finden Sie unter Aktualisieren oder Löschen einer Patch-Baseline < https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-baseline-update-or-delete.html >`__ im AWS Systems Manager Manager-Benutzerhandbuch.

- Einzelheiten zur API finden Sie UpdatePatchBaselinein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden einer vorhandenen Patch-Baseline zwei Patches als abgelehnt und ein Patch als genehmigt hinzugefügt.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Einzelheiten zur API finden Sie unter UpdatePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden einer vorhandenen Patch-Baseline zwei Patches als abgelehnt und ein Patch als genehmigt hinzugefügt.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
```

```
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- Einzelheiten zur API finden Sie unter UpdatePatchBaseline AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung von Systems Manager AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Systems Manager mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von Systems Manager oder in Kombination mit anderen aufrufen AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Einen Container-Service für VM-Instanzen erstellen
- Erste Schritte mit virtuellen Maschinen
- Systemmanagement einrichten

Einen Container-Service für VM-Instanzen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Einen Dienst erstellen und überwachen
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
```

```
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
```

```
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
```

```
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
```

```
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"
```

```
    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"
```

```
    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"
```

```
    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
```

```
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
```

```
    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
```

```
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"
```

```
    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
```

```
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition

- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Erste Schritte mit virtuellen Maschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Schlüsselpaares
- Eine Sicherheitsgruppe erstellen
- Anhalten und Starten Ihrer Instance
- Testen Sie die Elastic IP-Persistenz
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0
```

```
    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi
```

```
    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"
```

```
# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial'
 --output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
```

```
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value"
 \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
```

```
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]'
 'ResourceType=volume,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"
```

```
# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
```

```
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
```

```
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your
 instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - DisassociateAddress
  - GetParametersByPath
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - Wait

Systemmanagement einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- IAM-Berechtigungen für Systems Manager erstellen
- Erstellen einer IAM-Rolle für Systems Manager
- Systems Manager konfigurieren
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
```

```
    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"
```

```
        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
```

```
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
```

```
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
```

```
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
```

```
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
```

```
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
```

```
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
```

```
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
```

```
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi
```

```
# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF
```

```
echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
```

```
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy

- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetConfigurationManager

Codebeispiele für Amazon Textract mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Textract mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Textract – Weitere Informationen zu Amazon Textract.
- API-Referenz für Amazon Textract – Details zu allen verfügbaren Amazon-Textract-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Textract AWS SDKs
  - Aktionen für Amazon Textract mit AWS SDKs
    - Verwenden Sie AnalyzeDocumentmit einem AWS SDK oder CLI
    - Verwenden Sie DetectDocumentTextmit einem AWS SDK oder CLI
    - Verwenden Sie GetDocumentAnalysismit einem AWS SDK oder CLI
    - Verwenden Sie StartDocumentAnalysismit einem AWS SDK oder CLI
    - Verwenden Sie StartDocumentTextDetectionmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Textract AWS SDKs

- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK
- Beginnen Sie mit der Amazon Textract Textract-Dokumentenanalyse mit einem AWS SDK
- Erste Schritte mit der Textextraktion von Dokumenten

Grundlegende Beispiele für die Verwendung von Amazon Textract AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Textract mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Textract mit AWS SDKs
  - Verwenden Sie AnalyzeDocumentmit einem AWS SDK oder CLI
  - Verwenden Sie DetectDocumentTextmit einem AWS SDK oder CLI
  - Verwenden Sie GetDocumentAnalysismit einem AWS SDK oder CLI
  - Verwenden Sie StartDocumentAnalysismit einem AWS SDK oder CLI
  - Verwenden Sie StartDocumentTextDetectionmit einem AWS SDK oder CLI

Aktionen für Amazon Textract mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Textract Textract-Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Textract API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Textract AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Textract.

Beispiele

- Verwenden Sie AnalyzeDocumentmit einem AWS SDK oder CLI

- Verwenden Sie DetectDocumentTextmit einem AWS SDK oder CLI
- Verwenden Sie GetDocumentAnalysismit einem AWS SDK oder CLI
- Verwenden Sie StartDocumentAnalysismit einem AWS SDK oder CLI
- Verwenden Sie StartDocumentTextDetectionmit einem AWS SDK oder CLI

Verwenden Sie **`AnalyzeDocument`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `AnalyzeDocument` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Textextraktion von Dokumenten

CLI

AWS CLI

So analysieren Sie Text in einem Dokument

Das folgende Beispiel für `analyze-document` zeigt, wie Text in einem Dokument analysiert wird.

Linux/macOS:

```
aws textract analyze-document \
    --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
    --feature-types '["TABLES","FORMS"]'
```

Windows:

```
aws textract analyze-document \
    --document "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" \
    --feature-types "[\"TABLES\",\"FORMS\"]" \
    --region region-name
```

Ausgabe:

```
{
    "Blocks": [
```

```
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "87586964-d50d-43e2-ace5-8a890657b9a0",
                        "a1e72126-21d9-44f4-a8d6-5c385f9002ba",
                        "e889d012-8a6b-4d2e-b7cd-7a8b327d876a"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "c2227f12-b25d-4e1f-baea-1ee180d926b2"
        }
    ],
    "DocumentMetadata": {
        "Pages": 1
    }
```

```
}
```

Weitere Informationen finden Sie unter „Analysieren von Dokumenttext mit Amazon Textract“ im Entwicklerhandbuch für Amazon Textract.

- Einzelheiten zur API finden Sie AnalyzeDocumentin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be
 an image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);
```

```
            AnalyzeDocumentRequest analyzeDocumentRequest =
 AnalyzeDocumentRequest.builder()
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();

            AnalyzeDocumentResponse analyzeDocument =
 textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();

            while (blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +
 block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie AnalyzeDocumentin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
```

```
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def analyze_file(
        self, feature_types, *, document_file_name=None, document_bytes=None
    ):
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to
detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={"Bytes": document_bytes}, FeatureTypes=feature_types
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie AnalyzeDocumentin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Detects text and additional elements, such as forms or tables,"
   "in a local image file or from in-memory byte data."
   "The image must be in PNG or JPG format."


   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).

   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->analyzedocument(      "oo_result is returned for
testing purposes."
         io_document        = lo_document
```

```
          it_featuretypes    = lt_featuretypes ).
        LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
          IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
            MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
          ENDIF.
        ENDLOOP.
        MESSAGE 'Analyze document completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texhlquotaexceededex.
        MESSAGE 'Human loop quota exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.

      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie AnalyzeDocumentin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DetectDocumentText`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DetectDocumentText` verwendet wird.

CLI

AWS CLI

So erkennen Sie Text in einem Dokument

Im folgenden Beispiel für `detect-document-text` wird Text in einem Dokument erkannt.

Linux/macOS:

```
aws textract detect-document-text \
    --document '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
```

Windows:

```
aws textract detect-document-text \
    --document "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document\"}}" \
    --region region-name
```

Ausgabe:

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
```

```
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "896a9f10-9e70-4412-81ce-49ead73ed881",
                        "0da18623-dc4c-463d-a3d1-9ac050e9e720",
                        "167338d7-d38c-4760-91f1-79a8ec457bb2"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "21f0535e-60d5-4bc7-adf2-c05dd851fa25"
        },
        {
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "62490c26-37ea-49fa-8034-7a9ff9369c9c",
                        "1e4f3f21-05bd-4da9-ba10-15d01e66604c"
                    ]
                }
            ],
            "Confidence": 89.11581420898438,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33642634749412537,
                    "Top": 0.17169663310050964,
                    "Left": 0.13885067403316498,
                    "Height": 0.49159330129623413
                },
                "Polygon": [
                    {
                        "Y": 0.17169663310050964,
                        "X": 0.13885067403316498
                    },
                    {
                        "Y": 0.17169663310050964,
                        "X": 0.47527703642845154
                    },
                    {
                        "Y": 0.6632899641990662,
                        "X": 0.47527703642845154
```

```
                    },
                    {
                        "Y": 0.6632899641990662,
                        "X": 0.13885067403316498
                    }
                ]
            },
            "Text": "He llo,",
            "BlockType": "LINE",
            "Id": "896a9f10-9e70-4412-81ce-49ead73ed881"
        },
        {
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "19b28058-9516-4352-b929-64d7cef29daf"
                    ]
                }
            ],
            "Confidence": 85.5694351196289,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33182239532470703,
                    "Top": 0.23131252825260162,
                    "Left": 0.5091826915740967,
                    "Height": 0.3766750991344452
                },
                "Polygon": [
                    {
                        "Y": 0.23131252825260162,
                        "X": 0.5091826915740967
                    },
                    {
                        "Y": 0.23131252825260162,
                        "X": 0.8410050868988037
                    },
                    {
                        "Y": 0.607987642288208,
                        "X": 0.8410050868988037
                    },
                    {
                        "Y": 0.607987642288208,
                        "X": 0.5091826915740967
```

```
                    }
                ]
            },
            "Text": "worlc",
            "BlockType": "LINE",
            "Id": "0da18623-dc4c-463d-a3d1-9ac050e9e720"
        }
    ],
    "DocumentMetadata": {
        "Pages": 1
    }
}
```

Weitere Informationen finden Sie unter „Erkennen von Dokumenttext mit Amazon Textract“ im Entwicklerhandbuch für Amazon Textract.

- Einzelheiten zur API finden Sie DetectDocumentTextin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen Sie Text aus einem Eingabedokument.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be
 an image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen von Text aus einem Dokument, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
```

```
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains
 the document.\s

                    docName - The document name (must be an image, i.e.,
 book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }
```

```
    public static void detectDocTextS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DetectDocumentTextin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={"Bytes": document_bytes}
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
```

```
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie DetectDocumentTextin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Detects text in the input document."
   "Amazon Textract can detect lines of text and the words that make up a line
of text."
   "The input document must be in one of the following image formats: JPEG, PNG,
PDF, or TIFF."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).
   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->detectdocumenttext( io_document = lo_document ).
   "oo_result is returned for testing purposes."
       LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
```

```
          ENDIF.
        ENDLOOP.
        DATA(lo_metadata) = oo_result->get_documentmetadata( ).
        MESSAGE 'The number of pages in the document is ' && lo_metadata-
>ask_pages( ) TYPE 'I'.
        MESSAGE 'Detect document text completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DetectDocumentTextin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetDocumentAnalysis`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetDocumentAnalysis` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Dokumentenanalyse

CLI

AWS CLI

So erhalten Sie die Ergebnisse der asynchronen Textanalyse eines mehrseitigen Dokuments

Das folgende Beispiel für `get-document-analysis` zeigt, wie die Ergebnisse einer asynchronen Textanalyse eines mehrseitigen Dokuments abgerufen werden.

```
aws textract get-document-analysis \
    --job-id df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b \
    --max-results 1000
```

Ausgabe:

```
{
    "Blocks": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 1.0,
                    "Top": 0.0,
                    "Left": 0.0,
                    "Height": 1.0
                },
                "Polygon": [
                    {
                        "Y": 0.0,
                        "X": 0.0
                    },
                    {
                        "Y": 0.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 1.0
                    },
                    {
                        "Y": 1.0,
                        "X": 0.0
                    }
                ]
```

```
            },
            "Relationships": [
                {
                    "Type": "CHILD",
                    "Ids": [
                        "75966e64-81c2-4540-9649-d66ec341cd8f",
                        "bb099c24-8282-464c-a179-8a9fa0a057f0",
                        "5ebf522d-f9e4-4dc7-bfae-a288dc094595"
                    ]
                }
            ],
            "BlockType": "PAGE",
            "Id": "247c28ee-b63d-4aeb-9af0-5f7ea8ba109e",
            "Page": 1
        }
    ],
    "NextToken": "cY1W3eTFvoB0cH7YrKVudI4Gb0H8J0xAYLo8xI/JunCIPWCthaKQ+07n/
ElyutsSy0+1VOImoTRmP1zw4P0RFtaeV9Bzhnfedpx1YqwB4xaGDA==",
    "DocumentMetadata": {
        "Pages": 1
    },
    "JobStatus": "SUCCEEDED"
}
```

Weitere Informationen finden Sie unter Erkennen und Analysieren von Text in Multi-Page Dokumenten im Amazon Textract Developers Guide

- Einzelheiten zur API finden Sie unter GetDocumentAnalysis AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
```

```
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(JobId=job_id)
            job_status = response["JobStatus"]
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- Einzelheiten zur API finden Sie GetDocumentAnalysisin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets the results for an Amazon Textract"
   "asynchronous operation that analyzes text in a document."
   TRY.
       oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purposes."
       WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
         IF sy-index = 10.
           EXIT.               "Maximum 300 seconds.
         ENDIF.
         WAIT UP TO 30 SECONDS.
         oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
       ENDWHILE.

       DATA(lt_blocks) = oo_result->get_blocks( ).
       LOOP AT lt_blocks INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
         ENDIF.
       ENDLOOP.
       MESSAGE 'Document analysis retrieved.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidjobidex.
       MESSAGE 'Job ID is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
```

```
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetDocumentAnalysisin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartDocumentAnalysis`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartDocumentAnalysis` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit der Dokumentenanalyse

CLI

AWS CLI

So beginnen Sie mit der Analyse von Text in einem mehrseitigen Dokument

Das folgende Beispiel für `start-document-analysis` zeigt, wie die asynchrone Analyse von Text in einem mehrseitigen Dokument gestartet wird.

Linux/macOS:

```
aws textract start-document-analysis \
    --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}' \
    --feature-types '["TABLES","FORMS"]' \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Windows:

```
aws textract start-document-analysis \
    --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document
\"}}" \
    --feature-types "[\"TABLES\", \"FORMS\"]" \
```

```
    --region region-name \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Ausgabe:

```
{
    "JobId": "df7cf32ebbd2a5de113535fcf4d921926a701b09b4e7d089f3aebadb41e0712b"
}
```

Weitere Informationen finden Sie unter Erkennen und Analysieren von Text in Multi-Page Dokumenten im Amazon Textract Developers Guide

- Einzelheiten zur API finden Sie unter StartDocumentAnalysis AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import
 software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import
 software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import
 software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains
 the document.\s
                    docName - The document name (must be an image, for example,
 book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
```

```
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String
jobId) {
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
```

```
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
 textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Einzelheiten zur API finden Sie StartDocumentAnalysisin der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie einen asynchronen Auftrag zum Analysieren eines Dokuments.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_analysis_job(
        self,
        bucket_name,
        document_file_name,
        feature_types,
        sns_topic_arn,
        sns_role_arn,
    ):
        """
        Starts an asynchronous job to detect text and additional elements, such
 as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
 publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
 image.
        :param document_file_name: The name of the document image stored in
 Amazon S3.
        :param feature_types: The types of additional document features to
 detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
 topic
                               where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
 (IAM)
                              role that can be assumed by Textract and grants
 permission
```

```
                                to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name":
document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
                FeatureTypes=feature_types,
            )
            job_id = response["JobId"]
            logger.info(
                "Started text analysis job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Einzelheiten zur API finden Sie StartDocumentAnalysisin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Starts the asynchronous analysis of an input document for relationships"
   "between detected items such as key-value pairs, tables, and selection
elements."

   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start async document analysis."
   TRY.
       oo_result = lo_tex->startdocumentanalysis(      "oo_result is returned
for testing purposes."
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       DATA(lv_jobid) = oo_result->get_jobid( ).

       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartDocumentAnalysisin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartDocumentTextDetection`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartDocumentTextDetection` verwendet wird.

CLI

AWS CLI

So beginnen Sie mit der Erkennung von Text in einem mehrseitigen Dokument

Das folgende Beispiel für `start-document-text-detection` zeigt, wie die asynchrone Erkennung von Text in einem mehrseitigen Dokument gestartet wird.

Linux/macOS:

```
aws textract start-document-text-detection \
        --document-location '{"S3Object":{"Bucket":"bucket","Name":"document"}}'
 \
        --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleARN"
```

Windows:

```
aws textract start-document-text-detection \
    --document-location "{\"S3Object\":{\"Bucket\":\"bucket\",\"Name\":\"document
\"}}" \
    --region region-name \
    --notification-channel "SNSTopicArn=arn:snsTopic,RoleArn=roleArn"
```

Ausgabe:

```
{
    "JobId": "57849a3dc627d4df74123dca269d69f7b89329c870c65bb16c9fd63409d200b9"
}
```

Weitere Informationen finden Sie unter Erkennen und Analysieren von Text in Multi-Page Dokumenten im Amazon Textract Developers Guide

- Einzelheiten zur API finden Sie unter StartDocumentTextDetection AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie einen asynchronen Auftrag zum Erkennen von Text in Dokumenten.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_detection_job(
        self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn
    ):
        """
```

```
        Starts an asynchronous job to detect text elements in an image stored in
an
        Amazon S3 bucket. Textract publishes a notification to the specified
Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in
Amazon S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS
topic
                               where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management
(IAM)
                              role that can be assumed by Textract and grants
permission
                              to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name":
document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
            )
            job_id = response["JobId"]
            logger.info(
                "Started text detection job %s on %s.", job_id,
document_file_name
            )
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- Einzelheiten zur API finden Sie StartDocumentTextDetectionin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Starts the asynchronous detection of text in a document."
   "Amazon Textract can detect lines of text and the words that make up a line
of text."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).
   "Start document analysis."
   TRY.
       oo_result = lo_tex->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).
       DATA(lv_jobid) = oo_result->get_jobid( ).             "oo_result is
returned for testing purposes."
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidkmskeyex.
        MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartDocumentTextDetectionin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Textract AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Textract mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Textract oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen Sie eine Amazon-Textract-Explorer-Anwendung
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK
- Beginnen Sie mit der Amazon Textract Textract-Dokumentenanalyse mit einem AWS SDK
- Erste Schritte mit der Textextraktion von Dokumenten

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.

- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

Die folgenden Codebeispiele zeigen, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus ihrer Originalsprache übersetzt, ihre Stimmung ermittelt und aus dem übersetzten Text eine Audiodatei generiert.

.NET

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.

- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie der innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );
```

```
  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
```

```
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
```

```
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

Python

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service

(Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter TextractAndComprehendNotebook.ipynb.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Beginnen Sie mit der Amazon Textract Textract-Dokumentenanalyse mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie die asynchrone Analyse.
- Rufen Sie die Analyse von Dokumenten ab.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"Create ABAP objects for feature type."
"Add TABLES to return information about the tables."
"Add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes."
```

```
   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start document analysis."
   TRY.
       DATA(lo_start_result) = lo_tex->startdocumentanalysis(
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texlimitexceededex.
       MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
     CATCH /aws1/cx_texprovthruputexcdex.
       MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
     CATCH /aws1/cx_texthrottlingex.
       MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
     CATCH /aws1/cx_texunsupporteddocex.
       MESSAGE 'The document is not supported.' TYPE 'E'.
```

```
    ENDTRY.

    "Get job ID from the output."
    DATA(lv_jobid) = lo_start_result->get_jobid( ).

    "Wait for job to complete."
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).     "
oo_result is returned for testing purposes. "
    WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
      IF sy-index = 10.
        EXIT.               "Maximum 300 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).
    ENDWHILE.

    DATA(lt_blocks) = oo_result->get_blocks( ).
    LOOP AT lt_blocks INTO DATA(lo_block).
      IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
        MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
      ENDIF.
    ENDLOOP.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - GetDocumentAnalysis
  - StartDocumentAnalysis

Erste Schritte mit der Textextraktion von Dokumenten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines S3-Buckets
- Laden Sie ein Dokument auf S3 hoch
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "====================================================="
echo "This script will guide you through using Amazon Textract to analyze
 document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
```

```
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
 delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
 then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
```

```
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
```

```
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
 50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
```

```
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi
```

```
# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====================================================="
echo "Analysis Summary"
echo "====================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
 '{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
 echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
 echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
 echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
 echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
 | wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
 wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""
```

```
# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
echo "==================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "==================================================="
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
```

```
echo ""

trap - EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AnalyzeDocument
  - Cp
  - Hilfe
  - Mb
  - Rb
  - Rm

Codebeispiele für Amazon Transcribe mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Transcribe mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Transcribe – Weitere Informationen zu Amazon Transcribe.
- API-Referenz für Amazon Transcribe – Details zu allen verfügbaren Amazon-Transcribe-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Transcribe AWS SDKs

- Aktionen für Amazon Transcribe mit AWS SDKs
  - Verwenden Sie CreateVocabularymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteMedicalTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteVocabularymit einem AWS SDK oder CLI
  - Verwenden Sie GetTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie GetVocabularymit einem AWS SDK oder CLI
  - Verwenden Sie ListMedicalTranscriptionJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListTranscriptionJobsmit einem AWS SDK oder CLI
  - Verwenden Sie ListVocabulariesmit einem AWS SDK oder CLI
  - Verwenden Sie StartMedicalTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie StartTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie UpdateVocabularymit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Transcribe AWS SDKs
  - Erstellen einer Amazon-Transcribe-Streaming-App
  - Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK
  - Erstellen und verfeinern Sie ein benutzerdefiniertes Amazon Transcribe Transcribe-Vokabular mit einem AWS SDK
  - Transkribieren Sie Audio und rufen Sie Auftragsdaten mit Amazon Transcribe mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Transcribe AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Transcribe mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Transcribe mit AWS SDKs
  - Verwenden Sie CreateVocabularymit einem AWS SDK oder CLI
  - Verwenden Sie DeleteMedicalTranscriptionJobmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteTranscriptionJobmit einem AWS SDK oder CLI

- Verwenden Sie DeleteVocabularymit einem AWS SDK oder CLI
- Verwenden Sie GetTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie GetVocabularymit einem AWS SDK oder CLI
- Verwenden Sie ListMedicalTranscriptionJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListTranscriptionJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListVocabulariesmit einem AWS SDK oder CLI
- Verwenden Sie StartMedicalTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie StartTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie UpdateVocabularymit einem AWS SDK oder CLI

Aktionen für Amazon Transcribe mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Transcribe Transcribe-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, über den Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Transcribe API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Transcribe AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon Transcribe API-Referenz.

Beispiele

- Verwenden Sie CreateVocabularymit einem AWS SDK oder CLI
- Verwenden Sie DeleteMedicalTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie DeleteTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie DeleteVocabularymit einem AWS SDK oder CLI
- Verwenden Sie GetTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie GetVocabularymit einem AWS SDK oder CLI
- Verwenden Sie ListMedicalTranscriptionJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListTranscriptionJobsmit einem AWS SDK oder CLI
- Verwenden Sie ListVocabulariesmit einem AWS SDK oder CLI

- Verwenden Sie StartMedicalTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie StartTranscriptionJobmit einem AWS SDK oder CLI
- Verwenden Sie UpdateVocabularymit einem AWS SDK oder CLI

Verwenden Sie **`CreateVocabulary`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateVocabulary` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
                LanguageCode = languageCode,
```

```
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Einzelheiten zur API finden Sie CreateVocabularyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Erstellen eines benutzerdefinierten Vokabulars

Im folgenden Beispiel für `create-vocabulary` wird ein benutzerdefiniertes Vokabular erstellt. Um ein benutzerdefiniertes Vokabular zu erstellen, müssen Sie eine Textdatei mit allen Begriffen erstellt haben, die Sie genauer transkribieren möchten. Geben Sie für „vocabulary-file-uri“ den Amazon Simple Storage Service (Amazon S3)-URI der betreffenden Textdatei an. Geben Sie für „language-code“ den der Sprache Ihres benutzerdefinierten Vokabulars entsprechenden Sprachcode an. Geben Sie für „vocabulary-name“ die gewünschte Bezeichnung für Ihr benutzerdefiniertes Vokabular an.

```
aws transcribe create-vocabulary \
    --language-code language-code \
    --vocabulary-name cli-vocab-example \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-prefix/the-text-
file-for-the-custom-vocabulary.txt
```

Ausgabe:

```
{
    "VocabularyName": "cli-vocab-example",
    "LanguageCode": "language-code",
    "VocabularyState": "PENDING"
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie CreateVocabularyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
```

```
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
vocabulary_name)
        raise
    else:
        return response
```

- Einzelheiten zur API finden Sie CreateVocabularyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Custom vocabulary created.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
```

```
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie CreateVocabularyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteMedicalTranscriptionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteMedicalTranscriptionJob` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript
 associated with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
```

```
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteMedicalTranscriptionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Löschen eines medizinischen Transkriptionsauftrags

Im folgenden Beispiel für `delete-medical-transcription-job` wird ein medizinischer Transkriptionsauftrag gelöscht.

```
aws transcribe delete-medical-transcription-job \
    --medical-transcription-job-name medical-transcription-job-name
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie DeleteMedicalTranscriptionJobim Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie DeleteMedicalTranscriptionJobin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Einen Auftrag für medizinische Transkription löschen.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-
transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteMedicalTranscriptionJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`DeleteTranscriptionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTranscriptionJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with
the job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await
_amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Einzelheiten zur API finden Sie DeleteTranscriptionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Löschen eines Ihrer Transkriptionsaufträge

Im folgenden Beispiel für `delete-transcription-job` wird einer Ihrer Transkriptionsaufträge gelöscht.

```
aws transcribe delete-transcription-job \
    --transcription-job-name your-transcription-job
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie DeleteTranscriptionJobim Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie DeleteTranscriptionJobin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Transkriptionsauftrags.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};
```

```
export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie DeleteTranscriptionJobin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_job(job_name, transcribe_client):
```

```
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- Einzelheiten zur API finden Sie DeleteTranscriptionJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_tnb->deletetranscriptionjob( iv_job_name ).
        MESSAGE 'Transcription job deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
```

```
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteTranscriptionJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`DeleteVocabulary`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteVocabulary` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an existing custom vocabulary.
    /// </summary>
    /// <param name="vocabularyName">Name of the vocabulary to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
    {
        var response = await _amazonTranscribeService.DeleteVocabularyAsync(
            new DeleteVocabularyRequest
            {
                VocabularyName = vocabularyName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- Einzelheiten zur API finden Sie DeleteVocabularyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So löschen Sie ein benutzerdefiniertes Vokabular

Im folgenden Beispiel für `delete-vocabulary` wird ein benutzerdefiniertes Vokabular gelöscht.

```
aws transcribe delete-vocabulary \
    --vocabulary-name vocabulary-name
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Benutzerdefinierte Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteVocabularyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.
```

```
    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- Einzelheiten zur API finden Sie DeleteVocabularyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_tnb->deletevocabulary( iv_vocabulary_name ).
        MESSAGE 'Vocabulary deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie DeleteVocabularyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetTranscriptionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetTranscriptionJob` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars
- Audio transkribieren und Auftragsdaten abrufen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- Einzelheiten zur API finden Sie GetTranscriptionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Abrufen von Informationen zu einem bestimmten Transkriptionsauftrag

Im folgenden Beispiel für `get-transcription-job` werden Informationen zu einem bestimmten Transkriptionsauftrag abgerufen. Verwenden Sie den TranscriptFileUri Parameter, um auf die Transkriptionsergebnisse zuzugreifen. Verwenden Sie den MediaFileUri Parameter, um zu sehen, welche Audiodatei Sie mit diesem Job transkribiert haben. Sie können das Objekt „Settings“ verwenden, um die optionalen Features zu sehen, die Sie im Transkriptionsauftrag aktiviert haben.

```
aws transcribe get-transcription-job \
    --transcription-job-name your-transcription-job
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "your-transcription-job",
        "TranscriptionJobStatus": "COMPLETED",
        "LanguageCode": "language-code",
        "MediaSampleRateHertz": 48000,
        "MediaFormat": "mp4",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "Transcript": {
            "TranscriptFileUri": "https://Amazon-S3-file-location-of-
transcription-output"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "CompletionTime": "2020-09-18T22:28:21.197000+00:00",
        "Settings": {
```

```
            "ChannelIdentification": false,
            "ShowAlternatives": false
        },
        "IdentifyLanguage": true,
        "IdentifiedLanguageScore": 0.8672199249267578
    }
}
```

Weitere Informationen finden Sie unter Erste Schritte (AWS Befehlszeilenschnittstelle) im Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie unter GetTranscriptionJob AWS CLIBefehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- Einzelheiten zur API finden Sie GetTranscriptionJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_tnb->gettranscriptionjob( iv_job_name ).
    DATA(lo_job) = oo_result->get_transcriptionjob( ).
    MESSAGE 'Retrieved transcription job details.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
    RAISE EXCEPTION lo_not_found_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- Einzelheiten zur API finden Sie GetTranscriptionJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`GetVocabulary`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetVocabulary` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information about a custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.GetVocabularyAsync(
        new GetVocabularyRequest()
        {
            VocabularyName = vocabularyName
        });
    return response.VocabularyState;
}
```

- Einzelheiten zur API finden Sie GetVocabularyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Abrufen von Informationen zu einem benutzerdefinierten Vokabular

Im folgenden Beispiel für `get-vocabulary` werden Informationen zu einem zuvor erstellten benutzerdefinierten Vokabular abgerufen.

```
aws transcribe get-vocabulary \
    --vocabulary-name cli-vocab-1
```

Ausgabe:

```
{
    "VocabularyName": "cli-vocab-1",
    "LanguageCode": "language-code",
    "VocabularyState": "READY",
    "LastModifiedTime": "2020-09-19T23:22:32.836000+00:00",
    "DownloadUri": "https://link-to-download-the-text-file-used-to-create-your-
custom-vocabulary"
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetVocabularyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
```

```
    try:
        response =
transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- Einzelheiten zur API finden Sie GetVocabularyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_tnb->getvocabulary( iv_vocabulary_name ).
        MESSAGE 'Retrieved vocabulary details.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie GetVocabularyin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListMedicalTranscriptionJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListMedicalTranscriptionJobs` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// List medical transcription jobs, optionally with a name filter.
   /// </summary>
   /// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
   /// <returns>A list of summaries about medical transcription jobs.</returns>
   public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
       string? jobNameContains = null)
   {
       var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
           new ListMedicalTranscriptionJobsRequest()
           {
               JobNameContains = jobNameContains
           });
       return response.MedicalTranscriptionJobSummaries;
   }
```

- Einzelheiten zur API finden Sie ListMedicalTranscriptionJobsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Auflisten von medizinischen Transkriptionsaufträgen

Im folgenden `list-medical-transcription-jobs` Beispiel werden die medizinischen Transkriptionsaufträge aufgeführt, die Ihrem AWS Konto und Ihrer Region zugeordnet sind. Um weitere Informationen zu einem bestimmten Transkriptionsauftrag zu erhalten, kopieren Sie den Wert eines MedicalTranscriptionJobName Parameters in die Transkriptionsausgabe und geben Sie diesen Wert für die `MedicalTranscriptionJobName` Option des Befehls an. `get-medical-transcription-job` Um mehr Ihrer Transkriptionsaufträge zu sehen, kopieren Sie den Wert des NextToken Parameters, führen Sie den `list-medical-transcription-jobs` Befehl erneut aus und geben Sie diesen Wert in der Option an. `--next-token`

```
aws transcribe list-medical-transcription-jobs
```

Ausgabe:

```
{
    "NextToken": "3/PblzkiGhzjER3KHuQt2fmbPLF7cDYafjFMEoGn44ON/
gsuUSTIkGyanvRE6WMXFd/ZTEc2EZj+P9eii/
z1O2FDYli6RLI0WoRX4RwMisVrh9G0Kie0Y8ikBCdtqlZB10Wa9McC+ebOl
+LaDtZPC4u6ttoHLRlEfzqstHXSgapXg3tEBtm9piIaPB6MOM5BB6t86+qtmocTR/
qrteHZBBudhTfbCwhsxaqujHiiUvFdm3BQbKKWIW06yV9b+4f38oD2lVIan
+vfUs3gBYAl5VTDmXXzQPBQOHPjtwmFI+IWX15nSUjWuN3TUylHgPWzDaYT8qBtu0Z+3UG4V6b
+K2CC0XszXg5rBq9hYgNzy4XoFh/6s5DoSnzq49Q9xHgHdT2yBADFmvFK7myZBsj75+2vQZOSVpWUPy3WT/32zFAc
+mFYfUjtTZ8n/jq7aQEjQ42A
+X/7K6JgOcdVPtEg8PlDr5kgYYG3q3OmYXX37U3FZuJmnTI63VtIXsNnOU5eGoYObtpk00Nq9UkzgSJxqj84ZD5n
+S0EGy9ZUYBJRRcGeYUM3Q4DbSJfUwSAqcFdLIWZdp8qIREMQIBWy7BLwSdyqsQo2vRrd53hm5aWM7SVf6pPq6X/
IXR5+1eUOOD8/coaTT4ES2DerbV6RkV4o0VT1d0SdVX/
MmtkNG8nYj8PqU07w7988quh1ZP6D80veJS1q73tUUR9MjnGernW2tAnvnLNhdefBcD
+sZVfYq3iBMFY7wTy1P1G6NqW9GrYDYoX3tTPWlD7phpbVSyKrh/
PdYrps5UxnsGoA1b7L/FfAXDfUoGrGUB4N3JsPYXX9D++g+6gV1qBBs/
WfF934aKqfD6UTggm/zV3GAOWiBpfvAZRvEb924i6yGHyMC7y54O1ZAwSBupmI
+FFd13CaPO4kN1vJlth6aM5vUPXg4BpyUhtbRhwD/KxCvf9K0tLJGyL1A==",
    "MedicalTranscriptionJobSummaries": [
        {
            "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:17:27.016000+00:00",
```

```
            "StartTime": "2020-09-21T21:17:27.045000+00:00",
            "CompletionTime": "2020-09-21T21:17:59.561000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
            "CreationTime": "2020-09-21T21:01:14.569000+00:00",
            "StartTime": "2020-09-21T21:01:14.592000+00:00",
            "CompletionTime": "2020-09-21T21:01:43.606000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "DICTATION"
        },
        {
            "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T19:09:18.171000+00:00",
            "StartTime": "2020-09-21T19:09:18.199000+00:00",
            "CompletionTime": "2020-09-21T19:10:22.516000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
        {
            "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-21T18:43:37.157000+00:00",
            "StartTime": "2020-09-21T18:43:37.265000+00:00",
            "CompletionTime": "2020-09-21T18:44:21.192000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        },
```

```
        {
            "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
            "CreationTime": "2020-09-20T23:46:44.053000+00:00",
            "StartTime": "2020-09-20T23:46:44.081000+00:00",
            "CompletionTime": "2020-09-20T23:47:35.851000+00:00",
            "LanguageCode": "en-US",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "CUSTOMER_BUCKET",
            "Specialty": "PRIMARYCARE",
            "Type": "CONVERSATION"
        }
    ]
}
```

Weitere Informationen finden Sie unter https://docs.aws.amazon.com/transcribe/latest/dg/batch-med-transcription.html > im Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie ListMedicalTranscriptionJobsin der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Auflisten medizinischer Transkriptionsjobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListMedicalTranscriptionJobsin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`ListTranscriptionJobs`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListTranscriptionJobs` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
    /// <returns>A list of transcription job summaries.</returns>
    public async Task<List<TranscriptionJobSummary>>
ListTranscriptionJobs(string? jobNameContains = null)
    {
        var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
            new ListTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.TranscriptionJobSummaries;
    }
```

- Einzelheiten zur API finden Sie ListTranscriptionJobsin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Auflisten Ihrer Transkriptionsaufträge

Das folgende `list-transcription-jobs` Beispiel listet die Transkriptionsaufträge auf, die Ihrem AWS Konto und Ihrer Region zugeordnet sind.

```
aws transcribe list-transcription-jobs
```

Ausgabe:

```
{
    "NextToken": "NextToken",
    "TranscriptionJobSummaries": [
        {
            "TranscriptionJobName": "speak-id-job-1",
            "CreationTime": "2020-08-17T21:06:15.391000+00:00",
            "StartTime": "2020-08-17T21:06:15.416000+00:00",
            "CompletionTime": "2020-08-17T21:07:05.098000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "job-1",
            "CreationTime": "2020-08-17T20:50:24.207000+00:00",
            "StartTime": "2020-08-17T20:50:24.230000+00:00",
            "CompletionTime": "2020-08-17T20:52:18.737000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "sdk-test-job-4",
            "CreationTime": "2020-08-17T20:32:27.917000+00:00",
            "StartTime": "2020-08-17T20:32:27.956000+00:00",
            "CompletionTime": "2020-08-17T20:33:15.126000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
```

```
        {
            "TranscriptionJobName": "Diarization-speak-id",
            "CreationTime": "2020-08-10T22:10:09.066000+00:00",
            "StartTime": "2020-08-10T22:10:09.116000+00:00",
            "CompletionTime": "2020-08-10T22:26:48.172000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        },
        {
            "TranscriptionJobName": "your-transcription-job-name",
            "CreationTime": "2020-07-29T17:45:09.791000+00:00",
            "StartTime": "2020-07-29T17:45:09.826000+00:00",
            "CompletionTime": "2020-07-29T17:46:20.831000+00:00",
            "LanguageCode": "language-code",
            "TranscriptionJobStatus": "COMPLETED",
            "OutputLocationType": "SERVICE_BUCKET"
        }
    ]
}
```

Weitere Informationen finden Sie unter Erste Schritte (AWS Befehlszeilenschnittstelle) im Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie unter ListTranscriptionJobs AWS CLIBefehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient
transcribeClient) {
            ListTranscriptionJobsRequest listJobsRequest =
ListTranscriptionJobsRequest.builder()
                .build();


transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response ->
response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()

.transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
        }
    }
```

- Einzelheiten zur API finden Sie ListTranscriptionJobsin der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten von Transkriptionsaufträgen.

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params),
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie ListTranscriptionJobsin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in
 their
                        names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
 transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
```

```
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- Einzelheiten zur API finden Sie ListTranscriptionJobsin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF iv_job_filter IS NOT INITIAL.
          oo_result = lo_tnb->listtranscriptionjobs( iv_jobnamecontains =
iv_job_filter ).
        ELSE.
          oo_result = lo_tnb->listtranscriptionjobs( ).
        ENDIF.
        MESSAGE 'Retrieved transcription jobs list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListTranscriptionJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListVocabularies`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `ListVocabularies` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a
name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must
contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
nameContains = null,
        VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
```

```
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- Einzelheiten zur API finden Sie ListVocabularies in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Auflisten Ihrer benutzerdefinierten Vokabulare

Das folgende `list-vocabularies` Beispiel listet die benutzerdefinierten Vokabulare auf, die mit Ihrem AWS Konto und Ihrer Region verknüpft sind.

```
aws transcribe list-vocabularies
```

Ausgabe:

```
{
    "NextToken": "NextToken",
    "Vocabularies": [
        {
            "VocabularyName": "ards-test-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-27T22:00:27.330000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "sample-test",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T23:04:11.044000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-3-1",
            "LanguageCode": "language-code",
```

```
            "LastModifiedTime": "2020-04-24T22:12:22.277000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-test-2",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:53:50.455000+00:00",
            "VocabularyState": "READY"
        },
        {
            "VocabularyName": "CRLF-to-LF-1-1",
            "LanguageCode": "language-code",
            "LastModifiedTime": "2020-04-24T21:39:33.356000+00:00",
            "VocabularyState": "READY"
        }
    ]
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie ListVocabularies in der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
```

```
    """
    try:
        response =
transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- Einzelheiten zur API finden Sie ListVocabularies in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF iv_vocab_filter IS NOT INITIAL.
```

```
          oo_result = lo_tnb->listvocabularies( iv_namecontains =
 iv_vocab_filter ).
        ELSE.
          oo_result = lo_tnb->listvocabularies( ).
        ENDIF.
        MESSAGE 'Retrieved vocabularies list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- Einzelheiten zur API finden Sie ListVocabularies in der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartMedicalTranscriptionJob`** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartMedicalTranscriptionJob` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
```

```
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an
Amazon S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new
job.</returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
_amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode =
                    LanguageCode
                        .EnUS, // The value must be en-US for medical
transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of
the transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must
be set.
                Type = transcriptionType
            });
        return response.MedicalTranscriptionJob;
    }
```

- Einzelheiten zur API finden Sie StartMedicalTranscriptionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Transkribieren eines als Audiodatei gespeicherten medizinischen Diktats

Im folgenden Beispiel für `start-medical-transcription-job` wird eine Audiodatei transkribiert. Sie geben den Speicherort der Transkriptionsausgabe im Parameter `OutputBucketName` an.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfile.json
```

Inhalt von `myfile.json`:

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-
job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T00:35:22.256000+00:00",
        "CreationTime": "2020-09-20T00:35:22.218000+00:00",
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
```

```
}
```

Weitere Informationen finden Sie unter Übersicht über die Batch-Transkription im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 2: Transkribieren eines als Audiodatei gespeicherten Dialogs zwischen Arzt und Patient

Im folgenden Beispiel für `start-medical-transcription-job` wird eine Audiodatei mit einem Dialog zwischen Arzt und Patient transkribiert. Sie geben den Speicherort der Transkriptionsausgabe im OutputBucketName Parameter an.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysecondfile.json
```

Inhalt von `mysecondfile.json`:

```
{
    "MedicalTranscriptionJobName": "simple-dictation-medical-transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "simple-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:19:49.965000+00:00",
        "CreationTime": "2020-09-20T23:19:49.941000+00:00",
        "Specialty": "PRIMARYCARE",
```

```
        "Type": "CONVERSATION"
    }
}
```

Weitere Informationen finden Sie unter Übersicht über die Batch-Transkription im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 3: Transkribieren einer Mehrkanal-Audiodatei eines Dialogs zwischen Arzt und Patient

Im folgenden Beispiel für `start-medical-transcription-job` werden die Audiodaten aus jedem Kanal in der Audiodatei transkribiert und die einzelnen Transkriptionen von jedem Kanal zu einer einzigen Transkriptionsausgabe zusammengeführt. Sie geben den Speicherort der Transkriptionsausgabe im Parameter `OutputBucketName` an.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mythirdfile.json
```

Inhalt von `mythirdfile.json`:

```
{
    "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
        "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "Settings":{
          "ChannelIdentification": true
        }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "multichannel-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
```

```
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-20T23:46:44.081000+00:00",
        "CreationTime": "2020-09-20T23:46:44.053000+00:00",
        "Settings": {
            "ChannelIdentification": true
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Weitere Informationen finden Sie unter Kanalidentifizierung im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 4: Transkribieren einer Audiodatei eines Dialogs zwischen Arzt und Patient und Identifizieren der Sprecher in der Transkriptionsausgabe

Im folgenden Beispiel für `start-medical-transcription-job` wird eine Audiodatei transkribiert und die Sprache der einzelnen Sprecher wird in der Transkriptionsausgabe gekennzeichnet. Sie geben den Speicherort der Transkriptionsausgabe im Parameter `OutputBucketName` an.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfourthfile.json
```

Inhalt von `myfourthfile.json`:

```
{
    "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
    "Settings":{
        "ShowSpeakerLabels": true,
```

```
        "MaxSpeakerLabels": 2
        }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "speaker-id-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T18:43:37.265000+00:00",
        "CreationTime": "2020-09-21T18:43:37.157000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Weitere Informationen finden Sie unter Identifizieren von Sprechern im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 5: Transkribieren eines als Audiodatei gespeicherten medizinischen Gesprächs mit bis zu zwei Transkriptionsalternativen

Im folgenden Beispiel für `start-medical-transcription-job` werden bis zu zwei alternative Transkriptionen aus einer einzigen Audiodatei erstellt. Jeder Transkription ist ein gewisses Konfidenzniveau zugeordnet. Standardmäßig gibt Amazon Transcribe die Transkription mit dem höchsten Konfidenzniveau zurück. Sie können angeben, dass Amazon Transcribe zusätzliche Transkriptionen mit niedrigerem Konfidenzniveau zurückgeben soll. Sie geben den Speicherort der Transkriptionsausgabe im Parameter `OutputBucketName` an.

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://myfifthfile.json
```

Inhalt von `myfifthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "CONVERSATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "ShowAlternatives": true,
        "MaxAlternatives": 2
    }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T19:09:18.199000+00:00",
        "CreationTime": "2020-09-21T19:09:18.171000+00:00",
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "CONVERSATION"
    }
}
```

Weitere Informationen finden Sie unter Alternative Transkriptionen im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 6: Transkribieren einer Audiodatei eines medizinischen Diktats mit bis zu zwei alternativen Transkriptionen

Im folgenden Beispiel für `start-medical-transcription-job` wird eine Audiodatei transkribiert und zum Maskieren von unerwünschten Wörtern wird ein Vokabularfilter verwendet. Sie geben den Ort der Transkriptionsausgabe im Parameter an OutputBucketName .

```
aws transcribe start-medical-transcription-job \
    --cli-input-json file://mysixthfile.json
```

Inhalt von `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "alternatives-conversation-medical-
transcription-job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
          "ShowAlternatives": true,
          "MaxAlternatives": 2
    }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "alternatives-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:01:14.592000+00:00",
        "CreationTime": "2020-09-21T21:01:14.569000+00:00",
```

```
        "Settings": {
            "ShowAlternatives": true,
            "MaxAlternatives": 2
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

Weitere Informationen finden Sie unter Alternative Transkriptionen im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 7: Transkribieren einer Audiodatei eines medizinischen Diktats mit höherer Genauigkeit durch Verwendung eines benutzerdefinierten Vokabulars

Im folgenden Beispiel für `start-medical-transcription-job` wird eine Audiodatei transkribiert und zur Verbesserung der Transkriptionsgenauigkeit wird ein zuvor von Ihnen erstelltes benutzerdefiniertes medizinisches Vokabular verwendet. Sie geben den Speicherort der Transkriptionsausgabe im Parameter `OutputBucketName` an.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

Inhalt von `mysixthfile.json`:

```
{
    "MedicalTranscriptionJobName": "vocabulary-dictation-medical-transcription-
job",
    "LanguageCode": "language-code",
    "Specialty": "PRIMARYCARE",
    "Type": "DICTATION",
    "OutputBucketName":"amzn-s3-demo-bucket",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
    },
    "Settings":{
        "VocabularyName": "cli-medical-vocab-1"
    }
}
```

Ausgabe:

```
{
    "MedicalTranscriptionJob": {
        "MedicalTranscriptionJobName": "vocabulary-dictation-medical-
transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.extension"
        },
        "StartTime": "2020-09-21T21:17:27.045000+00:00",
        "CreationTime": "2020-09-21T21:17:27.016000+00:00",
        "Settings": {
            "VocabularyName": "cli-medical-vocab-1"
        },
        "Specialty": "PRIMARYCARE",
        "Type": "DICTATION"
    }
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte medizinische Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie unter StartMedicalTranscriptionJob AWS CLIBefehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
```

```
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Einen Auftrag für medizinische Transkription starten.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and
 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the
 same region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Einzelheiten zur API finden Sie StartMedicalTranscriptionJobin der AWS SDK für JavaScript API-Referenz.

Verwenden Sie **`StartTranscriptionJob`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `StartTranscriptionJob` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars
- Audio transkribieren und Auftragsdaten abrufen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon
 S3 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="languageCode">The language code of the media file, such as
 en-US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</
param>
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName,
 string mediaFileUri,
```

```
        MediaFormat mediaFormat, LanguageCode languageCode, string?
 vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- Einzelheiten zur API finden Sie StartTranscriptionJobin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Transkribieren einer Audiodatei

Im folgenden Beispiel für `start-transcription-job` wird Ihre Audiodatei transkribiert.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfile.json
```

Inhalt von `myfile.json`:

```
{
    "TranscriptionJobName": "cli-simple-transcription-job",
    "LanguageCode": "the-language-of-your-transcription-job",
```

```
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

Weitere Informationen finden Sie unter Erste Schritte (AWS Befehlszeilenschnittstelle) im Amazon Transcribe Developer Guide.

Beispiel 2: Transkribieren einer Mehrkanal-Audiodatei

Im folgenden Beispiel für `start-transcription-job` wird Ihre Mehrkanal-Audiodatei transkribiert.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysecondfile.json
```

Inhalt von `mysecondfile.json`:

```
{
    "TranscriptionJobName": "cli-channelid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "ChannelIdentification":true
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-channelid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
```

```
        "StartTime": "2020-09-17T16:07:56.817000+00:00",
        "CreationTime": "2020-09-17T16:07:56.784000+00:00",
        "Settings": {
            "ChannelIdentification": true
        }
    }
}
```

Weitere Informationen finden Sie unter Multi-Channel Audio transkribieren im Amazon Transcribe Developer Guide.

Beispiel 3: Transkribieren einer Audiodatei und Identifizieren der verschiedenen Sprecher

Im folgenden Beispiel für `start-transcription-job` wird Ihre Audiodatei transkribiert und die Sprecher werden in der Transkriptionsausgabe identifiziert.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mythirdfile.json
```

Inhalt von `mythirdfile.json`:

```
{
    "TranscriptionJobName": "cli-speakerid-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
    "ShowSpeakerLabels": true,
    "MaxSpeakerLabels": 2
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-speakerid-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
```

```
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-17T16:22:59.696000+00:00",
        "CreationTime": "2020-09-17T16:22:59.676000+00:00",
        "Settings": {
            "ShowSpeakerLabels": true,
            "MaxSpeakerLabels": 2
        }
    }
}
```

Weitere Informationen finden Sie unter Identifizieren von Sprechern im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 4: Transkribieren einer Audiodatei und Maskieren aller unerwünschten Wörter in der Transkriptionsausgabe

Im folgenden Beispiel für `start-transcription-job` wird Ihrer Audiodatei transkribiert und zum Maskieren von unerwünschten Wörtern wird ein zuvor von Ihnen erstellter Vokabularfilter verwendet.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfourthfile.json
```

Inhalt von `myfourthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-mask-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
          "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
        "VocabularyFilterMethod": "mask"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-mask-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "mask"
        }
    }
}
```

Weitere Informationen finden Sie unter Filtern von Transkriptionen im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 5: Transkribieren einer Audiodatei und Entfernen aller unerwünschten Wörter aus der Transkriptionsausgabe

Im folgenden Beispiel für `start-transcription-job` wird Ihrer Audiodatei transkribiert und zum Maskieren von unerwünschten Wörtern wird ein zuvor von Ihnen erstellter Vokabularfilter verwendet.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myfifthfile.json
```

Inhalt von `myfifthfile.json`:

```
{
    "TranscriptionJobName": "cli-filter-remove-job",
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyFilterName": "your-vocabulary-filter",
```

```
        "VocabularyFilterMethod": "remove"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-filter-remove-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyFilterName": "your-vocabulary-filter",
            "VocabularyFilterMethod": "remove"
        }
    }
}
```

Weitere Informationen finden Sie unter Filtern von Transkriptionen im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 6: Transkribieren einer Audiodatei mit höherer Genauigkeit durch Verwendung eines benutzerdefinierten Vokabulars

Im folgenden Beispiel für `start-transcription-job` wird Ihrer Audiodatei transkribiert und zum Maskieren von unerwünschten Wörtern wird ein zuvor von Ihnen erstellter Vokabularfilter verwendet.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mysixthfile.json
```

Inhalt von `mysixthfile.json`:

```
{
    "TranscriptionJobName": "cli-vocab-job",
```

```
    "LanguageCode": "the-language-of-your-transcription-job",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    },
    "Settings":{
        "VocabularyName": "your-vocabulary"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-vocab-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "the-language-of-your-transcription-job",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T16:36:18.568000+00:00",
        "CreationTime": "2020-09-18T16:36:18.547000+00:00",
        "Settings": {
            "VocabularyName": "your-vocabulary"
        }
    }
}
```

Weitere Informationen finden Sie unter Filtern von Transkriptionen im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 7: Identifizieren der Sprache einer Audiodatei und Transkribieren der Datei

Im folgenden Beispiel für `start-transcription-job` wird Ihrer Audiodatei transkribiert und zum Maskieren von unerwünschten Wörtern wird ein zuvor von Ihnen erstellter Vokabularfilter verwendet.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myseventhfile.json
```

Inhalt von `myseventhfile.json`:

```
{
    "TranscriptionJobName": "cli-identify-language-transcription-job",
    "IdentifyLanguage": true,
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-media-
file-name.file-extension"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-identify-language-transcription-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/Amazon-S3-prefix/your-
media-file-name.file-extension"
        },
        "StartTime": "2020-09-18T22:27:23.970000+00:00",
        "CreationTime": "2020-09-18T22:27:23.948000+00:00",
        "IdentifyLanguage": true
    }
}
```

Weitere Informationen finden Sie unter Identifizieren der Sprache im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 8: Transkribieren einer Audiodatei mit unkenntlich gemachten persönlich identifizierbaren Informationen

Im folgenden Beispiel für `start-transcription-job` wird Ihre Audiodatei transkribiert und die persönlich identifizierbaren Informationen werden in der Transkriptionsausgabe unkenntlich gemacht.

```
aws transcribe start-transcription-job \
    --cli-input-json file://myeighthfile.json
```

Inhalt von `myeigthfile.json`:

```
{
```

```
    "TranscriptionJobName": "cli-redaction-job",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
    },
    "ContentRedaction": {
        "RedactionOutput":"redacted",
        "RedactionType":"PII"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:49:13.195000+00:00",
        "CreationTime": "2020-09-25T23:49:13.176000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted"
        }
    }
}
```

Weitere Informationen finden Sie unter Automatische Inhaltsschwärzung im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 9: Generieren eines Transkripts mit unkenntlich gemachten persönlich identifizierbaren Informationen (PII) und eines ungeschwärzten Transkripts

Im folgenden Beispiel für `start-transcription-job` werden zwei Transkriptionen Ihrer Audiodatei generiert, eine mit unkenntlich gemachten persönlich identifizierbaren Informationen und die andere ohne Schwärzungen.

```
aws transcribe start-transcription-job \
```

```
    --cli-input-json file://myninthfile.json
```

Inhalt von `myninthfile.json`:

```
{
    "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
    "LanguageCode": "language-code",
    "Media": {
          "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-extension"
        },
    "ContentRedaction": {
        "RedactionOutput":"redacted_and_unredacted",
        "RedactionType":"PII"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-redaction-job-with-unredacted-transcript",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://Amazon-S3-Prefix/your-media-file.file-
extension"
        },
        "StartTime": "2020-09-25T23:59:47.677000+00:00",
        "CreationTime": "2020-09-25T23:59:47.653000+00:00",
        "ContentRedaction": {
            "RedactionType": "PII",
            "RedactionOutput": "redacted_and_unredacted"
        }
    }
}
```

Weitere Informationen finden Sie unter Automatische Inhaltsschwärzung im Amazon-Transcribe-Entwicklerhandbuch.

Beispiel 10: Verwenden eines benutzerdefinierten Sprachmodells, das Sie zuvor erstellt haben, um eine Audiodatei zu transkribieren

Im folgenden Beispiel für `start-transcription-job` wird Ihre Audiodatei mit einem benutzerdefinierten Sprachmodell transkribiert, das Sie zuvor erstellt haben.

```
aws transcribe start-transcription-job \
    --cli-input-json file://mytenthfile.json
```

Inhalt von `mytenthfile.json`:

```
{
    "TranscriptionJobName": "cli-clm-2-job-1",
    "LanguageCode": "language-code",
    "Media": {
        "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-extension"
    },
    "ModelSettings": {
        "LanguageModelName":"cli-clm-2"
    }
}
```

Ausgabe:

```
{
    "TranscriptionJob": {
        "TranscriptionJobName": "cli-clm-2-job-1",
        "TranscriptionJobStatus": "IN_PROGRESS",
        "LanguageCode": "language-code",
        "Media": {
            "MediaFileUri": "s3://amzn-s3-demo-bucket/your-audio-file.file-
extension"
        },
        "StartTime": "2020-09-28T17:56:01.835000+00:00",
        "CreationTime": "2020-09-28T17:56:01.801000+00:00",
        "ModelSettings": {
            "LanguageModelName": "cli-clm-2"
        }
    }
}
```

Weitere Informationen finden Sie unter Verbessern der Domain-Specific Transkriptionsgenauigkeit mit benutzerdefinierten Sprachmodellen im Amazon Transcribe Developer Guide.

- Einzelheiten zur API finden Sie StartTranscriptionJobin AWS CLI der Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einen Transkriptionsauftrag starten.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME",
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Einzelheiten zur API finden Sie StartTranscriptionJobin der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.
```

```
    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- Einzelheiten zur API finden Sie StartTranscriptionJobin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lo_media) = NEW /aws1/cl_tnbmedia( iv_mediafileuri = iv_media_uri ).
       DATA(lo_settings) = NEW /aws1/cl_tnbsettings( ).
       IF iv_vocabulary_name IS NOT INITIAL.
         lo_settings = NEW /aws1/cl_tnbsettings( iv_vocabularyname =
iv_vocabulary_name ).
       ENDIF.

       oo_result = lo_tnb->starttranscriptionjob(
         iv_transcriptionjobname = iv_job_name
         io_media = lo_media
         iv_mediaformat = iv_media_format
         iv_languagecode = iv_language_code
         io_settings = lo_settings ).

       MESSAGE 'Transcription job started.' TYPE 'I'.
     CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
       MESSAGE lo_bad_request_ex TYPE 'I'.
       RAISE EXCEPTION lo_bad_request_ex.
     CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
       MESSAGE lo_limit_ex TYPE 'I'.
       RAISE EXCEPTION lo_limit_ex.
     CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
       MESSAGE lo_internal_ex TYPE 'I'.
       RAISE EXCEPTION lo_internal_ex.
     CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
       MESSAGE lo_conflict_ex TYPE 'I'.
       RAISE EXCEPTION lo_conflict_ex.
   ENDTRY.
```

- Einzelheiten zur API finden Sie StartTranscriptionJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`UpdateVocabulary`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `UpdateVocabulary` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update a custom vocabulary with new values. Update overwrites all
existing information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
```

```
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- Einzelheiten zur API finden Sie UpdateVocabularyin der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Aktualisieren eines benutzerdefinierten Vokabular mit neuen Begriffen

Im folgenden Beispiel für `update-vocabulary` werden die Begriffe, die zur Erstellung eines benutzerdefinierten Vokabulars verwendet wurden, mit den von Ihnen angegebenen neuen Begriffen überschrieben. Voraussetzung: Um die Begriffe in einem benutzerdefinierten Wortschatz zu ersetzen, benötigen Sie eine Datei mit neuen Begriffen.

```
aws transcribe update-vocabulary \
    --vocabulary-file-uri s3://amzn-s3-demo-bucket/Amazon-S3-Prefix/custom-vocabulary.txt \
    --vocabulary-name custom-vocabulary \
    --language-code language-code
```

Ausgabe:

```
{
    "VocabularyName": "custom-vocabulary",
    "LanguageCode": "language",
    "VocabularyState": "PENDING"
}
```

Weitere Informationen finden Sie unter Benutzerdefinierte Vokabulare im Amazon-Transcribe-Entwicklerhandbuch.

- Einzelheiten zur API finden Sie UpdateVocabularyin der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise
```

- Einzelheiten zur API finden Sie UpdateVocabularyin AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Vocabulary updated.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
```

```
    ENDTRY.
```

- Einzelheiten zur API finden Sie UpdateVocabularyin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Transcribe AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Transcribe mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Transcribe oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen einer Amazon-Transcribe-Streaming-App
- Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK
- Erstellen und verfeinern Sie ein benutzerdefiniertes Amazon Transcribe Transcribe-Vokabular mit einem AWS SDK
- Transkribieren Sie Audio und rufen Sie Auftragsdaten mit Amazon Transcribe mithilfe eines AWS SDK

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Konvertieren Sie Text in Sprache und zurück in Text mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Klartext-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

Rust

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, die Audiodatei in einen Amazon S3 S3-Bucket hochzuladen, Amazon Transcribe zu verwenden, um diese Audiodatei in Text zu konvertieren und den Text anzuzeigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Erstellen und verfeinern Sie ein benutzerdefiniertes Amazon Transcribe Transcribe-Vokabular mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Laden Sie eine Audiodatei auf Amazon S3 hoch.
- Führen Sie einen Amazon Transcribe-Auftrag aus, um die Datei zu transkribieren und die Ergebnisse zu erhalten.
- Erstellen und verfeinern Sie ein benutzerdefiniertes Vokabular, um die Transkriptionsgenauigkeit zu verbessern.
- Führen Sie Aufträge mit benutzerdefinierten Vokabularen aus und erhalten Sie die Ergebnisse.

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Transkribieren Sie eine Audiodatei, die eine Lesung von Jabberwocky von Lewis Carroll enthält. Beginnen Sie damit, Funktionen zu erstellen, die Amazon Transcribe-Aktionen wrappen.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is
 started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.
```

```
    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
 transcribing
                             the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
```

```
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary
 processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
```

```
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.",
 vocabulary_name)
        raise
    else:
        return response



def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response =
 transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
 table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the
 vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in
 the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.",
 vocabulary_name)
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string
 in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
```

```
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Rufen Sie die Wrapper-Funktionen auf, um Audio ohne ein benutzerdefiniertes Vokabular und anschließend mit verschiedenen Versionen eines benutzerdefinierten Vokabulars zu transkribieren, um bessere Ergebnisse zu erzielen.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
```

```
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
        "mp3",
        "en-US",
```

```
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional
"
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])
```

```
    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)

    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob

- GetVocabulary
- ListVocabularies
- StartTranscriptionJob
- UpdateVocabulary

Transkribieren Sie Audio und rufen Sie Auftragsdaten mit Amazon Transcribe mithilfe eines AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Starten Sie einen Transkriptionsauftrag mit Amazon Transcribe.
- Warten Sie, bis der -Auftrag abgeschlossen wurde.
- Ermitteln Sie die URI, unter der das Transkript gespeichert ist.

Weitere Informationen finden Sie unter Erste Schritte mit Amazon Transcribe.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Transkribiert eine PCM-Datei.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
```

```
    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
```

```
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
 audioEventFromBuffer(audioBuffer);
```

```
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

Transkribiert Streaming-Audio vom Mikrofon Ihres Computers.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();
```

```
        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
```

```
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }
```

```
            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
```

```
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetTranscriptionJob
  - StartTranscriptionJob

Python

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import time
import boto3


def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
        Media={"MediaFileUri": file_uri},
        MediaFormat="wav",
        LanguageCode="en-US",
    )

    max_tries = 60
    while max_tries > 0:
```

```
        max_tries -= 1
        job =
 transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job["TranscriptionJob"]["TranscriptionJobStatus"]
        if job_status in ["COMPLETED", "FAILED"]:
            print(f"Job {job_name} is {job_status}.")
            if job_status == "COMPLETED":
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}."
                )
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - GetTranscriptionJob
  - StartTranscriptionJob

Codebeispiele für Amazon Transcribe Streaming mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Transcribe Streaming mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Transcribe Streaming – Weitere Informationen zu Amazon Transcribe Streaming.
- API-Referenz für Amazon Transcribe Streaming – Details zu allen verfügbaren Aktionen von Amazon Transcribe Streaming.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für Amazon Transcribe Streaming mit AWS SDKs
  - Aktionen für Amazon Transcribe Streaming mit AWS SDKs
    - Verwenden Sie StartMedicalStreamTranscriptionmit einem AWS SDK
    - Verwenden Sie StartStreamTranscriptionmit einem AWS SDK
- Szenarien für Amazon Transcribe Streaming mit AWS SDKs
  - Generieren Sie eine Transkription einer Datei mit einem AWS SDK
  - Generieren Sie mit einem Mikrofon eine Transkription AWS SDK

Grundlegende Beispiele für Amazon Transcribe Streaming mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Transcribe Streaming mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Transcribe Streaming mit AWS SDKs
  - Verwenden Sie StartMedicalStreamTranscriptionmit einem AWS SDK
  - Verwenden Sie StartStreamTranscriptionmit einem AWS SDK

Aktionen für Amazon Transcribe Streaming mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Transcribe Streaming-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Transcribe Streaming API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für Amazon Transcribe Streaming mit AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Transcribe.

Beispiele

- Verwenden Sie StartMedicalStreamTranscriptionmit einem AWS SDK
- Verwenden Sie StartStreamTranscriptionmit einem AWS SDK

Verwenden Sie **`StartMedicalStreamTranscription`**mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird`StartMedicalStreamTranscription`.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
To run this AWS code example, ensure that you have set up your development
 environment, including your AWS credentials.

 For information, see this documentation topic:

 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
This code demonstrates the process of starting a medical transcription job using
 the AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException
 {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
 {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            throw new LineUnavailableException("The audio system microphone line
 is not supported.");
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();
```

```
        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartMedicalStreamTranscriptionRequest
 getMedicalRequest(Integer mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
 transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or
 DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
 getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
 event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {
```

```
System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
```

```
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
```

```
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                .audioChunk(SdkBytes.fromByteBuffer(bb))
                .build();
        }
    }
}
```

- Einzelheiten zur API finden Sie StartMedicalStreamTranscriptionin der AWS SDK for Java 2.x API-Referenz.

Verwenden Sie **`StartStreamTranscription`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartStreamTranscription` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- Transkribieren einer Audiodatei
- Transkribieren von Audio von einem Mikrofon

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
int main() {
    Aws::SDKOptions options;
```

```
    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if
 the SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file
 transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
```

```
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
                    //the number of characters extracted by the last read()
operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
std::endl;
                            break;
                        }
                    }
                    else {
                        break;
```

```
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a
 file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" <<
 std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }
```

```
    Aws::ShutdownAPI(options);

    return 0;
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }
```

```
    private static InputStream getStreamFromMic() throws LineUnavailableException
{

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
```

```
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
```

```
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }
```

```
        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK for Java 2.x API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode:
TranscribeStreamingClientTypes.LanguageCode(rawValue: lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der API-Referenz zum AWS SDK für Swift.

Szenarien für Amazon Transcribe Streaming mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in Amazon Transcribe Streaming mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Transcribe Streaming oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Generieren Sie eine Transkription einer Datei mit einem AWS SDK
- Generieren Sie mit einem Mikrofon eine Transkription AWS SDK

Generieren Sie eine Transkription einer Datei mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie mit Amazon Transcribe-Streaming eine Transkription einer Quell-Audiodatei generiert wird.

C++

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if
 the SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
```

```
                []([(const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file
transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;
```

```
                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
 characters in the given string. It returns
                    //the number of characters extracted by the last read()
 operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
 std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a
 file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" <<
 std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
```

```
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK für C++ API-Referenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * To run this AWS code example, ensure that you have set up your development
```

```
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
```

```
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
```

```
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
 inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
 positive"));
            }

            demand.getAndAdd(n);
```

```
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }
```

```
        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK for Java 2.x API-Referenz.

Swift

SDK für Swift

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Transcribe Streaming, um die gesprochene Sprache in eine Audiodatei zu transkribieren.

```
/// An example that demonstrates how to watch an transcribe event stream to
/// transcribe audio from a file to the console.

import ArgumentParser
import AWSClientRuntime
import AWSTranscribeStreaming
import Foundation

/// Identify one of the media file formats supported by Amazon Transcribe.
enum TranscribeFormat: String, ExpressibleByArgument {
    case ogg = "ogg"
    case pcm = "pcm"
    case flac = "flac"
}
```

```
// -MARK: - Async command line tool

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Flag(help: "Show partial results")
    var showPartial = false
    @Option(help: "Language code to transcribe into")
    var lang: String = "en-US"
    @Option(help: "Format of the source audio file")
    var format: TranscribeFormat
    @Option(help: "Sample rate of the source audio file in Hertz")
    var sampleRate: Int = 16000
    @Option(help: "Path of the source audio file")
    var path: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "tsevents",
        abstract: """
        This example shows how to use event streaming with Amazon Transcribe.
        """,
        discussion: """
        """
    )

    /// Create and return an Amazon Transcribe audio stream from the file
    /// specified in the arguments.
    ///
    /// - Throws: Errors from `TranscribeError`.
    ///
    /// - Returns:
 `AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream, Error>`
    func createAudioStream() async throws
                ->
 AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream, Error> {

        let fileURL: URL = URL(fileURLWithPath: path)
        let audioData = try Data(contentsOf: fileURL)

        // Properties defining the size of audio chunks and the total size of
        // the audio file in bytes. You should try to send chunks that last on
        // average 125 milliseconds.
```

```
        let chunkSizeInMilliseconds = 125.0
        let chunkSize = Int(chunkSizeInMilliseconds  / 1000.0 *
Double(sampleRate) * 2.0)
        let audioDataSize = audioData.count

        // Create an audio stream from the source data. The stream's job is
        // to send the audio in chunks to Amazon Transcribe as
        // `AudioStream.audioevent` events.

        let audioStream =
AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
                                Error> { continuation in
            Task {
                var currentStart = 0
                var currentEnd = min(chunkSize, audioDataSize - currentStart)

                // Generate and send chunks of audio data as `audioevent`
                // events until the entire file has been sent. Each event is
                // yielded to the SDK after being created.

                while currentStart < audioDataSize {
                    let dataChunk = audioData[currentStart ..< currentEnd]

                    let audioEvent =
TranscribeStreamingClientTypes.AudioStream.audioevent(
                        .init(audioChunk: dataChunk)
                    )
                    let yieldResult = continuation.yield(audioEvent)
                    switch yieldResult {
                        case .enqueued(_):
                            // The chunk was successfully enqueued into the
                            // stream. The `remaining` parameter estimates how
                            // much room is left in the queue, but is ignored
here.
                            break
                        case .dropped(_):
                            // The chunk was dropped because the queue buffer
                            // is full. This will cause transcription errors.
                            print("Warning: Dropped audio! The transcription will
be incomplete.")
                        case .terminated:
                            print("Audio stream terminated.")
                            continuation.finish()
                            return
```

```
                        default:
                            print("Warning: Unrecognized response during audio
streaming.")
                    }

                    currentStart = currentEnd
                    currentEnd = min(currentStart + chunkSize, audioDataSize)
                }

                // Let the SDK's continuation block know the stream is over.

                continuation.finish()
            }
        }

        return audioStream
    }

    /// Run the transcription process.
    ///
    /// - Throws: An error from `TranscribeError`.
    func transcribe(encoding: TranscribeStreamingClientTypes.MediaEncoding) async
throws {
        // Create the Transcribe Streaming client.

        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode:
TranscribeStreamingClientTypes.LanguageCode(rawValue: lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

```
        // Iterate over the events in the returned transcript result stream.
        // Each `transcriptevent` contains a list of result fragments which
        // need to be concatenated together to build the final transcript.
        for try await event in output.transcriptResultStream! {
            switch event {
            case .transcriptevent(let event):
            for result in event.transcript?.results ?? [] {
                guard let transcript = result.alternatives?.first?.transcript
else {
                    continue
                }

                // If showing partial results is enabled and the result is
                // partial, show it. Partial results may be incomplete, and
                // may be inaccurate, with upcoming audio making the
                // transcription complete or by giving more context to make
                // transcription make more sense.

                if (result.isPartial && showPartial) {
                    print("[Partial] \(transcript)")
                }

                // When the complete fragment of transcribed text is ready,
                // print it. This could just as easily be used to draw the
                // text as a subtitle over a playing video, though timing
                // would need to be managed.

                if !result.isPartial {
                    if (showPartial) {
                        print("[Final  ] ", terminator: "")
                    }
                    print(transcript)
                }
            }
            default:
                print("Error: Unexpected message from Amazon Transcribe:")
            }
        }
    }

    /// Convert the value of the `--format` command line option into the
    /// corresponding Transcribe Streaming `MediaEncoding` type.
    ///
    /// - Returns: The `MediaEncoding` equivalent of the format specified on
```

```
    ///   the command line.
    func getMediaEncoding() -> TranscribeStreamingClientTypes.MediaEncoding {
        let mediaEncoding: TranscribeStreamingClientTypes.MediaEncoding

        switch format {
        case .flac:
            mediaEncoding = .flac
        case .ogg:
            mediaEncoding = .oggOpus
        case .pcm:
            mediaEncoding = .pcm
        }

        return mediaEncoding
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.transcribe(encoding: command.getMediaEncoding())
        } catch let error as TranscribeError {
            print("ERROR: \(error.errorDescription ?? "Unknown error")")
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

/// Errors thrown by the example's functions.
enum TranscribeError: Error {
    /// No transcription stream available.
    case noTranscriptionStream
    /// The source media file couldn't be read.
    case readError

    var errorDescription: String? {
```

```
        switch self {
        case .noTranscriptionStream:
            return "No transcription stream returned by Amazon Transcribe."
        case .readError:
            return "Unable to read the source audio file."
        }
    }
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der API-Referenz zum AWS SDK für Swift.

Generieren Sie mit einem Mikrofon eine Transkription AWS SDK

Das folgende Codebeispiel zeigt, wie mit Amazon Transcribe Streaming eine Transkription von einem Mikrofon generiert wird.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();
```

```
        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException
{

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
```

```
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully
 ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
```

```
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream
inputStream) {
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
```

```
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für Amazon Translate mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon Translate mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- Entwicklerhandbuch für Amazon Translate – Weitere Informationen zu Amazon Translate.
- API-Referenz für Amazon Translate – Details zu allen verfügbaren Amazon-Translate-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung von Amazon Translate AWS SDKs
  - Aktionen für Amazon Translate mit AWS SDKs
    - Verwenden Sie DescribeTextTranslationJobmit einem AWS SDK
    - Verwenden Sie ListTextTranslationJobsmit einem AWS SDK
    - Verwenden Sie StartTextTranslationJobmit einem AWS SDK
    - Verwenden Sie StopTextTranslationJobmit einem AWS SDK
    - Verwenden Sie TranslateTextmit einem AWS SDK oder CLI
- Szenarien für die Verwendung von Amazon Translate AWS SDKs
  - Erstellen einer Amazon-Transcribe-Streaming-App
  - Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher
  - Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
  - Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
  - Erste Schritte mit Amazon Translate Translate-Jobs mithilfe eines AWS SDK

Grundlegende Beispiele für die Verwendung von Amazon Translate AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Translate mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für Amazon Translate mit AWS SDKs
  - Verwenden Sie DescribeTextTranslationJobmit einem AWS SDK
  - Verwenden Sie ListTextTranslationJobsmit einem AWS SDK
  - Verwenden Sie StartTextTranslationJobmit einem AWS SDK
  - Verwenden Sie StopTextTranslationJobmit einem AWS SDK
  - Verwenden Sie TranslateTextmit einem AWS SDK oder CLI

Aktionen für Amazon Translate mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon Translate Translate-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon Translate API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung von Amazon Translate AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der API-Referenz für Amazon Translate.

Beispiele

- Verwenden Sie DescribeTextTranslationJobmit einem AWS SDK
- Verwenden Sie ListTextTranslationJobsmit einem AWS SDK
- Verwenden Sie StartTextTranslationJobmit einem AWS SDK
- Verwenden Sie StopTextTranslationJobmit einem AWS SDK
- Verwenden Sie TranslateTextmit einem AWS SDK oder CLI

Verwenden Sie **`DescribeTextTranslationJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `DescribeTextTranslationJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Übersetzungsaufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// The following example shows how to retrieve the details of
/// a text translation job using Amazon Translate.
/// </summary>
public class DescribeTextTranslation
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();

        // The Job Id is generated when the text translation job is started
        // with a call to the StartTextTranslationJob method.
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new DescribeTextTranslationJobRequest
        {
            JobId = jobId,
        };
```

```
            var jobProperties = await DescribeTranslationJobAsync(client,
 request);

            DisplayTranslationJobDetails(jobProperties);
        }

        /// <summary>
        /// Retrieve information about an Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client
 object.</param>
        /// <param name="request">The DescribeTextTranslationJobRequest object.</
param>
        /// <returns>The TextTranslationJobProperties object containing
        /// information about the text translation job..</returns>
        public static async Task<TextTranslationJobProperties>
 DescribeTranslationJobAsync(
            AmazonTranslateClient client,
            DescribeTextTranslationJobRequest request)
        {
            var response = await client.DescribeTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                return response.TextTranslationJobProperties;
            }
            else
            {
                return null;
            }
        }

        /// <summary>
        /// Displays the properties of the text translation job.
        /// </summary>
        /// <param name="jobProperties">The properties of the text translation
        /// job returned by the call to DescribeTextTranslationJobAsync.</param>
        public static void
 DisplayTranslationJobDetails(TextTranslationJobProperties jobProperties)
        {
            if (jobProperties is null)
            {
                Console.WriteLine("No text translation job properties found.");
                return;
```

```
            }

            // Display the details of the text translation job.
            Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
        }
    }
```

- Einzelheiten zur API finden Sie DescribeTextTranslationJobin der AWS SDK für .NET API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets the properties associated with an asynchronous batch translation job."
   "Includes properties such as name, ID, status, source and target languages,
and input/output Amazon Simple Storage Service (Amazon S3) buckets."
   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
returned for testing purposes."
         iv_jobid        = iv_jobid ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie DescribeTextTranslationJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`ListTextTranslationJobs`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `ListTextTranslationJobs` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.Translate;
   using Amazon.Translate.Model;

   /// <summary>
   /// List Amazon Translate translation jobs, along with details about each
job.
   /// </summary>
   public class ListTranslationJobs
   {
       public static async Task Main()
       {
           var client = new AmazonTranslateClient();
           var filter = new TextTranslationJobFilter
           {
               JobStatus = "COMPLETED",
           };

           var request = new ListTextTranslationJobsRequest
           {
               MaxResults = 10,
               Filter = filter,
           };
```

```
            await ListJobsAsync(client, request);
        }

        /// <summary>
        /// List Amazon Translate text translation jobs.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client
object.</param>
        /// <param name="request">An Amazon Translate
        /// ListTextTranslationJobsRequest object detailing which text
        /// translation jobs are of interest.</param>
        public static async Task ListJobsAsync(
            AmazonTranslateClient client,
            ListTextTranslationJobsRequest request)
        {
            ListTextTranslationJobsResponse response;

            do
            {
                response = await client.ListTextTranslationJobsAsync(request);

ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

                request.NextToken = response.NextToken;
            }
            while (response.NextToken is not null);
        }

        /// <summary>
        /// List existing translation job details.
        /// </summary>
        /// <param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
        {
            properties.ForEach(prop =>
            {
                Console.WriteLine($"{prop.JobId}: {prop.JobName}");
                Console.WriteLine($"Status: {prop.JobStatus}");
                Console.WriteLine($"Submitted time: {prop.SubmittedTime}");
            });
        }
```

```
    }
```

- Einzelheiten zur API finden Sie ListTextTranslationJobsin der AWS SDK für .NET API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets a list of the batch translation jobs that you have submitted."

   DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

   "Create an ABAP object for filtering using jobname."
   lo_filter = NEW #( iv_jobname = iv_jobname ).

   TRY.
       oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned
for testing purposes."
         io_filter        = lo_filter ).
       MESSAGE 'Jobs retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE
'E'.
     CATCH /aws1/cx_xl8invalidfilterex.
       MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie ListTextTranslationJobsin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StartTextTranslationJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StartTextTranslationJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Übersetzungsaufträgen

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// This example shows how to use Amazon Translate to process the files in
    /// an Amazon Simple Storage Service (Amazon S3) bucket. The translated
results
    /// will also be stored in an Amazon S3 bucket.
    /// </summary>
    public class BatchTranslate
    {
        public static async Task Main()
        {
            var contentType = "text/plain";
```

```
            // Set this variable to an S3 bucket location with a folder."
            // Input files must be in a folder and not at the bucket root."
            var s3InputUri = "s3://amzn-s3-demo-bucket1/FOLDER/";
            var s3OutputUri = "s3://amzn-s3-demo-bucket2/";

            // This role must have permissions to read the source bucket and to
 read and
            // write to the destination bucket where the translated text will be
 stored.
            var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

            var client = new AmazonTranslateClient();

            var inputConfig = new InputDataConfig
            {
                ContentType = contentType,
                S3Uri = s3InputUri,
            };

            var outputConfig = new OutputDataConfig
            {
                S3Uri = s3OutputUri,
            };

            var request = new StartTextTranslationJobRequest
            {
                JobName = "ExampleTranslationJob",
                DataAccessRoleArn = dataAccessRoleArn,
                InputDataConfig = inputConfig,
                OutputDataConfig = outputConfig,
                SourceLanguageCode = "en",
                TargetLanguageCodes = new List<string> { "fr" },
            };

            var response = await StartTextTranslationAsync(client, request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId}: {response.JobStatus}");
            }
        }
```

```
        /// <summary>
        /// Start the Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized AmazonTranslateClient object.</
param>
        /// <param name="request">The request object that includes details such
        /// as source and destination bucket names and the IAM Role that will
        /// be used to access the buckets.</param>
        /// <returns>The StartTextTranslationResponse object that includes the
        /// details of the request response.</returns>
        public static async Task<StartTextTranslationJobResponse>
 StartTextTranslationAsync(AmazonTranslateClient client,
 StartTextTranslationJobRequest request)
        {
            var response = await client.StartTextTranslationJobAsync(request);
            return response;
        }
    }
```

- Einzelheiten zur API finden Sie StartTextTranslationJobin der AWS SDK für .NET API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Starts an asynchronous batch translation job."
    "Use batch translation jobs to translate large volumes of text across
 multiple documents at once."

    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
```

```
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned
 for testing purposes."
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language
 of the source text into the requested target language.' TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StartTextTranslationJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`StopTextTranslationJob`**mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie `StopTextTranslationJob` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Shows how to stop a running Amazon Translation Service text translation
/// job.
/// </summary>
public class StopTextTranslationJob
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new StopTextTranslationJobRequest
        {
            JobId = jobId,
        };

        await StopTranslationJobAsync(client, request);
    }

    /// <summary>
```

```
        /// Sends a request to stop a text translation job.
        /// </summary>
        /// <param name="client">Initialized AmazonTrnslateClient object.</param>
        /// <param name="request">The request object to be passed to the
        /// StopTextJobAsync method.</param>
        public static async Task StopTranslationJobAsync(
            AmazonTranslateClient client,
            StopTextTranslationJobRequest request)
        {
            var response = await client.StopTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
            }
        }
    }
```

- Einzelheiten zur API finden Sie StopTextTranslationJobin der AWS SDK für .NET API-Referenz.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Stops an asynchronous batch translation job that is in progress."

    TRY.
        oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned
for testing purposes."
          iv_jobid        = iv_jobid ).
        MESSAGE 'Translation job stopped.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
```

```
        MESSAGE 'An internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    ENDTRY.
```

- Einzelheiten zur API finden Sie StopTextTranslationJobin der API-Referenz zum AWS SDK für SAP ABAP.

Verwenden Sie **`TranslateText`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `TranslateText` verwendet wird.

.NET

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Take text from a file stored a Amazon Simple Storage Service (Amazon S3)
/// object and translate it using the Amazon Transfer Service.
/// </summary>
public class TranslateText
{
    public static async Task Main()
```

```
        {
            // If the region you want to use is different from the region
            // defined for the default user, supply it as a parameter to the
            // Amazon Translate client object constructor.
            var client = new AmazonTranslateClient();

            // Set the source language to "auto" to request Amazon Translate to
            // automatically detect te language of the source text.

            // You can get a list of the languages supposed by Amazon Translate
            // in the Amazon Translate Developer's Guide here:
            //      https://docs.aws.amazon.com/translate/latest/dg/what-is.html
            string srcLang = "en"; // English.
            string destLang = "fr"; // French.

            // The Amazon Simple Storage Service (Amazon S3) bucket where the
            // source text file is stored.
            string srcBucket = "amzn-s3-demo-bucket";
            string srcTextFile = "source.txt";

            var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
            var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

            ShowText(srcText, destText);
        }

        /// <summary>
        /// Use the Amazon S3 TransferUtility to retrieve the text to translate
        /// from an object in an S3 bucket.
        /// </summary>
        /// <param name="srcBucket">The name of the S3 bucket where the
        /// text is stored.
        /// </param>
        /// <param name="srcTextFile">The key of the S3 object that
        /// contains the text to translate.</param>
        /// <returns>A string representing the source text.</returns>
        public static async Task<string> GetSourceTextAsync(string srcBucket,
string srcTextFile)
        {
            string srcText = string.Empty;

            var s3Client = new AmazonS3Client();
            TransferUtility utility = new TransferUtility(s3Client);
```

```
            using var stream = await utility.OpenStreamAsync(srcBucket,
 srcTextFile);

            StreamReader file = new System.IO.StreamReader(stream);

            srcText = file.ReadToEnd();
            return srcText;
        }

        /// <summary>
        /// Use the Amazon Translate Service to translate the document from the
        /// source language to the specified destination language.
        /// </summary>
        /// <param name="client">The Amazon Translate Service client used to
        /// perform the translation.</param>
        /// <param name="srcLang">The language of the source text.</param>
        /// <param name="destLang">The destination language for the translated
        /// text.</param>
        /// <param name="text">A string representing the text to ranslate.</
param>
        /// <returns>The text that has been translated to the destination
        /// language.</returns>
        public static async Task<string>
 TranslatingTextAsync(AmazonTranslateClient client, string srcLang, string
 destLang, string text)
        {
            var request = new TranslateTextRequest
            {
                SourceLanguageCode = srcLang,
                TargetLanguageCode = destLang,
                Text = text,
            };

            var response = await client.TranslateTextAsync(request);

            return response.TranslatedText;
        }

        /// <summary>
        /// Show the original text followed by the translated text.
        /// </summary>
        /// <param name="srcText">The original text to be translated.</param>
        /// <param name="destText">The translated text.</param>
```

```
        public static void ShowText(string srcText, string destText)
        {
            Console.WriteLine("Source text:");
            Console.WriteLine(srcText);
            Console.WriteLine();
            Console.WriteLine("Translated text:");
            Console.WriteLine(destText);
        }
    }
```

- Einzelheiten zur API finden Sie TranslateTextin der AWS SDK für .NET API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Konvertiert den angegebenen englischen Text in Französisch. Der zu konvertierende Text kann auch als Parameter -Text übergeben werden.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Einzelheiten zur API finden Sie unter TranslateText AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Konvertiert den angegebenen englischen Text in Französisch. Der zu konvertierende Text kann auch als Parameter -Text übergeben werden.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- Einzelheiten zur API finden Sie unter TranslateText AWS -Tools für PowerShellCmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Translates input text from the source language to the target language."
   TRY.
       oo_result = lo_xl8->translatetext(      "oo_result is returned for
testing purposes."
         iv_text        = iv_text
           iv_sourcelanguagecode = iv_sourcelanguagecode
           iv_targetlanguagecode = iv_targetlanguagecode ).
       MESSAGE 'Translation completed.' TYPE 'I'.
     CATCH /aws1/cx_xl8detectedlanguage00.
       MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8serviceunavailex.
       MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE
'E'.
     CATCH /aws1/cx_xl8textsizelmtexcdex.
       MESSAGE 'The size of the text you submitted exceeds the size limit. '
TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
     CATCH /aws1/cx_xl8unsuppedlanguage00.
       MESSAGE 'Amazon Translate does not support translation from the language
of the source text into the requested target language. ' TYPE 'E'.
   ENDTRY.
```

- Einzelheiten zur API finden Sie TranslateTextin der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien für die Verwendung von Amazon Translate AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon Translate mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon Translate oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erstellen einer Amazon-Transcribe-Streaming-App
- Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher
- Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt
- Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert
- Erste Schritte mit Amazon Translate Translate-Jobs mithilfe eines AWS SDK

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Erstellen eines Amazon Lex-Chatbots zur Einbindung Ihrer Website-Besucher

Die folgenden Codebeispiele zeigen, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

Java

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel „Einen Amazon Lex-Chatbot erstellen“ im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Erstellen einer Publish- und Abonnement-Anwendung, die Nachrichten übersetzt

Die folgende Code-Beispiele zeigen, wie man eine Anwendung erstellt, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

.NET

SDK für .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Java

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung des Beispiels, das die Java Async API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Kotlin

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Anwendung, die Kundenfeedback analysiert und Audio generiert

Die folgenden Codebeispiele zeigen, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus ihrer Originalsprache übersetzt, ihre Stimmung ermittelt und aus dem übersetzten Text eine Audiodatei generiert.

.NET

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.

- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract

- Amazon Translate

JavaScript

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie der innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
```

```
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
```

```
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
```

```
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Ruby

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Erste Schritte mit Amazon Translate Translate-Jobs mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen asynchronen Stapelübersetzungsauftrag.
- Warten Sie, bis der asynchrone Auftrag abgeschlossen wurde.
- Beschreiben Sie den asynchronen Job.

SAP ABAP

SDK für SAP ABAP

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
```

```
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time. '
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language
 of the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8-
>describetexttranslationjob( iv_jobid = lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    ENDWHILE.

    TRY.
        oo_result = lo_xl8->describetexttranslationjob(      "oo_result is
 returned for testing purposes."
          iv_jobid        = lv_jobid ).
        MESSAGE 'Job description retrieved.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE
 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - DescribeTextTranslationJob
  - StartTextTranslationJob

Codebeispiele für VPC Lattice mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon VPC Lattice mit einem AWS Software Development Kit (SDK) verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- VPC Lattice Developer Guide — Weitere Informationen zu VPC Lattice.
- VPC Lattice API-Referenz — Details zu allen verfügbaren VPC Lattice-Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Szenarien für die Verwendung von VPC Lattice AWS SDKs
  - Ein Service Networking Mesh erstellen und verwalten

Szenarien für die Verwendung von VPC Lattice AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in VPC Lattice mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von VPC Lattice oder in Kombination mit anderen aufrufen. AWS-Services Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Ein Service Networking Mesh erstellen und verwalten

Ein Service Networking Mesh erstellen und verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ein Servicenetzwerk erstellen
- Einen Service erstellen
- Verfügbare VPCs auflisten
- Sicherheitsgruppen für die ausgewählte VPC auflisten
- Listet die Dienstzuordnungen auf
- VPC-Zuordnungen auflisten
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
```

```
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a
 "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
```

```
        echo "ERROR: $param_name contains invalid characters" | tee -a
 "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
    validate_input "$resource_id" "resource_id" || return 1
    validate_input "$desired_status" "desired_status" || return 1

    echo "Waiting for $resource_type $resource_id to be in state
 $desired_status..." | tee -a "$LOG_FILE"

    while [ "$attempt" -le "$max_attempts" ]; do
        echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

        # Run the command to get the status and capture the output
        status_output=$(eval "$command" 2>&1) || true
        echo "$status_output" >> "$LOG_FILE"

        # For service networks, they do not have a status field in the output
        # We'll consider them active if we can retrieve them
        if [[ "$resource_type" == "Service Network" ]]; then
            if [[ "$status_output" == *"$resource_id"* ]]; then
                echo "$resource_type $resource_id is now active" | tee -a
 "$LOG_FILE"
                return 0
            fi
        else
            # For other resources, extract the status field
            status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
 $4}' || true)
```

```
            echo "Current status: $status" >> "$LOG_FILE"

            if [[ "$status" == "$desired_status" ]]; then
                echo "$resource_type $resource_id is now in state
 $desired_status" | tee -a "$LOG_FILE"
                return 0
            elif [[ "$status" == *"FAIL"* ]]; then
                echo "ERROR: $resource_type $resource_id failed to reach desired
 state. Current status: $status" | tee -a "$LOG_FILE"
                return 1
            fi
        fi

        echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
 "$LOG_FILE"
        sleep 10
        ((attempt++))
    done

    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES
```

```
echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/
dev/null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-
lattice get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --
output json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name
 '$SERVICE_NAME' --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-
vpc-lattice-gs --output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
```

```
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service
 --service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"

SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
service-association --service-identifier '$SERVICE_ID' --service-network-
identifier '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a
 "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
```

```
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"

VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output
 text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')
```

```
    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?

        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a
 "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE"
 "aws vpc-lattice get-service-network-vpc-association --service-network-vpc-
association-identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
```

```
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee
 -a "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --
output json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --
service-network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
```

```
log_command "aws vpc-lattice delete-service-network-service-association --
service-network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'"
 || true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"

echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService

- GetServiceNetwork
- GetServiceNetworkServiceAssociation
- GetServiceNetworkVpcAssociation
- ListServiceNetworkServiceAssociations
- ListServiceNetworkVpcAssociations

Codebeispiele für AWS WAFV2 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS WAFV2 mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- AWS WAFV2 Entwicklerhandbuch — Weitere Informationen zu AWS WAFV2.
- AWS WAFV2 API-Referenz — Details zu allen verfügbaren AWS WAFV2 Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für AWS WAFV2 verwenden AWS SDKs
  - Aktionen für AWS WAFV2 verwenden AWS SDKs
    - Verwendung von CreateWebAcl mit einer CLI
- Szenarien für AWS WAFV2 verwenden AWS SDKs
  - Erste Schritte mit Firewalls für Webanwendungen

Grundlegende Beispiele für AWS WAFV2 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS WAFV2 mit AWS SDKs verwendet werden.

Beispiele

- Aktionen für AWS WAFV2 verwenden AWS SDKs
  - Verwendung von CreateWebAcl mit einer CLI

Aktionen für AWS WAFV2 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne AWS WAFV2 Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die AWS WAFV2 API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für AWS WAFV2 verwenden AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS WAFV2 -API-Referenz.

Beispiele

- Verwendung von CreateWebAcl mit einer CLI

Verwendung von **`CreateWebAcl`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateWebAcl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Erste Schritte mit Webanwendungs-Firewalls

CLI

AWS CLI

Um eine Web-ACL zu erstellen

Mit dem folgenden `create-web-acl`-Befehl wird eine Web-ACL für regionale Verwendung erstellt. Die Regelanweisungen für die Web-ACL werden in einer JSON-formatted Datei bereitgestellt.

```
aws wafv2 create-web-acl \
    --name TestWebAcl \
    --scope REGIONAL \
    --default-action Allow={} \
    --visibility-
config SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=TestWebAclMet
 \
    --rules file://waf-rule.json \
    --region us-west-2
```

Der Inhalt von file://waf-rule.json:

```
[
    {
        "Name":"basic-rule",
        "Priority":0,
        "Statement":{
            "AndStatement":{
                "Statements":[
                    {
                        "ByteMatchStatement":{
                            "SearchString":"example.com",
                            "FieldToMatch":{
                                "SingleHeader":{
                                    "Name":"host"
                                }
                            },
                            "TextTransformations":[
                                {
                                    "Priority":0,
                                    "Type":"LOWERCASE"
                                }
                            ],
                            "PositionalConstraint":"EXACTLY"
                        }
                    },
                    {
                        "GeoMatchStatement":{
                            "CountryCodes":[
```

```
                                "US",
                                "IN"
                            ]
                        }
                    }
                ]
            }
        },
        "Action":{
            "Allow":{

            }
        },
        "VisibilityConfig":{
            "SampledRequestsEnabled":true,
            "CloudWatchMetricsEnabled":true,
            "MetricName":"basic-rule"
        }
    }
]
```

Ausgabe:

```
{
    "Summary":{
        "ARN":"arn:aws:wafv2:us-west-2:123456789012:regional/webacl/TestWebAcl/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "Description":"",
        "Name":"TestWebAcl",
        "LockToken":"2294b3a1-eb60-4aa0-a86f-a3ae04329de9",
        "Id":"a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    }
}
```

Weitere Informationen finden Sie unter Verwaltung und Verwendung einer Web Access Control List (Web ACL) im AWS WAF, AWS Firewall Manager und AWS Shield Advanced Developer Guide.

- Einzelheiten zur API finden Sie CreateWebAclin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl erstellt eine neue Web-ACL mit dem Namen „waf-test“. Bitte beachten Sie, dass 'DefaultAction' gemäß der Dokumentation zur Service-API eine erforderliche Eigenschaft ist. Daher sollte ein Wert für entweder '- DefaultAction _Allow' oder and/or '- DefaultAction _Block 'angegeben werden. Da '- DefaultAction _Allow' und '- DefaultAction _Block 'nicht die erforderlichen Eigenschaften sind, könnte der Wert '@ {}' als Platzhalter verwendet werden, wie im obigen Beispiel gezeigt.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Ausgabe:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Einzelheiten zur API finden Sie unter CreateWebAclCmdlet-Referenz (V4).AWS -Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl erstellt eine neue Web-ACL mit dem Namen „waf-test“. Bitte beachten Sie, dass 'DefaultAction' gemäß der Dokumentation zur Service-API eine erforderliche Eigenschaft ist. Daher sollte ein Wert für entweder '- DefaultAction _Allow' oder and/or '- DefaultAction _Block 'angegeben werden. Da '- DefaultAction _Allow' und '- DefaultAction _Block 'nicht die erforderlichen Eigenschaften sind, könnte der Wert '@ {}' als Platzhalter verwendet werden, wie im obigen Beispiel gezeigt.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Ausgabe:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- Einzelheiten zur API finden Sie unter CreateWebAclCmdlet-Referenz (V5).AWS -Tools für PowerShell

Szenarien für AWS WAFV2 verwenden AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien AWS WAFV2 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb AWS WAFV2 oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Erste Schritte mit Firewalls für Webanwendungen

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleich von Zeichenketten hinzu
- Fügen Sie verwaltete Regeln hinzu
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "AWS WAF Getting Started Tutorial"
echo "==================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
```

```
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi
```

```
        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
```

```
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
```

```
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
```

```
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
```

```
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
```

```
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
```

```
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
```

```
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
```

```
    echo ""
    echo "======================================================"
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "======================================================"

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi
```

```
# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Codebeispiele für die WorkSpaces Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Amazon WorkSpaces mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Weitere -Quellen

- WorkSpaces Entwicklerhandbuch — Weitere Informationen zu WorkSpaces.
- WorkSpaces API-Referenz — Details zu allen verfügbaren WorkSpaces Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung WorkSpaces AWS SDKs
  - Aktionen zur Verwendung WorkSpaces AWS SDKs
    - Verwendung von CreateTags mit einer CLI
    - Verwendung von CreateWorkspaces mit einer CLI
    - Verwendung von DeleteTags mit einer CLI
    - Verwendung von DescribeTags mit einer CLI
    - Verwendung von DescribeWorkspaceBundles mit einer CLI
    - Verwendung von DescribeWorkspaceDirectories mit einer CLI
    - Verwendung von DescribeWorkspaces mit einer CLI
    - Verwendung von DescribeWorkspacesConnectionStatus mit einer CLI
    - Verwendung von ModifyWorkspaceCreationProperties mit einer CLI
    - Verwendung von ModifyWorkspaceProperties mit einer CLI
    - Verwendung von ModifyWorkspaceState mit einer CLI

    - Verwendung von RebootWorkspaces mit einer CLI
    - Verwendung von RebuildWorkspaces mit einer CLI
    - Verwendung von RegisterWorkspaceDirectory mit einer CLI
    - Verwendung von StopWorkspaces mit einer CLI
    - Verwendung von TerminateWorkspaces mit einer CLI
- Szenarien für die Verwendung WorkSpaces AWS SDKs
  - Persönliche virtuelle Desktops erstellen und verwalten

Grundlegende Beispiele für die Verwendung WorkSpaces AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon WorkSpaces mit AWS SDKs verwenden können.

Beispiele

- Aktionen zur Verwendung WorkSpaces AWS SDKs
  - Verwendung von CreateTags mit einer CLI
  - Verwendung von CreateWorkspaces mit einer CLI
  - Verwendung von DeleteTags mit einer CLI
  - Verwendung von DescribeTags mit einer CLI
  - Verwendung von DescribeWorkspaceBundles mit einer CLI
  - Verwendung von DescribeWorkspaceDirectories mit einer CLI
  - Verwendung von DescribeWorkspaces mit einer CLI
  - Verwendung von DescribeWorkspacesConnectionStatus mit einer CLI
  - Verwendung von ModifyWorkspaceCreationProperties mit einer CLI
  - Verwendung von ModifyWorkspaceProperties mit einer CLI
  - Verwendung von ModifyWorkspaceState mit einer CLI
  - Verwendung von RebootWorkspaces mit einer CLI
  - Verwendung von RebuildWorkspaces mit einer CLI
  - Verwendung von RegisterWorkspaceDirectory mit einer CLI
  - Verwendung von StopWorkspaces mit einer CLI
  - Verwendung von TerminateWorkspaces mit einer CLI

Aktionen zur Verwendung WorkSpaces AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne WorkSpaces Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die WorkSpaces API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter Szenarien für die Verwendung WorkSpaces AWS SDKs anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der Amazon WorkSpaces API-Referenz.

Beispiele

- Verwendung von CreateTags mit einer CLI
- Verwendung von CreateWorkspaces mit einer CLI
- Verwendung von DeleteTags mit einer CLI
- Verwendung von DescribeTags mit einer CLI
- Verwendung von DescribeWorkspaceBundles mit einer CLI
- Verwendung von DescribeWorkspaceDirectories mit einer CLI
- Verwendung von DescribeWorkspaces mit einer CLI
- Verwendung von DescribeWorkspacesConnectionStatus mit einer CLI
- Verwendung von ModifyWorkspaceCreationProperties mit einer CLI
- Verwendung von ModifyWorkspaceProperties mit einer CLI
- Verwendung von ModifyWorkspaceState mit einer CLI
- Verwendung von RebootWorkspaces mit einer CLI
- Verwendung von RebuildWorkspaces mit einer CLI
- Verwendung von RegisterWorkspaceDirectory mit einer CLI
- Verwendung von StopWorkspaces mit einer CLI
- Verwendung von TerminateWorkspaces mit einer CLI

Verwendung von **`CreateTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateTags` verwendet wird.

CLI

AWS CLI

Um Tags zu einem hinzuzufügen WorkSpace

Im folgenden `create-tags` Beispiel werden die angegebenen Tags zu den angegebenen hinzugefügt WorkSpace.

```
aws workspaces create-tags \
    --resource-id ws-dk1xzr417 \
    --tags Key=Department,Value=Finance
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter WorkSpaces Tag-Ressourcen im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie CreateTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** ein neues Tag hinzugefügt. Das Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** mehrere Tags hinzugefügt. Ein Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**; das andere Tag hat den Tag-Schlüssel „Stage“ und den Schlüsselwert „Test“.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
```

```
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Einzelheiten zur API finden Sie unter CreateTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** ein neues Tag hinzugefügt. Das Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** mehrere Tags hinzugefügt. Ein Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**; das andere Tag hat den Tag-Schlüssel „Stage“ und den Schlüsselwert „Test“.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- Einzelheiten zur API finden Sie unter CreateTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`CreateWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `CreateWorkspaces` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

Beispiel 1: Um ein zu erstellen AlwaysOn WorkSpace

Im folgenden `create-workspaces` Beispiel wird eine AlwaysOn WorkSpace für den angegebenen Benutzer erstellt, wobei das angegebene Verzeichnis und das angegebene Bundle verwendet werden.

```
aws workspaces create-workspaces \
    --workspaces DirectoryId=d-926722edaf,UserName=Mateo,BundleId=wsb-0zsvgp8fc
```

Ausgabe:

```
{
    "FailedRequests": [],
    "PendingRequests": [
        {
            "WorkspaceId": "ws-kcqms853t",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mateo",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc"
        }
    ]
}
```

Beispiel 2: Um ein zu erstellen AutoStop WorkSpace

Im folgenden `create-workspaces` Beispiel wird eine AutoStop WorkSpace für den angegebenen Benutzer erstellt, wobei das angegebene Verzeichnis und das angegebene Bundle verwendet werden.

```
aws workspaces create-workspaces \
    --
workspaces DirectoryId=d-926722edaf,UserName=Mary,BundleId=wsb-0zsvgp8fc,WorkspaceProperti
```

Ausgabe:

```
{
```

```
    "FailedRequests": [],
    "PendingRequests": [
        {
            "WorkspaceId": "ws-dk1xzr417",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mary",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc"
        }
    ]
}
```

Beispiel 3: Um ein vom Benutzer entkoppeltes Objekt zu erstellen WorkSpace

Im folgenden `create-workspaces` Beispiel wird eine benutzerentkoppelte Datei erstellt, WorkSpace indem der Benutzername auf gesetzt und ein WorkSpace Name`[UNDEFINED]`, eine Verzeichnis-ID und eine Bundle-ID angegeben werden.

```
aws workspaces create-workspaces \
    --workspaces
 DirectoryId=d-926722edaf,UserName='"[UNDEFINED]"',WorkspaceName=MaryWorkspace1,BundleId=
```

Ausgabe:

```
{
    "FailedRequests": [],
    "PendingRequests": [
        {
            "WorkspaceId": "ws-abcd1234",
            "DirectoryId": "d-926722edaf",
            "UserName": "[UNDEFINED]",
            "State": "PENDING",
            "BundleId": "wsb-0zsvgp8fc",
            "WorkspaceName": "MaryWorkspace1"
        }
    ]
}
```

Weitere Informationen finden Sie unter Launch a virtual desktop im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie CreateWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Erstellen Sie eine WorkSpace für das bereitgestellte Bundle, Verzeichnis und Benutzer.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Beispiel 2: In diesem Beispiel werden mehrere erstellt WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Einzelheiten zur API finden Sie unter CreateWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Erstellen Sie eine WorkSpace für das bereitgestellte Bundle, Verzeichnis und Benutzer.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Beispiel 2: In diesem Beispiel werden mehrere erstellt WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- Einzelheiten zur API finden Sie unter CreateWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DeleteTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteTags` verwendet wird.

CLI

AWS CLI

Um ein Tag aus einem zu löschen WorkSpace

Im folgenden `delete-tags` Beispiel wird das angegebene Tag aus dem angegebenen WorkSpace Tag gelöscht.

```
aws workspaces delete-tags \
    --resource-id ws-dk1xzr417 \
    --tag-keys Department
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter WorkSpaces Tag-Ressourcen im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie DeleteTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das dem Workspace zugeordnete Tag entfernt

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das dem Workspace zugeordnete Tag entfernt

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Einzelheiten zur API finden Sie unter DeleteTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeTags`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeTags` verwendet wird.

CLI

AWS CLI

Um die Tags für ein zu beschreiben WorkSpace

Das folgende `describe-tags` Beispiel beschreibt die Tags für die angegebenen WorkSpace.

```
aws workspaces describe-tags \
    --resource-id ws-dk1xzr417
```

Ausgabe:

```
{
    "TagList": [
        {
            "Key": "Department",
            "Value": "Finance"
        }
    ]
}
```

Weitere Informationen finden Sie unter WorkSpaces Tag-Ressourcen im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie DescribeTagsin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das Tag für den angegebenen Workspace abgerufen

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Ausgabe:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das Tag für den angegebenen Workspace abgerufen

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Ausgabe:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- Einzelheiten zur API finden Sie unter DescribeTags AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeWorkspaceBundles`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeWorkspaceBundles` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

So listen Sie die von Amazon bereitgestellten Bundles auf

Im folgenden Beispiel für `describe-workspace-bundles` werden die Namen und IDs der von Amazon bereitgestellten Bundles im Tabellenformat aufgelistet und nach Namen sortiert.

```
aws workspaces describe-workspace-bundles \
    --owner AMAZON \
    --query "Bundles[*].[Name, BundleId]"
```

Ausgabe:

```
[
    [
        "Standard with Amazon Linux 2",
        "wsb-clj85qzj1"
    ],
    [
        "Performance with Windows 10 (Server 2016 based)",
        "wsb-gm4d5tx2v"
    ],
    [
        "PowerPro with Windows 7",
        "wsb-1pzkp0bx4"
    ],
    [
        "Power with Amazon Linux 2",
        "wsb-2bs6k5lgn"
    ],
    [
        "Graphics with Windows 10 (Server 2019 based)",
        "wsb-03gyjnfyy"
    ],
    ...
```

```
]
```

Weitere Informationen finden Sie unter WorkSpaces Bundles und Images im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie DescribeWorkspaceBundlesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden Details zu allen Workspace-Bundles in der aktuellen Region abgerufen

```
Get-WKSWorkspaceBundle
```

Ausgabe:

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceBundles AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden Details zu allen Workspace-Bundles in der aktuellen Region abgerufen

```
Get-WKSWorkspaceBundle
```

Ausgabe:

```
BundleId        : wsb-sfhdgv342
```

```
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceBundles AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeWorkspaceDirectories`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeWorkspaceDirectories` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

So beschreiben Sie ein registriertes Verzeichnis

Im folgenden Beispiel für `describe-workspace-directories` wird das angegebene registrierte Verzeichnis beschrieben.

```
aws workspaces describe-workspace-directories \
    --directory-ids d-926722edaf
```

Ausgabe:

```
{
    "Directories": [
        {
            "DirectoryId": "d-926722edaf",
            "Alias": "d-926722edaf",
            "DirectoryName": "example.com",
```

```
            "RegistrationCode": "WSpdx+9RJ8JT",
            "SubnetIds": [
                "subnet-9d19c4c6",
                "subnet-500d5819"
            ],
            "DnsIpAddresses": [
                "172.16.1.140",
                "172.16.0.30"
            ],
            "CustomerUserName": "Administrator",
            "IamRoleId": "arn:aws:iam::123456789012:role/workspaces_DefaultRole",
            "DirectoryType": "SIMPLE_AD",
            "WorkspaceSecurityGroupId": "sg-0d89e927e5645d7c5",
            "State": "REGISTERED",
            "WorkspaceCreationProperties": {
                "EnableInternetAccess": false,
                "UserEnabledAsLocalAdministrator": true,
                "EnableMaintenanceMode": true
            },
            "WorkspaceAccessProperties": {
                "DeviceTypeWindows": "ALLOW",
                "DeviceTypeOsx": "ALLOW",
                "DeviceTypeWeb": "DENY",
                "DeviceTypeIos": "ALLOW",
                "DeviceTypeAndroid": "ALLOW",
                "DeviceTypeChromeOs": "ALLOW",
                "DeviceTypeZeroClient": "ALLOW",
                "DeviceTypeLinux": "DENY"
            },
            "Tenancy": "SHARED",
            "SelfservicePermissions": {
                "RestartWorkspace": "ENABLED",
                "IncreaseVolumeSize": "DISABLED",
                "ChangeComputeType": "DISABLED",
                "SwitchRunningMode": "DISABLED",
                "RebuildWorkspace": "DISABLED"
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Verzeichnisse für WorkSpaces Personal verwalten im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie DescribeWorkspaceDirectoriesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Verzeichnisdetails für registrierte Verzeichnisse aufgeführt

```
Get-WKSWorkspaceDirectory
```

Ausgabe:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/
workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceDirectories AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Verzeichnisdetails für registrierte Verzeichnisse aufgeführt

```
Get-WKSWorkspaceDirectory
```

Ausgabe:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/
workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaceDirectories AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeWorkspaces` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

Um ein zu beschreiben WorkSpace

Das folgende `describe-workspaces` Beispiel beschreibt den angegebenen WorkSpace.

```
aws workspaces describe-workspaces \
    --workspace-ids ws-dk1xzr417
```

Ausgabe:

```
{
    "Workspaces": [
        {
            "WorkspaceId": "ws-dk1xzr417",
            "DirectoryId": "d-926722edaf",
            "UserName": "Mary",
            "IpAddress": "172.16.0.175",
            "State": "STOPPED",
            "BundleId": "wsb-0zsvgp8fc",
            "SubnetId": "subnet-500d5819",
            "ComputerName": "WSAMZN-RBSLTTD9",
            "WorkspaceProperties": {
                "RunningMode": "AUTO_STOP",
                "RunningModeAutoStopTimeoutInMinutes": 60,
                "RootVolumeSizeGib": 80,
                "UserVolumeSizeGib": 10,
                "ComputeTypeName": "VALUE"
            },
            "ModificationStates": []
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten Ihres WorkSpaces im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie DescribeWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Ruft Details aller Ihrer Verbindungen WorkSpaces zur Pipeline ab.

```
Get-WKSWorkspace
```

Ausgabe:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Beispiel 2: Dieser Befehl zeigt die Werte der untergeordneten Eigenschaften von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Weitere Informationen zu den untergeordneten Eigenschaften von finden Sie **`WorkSpaceProperties`** unter https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties
```

Ausgabe:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Beispiel 3: Dieser Befehl zeigt den Wert der untergeordneten Eigenschaft **`RootVolumeSizeGib`** von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Die Größe des Root-Volumes in GiB beträgt 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Ausgabe:

```
        80
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Ruft Details aller Ihrer Verbindungen WorkSpaces zur Pipeline ab.

```
Get-WKSWorkspace
```

Ausgabe:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Beispiel 2: Dieser Befehl zeigt die Werte der untergeordneten Eigenschaften von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Weitere Informationen zu den untergeordneten Eigenschaften von finden Sie **`WorkSpaceProperties`** unter https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties
```

Ausgabe:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
```

```
        RunningMode                          : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                    : 50
```

Beispiel 3: Dieser Befehl zeigt den Wert der untergeordneten Eigenschaft **`RootVolumeSizeGib`** von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Die Größe des Root-Volumes in GiB beträgt 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Ausgabe:

```
        80
```

- Einzelheiten zur API finden Sie unter DescribeWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`DescribeWorkspacesConnectionStatus`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeWorkspacesConnectionStatus` verwendet wird.

CLI

AWS CLI

Um den Verbindungsstatus eines zu beschreiben WorkSpace

Das folgende `describe-workspaces-connection-status` Beispiel beschreibt den Verbindungsstatus des angegebenen WorkSpace.

```
aws workspaces describe-workspaces-connection-status \
    --workspace-ids ws-dk1xzr417
```

Ausgabe:

```
{
    "WorkspacesConnectionStatus": [
```

```
        {
            "WorkspaceId": "ws-dk1xzr417",
            "ConnectionState": "CONNECTED",
            "ConnectionStateCheckTimestamp": 1662526214.744
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwalten Ihres WorkSpaces im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie DescribeWorkspacesConnectionStatusin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Verbindungsstatus für den angegebenen Workspace abgerufen

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Einzelheiten zur API finden Sie unter DescribeWorkspacesConnectionStatus AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Verbindungsstatus für den angegebenen Workspace abgerufen

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- Einzelheiten zur API finden Sie unter DescribeWorkspacesConnectionStatus AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyWorkspaceCreationProperties`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyWorkspaceCreationProperties` verwendet wird.

CLI

AWS CLI

Um die WorkSpace Erstellungseigenschaft eines Verzeichnisses zu ändern

Im folgenden Beispiel für `modify-workspace-creation-properties` wird die `EnableInternetAccess`-Eigenschaft für das angegebene Verzeichnis aktiviert. Dies ermöglicht die automatische Zuweisung von öffentlichen IP-Adressen für das Verzeichnis, das für das Verzeichnis WorkSpaces erstellt wurde.

```
aws workspaces modify-workspace-creation-properties \
    --resource-id d-926722edaf \
    --workspace-creation-properties EnableInternetAccess=true
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Verzeichnisdetails für Sie aktualisieren WorkSpaces im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie ModifyWorkspaceCreationPropertiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden beim Erstellen eines Workspace die Standardwerte für Internetzugriff und Wartungsmodus auf „true“ gesetzt

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceCreationProperties AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden beim Erstellen eines Workspace die Standardwerte für Internetzugriff und Wartungsmodus auf „true“ gesetzt

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceCreationProperties AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyWorkspaceProperties`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyWorkspaceProperties` verwendet wird.

CLI

AWS CLI

Um den Betriebsmodus eines zu ändern WorkSpace

Im folgenden `modify-workspace-properties` Beispiel wird der angegebene Laufmodus WorkSpace auf gesetzt`AUTO_STOP`.

```
aws workspaces modify-workspace-properties \
    --workspace-id ws-dk1xzr417 \
    --workspace-properties RunningMode=AUTO_STOP
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Modify a WorkSpace im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie ModifyWorkspacePropertiesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Eigenschaft Workspace Running Mode für den angegebenen Workspace auf Auto Stop geändert

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceProperties AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Eigenschaft Workspace Running Mode für den angegebenen Workspace auf Auto Stop geändert

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceProperties AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`ModifyWorkspaceState`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `ModifyWorkspaceState` verwendet wird.

CLI

AWS CLI

Um den Status eines zu ändern WorkSpace

Im folgenden `modify-workspace-state` Beispiel wird der angegebene Status WorkSpace auf gesetzt`ADMIN_MAINTENANCE`.

```
aws workspaces modify-workspace-state \
    --workspace-id ws-dk1xzr417 \
    --workspace-state ADMIN_MAINTENANCE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter WorkSpace Wartung im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie ModifyWorkspaceStatein der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Workspace auf „Verfügbar“ geändert

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceState AVAILABLE
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceState AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Workspace auf „Verfügbar“ geändert

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceState AVAILABLE
```

- Einzelheiten zur API finden Sie unter ModifyWorkspaceState AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RebootWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RebootWorkspaces` verwendet wird.

CLI

AWS CLI

Um einen neu zu starten WorkSpace

Im folgenden `reboot-workspaces` Beispiel wird der angegebene WorkSpace neu gestartet.

```
aws workspaces reboot-workspaces \
    --reboot-workspace-requests ws-dk1xzr417
```

Ausgabe:

```
{
    "FailedRequests": []
}
```

Weitere Informationen finden Sie unter Reboot a WorkSpace im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie RebootWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Startet den angegebenen WorkSpace neu.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Startet mehrere neu. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Beispiel 3: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId -Parameter von weiter Restart-WKSWorkspace, wodurch der neu gestartet WorkSpaces wird.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter RebootWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Startet den angegebenen neu. WorkSpace

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Startet mehrere neu. WorkSpaces

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Beispiel 3: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId -Parameter von weiter Restart-WKSWorkspace, wodurch der neu gestartet WorkSpaces wird.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter RebootWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RebuildWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RebuildWorkspaces` verwendet wird.

CLI

AWS CLI

Um ein neu aufzubauen WorkSpace

Im folgenden `rebuild-workspaces` Beispiel wird das angegebene WorkSpace neu erstellt.

```
aws workspaces rebuild-workspaces \
    --rebuild-workspace-requests ws-dk1xzr417
```

Ausgabe:

```
{
    "FailedRequests": []
}
```

Weitere Informationen finden Sie unter Rebuild a WorkSpace im Amazon WorkSpaces Administration Guide.

- Einzelheiten zur API finden Sie RebuildWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Baut die angegebene WorkSpace Datei neu auf.

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId Parameter - von weiter Reset-WKSWorkspace, wodurch der neu erstellt WorkSpaces wird.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter RebuildWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Baut die angegebene Datei neu auf. WorkSpace

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId Parameter - von weiter Reset-WKSWorkspace, wodurch der neu erstellt WorkSpaces wird.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- Einzelheiten zur API finden Sie unter RebuildWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`RegisterWorkspaceDirectory`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `RegisterWorkspaceDirectory` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

Um ein Verzeichnis zu registrieren

Im folgenden `register-workspace-directory` Beispiel wird das angegebene Verzeichnis für die Verwendung mit Amazon registriert WorkSpaces.

```
aws workspaces register-workspace-directory \
    --directory-id d-926722edaf
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Registrieren eines vorhandenen AWS Directory Service Service-Verzeichnisses bei WorkSpaces Personal im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie RegisterWorkspaceDirectoryin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird das angegebene Verzeichnis für Workspaces Service registriert

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Einzelheiten zur API finden Sie unter RegisterWorkspaceDirectory AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Verzeichnis für Workspaces Service registriert

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- Einzelheiten zur API finden Sie unter RegisterWorkspaceDirectory AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`StopWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `StopWorkspaces` verwendet wird.

CLI

AWS CLI

Um ein zu stoppen AutoStop WorkSpace

Im folgenden `stop-workspaces` Beispiel wird der angegebene Vorgang beendet WorkSpace. Der WorkSpace muss den Betriebsmodus haben`AutoStop`.

```
aws workspaces stop-workspaces \
    --stop-workspace-requests WorkspaceId=ws-dk1xzr417
```

Ausgabe:

```
{
    "FailedRequests": []
}
```

Weitere Informationen finden Sie unter Stopp und Start AutoStop WorkSpace im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie StopWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Stoppt mehrere WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - weiter, Stop-WKSWorkspace wodurch der WorkSpaces gestoppt wird.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie StopRequest Objekte übergeben werden, die definieren, dass gestoppt werden WorkSpaces sollen.

```
$arrRequest = @()
```

```
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Einzelheiten zur API finden Sie unter StopWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Stoppt mehrere WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - weiter, Stop-WKSWorkspace wodurch der WorkSpaces gestoppt wird.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie StopRequest Objekte übergeben werden, die definieren, dass gestoppt werden WorkSpaces sollen.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- Einzelheiten zur API finden Sie unter StopWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Verwendung von **`TerminateWorkspaces`** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `TerminateWorkspaces` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- Persönliche virtuelle Desktops erstellen und verwalten

CLI

AWS CLI

Um ein zu beenden WorkSpace

Im folgenden Beispiel für `terminate-workspaces` wird der angegebene WorkSpace beendet.

```
aws workspaces terminate-workspaces \
    --terminate-workspace-requests ws-dk1xzr417
```

Ausgabe:

```
{
    "FailedRequests": []
}
```

Weitere Informationen finden Sie unter Delete a WorkSpace im WorkSpaces Amazon-Administratorhandbuch.

- Einzelheiten zur API finden Sie TerminateWorkspacesin der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Beendet mehrere WorkSpaces. Die Verwendung der Option -Force verhindert, dass das Cmdlet zur Bestätigung auffordert.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Beispiel 2: Ruft die Auflistung aller Ihrer IDs ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId -Parameter von, wodurch alle von Remove-WKSWorkspace beendet werden. WorkSpaces Das Cmdlet gibt eine Eingabeaufforderung aus, bevor jedes

Cmdlet beendet wird. WorkSpace Um die Bestätigungsaufforderung zu unterdrücken, fügen Sie den Schalter -Force hinzu.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie TerminateRequest Objekte übergeben werden, die definieren, dass sie beendet werden WorkSpaces sollen. Das Cmdlet fordert Sie zur Bestätigung auf, bevor es fortfährt, es sei denn, der Schalterparameter -Force ist ebenfalls angegeben.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Einzelheiten zur API finden Sie unter TerminateWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Beendet mehrere WorkSpaces. Die Verwendung der Option -Force verhindert, dass das Cmdlet zur Bestätigung auffordert.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Beispiel 2: Ruft die Auflistung aller Ihrer IDs ab WorkSpaces und leitet die IDs über die Pipeline an den WorkSpaceId -Parameter von, wodurch alle von Remove-WKSWorkspace beendet werden. WorkSpaces Das Cmdlet gibt eine Eingabeaufforderung aus, bevor jedes Cmdlet beendet wird. WorkSpace Um die Bestätigungsaufforderung zu unterdrücken, fügen Sie den Schalter -Force hinzu.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie TerminateRequest Objekte übergeben werden, die definieren, dass sie beendet werden WorkSpaces sollen. Das Cmdlet fordert Sie zur

Bestätigung auf, bevor es fortfährt, es sei denn, der Schalterparameter -Force ist ebenfalls angegeben.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- Einzelheiten zur API finden Sie unter TerminateWorkspaces AWS -Tools für PowerShellCmdlet-Referenz (V5).

Szenarien für die Verwendung WorkSpaces AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien WorkSpaces mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb WorkSpaces oder in Kombination mit anderen AWS-Services aufrufen. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- Persönliche virtuelle Desktops erstellen und verwalten

Persönliche virtuelle Desktops erstellen und verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie WorkSpaces die Verfügbarkeit in Ihrer Region
- Verfügbare WorkSpaces Pakete auflisten
- Erstellen Sie ein WorkSpace
- Überprüfen Sie den Status Ihres WorkSpace
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a WorkSpace in WorkSpaces Personal
# This script follows the workflow described in the AWS documentation
# https://docs.aws.amazon.com/workspaces/latest/adminguide/create-workspaces-
personal.html

# Set up logging
LOG_FILE="workspaces_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting WorkSpaces creation script"
echo "==============================================="

# Initialize resource tracking array
declare -a CREATED_RESOURCES

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    exit 1
}

# Function to check if a command succeeded
check_command() {
    # Check for ResourceNotFound.User error specifically
    if echo "$1" | grep -q "ResourceNotFound.User"; then
```

```
        echo ""
        echo "ERROR: User not found in the directory."
        echo ""
        echo "This error occurs when the specified username doesn't exist in the
 directory."
        echo ""
        echo "To resolve this issue:"
        echo "1. Ensure the user exists in the directory before creating a
 WorkSpace."
        echo "2. For Simple AD and AWS Managed Microsoft AD:"
        echo "   - Connect to a directory-joined instance"
        echo "   - Use Active Directory tools to create the user"
        echo "   - See: https://docs.aws.amazon.com/workspaces/latest/adminguide/
manage-users.html"
        echo ""
        echo "3. For AD Connector:"
        echo "   - Create the user in your on-premises Active Directory"
        echo "   - Ensure proper synchronization with the AD Connector"
        echo ""
        echo "4. Alternatively, you can use the AWS Console to create a
 WorkSpace,"
        echo "   which can create the user automatically in some directory
 types."
        echo ""
        handle_error "User '$USERNAME' not found in directory '$DIRECTORY_ID'"
    # Check for other errors
    elif echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Step 0: Select AWS region
echo ""
echo "==============================================="
echo "AWS REGION SELECTION"
echo "==============================================="
echo "Enter the AWS region to use (e.g., us-east-1, us-west-2):"
AWS_REGION="${AWS_REGION:-us-east-1}"

if [ -z "$AWS_REGION" ]; then
    handle_error "Region cannot be empty"
fi

export AWS_DEFAULT_REGION="$AWS_REGION"
```

```
echo "Using AWS region: $AWS_REGION"

# Step 1: Prompt for directory ID
echo ""
echo "==============================================="
echo "DIRECTORY SELECTION"
echo "==============================================="
echo "Listing available directories..."

DIRECTORIES_OUTPUT=$(aws workspaces describe-workspace-directories --output json)
check_command "$DIRECTORIES_OUTPUT"
echo "$DIRECTORIES_OUTPUT"

# Extract directory IDs and display them
DIRECTORY_IDS=$(echo "$DIRECTORIES_OUTPUT" | grep -o '"DirectoryId": "[^"]*' |
 cut -d'"' -f4)

if [ -z "$DIRECTORY_IDS" ]; then
    echo "No directories found. Please create a directory first using AWS
 Directory Service."
    echo "For more information, see: https://docs.aws.amazon.com/workspaces/
latest/adminguide/register-deregister-directory.html"
    exit 1
fi

echo ""
echo "Available directory IDs:"
echo "$DIRECTORY_IDS"
echo ""
echo "Enter the directory ID you want to use:"
DIRECTORY_ID=$(echo "$DIRECTORY_IDS" | head -1)

# Validate directory ID
if ! echo "$DIRECTORY_IDS" | grep -q "$DIRECTORY_ID"; then
    echo "Directory ID $DIRECTORY_ID not found in the list of available
 directories."
    echo "Please check the ID and try again."
    exit 1
fi

echo "Selected directory ID: $DIRECTORY_ID"

# Step 2: Check if directory is registered with WorkSpaces
echo ""
```

```
echo "==============================================="
echo "CHECKING DIRECTORY REGISTRATION"
echo "==============================================="

REGISTERED=$(echo "$DIRECTORIES_OUTPUT" | grep -A 5 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -c "\"State\": \"REGISTERED\"")

if [ "$REGISTERED" -eq 0 ]; then
    echo "Directory $DIRECTORY_ID is not registered with WorkSpaces. Registering
 now..."
    REGISTER_OUTPUT=$(aws workspaces register-workspace-directory --directory-id
 "$DIRECTORY_ID")
    check_command "$REGISTER_OUTPUT"
    echo "Directory registration initiated. This may take a few minutes."

    # Add to resource tracking
    CREATED_RESOURCES+=("Directory registration: $DIRECTORY_ID")

    # Wait for directory to be registered
    echo "Waiting for directory registration to complete..."
    sleep 30

    # Check registration status
    REGISTRATION_CHECK=$(aws workspaces describe-workspace-directories --
directory-ids "$DIRECTORY_ID")
    check_command "$REGISTRATION_CHECK"

    REGISTRATION_STATE=$(echo "$REGISTRATION_CHECK" | grep -o '"State": "[^"]*' |
 cut -d'"' -f4)
    if [ "$REGISTRATION_STATE" != "REGISTERED" ]; then
        echo "Directory registration is still in progress. Current state:
 $REGISTRATION_STATE"
        echo "Please check the AWS console for the final status."
        echo "You may need to wait a few minutes before proceeding."
    else
        echo "Directory successfully registered with WorkSpaces."
    fi
else
    echo "Directory $DIRECTORY_ID is already registered with WorkSpaces."
fi

# Get directory type to provide appropriate user guidance
DIRECTORY_TYPE=$(echo "$DIRECTORIES_OUTPUT" | grep -A 10 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -o '"DirectoryType": "[^"]*' | cut -d'"' -f4)
```

```
echo "Directory type: $DIRECTORY_TYPE"

# Display user creation guidance based on directory type
echo ""
echo "==============================================="
echo "USER CREATION GUIDANCE"
echo "==============================================="
case "$DIRECTORY_TYPE" in
    "SimpleAD" | "MicrosoftAD")
        echo "For $DIRECTORY_TYPE, users must be created using Active Directory
 tools."
        echo "1. Connect to a directory-joined EC2 instance"
        echo "2. Use Active Directory Users and Computers to create users"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
    "ADConnector")
        echo "For AD Connector, users must exist in your on-premises Active
 Directory."
        echo "1. Create the user in your on-premises Active Directory"
        echo "2. Ensure the user is in an OU that is within the scope of your AD
 Connector"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
directoryservice/latest/admin-guide/ad_connector_management.html"
        ;;
    *)
        echo "For this directory type, ensure users exist before creating
 WorkSpaces."
        echo "For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
esac
echo ""

# Step 3: List available bundles
echo ""
echo "==============================================="
echo "BUNDLE SELECTION"
echo "==============================================="
echo "Listing available WorkSpace bundles..."

# Get bundles with a format that's easier to parse
```

```
BUNDLES_OUTPUT=$(aws workspaces describe-workspace-
bundles --owner AMAZON --output text --query "Bundles[*].
[BundleId,Name,ComputeType.Name,RootStorage.Capacity,UserStorage.Capacity]")
check_command "$BUNDLES_OUTPUT"

# Extract bundle information and display in a numbered list
echo "Available bundles:"
echo "-----------------"
echo "NUM | BUNDLE ID | NAME | COMPUTE TYPE | ROOT STORAGE | USER STORAGE"
echo "-------------------------------------------------------------------"

# Create arrays to store bundle information
declare -a BUNDLE_IDS
declare -a BUNDLE_NAMES

# Process the output to extract bundle information
COUNT=1
while IFS=$'\t' read -r BUNDLE_ID BUNDLE_NAME COMPUTE_TYPE ROOT_STORAGE
 USER_STORAGE || [[ -n "$BUNDLE_ID" ]]; do
    # Store in arrays
    BUNDLE_IDS[$COUNT]="$BUNDLE_ID"
    BUNDLE_NAMES[$COUNT]="$BUNDLE_NAME"

    # Display with number
    echo "$COUNT | $BUNDLE_ID | $BUNDLE_NAME | $COMPUTE_TYPE | $ROOT_STORAGE GB |
 $USER_STORAGE GB"

    ((COUNT++))
done <<< "$BUNDLES_OUTPUT"

# Prompt for selection
echo ""
echo "Enter the number of the bundle you want to use (1-$((COUNT-1))):"
BUNDLE_SELECTION="1"

# Validate selection
if ! [[ "$BUNDLE_SELECTION" =~ ^[0-9]+$ ]] || [ "$BUNDLE_SELECTION" -lt 1 ] ||
 [ "$BUNDLE_SELECTION" -ge "$COUNT" ]; then
    handle_error "Invalid bundle selection. Please enter a number between 1 and
 $((COUNT-1))."
fi

# Get the selected bundle ID
BUNDLE_ID="${BUNDLE_IDS[$BUNDLE_SELECTION]}"
```

```
BUNDLE_NAME="${BUNDLE_NAMES[$BUNDLE_SELECTION]}"

echo "Selected bundle: $BUNDLE_NAME (ID: $BUNDLE_ID)"

# Step 4: Prompt for username
echo ""
echo "================================================"
echo "USER INFORMATION"
echo "================================================"
echo "Enter the username for the WorkSpace:"
USERNAME="testuser-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n 1)"

echo "NOTE: The user must already exist in the directory for the WorkSpace
 creation to succeed."
echo "If you're using Simple AD or AWS Managed Microsoft AD, the user must be
 created using Active Directory tools."
echo "If you're using AD Connector, the user must exist in your on-premises
 Active Directory."
echo ""

echo "Enter the user's first name:"
FIRST_NAME="Test"

echo "Enter the user's last name:"
LAST_NAME="User"

echo "Enter the user's email address:"
EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"

# Step 5: Choose running mode
echo ""
echo "================================================"
echo "RUNNING MODE SELECTION"
echo "================================================"
echo "Select running mode:"
echo "1. AlwaysOn (billed monthly)"
echo "2. AutoStop (billed hourly)"
RUNNING_MODE_CHOICE="1"

if [ "$RUNNING_MODE_CHOICE" = "1" ]; then
    RUNNING_MODE="ALWAYS_ON"
    AUTO_STOP_TIMEOUT=""
else
```

```
    RUNNING_MODE="AUTO_STOP"
    AUTO_STOP_TIMEOUT=60
fi

echo "Selected running mode: $RUNNING_MODE"

# Step 6: Add tags (optional)
echo ""
echo "================================================"
echo "TAGS (OPTIONAL)"
echo "================================================"
echo "Would you like to add tags to your WorkSpace? (y/n):"
ADD_TAGS="n"

TAGS_JSON=""
if [ "$ADD_TAGS" = "y" ] || [ "$ADD_TAGS" = "Y" ]; then
    echo "Enter tag key (e.g., Department):"
    read -r TAG_KEY

    echo "Enter tag value (e.g., IT):"
    read -r TAG_VALUE

    TAGS_JSON="[{\"Key\":\"$TAG_KEY\",\"Value\":\"$TAG_VALUE\"}]"
fi

# Step 7: Create the WorkSpace
echo ""
echo "================================================"
echo "CREATING WORKSPACE"
echo "================================================"
echo "Creating WorkSpace with the following parameters:"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Bundle ID: $BUNDLE_ID"
echo "Running Mode: $RUNNING_MODE"
if [ -n "$TAGS_JSON" ]; then
    echo "Tags: $TAG_KEY=$TAG_VALUE"
fi

# Create JSON for workspace properties
if [ "$RUNNING_MODE" = "AUTO_STOP" ]; then
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\",
\"RunningModeAutoStopTimeoutInMinutes\":$AUTO_STOP_TIMEOUT}"
else
```

```
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\"}"
fi

# Create JSON for workspaces parameter
WORKSPACE_JSON="{\"DirectoryId\":\"$DIRECTORY_ID\",\"UserName\":\"$USERNAME\",
\"BundleId\":\"$BUNDLE_ID\",\"WorkspaceProperties\":$PROPERTIES_JSON"

# Add tags if specified
if [ -n "$TAGS_JSON" ]; then
    USER_TAGS=$(echo "$TAGS_JSON" | sed 's/^\[//;s/\]$//')
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[$USER_TAGS,{\"Key\":\"project\",
\"Value\":\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal
\"}]"
else
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[{\"Key\":\"project\",\"Value\":
\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
fi

# Close the JSON object
WORKSPACE_JSON="$WORKSPACE_JSON}"

# Construct the create-workspaces command
CREATE_COMMAND="aws workspaces create-workspaces --workspaces '$WORKSPACE_JSON'"

echo "Executing: $CREATE_COMMAND"
CREATE_OUTPUT=$(eval "$CREATE_COMMAND")
check_command "$CREATE_OUTPUT"
echo "$CREATE_OUTPUT"

# Extract WorkSpace ID
WORKSPACE_ID=$(echo "$CREATE_OUTPUT" | grep -o '"WorkspaceId": "[^"]*' | head -1
 | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    handle_error "Failed to extract WorkSpace ID from creation output."
fi

echo "WorkSpace creation initiated. WorkSpace ID: $WORKSPACE_ID"
CREATED_RESOURCES+=("WorkSpace: $WORKSPACE_ID")

# Step 8: Check WorkSpace status
echo ""
echo "==============================================="
echo "CHECKING WORKSPACE STATUS"
```

```
echo "==============================================="
echo "Checking status of WorkSpace $WORKSPACE_ID..."

# Initial status check
STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
check_command "$STATUS_OUTPUT"
echo "$STATUS_OUTPUT"

WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 |
 cut -d'"' -f4)
echo "Current WorkSpace state: $WORKSPACE_STATE"

# Wait for WorkSpace to be available (this can take 20+ minutes)
echo ""
echo "WorkSpace creation is in progress. This can take 20+ minutes."
echo "The script will check the status every 60 seconds."
echo "Press Ctrl+C to exit the script at any time. The WorkSpace will continue to
 be created."

while [ "$WORKSPACE_STATE" = "PENDING" ]; do
    echo "$(date): WorkSpace state is still PENDING. Waiting 60 seconds before
 checking again..."
    sleep 60

    STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
    check_command "$STATUS_OUTPUT"

    WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1
 | cut -d'"' -f4)
    echo "$(date): Current WorkSpace state: $WORKSPACE_STATE"

    # If state is ERROR or UNHEALTHY, exit
    if [ "$WORKSPACE_STATE" = "ERROR" ] || [ "$WORKSPACE_STATE" = "UNHEALTHY" ];
 then
        handle_error "WorkSpace creation failed. Final state: $WORKSPACE_STATE"
    fi

    # If state is AVAILABLE, break the loop
    if [ "$WORKSPACE_STATE" = "AVAILABLE" ]; then
        break
    fi
done
```

```
# Step 9: Display WorkSpace information
echo ""
echo "================================================"
echo "WORKSPACE CREATION COMPLETE"
echo "================================================"
echo "WorkSpace has been successfully created!"
echo "WorkSpace ID: $WORKSPACE_ID"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Running Mode: $RUNNING_MODE"

# Step 10: Remind about invitation emails
echo ""
echo "================================================"
echo "INVITATION EMAILS"
echo "================================================"
echo "IMPORTANT: If you're using AD Connector or a trust relationship, or if the
 user already exists in Active Directory,"
echo "invitation emails are not sent automatically. You'll need to manually send
 an invitation email."
echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/manage-workspaces-users.html#send-invitation"

# Step 11: Cleanup confirmation
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

echo ""
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"

if [ "$CLEANUP_CHOICE" = "y" ] || [ "$CLEANUP_CHOICE" = "Y" ]; then
    echo ""
    echo "================================================"
    echo "CLEANING UP RESOURCES"
    echo "================================================"
```

```
    # Terminate WorkSpace
    if [ -n "$WORKSPACE_ID" ]; then
        echo "Terminating WorkSpace $WORKSPACE_ID..."
        TERMINATE_OUTPUT=$(aws workspaces terminate-workspaces --terminate-
workspace-requests WorkspaceId="$WORKSPACE_ID")
        check_command "$TERMINATE_OUTPUT"
        echo "$TERMINATE_OUTPUT"
        echo "WorkSpace termination initiated. This may take a few minutes."
    fi

    # Deregister directory (only if we registered it in this script)
    if [[ " ${CREATED_RESOURCES[*]} " == *"Directory registration:
 $DIRECTORY_ID"* ]]; then
        echo "Deregistering directory $DIRECTORY_ID from WorkSpaces..."
        DEREGISTER_OUTPUT=$(aws workspaces deregister-workspace-directory --
directory-id "$DIRECTORY_ID")
        check_command "$DEREGISTER_OUTPUT"
        echo "$DEREGISTER_OUTPUT"
        echo "Directory deregistration initiated. This may take a few minutes."
    fi

    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo ""
echo "================================================"
echo "SCRIPT COMPLETED"
echo "================================================"
echo "Log file: $LOG_FILE"
echo "Thank you for using the WorkSpaces creation script!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
  - CreateWorkspaces
  - DeregisterWorkspaceDirectory
  - DescribeWorkspaceBundles
  - DescribeWorkspaceDirectories
  - DescribeWorkspaces

- RegisterWorkspaceDirectory
- TerminateWorkspaces

Codebeispiele für die X-Ray Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie es AWS X-Ray mit einem AWS Software Development Kit (SDK) verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Weitere -Quellen

- X-Ray Entwicklerhandbuch — Weitere Informationen zu X-Ray.
- X-Ray API-Referenz — Details zu allen verfügbaren X-Ray Aktionen.
- AWS Developer Center — Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Grundlegende Beispiele für die Verwendung X-Ray AWS SDKs
  - Aktionen zur Verwendung X-Ray AWS SDKs
    - Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
    - Verwenden Sie CreateSamplingRulemit einem AWS SDK oder CLI
    - Verwenden Sie DeleteGroupmit einem AWS SDK oder CLI
    - Verwenden Sie DeleteSamplingRulemit einem AWS SDK oder CLI
    - Verwenden Sie GetGroupsmit einem AWS SDK oder CLI
    - Verwenden Sie GetSamplingRulesmit einem AWS SDK oder CLI
    - Verwenden Sie GetServiceGraphmit einem AWS SDK oder CLI

Grundlegende Beispiele für die Verwendung X-Ray AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von AWS X-Ray mit AWS SDKs verwendet werden.

Beispiele

- Aktionen zur Verwendung X-Ray AWS SDKs
  - Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
  - Verwenden Sie CreateSamplingRulemit einem AWS SDK oder CLI
  - Verwenden Sie DeleteGroupmit einem AWS SDK oder CLI
  - Verwenden Sie DeleteSamplingRulemit einem AWS SDK oder CLI
  - Verwenden Sie GetGroupsmit einem AWS SDK oder CLI
  - Verwenden Sie GetSamplingRulesmit einem AWS SDK oder CLI
  - Verwenden Sie GetServiceGraphmit einem AWS SDK oder CLI

Aktionen zur Verwendung X-Ray AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne X-Ray Aktionen mit AWS SDKs ausgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der AWS X-Ray -API-Referenz.

Beispiele

- Verwenden Sie CreateGroupmit einem AWS SDK oder CLI
- Verwenden Sie CreateSamplingRulemit einem AWS SDK oder CLI
- Verwenden Sie DeleteGroupmit einem AWS SDK oder CLI
- Verwenden Sie DeleteSamplingRulemit einem AWS SDK oder CLI
- Verwenden Sie GetGroupsmit einem AWS SDK oder CLI
- Verwenden Sie GetSamplingRulesmit einem AWS SDK oder CLI
- Verwenden Sie GetServiceGraphmit einem AWS SDK oder CLI

Verwenden Sie **`CreateGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateGroup` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Gruppe

Im folgenden Beispiel für `create-group` wird eine Gruppenressource namens `AdminGroup` erstellt. Die Gruppe erhält einen Filterausdruck, der die Kriterien der Gruppe als Segment definiert, das sich auf einen bestimmten Service bezieht, der eine Störung oder einen Fehler verursacht.

```
aws xray create-group \
   --group-name "AdminGroup" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}"
```

Ausgabe:

```
{
    "GroupName": "AdminGroup",
    "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/AdminGroup/123456789",
    "FilterExpression": "service(\"mydomain.com\") {fault OR error}"
}
```

Weitere Informationen finden Sie unter Konfiguration von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie CreateGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateGroupRequest;
import software.amazon.awssdk.services.xray.model.CreateGroupResponse;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to create\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createNewGroup(xRayClient, groupName);
    }

    public static void createNewGroup(XRayClient xRayClient, String groupName) {
        try {
            CreateGroupRequest groupRequest = CreateGroupRequest.builder()
```

```
                    .filterExpression("fault = true AND http.url CONTAINS
 \"example/game\" AND responsetime >= 5")
                    .groupName(groupName)
                    .build();

            CreateGroupResponse groupResponse =
 xRayClient.createGroup(groupRequest);
            System.out.println("The Group ARN is " +
 groupResponse.group().groupARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGroup(groupNameVal: String?) {
    val groupRequest =
        CreateGroupRequest {
            filterExpression = "fault = true AND http.url CONTAINS \"example/game
\" AND responsetime >= 5"
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val groupResponse = xRayClient.createGroup(groupRequest)
        println("The Group ARN is " + (groupResponse.group?.groupArn))
    }
```

```
}
```

- Einzelheiten zur API finden Sie CreateGroupin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`CreateSamplingRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `CreateSamplingRule` verwendet wird.

CLI

AWS CLI

So erstellen Sie eine Sampling-Regel

Im folgenden Beispiel für `create-sampling-rule` wird eine Regel zur Steuerung des Sampling-Verhaltens für instrumentierte Anwendungen erstellt. Die Regeln werden in einer JSON-Datei bereitgestellt. Die meisten Felder für die Sampling-Regel sind erforderlich, um die Regel zu erstellen.

```
aws xray create-sampling-rule \
    --cli-input-json file://9000-base-scorekeep.json
```

Inhalt von `9000-base-scorekeep.json`:

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
}
```

Ausgabe:

```
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "base-scorekeep",
            "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/base-
scorekeep",
            "ResourceARN": "*",
            "Priority": 9000,
            "FixedRate": 0.1,
            "ReservoirSize": 5,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "*",
            "URLPath": "*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574410.0,
        "ModifiedAt": 1530574410.0
    }
}
```

Weitere Informationen finden Sie unter Konfiguration von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie CreateSamplingRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleRequest;
```

```
import software.amazon.awssdk.services.xray.model.SamplingRule;
import software.amazon.awssdk.services.xray.model.XRayException;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createRule(xRayClient, ruleName);
    }

    public static void createRule(XRayClient xRayClient, String ruleName) {
        try {

            SamplingRule rule = SamplingRule.builder()
                    .ruleName(ruleName)
                    .priority(1)
```

```
                    .httpMethod("*")
                    .serviceType("*")
                    .serviceName("*")
                    .urlPath("*")
                    .version(1)
                    .host("*")
                    .resourceARN("*")
                    .build();

            CreateSamplingRuleRequest ruleRequest =
 CreateSamplingRuleRequest.builder()
                    .samplingRule(rule)
                    .build();

            CreateSamplingRuleResponse ruleResponse =
 xRayClient.createSamplingRule(ruleRequest);
            System.out.println(
                    "The ARN of the new rule is " +
 ruleResponse.samplingRuleRecord().samplingRule().ruleARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie CreateSamplingRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createRule(ruleNameVal: String?) {
    val rule =
        SamplingRule {
            ruleName = ruleNameVal
            priority = 1
            httpMethod = "*"
            serviceType = "*"
            serviceName = "*"
            urlPath = "*"
            version = 1
            host = "*"
            resourceArn = "*"
        }

    val ruleRequest =
        CreateSamplingRuleRequest {
            samplingRule = rule
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val ruleResponse: CreateSamplingRuleResponse =
 xRayClient.createSamplingRule(ruleRequest)
        println("The ARN of the new rule is
 ${ruleResponse.samplingRuleRecord?.samplingRule?.ruleArn}")
    }
}
```

- Einzelheiten zur API finden Sie CreateSamplingRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteGroup`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteGroup` verwendet wird.

CLI

AWS CLI

Um eine Gruppe zu löschen

Im folgenden Beispiel für `delete-group` wird die angegebene Gruppenressourcen gelöscht.

```
aws xray delete-group \
    --group-name "AdminGroup" \
    --group-arn "arn:aws:xray:us-east-2:123456789012:group/AdminGroup/123456789"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter Konfiguration von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteGroupin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteGroupRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>
```

```
                Where:
                   groupName - The name of the group to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteSpecificGroup(xRayClient, groupName);
    }

    public static void deleteSpecificGroup(XRayClient xRayClient, String
 groupName) {
        try {
            DeleteGroupRequest groupRequest = DeleteGroupRequest.builder()
                    .groupName(groupName)
                    .build();

            xRayClient.deleteGroup(groupRequest);
            System.out.println(groupName + " was deleted!");

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteGroupin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificGroup(groupNameVal: String) {
    val groupRequest =
        DeleteGroupRequest {
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteGroup(groupRequest)
        println("$groupNameVal was deleted!")
    }
}
```

- Einzelheiten zur API finden Sie DeleteGroupin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`DeleteSamplingRule`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DeleteSamplingRule` verwendet wird.

CLI

AWS CLI

So löschen Sie eine Sampling-Regel

Im folgenden Beispiel für `delete-sampling-rule` wird die angegebene Sampling-Regel gelöscht. Sie können die Gruppe angeben, indem Sie entweder den Gruppennamen oder den Gruppen-ARN verwenden.

```
aws xray delete-sampling-rule \
    --rule-name polling-scorekeep
```

Ausgabe:

```
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-west-2:123456789012:sampling-rule/
polling-scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
```

Weitere Informationen finden Sie unter Konfiguration von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie DeleteSamplingRulein der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
```

```
import software.amazon.awssdk.services.xray.model.DeleteSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteRule(xRayClient, ruleName);
    }

    public static void deleteRule(XRayClient xRayClient, String ruleName) {
        try {
            DeleteSamplingRuleRequest ruleRequest =
 DeleteSamplingRuleRequest.builder()
                    .ruleName(ruleName)
                    .build();
```

```
            xRayClient.deleteSamplingRule(ruleRequest);
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie DeleteSamplingRulein der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteRule(ruleNameVal: String?) {
    val ruleRequest =
        DeleteSamplingRuleRequest {
            ruleName = ruleNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteSamplingRule(ruleRequest)
        println("$ruleNameVal was deleted")
    }
}
```

- Einzelheiten zur API finden Sie DeleteSamplingRulein der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetGroups`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetGroups` verwendet wird.

CLI

AWS CLI

So rufen Sie alle Gruppen ab

Im folgenden Beispiel werden Details für alle aktiven Gruppen angezeigt.

```
aws xray get-groups
```

Ausgabe:

```
{
    "Groups": [
        {
            "GroupName": "AdminGroup",
            "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
AdminGroup/123456789",
            "FilterExpression": "service(\"example.com\") {fault OR error}"
        },
        {
            "GroupName": "SDETGroup",
            "GroupARN": "arn:aws:xray:us-west-2:123456789012:group/
SDETGroup/987654321",
            "FilterExpression": "responsetime > 2"
        }
    ]
}
```

Weitere Informationen finden Sie unter Konfiguration von Sampling-, Gruppen- und Verschlüsselungseinstellungen mit der AWS X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetGroupsin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetGroupsResponse;
import software.amazon.awssdk.services.xray.model.GroupSummary;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetGroups {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        getAllGroups(xRayClient);
    }

    public static void getAllGroups(XRayClient xRayClient) {
        try {
            GetGroupsResponse groupsResponse = xRayClient.getGroups();
            List<GroupSummary> groups = groupsResponse.groups();
            for (GroupSummary group : groups) {
```

```
                System.out.println("The AWS XRay group name is " +
 group.groupName());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetGroupsin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAllGroups() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getGroups(GetGroupsRequest {})
        response.groups?.forEach { group ->
            println("The AWS X-Ray group name is ${group.groupName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetGroupsin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetSamplingRules`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetSamplingRules` verwendet wird.

CLI

AWS CLI

So rufen Sie alle Sampling-Regeln ab

Im folgenden Beispiel für `get-sampling-rules` werden Details zu allen verfügbaren Sampling-Regeln angezeigt:

```
aws xray get-sampling-rules
```

Ausgabe:

```
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1530558121.0
        },
        {
            "SamplingRule": {
                "RuleName": "base-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-
scorekeep",
                "ResourceARN": "*",
                "Priority": 9000,
                "FixedRate": 0.1,
                "ReservoirSize": 2,
```

```
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530573954.0,
            "ModifiedAt": 1530920505.0
        },
        {
            "SamplingRule": {
                "RuleName": "polling-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-
scorekeep",
                "ResourceARN": "*",
                "Priority": 5000,
                "FixedRate": 0.003,
                "ReservoirSize": 0,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "GET",
                "URLPath": "/api/state/*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530918163.0,
            "ModifiedAt": 1530918163.0
        }
    ]
}
```

Weitere Informationen finden Sie unter Verwenden von Sampling-Regeln mit der X-Ray API im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetSamplingRulesin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetSamplingRulesResponse;
import software.amazon.awssdk.services.xray.model.SamplingRuleRecord;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSamplingRules {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        getRules(xRayClient);
    }

    public static void getRules(XRayClient xRayClient) {
        try {
            GetSamplingRulesResponse response = xRayClient.getSamplingRules(r ->
 r.build());
```

```
            List<SamplingRuleRecord> records = response.samplingRuleRecords();

            for (SamplingRuleRecord record : records) {
                System.out.println("The rule name is: " +
 record.samplingRule().ruleName());
                System.out.println("The related service is: " +
 record.samplingRule().serviceName());
            }
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetSamplingRulesin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getRules() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getSamplingRules(GetSamplingRulesRequest {})
        response.samplingRuleRecords?.forEach { record ->
            println("The rule name is ${record.samplingRule?.ruleName}")
            println("The related service is:
 ${record.samplingRule?.serviceName}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetSamplingRulesin der API-Referenz zum AWS SDK für Kotlin.

Verwenden Sie **`GetServiceGraph`**mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetServiceGraph` verwendet wird.

CLI

AWS CLI

So rufen Sie ein Service-Diagramm ab

Im folgenden Beispiel wird ein Dokument innerhalb eines bestimmten Zeitraums angezeigt, das Services zur Verarbeitung eingehender Anforderungen beschreibt sowie die Downstream-Services, die diese aufgrund dessen aufrufen:

```
aws xray get-service-graph \
    --start-time 1568835392.0
    --end-time 1568835446.0
```

Ausgabe:

```
{
    "Services": [
        {
            "ReferenceId": 0,
            "Name": "Scorekeep",
            "Names": [
                "Scorekeep"
            ],
            "Root": true,
            "Type": "AWS::ElasticBeanstalk::Environment",
            "State": "active",
            "StartTime": 1568835392.0,
            "EndTime": 1568835446.0,
            "Edges": [
                {
                    "ReferenceId": 1,
                    "StartTime": 1568835392.0,
                    "EndTime": 1568835446.0,
                    "SummaryStatistics": {
```

```
                "OkCount": 14,
                "ErrorStatistics": {
                    "ThrottleCount": 0,
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "FaultStatistics": {
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "TotalCount": 14,
                "TotalResponseTime": 0.13
            },
            "ResponseTimeHistogram": [
                {
                    "Value": 0.008,
                    "Count": 1
                },
                {
                    "Value": 0.005,
                    "Count": 7
                },
                {
                    "Value": 0.009,
                    "Count": 1
                },
                {
                    "Value": 0.021,
                    "Count": 1
                },
                {
                    "Value": 0.038,
                    "Count": 1
                },
                {
                    "Value": 0.007,
                    "Count": 1
                },
                {
                    "Value": 0.006,
                    "Count": 2
                }
            ],
            "Aliases": []
```

```
                },

                ... TRUNCATED FOR BREVITY ...

            ]
        }
    ],
    "StartTime": 1568835392.0,
    "EndTime": 1568835446.0,
    "ContainsOldGroupVersions": false
}
```

Weitere Informationen finden Sie unter Verwenden der AWS X-Ray API mit der AWS CLI im AWS X-Ray Entwicklerhandbuch.

- Einzelheiten zur API finden Sie GetServiceGraphin der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetServiceGraphRequest;
import software.amazon.awssdk.services.xray.model.GetServiceGraphResponse;
import software.amazon.awssdk.services.xray.model.Service;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.time.LocalDateTime;
import java.time.Instant;
import java.time.ZoneId;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetServiceGraph {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of a group based on which you want to
 generate a graph.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();
        getGraph(xRayClient, groupName);
    }

    public static void getGraph(XRayClient xRayClient, String groupName) {
        try {
            // The Instant values have to be 6 hours apart.
            LocalDateTime localDateTime =
 LocalDateTime.parse("2022-03-09T06:00:00");
            Instant start = localDateTime.atZone(ZoneId.of("America/
New_York")).toInstant();

            LocalDateTime localDateTime2 =
 LocalDateTime.parse("2022-03-09T12:00:00");
            Instant end = localDateTime2.atZone(ZoneId.of("America/
New_York")).toInstant();
```

```
            GetServiceGraphRequest getServiceGraphRequest =
 GetServiceGraphRequest.builder()
                    .groupName(groupName)
                    .startTime(start)
                    .endTime(end)
                    .build();

            GetServiceGraphResponse graphResponse =
 xRayClient.getServiceGraph(getServiceGraphRequest);
            List<Service> services = graphResponse.services();
            for (Service service : services) {
                System.out.println("The name of the service is  " +
 service.name());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Einzelheiten zur API finden Sie GetServiceGraphin der AWS SDK for Java 2.x API-Referenz.

Kotlin

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getGraph(groupNameVal: String?) {
    val time = aws.smithy.kotlin.runtime.time.Instant
    val getServiceGraphRequest =
        GetServiceGraphRequest {
            groupName = groupNameVal
```

```
            this.startTime = time.now()
            endTime = time.now()
        }
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getServiceGraph(getServiceGraphRequest)
        response.services?.forEach { service ->
            println("The name of the service is  ${service.name}")
        }
    }
}
```

- Einzelheiten zur API finden Sie GetServiceGraphin der API-Referenz zum AWS SDK für Kotlin.

Codebeispiele von SDK unter Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen die Verwendung AWS-Services mit einem AWS Software Development Kit (SDK). Dieser Abschnitt ist nach SDK-Sprachen gegliedert.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Codebeispiele

- Codebeispiele für SDK für .NET
- Codebeispiele für SDK für .NET (v4)
- Codebeispiele für AWS CLI mit Bash-Skript
- Codebeispiele für SDK für C++
- Codebeispiele für SDK für Go V2
- Codebeispiele für SDK für JavaScript (v2)
- Codebeispiele für SDK für JavaScript (v3)
- Codebeispiele für SDK für Java 2.x
- Codebeispiele für SDK für Kotlin
- Codebeispiele für SDK für PHP

- Codebeispiele für Tools für V5 PowerShell
- Codebeispiele für SDK für Python
- Codebeispiele für SDK für Python (Boto3)
- Codebeispiele für SDK für Ruby
- Codebeispiele für SDK für Rust
- Codebeispiele für SDK für SAP ABAP
- Codebeispiele für SDK für Swift

Codebeispiele für SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für .NET with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für .NET Entwicklerhandbuch — Weitere Informationen zur Verwendung von.NET mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele unter Verwendung SDK für .NET
- API Gateway-Beispiele mit SDK für .NET

- Aurora-Beispiele mit SDK für .NET
- Beispiele für Auto Scaling mit SDK für .NET
- Beispiele für Amazon Bedrock Runtime mit SDK für .NET
- CloudWatch Protokolliert Beispiele mit SDK für .NET
- Beispiele für Amazon Cognito Identity Provider unter SDK für .NET
- Amazon Comprehend Beispiele mit SDK für .NET
- Beispiele für Amazon DocumentDB unter Verwendung SDK für .NET
- DynamoDB-Beispiele mit SDK für .NET
- Amazon EC2-Beispiele mit SDK für .NET
- Amazon ECS-Beispiele mit SDK für .NET
- Beispiele für Elastic Load Balancing — Version 2 mit SDK für .NET
- EventBridge Beispiele mit SDK für .NET
- EventBridge Scheduler-Beispiele mit SDK für .NET
- Beispiele für Amazon Glacier mit SDK für .NET
- AWS Glue Beispiele unter Verwendung SDK für .NET
- IAM-Beispiele mit SDK für .NET
- Beispiele für Amazon Keyspaces unter Verwendung SDK für .NET
- Kinesis-Beispiele mit SDK für .NET
- AWS KMS Beispiele mit SDK für .NET
- Lambda-Beispiele mit SDK für .NET
- MediaConvert Beispiele unter Verwendung SDK für .NET
- Amazon MSK-Beispiele mit SDK für .NET
- Beispiele für Organisationen mit SDK für .NET
- Partner Central-Beispiele mit SDK für .NET
- Amazon Pinpoint-Beispiele mit SDK für .NET
- Beispiele für Amazon Polly mit SDK für .NET
- Beispiele für Amazon RDS unter Verwendung SDK für .NET
- Beispiele für Amazon RDS Data Service unter SDK für .NET
- Beispiele für Amazon Rekognition mit SDK für .NET
- Beispiele für die Domainregistrierung von Route 53 unter Verwendung SDK für .NET

- Amazon S3-Beispiele mit SDK für .NET
- SageMaker KI-Beispiele unter Verwendung SDK für .NET
- Secrets Manager-Beispiele mit SDK für .NET
- Amazon SES-Beispiele mit SDK für .NET
- Beispiele für Amazon SES API v2 unter Verwendung SDK für .NET
- Amazon SNS-Beispiele unter Verwendung SDK für .NET
- Amazon SQS-Beispiele mit SDK für .NET
- Beispiele für Step Functions mit SDK für .NET
- AWS STS Beispiele unter Verwendung SDK für .NET
- Support Beispiele unter Verwendung SDK für .NET
- Beispiele für Amazon Textract mit SDK für .NET
- Amazon Transcribe Beispiele mit SDK für .NET
- Beispiele für Amazon Translate mit SDK für .NET

ACM-Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von ACM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCertificate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;

namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
        // certificate using the AWS Certificate Manager (ACM) service.

        // Specify your AWS Region (an example Region is shown).
        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new
 Amazon.CertificateManager.AmazonCertificateManagerClient(ACMRegion);

            var describeCertificateReq = new DescribeCertificateRequest();
            // The ARN used here is just an example. Replace it with the ARN of
            // a certificate that exists on your account.
            describeCertificateReq.CertificateArn =
                "arn:aws:acm:us-
east-1:123456789012:certificate/8cfd7dae-9b6a-2d07-92bc-1c309EXAMPLE";

            var certificateDetailResp =
                DescribeCertificateResponseAsync(client: _client, request:
 describeCertificateReq);
            var certificateDetail = certificateDetailResp.Result.Certificate;
```

```
            if (certificateDetail is not null)
            {
                DisplayCertificateDetails(certificateDetail);
            }
        }

        /// <summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        /// </summary>
        /// <param name="certificateDetail">The object that contains details
        /// returned from the call to DescribeCertificateAsync.</param>
        static void DisplayCertificateDetails(CertificateDetail certificateDetail)
        {
            Console.WriteLine("\nCertificate Details: ");
            Console.WriteLine($"Certificate Domain:
{certificateDetail.DomainName}");
            Console.WriteLine($"Certificate Arn:
{certificateDetail.CertificateArn}");
            Console.WriteLine($"Certificate Subject: {certificateDetail.Subject}");
            Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
            foreach (var san in certificateDetail.SubjectAlternativeNames)
            {
                Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with the ACM service certificate.
        /// </summary>
        /// <param name="client">An AmazonCertificateManagerClient object
        /// used to call DescribeCertificateResponse.</param>
        /// <param name="request">The DescribeCertificateRequest object that
        /// will be passed to the method call.</param>
        /// <returns></returns>
        static async Task<DescribeCertificateResponse>
DescribeCertificateResponseAsync(
            AmazonCertificateManagerClient client, DescribeCertificateRequest
request)
        {
            var response = new DescribeCertificateResponse();

            try
            {
```

```
                response = await client.DescribeCertificateAsync(request);
            }
            catch (InvalidArnException)
            {
                Console.WriteLine($"Error: The ARN specified is invalid.");
            }
            catch (ResourceNotFoundException)
            {
                Console.WriteLine($"Error: The specified certificate could not be
 found.");
            }

            return response;
        }
    }

}
```

- API-Details finden Sie DescribeCertificate in der AWS SDK für .NET API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;

namespace ListCertificates
```

```
{
    // The following example retrieves and displays a list of the
    // certificates defined for the default account using the AWS
    // Certificate Manager (ACM) service.
    class ListCertificates
    {
        // Specify your AWS Region (an example Region is shown).

        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            _client = new AmazonCertificateManagerClient(ACMRegion);
            var certificateList = ListCertificatesResponseAsync(client: _client);

            Console.WriteLine("Certificate Summary List\n");

            foreach (var certificate in
 certificateList.Result.CertificateSummaryList)
            {
                Console.WriteLine($"Certificate Domain: {certificate.DomainName}");
                Console.WriteLine($"Certificate ARN:
 {certificate.CertificateArn}\n");
            }
        }

        /// <summary>
        /// Retrieves a list of the certificates defined in this Region.
        /// </summary>
        /// <param name="client">The ACM client object passed to the
        /// ListCertificateResAsync method call.</param>
        /// <param name="request"></param>
        /// <returns>The ListCertificatesResponse.</returns>
        static async Task<ListCertificatesResponse> ListCertificatesResponseAsync(
            AmazonCertificateManagerClient client)
        {
            var request = new ListCertificatesRequest();

            var response = await client.ListCertificatesAsync(request);
            return response;
        }
    }
}
```

- API-Details finden Sie ListCertificates in der AWS SDK für .NET API-Referenz.

API Gateway-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von API Gateway Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden AWS. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien
- AWS Beiträge der Gemeinschaft

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

> Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für .NET

Zeigt, wie eine Serverless-Anwendung, die aus einem API Gateway mit Lambda und DynamoDB besteht, mithilfe des .NET-SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Aurora-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie Sie das verwenden AWS SDK für .NET , um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend erstellt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Auto Scaling mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AttachLoadBalancerTargetGroups`

Das folgende Codebeispiel zeigt die Verwendung`AttachLoadBalancerTargetGroups`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
```

```
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- API-Details finden Sie AttachLoadBalancerTargetGroups in der AWS SDK für .NET API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
```

```
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
```

```
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);
```

```
            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
```

```
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
```

```
                            $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                            $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
```

```
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");
```

```
        Console.WriteLine($"\nNow, checking target health indicates that the
 instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
 the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
```

```
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
```

```
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
```

```
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
```

```
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
                                        "]" +
                                        "}," +
                                   "\"Action\": \"sts:AssumeRole\"" +
                                   "}]" +
                               "}";

       var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

       var policyArn = "";

       try
       {
           var createPolicyResult = await _amazonIam.CreatePolicyAsync(
               new CreatePolicyRequest
               {
                   PolicyName = policyName,
                   PolicyDocument = policyDocument
               });
```

```
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
```

```
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
```

```
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
```

```
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
```

```
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                  $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
```

```
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
```

```
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
```

```
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
```

```
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
```

```
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
    /// If you do, be sure to remove public access when you're done.
```

```
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }
```

```
    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
```

```
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
```

```
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
```

```
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
```

```
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
```

```
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
```

```
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
```

```
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
```

```
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Beispiele für Amazon Bedrock Runtime mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Szenarien
- Amazon Nova
- Amazon Nova Canvas
- Stable Image Core

Erste Schritte

Hallo Amazon Bedrock Runtime

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock Runtime beginnen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;
```

```
/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
```

```
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Szenarien

Erstellen einer Playground-Anwendung für die Interaktion mit Amazon-Bedrock-Basismodellen

Das folgende Codebeispiel zeigt, wie Sie Playgrounds erstellen, um mit Amazon-Bedrock-Basismodellen über verschiedene Modalitäten zu interagieren.

SDK für .NET

.NET Foundation Model (FM) Playground ist eine .NET MAUI Blazor-Beispielanwendung, die zeigt, wie Amazon Bedrock aus C#-Code verwendet wird. Dieses Beispiel zeigt, wie .NET- und C#-Entwickler Amazon Bedrock verwenden können, um generative AI-enabled Anwendungen zu erstellen. Sie können die Amazon-Bedrock-Basismodelle mithilfe der folgenden vier Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Voice-Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
```

```
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
   */

    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and refer
 to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your purpose.
        - Never claim to search online, access external data, or use tools besides
 Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
```

```
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content = new
List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
```

```
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
 message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
 SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
 the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
 system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
 necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
 Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
 modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
 infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
 Please try again.");
        }
```

```
        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
        {
            // If the stop reason is "tool_use", forward everything to the tool use
 handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
 maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
 and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
 request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
 List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
```

```
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }

            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
```

```
    /// Invokes the specified tool with the given payload and returns the tool's
 response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input data.</
param>
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
 _weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
 inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
 weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
 not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content = new
 { error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
 weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
```

```
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }

        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of a
city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.
```

```
        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
```

```
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-schema/
reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
```

```
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
```

```
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();

            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data: {Message}",
 e.Message);
            throw;
        }
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
```

```
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };
```

```
            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// Use the Converse API to send a text message to Amazon Nova.

using System;
using System.Collections.Generic;
using Amazon;
```

```
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

Senden Sie mithilfe der Converse-API von Bedrock und einer Tool-Konfiguration eine Konversation mit Nachrichten an Amazon Nova.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
```

```
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Amazon Nova
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Amazon Nova Lite.
var modelId = "amazon.nova-lite-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
```

```
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für .NET API-Referenz.

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen

zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Http;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;

namespace ConverseToolScenario;

public static class ConverseToolScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This demo illustrates a tool use scenario using Amazon Bedrock's Converse API
 and a weather tool.
    The script interacts with a foundation model on Amazon Bedrock to provide
 weather information based on user
    input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
   */
```

```
    public static BedrockActionsWrapper _bedrockActionsWrapper = null!;
    public static WeatherTool _weatherTool = null!;
    public static bool _interactive = true;

    // Change this string to use a different model with Converse API.
    private static string model_id = "amazon.nova-lite-v1:0";

    private static string system_prompt = @"
        You are a weather assistant that provides current weather data for user-
specified locations using only
        the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
        If the user specifies a state, country, or region, infer the locations of
 cities within that state.
        If the user provides coordinates, infer the approximate location and refer
 to it in your response.
        To use the tool, you strictly apply the provided tool specification.

        - Explain your step-by-step process, and give brief updates before each
 step.
        - Only use the Weather_Tool for data. Never guess or make up information.
        - Repeat the tool use for subsequent requests if necessary.
        - If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.
        - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
          emojis where appropriate.
        - Only respond to weather queries. Remind off-topic users of your purpose.
        - Never claim to search online, access external data, or use tools besides
 Weather_Tool.
        - Complete the entire process until you have all required data before
 sending the complete response.
    "
    ;

    private static string default_prompt = "What is the weather like in Seattle?";

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int max_recursions = 5;

    public static async Task Main(string[] args)
    {
```

```
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Error)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddHttpClient()
                    .AddSingleton<IAmazonBedrockRuntime>(_ => new
AmazonBedrockRuntimeClient(RegionEndpoint.USEast1)) // Specify a region that has
access to the chosen model.
                    .AddTransient<BedrockActionsWrapper>()
                    .AddTransient<WeatherTool>()
                    .RemoveAll<IHttpMessageHandlerBuilderFilter>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            await RunConversationAsync();

        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            Console.WriteLine(new string('-', 80));
        }
        finally
        {
            Console.WriteLine(
                "Amazon Bedrock Converse API with Tool Use Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
```

```
    {
        _bedrockActionsWrapper =
host.Services.GetRequiredService<BedrockActionsWrapper>();
        _weatherTool = host.Services.GetRequiredService<WeatherTool>();
    }

    /// <summary>
    /// Starts the conversation with the user and handles the interaction with
Bedrock.
    /// </summary>
    /// <returns>The conversation array.</returns>
    public static async Task<List<Message>> RunConversationAsync()
    {
        // Print the greeting and a short user guide
        PrintHeader();

        // Start with an empty conversation
        var conversation = new List<Message>();

        // Get the first user input
        var userInput = await GetUserInputAsync();

        while (userInput != null)
        {
            // Create a new message with the user input and append it to the
conversation
            var message = new Message { Role = ConversationRole.User, Content = new
List<ContentBlock> { new ContentBlock { Text = userInput } } };
            conversation.Add(message);

            // Send the conversation to Amazon Bedrock
            var bedrockResponse = await SendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0
            await ProcessModelResponseAsync(bedrockResponse, conversation,
max_recursions);

            // Repeat the loop until the user decides to exit the application
            userInput = await GetUserInputAsync();
        }

        PrintFooter();
        return conversation;
```

```
    }

    /// <summary>
    /// Sends the conversation, the system prompt, and the tool spec to Amazon
 Bedrock, and returns the response.
    /// </summary>
    /// <param name="conversation">The conversation history including the next
 message to send.</param>
    /// <returns>The response from Amazon Bedrock.</returns>
    private static async Task<ConverseResponse>
 SendConversationToBedrock(List<Message> conversation)
    {
        Console.WriteLine("\tCalling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
 the response
        return await _bedrockActionsWrapper.SendConverseRequestAsync(model_id,
 system_prompt, conversation, _weatherTool.GetToolSpec());
    }

    /// <summary>
    /// Processes the response received via Amazon Bedrock and performs the
 necessary actions based on the stop reason.
    /// </summary>
    /// <param name="modelResponse">The model's response returned via Amazon
 Bedrock.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    private static async Task ProcessModelResponseAsync(ConverseResponse
 modelResponse, List<Message> conversation, int maxRecursion)
    {
        if (maxRecursion <= 0)
        {
            // Stop the process, the number of recursive calls could indicate an
 infinite loop
            Console.WriteLine("\tWarning: Maximum number of recursions reached.
 Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.Add(modelResponse.Output.Message);

        if (modelResponse.StopReason == "tool_use")
```

```
        {
            // If the stop reason is "tool_use", forward everything to the tool use
 handler
            await HandleToolUseAsync(modelResponse.Output, conversation,
 maxRecursion - 1);
        }

        if (modelResponse.StopReason == "end_turn")
        {
            // If the stop reason is "end_turn", print the model's response text,
 and finish the process
            PrintModelResponse(modelResponse.Output.Message.Content[0].Text);
            if (!_interactive)
            {
                default_prompt = "x";
            }
        }
    }

    /// <summary>
    /// Handles the tool use case by invoking the specified tool and sending the
 tool's response back to Bedrock.
    /// The tool response is appended to the conversation, and the conversation is
 sent back to Amazon Bedrock for further processing.
    /// </summary>
    /// <param name="modelResponse">The model's response containing the tool use
 request.</param>
    /// <param name="conversation">The conversation history.</param>
    /// <param name="maxRecursion">The maximum number of recursive calls allowed.</
param>
    public static async Task HandleToolUseAsync(ConverseOutput modelResponse,
 List<Message> conversation, int maxRecursion)
    {
        // Initialize an empty list of tool results
        var toolResults = new List<ContentBlock>();

        // The model's response can consist of multiple content blocks
        foreach (var contentBlock in modelResponse.Message.Content)
        {
            if (!String.IsNullOrEmpty(contentBlock.Text))
            {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.Text);
            }
```

```
            if (contentBlock.ToolUse != null)
            {
                // If the content block is a tool use request, forward it to the
 tool
                var toolResponse = await InvokeTool(contentBlock.ToolUse);

                // Add the tool use ID and the tool's response to the list of
 results
                toolResults.Add(new ContentBlock
                {
                    ToolResult = new ToolResultBlock()
                    {
                        ToolUseId = toolResponse.ToolUseId,
                        Content = new List<ToolResultContentBlock>()
                            { new ToolResultContentBlock { Json =
 toolResponse.Content } }
                    }
                });
            }
        }

        // Embed the tool results in a new user message
        var message = new Message() { Role = ConversationRole.User, Content =
 toolResults };

        // Append the new message to the ongoing conversation
        conversation.Add(message);

        // Send the conversation to Amazon Bedrock
        var response = await SendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
 final response or the recursion counter has reached 0
        await ProcessModelResponseAsync(response, conversation, maxRecursion);
    }

    /// <summary>
    /// Invokes the specified tool with the given payload and returns the tool's
 response.
    /// If the requested tool does not exist, an error message is returned.
    /// </summary>
    /// <param name="payload">The payload containing the tool name and input data.</
param>
```

```
    /// <returns>The tool's response or an error message.</returns>
    public static async Task<ToolResponse> InvokeTool(ToolUseBlock payload)
    {
        var toolName = payload.Name;

        if (toolName == "Weather_Tool")
        {
            var inputData = payload.Input.AsDictionary();
            PrintToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            var weatherResponse = await
_weatherTool.FetchWeatherDataAsync(inputData["latitude"].ToString(),
inputData["longitude"].ToString());
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content =
weatherResponse };
        }
        else
        {
            var errorMessage = $"\tThe requested tool with name '{toolName}' does
not exist.";
            return new ToolResponse { ToolUseId = payload.ToolUseId, Content = new
{ error = true, message = errorMessage } };
        }
    }


    /// <summary>
    /// Prompts the user for input and returns the user's response.
    /// Returns null if the user enters 'x' to exit.
    /// </summary>
    /// <param name="prompt">The prompt to display to the user.</param>
    /// <returns>The user's input or null if the user chooses to exit.</returns>
    private static async Task<string?> GetUserInputAsync(string prompt = "\tYour
weather info request:")
    {
        var userInput = default_prompt;
        if (_interactive)
        {
            Console.WriteLine(new string('*', 80));
            Console.WriteLine($"{prompt} (x to exit): \n\t");
            userInput = Console.ReadLine();
        }
```

```
        if (string.IsNullOrWhiteSpace(userInput))
        {
            prompt = "\tPlease enter your weather info request, e.g. the name of a
city";
            return await GetUserInputAsync(prompt);
        }

        if (userInput.ToLowerInvariant() == "x")
        {
            return null;
        }

        return userInput;
    }

    /// <summary>
    /// Logs the welcome message and usage guide for the tool use demo.
    /// </summary>
    public static void PrintHeader()
    {
        Console.WriteLine(@"
        ==================================================
        Welcome to the Amazon Bedrock Tool Use demo!
        ==================================================

        This assistant provides current weather information for user-specified
locations.
        You can ask for weather details by providing the location name or
coordinates. Weather information
        will be provided using a custom Tool and open-meteo API.

        Example queries:
        - What's the weather like in New York?
        - Current weather for latitude 40.70, longitude -74.01
        - Is it warmer in Rome or Barcelona today?

        To exit the program, simply type 'x' and press Enter.

        P.S.: You're not limited to single locations, or even to using English!
        Have fun and experiment with the app!
        ");
    }

    /// <summary>
```

```
    /// Logs the footer information for the tool use demo.
    /// </summary>
    public static void PrintFooter()
    {
        Console.WriteLine(@"
        ==================================================
        Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you
        learned something new, or got some inspiration for your own apps today!

        For more Bedrock examples in different programming languages, have a look
 at:
        https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
        ==================================================
        ");
    }

    /// <summary>
    /// Logs information about the tool use.
    /// </summary>
    /// <param name="toolName">The name of the tool being used.</param>
    /// <param name="inputData">The input data for the tool.</param>
    public static void PrintToolUse(string toolName, Dictionary<string, Document>
 inputData)
    {
        Console.WriteLine($"\n\tInvoking tool: {toolName} with input:
 {inputData["latitude"].ToString()}, {inputData["longitude"].ToString()}...\n");
    }

    /// <summary>
    /// Logs the model's response.
    /// </summary>
    /// <param name="message">The model's response message.</param>
    public static void PrintModelResponse(string message)
    {
        Console.WriteLine("\tThe model's response:\n");
        Console.WriteLine(message);
        Console.WriteLine();
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
using Amazon.BedrockRuntime.Model;
using Amazon.Runtime.Documents;
using Microsoft.Extensions.Logging;

namespace ConverseToolScenario;

/// <summary>
/// Weather tool that will be invoked when requested by the Bedrock response.
/// </summary>
public class WeatherTool
{
    private readonly ILogger<WeatherTool> _logger;
    private readonly IHttpClientFactory _httpClientFactory;

    public WeatherTool(ILogger<WeatherTool> logger, IHttpClientFactory
 httpClientFactory)
    {
        _logger = logger;
        _httpClientFactory = httpClientFactory;
    }

    /// <summary>
    /// Returns the JSON Schema specification for the Weather tool. The tool
 specification
    /// defines the input schema and describes the tool's functionality.
    /// For more information, see https://json-schema.org/understanding-json-schema/
reference.
    /// </summary>
    /// <returns>The tool specification for the Weather tool.</returns>
    public ToolSpecification GetToolSpec()
    {
        ToolSpecification toolSpecification = new ToolSpecification();

        toolSpecification.Name = "Weather_Tool";
        toolSpecification.Description = "Get the current weather for a given
 location, based on its WGS84 coordinates.";

        Document toolSpecDocument = Document.FromObject(
            new
```

```
            {
                type = "object",
                properties = new
                {
                    latitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 latitude of the location."
                    },
                    longitude = new
                    {
                        type = "string",
                        description = "Geographical WGS84 longitude of the
 location."
                    }
                },
                required = new[] { "latitude", "longitude" }
            });

        toolSpecification.InputSchema = new ToolInputSchema() { Json =
 toolSpecDocument };
        return toolSpecification;
    }

    /// <summary>
    /// Fetches weather data for the given latitude and longitude using the Open-
Meteo API.
    /// Returns the weather data or an error message if the request fails.
    /// </summary>
    /// <param name="latitude">The latitude of the location.</param>
    /// <param name="longitude">The longitude of the location.</param>
    /// <returns>The weather data or an error message.</returns>
    public async Task<Document> FetchWeatherDataAsync(string latitude, string
 longitude)
    {
        string endpoint = "https://api.open-meteo.com/v1/forecast";

        try
        {
            var httpClient = _httpClientFactory.CreateClient();
            var response = await httpClient.GetAsync($"{endpoint}?
latitude={latitude}&longitude={longitude}&current_weather=True");
            response.EnsureSuccessStatusCode();
            var weatherData = await response.Content.ReadAsStringAsync();
```

```
            Document weatherDocument = Document.FromObject(
                new { weather_data = weatherData });

            return weatherDocument;
        }
        catch (HttpRequestException e)
        {
            _logger.LogError(e, "Error fetching weather data: {Message}",
 e.Message);
            throw;
        }
        catch (Exception e)
        {
            _logger.LogError(e, "Unexpected error fetching weather data: {Message}",
 e.Message);
            throw;
        }
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
/// <summary>
/// Wrapper class for interacting with the Amazon Bedrock Converse API.
/// </summary>
public class BedrockActionsWrapper
{
    private readonly IAmazonBedrockRuntime _bedrockClient;
    private readonly ILogger<BedrockActionsWrapper> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="BedrockActionsWrapper"/> class.
    /// </summary>
    /// <param name="bedrockClient">The Bedrock Converse API client.</param>
    /// <param name="logger">The logger instance.</param>
    public BedrockActionsWrapper(IAmazonBedrockRuntime bedrockClient,
 ILogger<BedrockActionsWrapper> logger)
    {
        _bedrockClient = bedrockClient;
        _logger = logger;
```

```
    }

    /// <summary>
    /// Sends a Converse request to the Amazon Bedrock Converse API.
    /// </summary>
    /// <param name="modelId">The Bedrock Model Id.</param>
    /// <param name="systemPrompt">A system prompt instruction.</param>
    /// <param name="conversation">The array of messages in the conversation.</
param>
    /// <param name="toolSpec">The specification for a tool.</param>
    /// <returns>The response of the model.</returns>
    public async Task<ConverseResponse> SendConverseRequestAsync(string modelId,
 string systemPrompt, List<Message> conversation, ToolSpecification toolSpec)
    {
        try
        {
            var request = new ConverseRequest()
            {
                ModelId = modelId,
                System = new List<SystemContentBlock>()
                {
                    new SystemContentBlock()
                    {
                        Text = systemPrompt
                    }
                },
                Messages = conversation,
                ToolConfig = new ToolConfiguration()
                {
                    Tools = new List<Tool>()
                    {
                        new Tool()
                        {
                            ToolSpec = toolSpec
                        }
                    }
                }
            };

            var response = await _bedrockClient.ConverseAsync(request);

            return response;
        }
        catch (ModelNotReadyException ex)
```

```
        {
            _logger.LogError(ex, "Model not ready, please wait and try again.");
            throw;
        }
        catch (AmazonBedrockRuntimeException ex)
        {
            _logger.LogError(ex, "Error occurred while sending Converse request.");
            throw;
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Amazon Nova Canvas

InvokeModel

Das folgende Codebeispiel zeigt, wie Amazon Nova Canvas auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas.

using System;
using System.IO;
using System.Text.Json;
using System.Text.Json.Nodes;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;
```

```
// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID.
var modelId = "amazon.nova-canvas-v1:0";

// Define the image generation prompt for the model.
var prompt = "A stylized picture of a cute old steampunk robot.";

// Create a random seed between 0 and 858,993,459
int seed = new Random().Next(0, 858993460);

//Format the request payload using the model's native structure.
var nativeRequest = JsonSerializer.Serialize(new
{
    taskType = "TEXT_IMAGE",
    textToImageParams = new
    {
        text = prompt
    },
    imageGenerationConfig = new
    {
        seed,
        quality = "standard",
        width = 512,
        height = 512,
        numberOfImages = 1
    }
});

// Create a request with the model ID and the model's native request payload.
var request = new InvokeModelRequest()
{
    ModelId = modelId,
    Body = new MemoryStream(System.Text.Encoding.UTF8.GetBytes(nativeRequest)),
    ContentType = "application/json"
};

try
{
    // Send the request to the Bedrock Runtime and wait for the response.
    var response = await client.InvokeModelAsync(request);

    // Decode the response body.
```

```
    var modelResponse = await JsonNode.ParseAsync(response.Body);

    // Extract the image data.
    var base64Image = modelResponse["images"]?[0].ToString() ?? "";

    // Save the image in a local folder
    string savedPath = AmazonNovaCanvas.InvokeModel.SaveBase64Image(base64Image);
    Console.WriteLine($"Image saved to: {savedPath}");
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie InvokeModel in der AWS SDK für .NET API-Referenz.

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Image mit Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to run
 an inference based on the provided input.
        /// </summary>
```

```
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable Image
 Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string prompt,
 int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
```

```
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code " +
response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- API-Details finden Sie InvokeModel in der AWS SDK für .NET API-Referenz.

CloudWatch Protokolliert Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssociateKmsKey`

Das folgende Codebeispiel zeigt die Verwendung`AssociateKmsKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to associate an AWS Key Management Service (AWS KMS) key with
    /// an Amazon CloudWatch Logs log group.
    /// </summary>
    public class AssociateKmsKey
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            string groupName = "cloudwatchlogs-example-loggroup";

            var request = new AssociateKmsKeyRequest
            {
                KmsKeyId = kmsKeyId,
                LogGroupName = groupName,
            };

            var response = await client.AssociateKmsKeyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
```

```
                Console.WriteLine($"Successfully associated KMS key ID: {kmsKeyId}
with log group: {groupName}.");
            }
            else
            {
                Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
            }
        }
    }
```

- API-Details finden Sie AssociateKmsKey in der AWS SDK für .NET API-Referenz.

`CancelExportTask`

Das folgende Codebeispiel zeigt die Verwendung`CancelExportTask`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task.
/// </summary>
public class CancelExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
```

```
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskId = "exampleTaskId";

            var request = new CancelExportTaskRequest
            {
                TaskId = taskId,
            };

            var response = await client.CancelExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{taskId} successfully canceled.");
            }
            else
            {
                Console.WriteLine($"{taskId} could not be canceled.");
            }
        }
    }
```

- API-Details finden Sie CancelExportTask in der AWS SDK für .NET API-Referenz.

`CreateExportTask`

Das folgende Codebeispiel zeigt die Verwendung`CreateExportTask`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;
```

```
    /// <summary>
    /// Shows how to create an Export Task to export the contents of the Amazon
    /// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon S3)
    /// bucket.
    /// </summary>
    public class CreateExportTask
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();
            string taskName = "export-task-example";
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string destination = "amzn-s3-demo-bucket";
            var fromTime = 1437584472382;
            var toTime = 1437584472833;

            var request = new CreateExportTaskRequest
            {
                From = fromTime,
                To = toTime,
                TaskName = taskName,
                LogGroupName = logGroupName,
                Destination = destination,
            };

            var response = await client.CreateExportTaskAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The task, {taskName} with ID: " +
                                  $"{response.TaskId} has been created
successfully.");
            }
        }
    }
```

- API-Details finden Sie CreateExportTask in der AWS SDK für .NET API-Referenz.

`CreateLogGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateLogGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Shows how to create an Amazon CloudWatch Logs log group.
    /// </summary>
    public class CreateLogGroup
    {
        public static async Task Main()
        {
            // This client object will be associated with the same AWS Region
            // as the default user on this system. If you need to use a
            // different AWS Region, pass it as a parameter to the client
            // constructor.
            var client = new AmazonCloudWatchLogsClient();

            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new CreateLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.CreateLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
```

```
            }
            else
            {
                Console.WriteLine("Could not create log group.");
            }
        }
    }
```

- API-Details finden Sie CreateLogGroup in der AWS SDK für .NET API-Referenz.

`CreateLogStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateLogStream`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
```

```
            string logGroupName = "cloudwatchlogs-example-loggroup";
            string logStreamName = "cloudwatchlogs-example-logstream";

            var request = new CreateLogStreamRequest
            {
                LogGroupName = logGroupName,
                LogStreamName = logStreamName,
            };

            var response = await client.CreateLogStreamAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
            }
            else
            {
                Console.WriteLine("Could not create stream.");
            }
        }
    }
```

- API-Details finden Sie CreateLogStream in der AWS SDK für .NET API-Referenz.

`DeleteLogGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLogGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.CloudWatchLogs;
```

```
    using Amazon.CloudWatchLogs.Model;

    /// <summary>
    /// Uses the Amazon CloudWatch Logs Service to delete an existing
    /// CloudWatch Logs log group.
    /// </summary>
    public class DeleteLogGroup
    {
        public static async Task Main()
        {
            var client = new AmazonCloudWatchLogsClient();
            string logGroupName = "cloudwatchlogs-example-loggroup";

            var request = new DeleteLogGroupRequest
            {
                LogGroupName = logGroupName,
            };

            var response = await client.DeleteLogGroupAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
            }
        }
    }
```

- API-Details finden Sie DeleteLogGroup in der AWS SDK für .NET API-Referenz.

`DescribeExportTasks`

Das folgende Codebeispiel zeigt die Verwendung`DescribeExportTasks`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
                Console.WriteLine($"{t.TaskName} with ID: {t.TaskId} has status:
{t.Status}");
            });
        }
        while (response.NextToken is not null);
    }
}
```

- API-Details finden Sie DescribeExportTasks in der AWS SDK für .NET API-Referenz.

`DescribeLogGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeLogGroups`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
        // user. If you need to work with resources in another
        // AWS Region than the one defined for the default user,
        // pass the AWS Region as a parameter to the client constructor.
        var client = new AmazonCloudWatchLogsClient();

        bool done = false;
        string newToken = null;

        var request = new DescribeLogGroupsRequest
        {
            Limit = 5,
        };

        DescribeLogGroupsResponse response;

        do
        {
```

```
                if (newToken is not null)
                {
                    request.NextToken = newToken;
                }

                response = await client.DescribeLogGroupsAsync(request);

                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"{lg.LogGroupName} is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on: {lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });

                if (response.NextToken is null)
                {
                    done = true;
                }
                else
                {
                    newToken = response.NextToken;
                }
            }
            while (!done);
        }
    }
```

- API-Details finden Sie DescribeLogGroups in der AWS SDK für .NET API-Referenz.

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für .NET

Binden Sie die erforderlichen Dateien ein.

```
using Amazon;
using Amazon.CloudWatchLogs;
```

```
using Amazon.CloudWatchLogs.Model;
```

Starten Sie die Live-Tail-Sitzung.

```
            var client = new AmazonCloudWatchLogsClient();
            var request = new StartLiveTailRequest
            {
                LogGroupIdentifiers = logGroupIdentifiers,
                LogStreamNames = logStreamNames,
                LogEventFilterPattern = filterPattern,
            };

            var response = await client.StartLiveTailAsync(request);

            // Catch if request fails
            if (response.HttpStatusCode != System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Failed to start live tail session");
                return;
            }
```

Sie können die Ereignisse aus der Live Tail-Sitzung auf zwei Arten handhaben:

```
            /* Method 1
            * 1). Asynchronously loop through the event stream
            * 2). Set a timer to dispose the stream and stop the Live Tail session
 at the end.
            */
            var eventStream = response.ResponseStream;
            var task = Task.Run(() =>
            {
                foreach (var item in eventStream)
                {
                    if (item is LiveTailSessionUpdate liveTailSessionUpdate)
                    {
                        foreach (var sessionResult in
 liveTailSessionUpdate.SessionResults)
                        {
                            Console.WriteLine("Message : {0}",
 sessionResult.Message);
                        }
```

```
                    }
                    if (item is LiveTailSessionStart)
                    {
                        Console.WriteLine("Live Tail session started");
                    }
                    // On-stream exceptions are processed here
                    if (item is CloudWatchLogsEventStreamException)
                    {
                        Console.WriteLine($"ERROR: {item}");
                    }
                }
            });
            // Close the stream to stop the session after a timeout
            if (!task.Wait(TimeSpan.FromSeconds(10))){
                eventStream.Dispose();
                Console.WriteLine("End of line");
            }
```

```
            /* Method 2
            * 1). Add event handlers to each event variable
            * 2). Start processing the stream and wait for a timeout using
AutoResetEvent
            */
            AutoResetEvent endEvent = new AutoResetEvent(false);
            var eventStream = response.ResponseStream;
            using (eventStream) // automatically disposes the stream to stop the
session after execution finishes
            {
                eventStream.SessionStartReceived += (sender, e) =>
                {
                    Console.WriteLine("LiveTail session started");
                };
                eventStream.SessionUpdateReceived += (sender, e) =>
                {
                    foreach (LiveTailSessionLogEvent logEvent in
e.EventStreamEvent.SessionResults){
                        Console.WriteLine("Message: {0}", logEvent.Message);
                    }
                };
                // On-stream exceptions are captured here
                eventStream.ExceptionReceived += (sender, e) =>
                {
```

```
                    Console.WriteLine($"ERROR: {e.EventStreamException.Message}");
                };

                eventStream.StartProcessing();
                // Stream events for this amount of time.
                endEvent.WaitOne(TimeSpan.FromSeconds(10));
                Console.WriteLine("End of line");
            }
```

- API-Details finden Sie StartLiveTail in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with administrator
access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- API-Details finden Sie AdminGetUser in der AWS SDK für .NET API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK für .NET API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
```

```
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK für .NET API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");
```

```
        return tokenResponse.Session;
    }
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK für .NET API-Referenz.

`ConfirmDevice`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmDevice`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string deviceKey,
string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- API-Details finden Sie ConfirmDevice in der AWS SDK für .NET API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code, string
userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK für .NET API-Referenz.

`InitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`InitiateAuth`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- API-Details finden Sie InitiateAuth in der AWS SDK für .NET API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();

    var usersPaginator = _cognitoService.Paginators.ListUsers(request);
    await foreach (var response in usersPaginator.Responses)
    {
        users.AddRange(response.Users);
    }

    return users;
}
```

- API-Details finden Sie ListUsers in der AWS SDK für .NET API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK für .NET API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
```

```
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie SignUp in der AWS SDK für .NET API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK für .NET API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
```

```
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address: {email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }
```

```
        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator: ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google Authenticator:
 ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}
```

```
using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
```

```
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new AdminRespondToAuthChallengeRequest
        {
```

```
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
```

```
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AassociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
```

```
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code, string
 userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };
```

```
        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string deviceKey,
 string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
```

```
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with administrator
access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
```

```
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Amazon Comprehend Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DetectDominantLanguage`

Das folgende Codebeispiel zeigt die Verwendung. `DetectDominantLanguage`

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example calls the Amazon Comprehend service to determine the
/// dominant language.
/// </summary>
public static class DetectDominantLanguage
{
    /// <summary>
    /// Calls Amazon Comprehend to determine the dominant language used in
    /// the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle.";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        Console.WriteLine("Calling DetectDominantLanguage\n");
        var detectDominantLanguageRequest = new DetectDominantLanguageRequest()
        {
            Text = text,
        };

        var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
        foreach (var dl in detectDominantLanguageResponse.Languages)
        {
            Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
{dl.Score}");
```

```
            }

            Console.WriteLine("Done");
        }
    }
```

- API-Details finden Sie DetectDominantLanguage in der AWS SDK für .NET API-Referenz.

`DetectEntities`

Das folgende Codebeispiel zeigt die Verwendung`DetectEntities`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;

    /// <summary>
    /// This example shows how to use the AmazonComprehend service detect any
    /// entities in submitted text.
    /// </summary>
    public static class DetectEntities
    {
        /// <summary>
        /// The main method calls the DetectEntitiesAsync method to find any
        /// entities in the sample code.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";
```

```
            var comprehendClient = new AmazonComprehendClient();

            Console.WriteLine("Calling DetectEntities\n");
            var detectEntitiesRequest = new DetectEntitiesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

            foreach (var e in detectEntitiesResponse.Entities)
            {
                Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
{e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API-Details finden Sie DetectEntities in der AWS SDK für .NET API-Referenz.

`DetectKeyPhrases`

Das folgende Codebeispiel zeigt die Verwendung`DetectKeyPhrases`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;
```

```
    /// <summary>
    /// This example shows how to use the Amazon Comprehend service to
    /// search text for key phrases.
    /// </summary>
    public static class DetectKeyPhrase
    {
        /// <summary>
        /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
        /// to detect any key phrases in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectKeyPhrases");
            var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
            foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
            {
                Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score}, BeginOffset:
{kp.BeginOffset}, EndOffset: {kp.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API-Details finden Sie DetectKeyPhrases in der AWS SDK für .NET API-Referenz.

`DetectPiiEntities`

Das folgende Codebeispiel zeigt die Verwendung`DetectPiiEntities`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method.
/// </summary>
public class DetectingPII
{
    /// <summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally dientifiable information within the supplied text.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();
        var text = @"Hello Paul Santos. The latest statement for your
                    credit card account 1111-0000-1111-0000 was
                    mailed to 123 Any Street, Seattle, WA 98109.";

        var request = new DetectPiiEntitiesRequest
        {
            Text = text,
            LanguageCode = "EN",
        };

        var response = await comprehendClient.DetectPiiEntitiesAsync(request);

        if (response.Entities.Count > 0)
        {
            foreach (var entity in response.Entities)
```

```
                {
                    var entityValue = text.Substring(entity.BeginOffset,
entity.EndOffset - entity.BeginOffset);
                    Console.WriteLine($"{entity.Type}: {entityValue}");
                }
            }
        }
    }
```

- API-Details finden Sie DetectPiiEntities in der AWS SDK für .NET API-Referenz.

`DetectSentiment`

Das folgende Codebeispiel zeigt die Verwendung`DetectSentiment`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
    /// This method calls the DetetectSentimentAsync method to analyze the
    /// supplied text and determine the overal sentiment.
    /// </summary>
    public static async Task Main()
```

```
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

            // Call DetectKeyPhrases API
            Console.WriteLine("Calling DetectSentiment");
            var detectSentimentRequest = new DetectSentimentRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
            Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
            Console.WriteLine("Done");
        }
    }
```

- API-Details finden Sie DetectSentiment in der AWS SDK für .NET API-Referenz.

`DetectSyntax`

Das folgende Codebeispiel zeigt die Verwendung`DetectSyntax`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Comprehend;
    using Amazon.Comprehend.Model;
```

```
    /// <summary>
    /// This example shows how to use Amazon Comprehend to detect syntax
    /// elements by calling the DetectSyntaxAsync method.
    /// </summary>
    public class DetectingSyntax
    {
        /// <summary>
        /// This method calls DetectSynaxAsync to identify the syntax elements
        /// in the sample text.
        /// </summary>
        public static async Task Main()
        {
            string text = "It is raining today in Seattle";

            var comprehendClient = new AmazonComprehendClient();

            // Call DetectSyntax API
            Console.WriteLine("Calling DetectSyntaxAsync\n");
            var detectSyntaxRequest = new DetectSyntaxRequest()
            {
                Text = text,
                LanguageCode = "en",
            };
            DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
            foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
            {
                Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
{s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
            }

            Console.WriteLine("Done");
        }
    }
```

- API-Details finden Sie DetectSyntax in der AWS SDK für .NET API-Referenz.

StartTopicsDetectionJob

Das folgende Codebeispiel zeigt die VerwendungStartTopicsDetectionJob.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NEt
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
    /// <summary>
    /// This methos calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();

        string inputS3Uri = "s3://input bucket/input path";
        InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
        string outputS3Uri = "s3://output bucket/output path";
        string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
        int numberOfTopics = 10;

        var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
        {
            InputDataConfig = new InputDataConfig()
            {
```

```
                    S3Uri = inputS3Uri,
                    InputFormat = inputDocFormat,
                },
                OutputDataConfig = new OutputDataConfig()
                {
                    S3Uri = outputS3Uri,
                },
                DataAccessRoleArn = dataAccessRoleArn,
                NumberOfTopics = numberOfTopics,
            };

            var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

            var jobId = startTopicsDetectionJobResponse.JobId;
            Console.WriteLine("JobId: " + jobId);

            var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
            {
                JobId = jobId,
            };

            var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

            var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
            foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
            {
                PrintJobProperties(props);
            }
        }

        /// <summary>
        /// This method is a helper method that displays the job properties
        /// from the call to StartTopicsDetectionJobRequest.
        /// </summary>
        /// <param name="props">A list of properties from the call to
        /// StartTopicsDetectionJobRequest.</param>
```

```
        private static void PrintJobProperties(TopicsDetectionJobProperties props)
        {
            Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
            Console.WriteLine($"NumberOfTopics: {props.NumberOfTopics}\nInputS3Uri:
{props.InputDataConfig.S3Uri}");
            Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
        }
    }
```

- API-Details finden Sie StartTopicsDetectionJob in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon DocumentDB unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon-DocumentDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
```

```
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted into
 a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }

     private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
 ILambdaContext context)
    {

        var eventData = record.Event;
        var operationType = eventData.OperationType;
        var databaseName = eventData.Ns.Db;
        var collectionName = eventData.Ns.Coll;
        var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
 JsonSerializerOptions { WriteIndented = true });

        context.Logger.LogLine($"Operation type: {operationType}");
        context.Logger.LogLine($"Database: {databaseName}");
        context.Logger.LogLine($"Collection: {collectionName}");
        context.Logger.LogLine($"Full document:\n{fullDocument}");
```

```
    }



    public class Event
    {
        [JsonPropertyName("eventSourceArn")]
        public string EventSourceArn { get; set; }

        [JsonPropertyName("events")]
        public List<DocumentDBEventRecord> Events { get; set; }

        [JsonPropertyName("eventSource")]
        public string EventSource { get; set; }
    }

    public class DocumentDBEventRecord
    {
        [JsonPropertyName("event")]
        public EventData Event { get; set; }
    }

    public class EventData
    {
        [JsonPropertyName("_id")]
        public IdData Id { get; set; }

        [JsonPropertyName("clusterTime")]
        public ClusterTime ClusterTime { get; set; }

        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
```

```
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

DynamoDB-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden. AWS Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
```

```
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);
```

```
            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
= ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
```

```
                    },
                },

                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {
                        Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
        /// <summary>
```

```
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
 AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
```

```
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
```

```
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für .NET API-Referenz.

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;

namespace LowLevelBatchGet
{
    public class LowLevelBatchGet
    {
        private static readonly string _table1Name = "Forum";
        private static readonly string _table2Name = "Thread";

        public static async void RetrieveMultipleItemsBatchGet(AmazonDynamoDBClient
 client)
        {
            var request = new BatchGetItemRequest
            {
                RequestItems = new Dictionary<string, KeysAndAttributes>()
            {
                { _table1Name,
                  new KeysAndAttributes
                  {
                      Keys = new List<Dictionary<string, AttributeValue> >()
                      {
                          new Dictionary<string, AttributeValue>()
                          {
                              { "Name", new AttributeValue {
                            S = "Amazon DynamoDB"
                        } }
                          },
                          new Dictionary<string, AttributeValue>()
                          {
                              { "Name", new AttributeValue {
                            S = "Amazon S3"
```

```
                        } }
                          }
                      }
                }},
            {
                _table2Name,
                new KeysAndAttributes
                {
                    Keys = new List<Dictionary<string, AttributeValue> >()
                    {
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon DynamoDB"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "DynamoDB Thread 1"
                              } }
                        },
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon DynamoDB"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "DynamoDB Thread 2"
                              } }
                        },
                        new Dictionary<string, AttributeValue>()
                        {
                            { "ForumName", new AttributeValue {
                                  S = "Amazon S3"
                              } },
                            { "Subject", new AttributeValue {
                                  S = "S3 Thread 1"
                              } }
                        }
                    }
                }
            }
        }
        };

        BatchGetItemResponse response;
```

```
            do
            {
                Console.WriteLine("Making request");
                response = await client.BatchGetItemAsync(request);

                // Check the response.
                var responses = response.Responses; // Attribute list in the
response.

                foreach (var tableResponse in responses)
                {
                    var tableResults = tableResponse.Value;
                    Console.WriteLine("Items retrieved from table {0}",
tableResponse.Key);
                    foreach (var item1 in tableResults)
                    {
                        PrintItem(item1);
                    }
                }

                // Any unprocessed keys? could happen if you exceed
ProvisionedThroughput or some other error.
                Dictionary<string, KeysAndAttributes> unprocessedKeys =
response.UnprocessedKeys;
                foreach (var unprocessedTableKeys in unprocessedKeys)
                {
                    // Print table name.
                    Console.WriteLine(unprocessedTableKeys.Key);
                    // Print unprocessed primary keys.
                    foreach (var key in unprocessedTableKeys.Value.Keys)
                    {
                        PrintItem(key);
                    }
                }

                request.RequestItems = unprocessedKeys;
            } while (response.UnprocessedKeys.Count > 0);
        }

        private static void PrintItem(Dictionary<string, AttributeValue>
attributeList)
        {
            foreach (KeyValuePair<string, AttributeValue> kvp in attributeList)
            {
```

```
                string attributeName = kvp.Key;
                AttributeValue value = kvp.Value;

                Console.WriteLine(
                    attributeName + " " +
                    (value.S == null ? "" : "S=[" + value.S + "]") +
                    (value.N == null ? "" : "N=[" + value.N + "]") +
                    (value.SS == null ? "" : "SS=[" + string.Join(",",
 value.SS.ToArray()) + "]") +
                    (value.NS == null ? "" : "NS=[" + string.Join(",",
 value.NS.ToArray()) + "]")
                    );
            }
            Console.WriteLine("************************************************");
        }

        static void Main()
        {
            var client = new AmazonDynamoDBClient();

            RetrieveMultipleItemsBatchGet(client);
        }
    }
}
```

- API-Details finden Sie BatchGetItem in der AWS SDK für .NET API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    private static async Task GetTableInformation()
    {
```

```
        Console.WriteLine("\n*** Retrieving table information ***");

        var response = await Client.DescribeTableAsync(new DescribeTableRequest
        {
            TableName = ExampleTableName
        });

        var table = response.Table;
        Console.WriteLine($"Name: {table.TableName}");
        Console.WriteLine($"# of items: {table.ItemCount}");

    }
```

- API-Details finden Sie DescribeTable in der AWS SDK für .NET API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
```

```
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
```

```
                Parameters = parameters,
            });

            return response.Items;
        }
```

Verwenden einer SELECT-Anweisung, um ein Liste an Elementen abzurufen.

```
        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
```

```
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
 producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
 = ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
 ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

Verwenden einer DELETE-Anweisung, um einen einzelnen Film zu löschen.

```
        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
 movieTitle, int year)
```

```
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für .NET API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    private static async Task ListMyTables()
    {
        Console.WriteLine("\n*** Listing tables ***");

        string lastTableNameEvaluated = null;
        do
        {
            var response = await Client.ListTablesAsync(new ListTablesRequest
```

```
            {
                Limit = 2,
                ExclusiveStartTableName = lastTableNameEvaluated
            });

            foreach (var name in response.TableNames)
            {
                Console.WriteLine(name);
            }

            lastTableNameEvaluated = response.LastEvaluatedTableName;
        } while (lastTableNameEvaluated != null);
    }
```

- API-Details finden Sie ListTables in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie man die Amazon-DynamoDB-.NET-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
const string movieFileName = @"..\..\..\..\..\..\..\..\resources\sample_files
\movies.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
{
    Console.WriteLine($"Successfully created table: {tableName}.");
}

WaitForEnter();

// Add movie information to the table from moviedata.json. See the
// instructions at the top of this file to download the JSON file.
Console.WriteLine($"Inserting movies into the new table. Please wait...");
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);
if (success)
{
    Console.WriteLine("Movies successfully added to the table.");
}
else
{
    Console.WriteLine("Movies could not be added to the table.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var title1 = "Star Wars";
var year1 = 1977;
```

```
var title2 = "Wizard of Oz";
var year2 = 1939;

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1,
 year2);
if (success)
{
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");
}
else
{
    Console.WriteLine("Select statement failed.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var producer1 = "LucasFilm";
var producer2 = "MGM";

Console.WriteLine($"Updating two movies with producer information: {title1} and
 {title2}.");
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1, year1,
 producer2, title2, year2);
if (success)
{
    Console.WriteLine($"Successfully updated {title1} and {title2}.");
}
else
{
    Console.WriteLine("Update failed.");
}

WaitForEnter();

// Delete multiple movies by using the BatchExecute statement.
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2,
 year2);

if (success)
{
```

```
    Console.WriteLine($"Deleted {title1} and {title2}");
}
else
{
    Console.WriteLine($"could not delete {title1} or {title2}");
}

WaitForEnter();

// DNow that the PartiQL Batch scenario is complete, delete the movie table.
success = await DynamoDBMethods.DeleteTableAsync(tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
void DisplayInstructions()
{
    Console.Clear();
    Console.WriteLine();
    Console.Write(new string(' ', 24));
    Console.WriteLine("DynamoDB PartiQL Basics Example");
    Console.WriteLine(SepBar);
    Console.WriteLine("This demo application shows the basics of using Amazon
 DynamoDB with the AWS SDK for");
    Console.WriteLine(".NET version 3.7 and .NET 6.");
    Console.WriteLine(SepBar);
    Console.WriteLine("Creates a table by using the CreateTable method.");
    Console.WriteLine("Gets multiple movies by using a PartiQL SELECT statement.");
    Console.WriteLine("Updates multiple movies by using the ExecuteBatch method.");
    Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE
 statement.");
    Console.WriteLine("Cleans up the resources created for the demo by deleting the
 table.");
    Console.WriteLine(SepBar);
```

```
    WaitForEnter();
}

/// <summary>
/// Simple method to wait for the <Enter> key to be pressed.
/// </summary>
void WaitForEnter()
{
    Console.WriteLine("\nPress <Enter> to continue.");
    Console.Write(SepBar);
    _ = Console.ReadLine();
}


        /// <summary>
        /// Gets movies from the movie table by
        /// using an Amazon DynamoDB PartiQL SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year1">The year of the first movie.</param>
        /// <param name="year2">The year of the second movie.</param>
        /// <returns>True if successful.</returns>
        public static async Task<bool> GetBatch(
            string tableName,
            string title1,
            string title2,
            int year1,
            int year2)
        {
            var getBatch = $"SELECT * FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },
```

```
                new BatchStatementRequest
                {
                    Statement = getBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            if (response.Responses.Count > 0)
            {
                response.Responses.ForEach(r =>
                {
                    if (r.Item.Any())
                    {
                        Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
                    }
                });
                return true;
            }
            else
            {
                Console.WriteLine($"Couldn't find either {title1} or {title2}.");
                return false;
            }

        }

        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table into which the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
```

```
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";
                var statements = new List<BatchStatementRequest>();

                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });
```

```
                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }

        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
```

```
            }
        }

        /// <summary>
        /// Updates information for multiple movies.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// movies to be updated.</param>
        /// <param name="producer1">The producer name for the first movie
        /// to update.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year that the first movie was released.</param>
        /// <param name="producer2">The producer name for the second
        /// movie to update.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year that the second movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the update.</
returns>
        public static async Task<bool> UpdateBatch(
            string tableName,
            string producer1,
            string title1,
            int year1,
            string producer2,
            string title2,
            int year2)
        {

            string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
 AND year = ?";
            var statements = new List<BatchStatementRequest>
            {
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer1 },
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
```

```
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = producer2 },
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
 BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }

        /// <summary>
        /// Deletes multiple movies using a PartiQL BatchExecuteAsync
        /// statement.
        /// </summary>
        /// <param name="tableName">The name of the table containing the
        /// moves that will be deleted.</param>
        /// <param name="title1">The title of the first movie.</param>
        /// <param name="year1">The year the first movie was released.</param>
        /// <param name="title2">The title of the second movie.</param>
        /// <param name="year2">The year the second movie was released.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeleteBatch(
            string tableName,
            string title1,
            int year1,
            string title2,
            int year2)
        {

            string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
 = ?";
            var statements = new List<BatchStatementRequest>
            {
```

```
                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title1 },
                        new AttributeValue { N = year1.ToString() },
                    },
                },

                new BatchStatementRequest
                {
                    Statement = updateBatch,
                    Parameters = new List<AttributeValue>
                    {
                        new AttributeValue { S = title2 },
                        new AttributeValue { N = year2.ToString() },
                    },
                }
            };

            var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
            {
                Statements = statements,
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für .NET API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
        private static readonly AmazonDynamoDBClient Client = new
 AmazonDynamoDBClient();


        /// <summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table where the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
 movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
                string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
 'year': ?}}";
                var statements = new List<BatchStatementRequest>();
```

```
                try
                {
                    for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                    {
                        for (var i = indexOffset; i < indexOffset + 25; i++)
                        {
                            statements.Add(new BatchStatementRequest
                            {
                                Statement = insertBatch,
                                Parameters = new List<AttributeValue>
                                {
                                    new AttributeValue { S = movies[i].Title },
                                    new AttributeValue { N =
movies[i].Year.ToString() },
                                },
                            });
                        }

                        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                        {
                            Statements = statements,
                        });

                        // Wait between batches for movies to be successfully added.
                        System.Threading.Thread.Sleep(3000);

                        success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                        // Clear the list of statements for the next batch.
                        statements.Clear();
                    }
                }
                catch (AmazonDynamoDBException ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            return success;
        }
```

```
        /// <summary>
        /// Loads the contents of a JSON file into a list of movies to be
        /// added to the DynamoDB table.
        /// </summary>
        /// <param name="movieFileName">The full path to the JSON file.</param>
        /// <returns>A generic list of movie objects.</returns>
        public static List<Movie> ImportMovies(string movieFileName)
        {
            if (!File.Exists(movieFileName))
            {
                return null!;
            }

            using var sr = new StreamReader(movieFileName);
            string json = sr.ReadToEnd();
            var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

            if (allMovies is not null)
            {
                // Return the first 250 entries.
                return allMovies.GetRange(0, 250);
            }
            else
            {
                return null!;
            }
        }



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
```

```
                new AttributeValue { S = movieTitle },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Retrieve multiple movies by year using a SELECT statement.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="year">The year the movies were released.</param>
        /// <returns></returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { N = year.ToString() },
            };

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }


        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
```

```
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";
```

```
            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

```
        /// <summary>
        /// Displays the list of movies returned from a database query.
        /// </summary>
        /// <param name="items">The list of movie information to display.</param>
        private static void DisplayMovies(List<Dictionary<string, AttributeValue>>
 items)
        {
            if (items.Count > 0)
            {
                Console.WriteLine($"Found {items.Count} movies.");
                items.ForEach(item =>
 Console.WriteLine($"{item["year"].N}\t{item["title"].S}"));
            }
            else
            {
                Console.WriteLine($"Didn't find a movie that matched the supplied
 criteria.");
            }
        }


    }
}



        /// <summary>
        /// Uses a PartiQL SELECT statement to retrieve a single movie from the
        /// movie database.
        /// </summary>
        /// <param name="tableName">The name of the movie table.</param>
        /// <param name="movieTitle">The title of the movie to retrieve.</param>
        /// <returns>A list of movie data. If no movie matches the supplied
        /// title, the list is empty.</returns>
        public static async Task<List<Dictionary<string, AttributeValue>>>
 GetSingleMovie(string tableName, string movieTitle)
        {
            string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
            var parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
            };
```

```
            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = selectSingle,
                Parameters = parameters,
            });

            return response.Items;
        }



        /// <summary>
        /// Inserts a single movie into the movies table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to insert.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the INSERT operation.</returns>
        public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
        {
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,
'year': ?}}";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
```

```
        /// Updates a single movie in the table, adding information for the
        /// producer.
        /// </summary>
        /// <param name="tableName">the name of the table.</param>
        /// <param name="producer">The name of the producer.</param>
        /// <param name="movieTitle">The movie title.</param>
        /// <param name="year">The year the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// UPDATE operation.</returns>
        public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
        {
            string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = insertSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = producer },
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }



        /// <summary>
        /// Deletes a single movie from the table.
        /// </summary>
        /// <param name="tableName">The name of the table.</param>
        /// <param name="movieTitle">The title of the movie to delete.</param>
        /// <param name="year">The year that the movie was released.</param>
        /// <returns>A Boolean value that indicates the success of the
        /// DELETE operation.</returns>
        public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
        {
```

```
            var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

            var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
            {
                Statement = deleteSingle,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = movieTitle },
                    new AttributeValue { N = year.ToString() },
                },
            });

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für .NET API-Referenz.

Verwenden eines Dokumentmodells

Das folgende Codebeispiel zeigt, wie CRUD (Create, Read, Update, Delete) und Batch-Operationen mithilfe eines Dokumentmodells für DynamoDB und eines SDK ausgeführt werden. AWS

Weitere Informationen finden Sie unter Dokumentmodell.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie CRUD-Operationen unter Verwendung eines Dokumentmodells durch.

```
    /// <summary>
    /// Performs CRUD operations on an Amazon DynamoDB table.
    /// </summary>
```

```
    public class MidlevelItemCRUD
    {
        public static async Task Main()
        {
            var tableName = "ProductCatalog";
            var sampleBookId = 555;

            var client = new AmazonDynamoDBClient();
            var productCatalog = LoadTable(client, tableName);

            await CreateBookItem(productCatalog, sampleBookId);
            RetrieveBook(productCatalog, sampleBookId);

            // Couple of sample updates.
            UpdateMultipleAttributes(productCatalog, sampleBookId);
            UpdateBookPriceConditionally(productCatalog, sampleBookId);

            // Delete.
            await DeleteBook(productCatalog, sampleBookId);
        }

        /// <summary>
        /// Loads the contents of a DynamoDB table.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        /// <param name="tableName">The name of the table to load.</param>
        /// <returns>A DynamoDB table object.</returns>
        public static Table LoadTable(IAmazonDynamoDB client, string tableName)
        {
            Table productCatalog = Table.LoadTable(client, tableName);
            return productCatalog;
        }

        /// <summary>
        /// Creates an example book item and adds it to the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async Task CreateBookItem(Table productCatalog, int
sampleBookId)
        {
            Console.WriteLine("\n*** Executing CreateBookItem() ***");
```

```
            var book = new Document
            {
                ["Id"] = sampleBookId,
                ["Title"] = "Book " + sampleBookId,
                ["Price"] = 19.99,
                ["ISBN"] = "111-1111111111",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["PageCount"] = 500,
                ["Dimensions"] = "8.5x11x.5",
                ["InPublication"] = new DynamoDBBool(true),
                ["InStock"] = new DynamoDBBool(false),
                ["QuantityOnHand"] = 0,
            };

            // Adds the book to the ProductCatalog table.
            await productCatalog.PutItemAsync(book);
        }

        /// <summary>
        /// Retrieves an item, a book, from the DynamoDB ProductCatalog table.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void RetrieveBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing RetrieveBook() ***");

            // Optional configuration.
            var config = new GetItemOperationConfig
            {
                AttributesToGet = new List<string> { "Id", "ISBN", "Title",
"Authors", "Price" },
                ConsistentRead = true,
            };

            Document document = await productCatalog.GetItemAsync(sampleBookId,
config);
            Console.WriteLine("RetrieveBook: Printing book retrieved...");
            PrintDocument(document);
        }
```

```
        /// <summary>
        /// Updates multiple attributes for a book and writes the changes to the
        /// DynamoDB table ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateMultipleAttributes(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\nUpdating multiple attributes....");
            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,

                // List of attribute updates.
                // The following replaces the existing authors list.
                ["Authors"] = new List<string> { "Author x", "Author y" },
                ["newAttribute"] = "New Value",
                ["ISBN"] = null, // Remove it.
            };

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                // Gets updated item in response.
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateMultipleAttributes: Printing item after
updates ...");
            PrintDocument(updatedBook);
        }

        /// <summary>
        /// Updates a book item if it meets the specified criteria.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
```

```
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
        public static async void UpdateBookPriceConditionally(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing UpdateBookPriceConditionally() ***");

            int partitionKey = sampleBookId;

            var book = new Document
            {
                ["Id"] = partitionKey,
                ["Price"] = 29.99,
            };

            // For conditional price update, creating a condition expression.
            var expr = new Expression
            {
                ExpressionStatement = "Price = :val",
            };
            expr.ExpressionAttributeValues[":val"] = 19.00;

            // Optional parameters.
            var config = new UpdateItemOperationConfig
            {
                ConditionalExpression = expr,
                ReturnValues = ReturnValues.AllNewAttributes,
            };

            Document updatedBook = await productCatalog.UpdateItemAsync(book,
config);
            Console.WriteLine("UpdateBookPriceConditionally: Printing item whose
price was conditionally updated");
            PrintDocument(updatedBook);
        }

        /// <summary>
        /// Deletes the book with the supplied Id value from the DynamoDB table
        /// ProductCatalog.
        /// </summary>
        /// <param name="productCatalog">A DynamoDB table object.</param>
        /// <param name="sampleBookId">An integer value representing the book's
ID.</param>
```

```
        public static async Task DeleteBook(
          Table productCatalog,
          int sampleBookId)
        {
            Console.WriteLine("\n*** Executing DeleteBook() ***");

            // Optional configuration.
            var config = new DeleteItemOperationConfig
            {
                // Returns the deleted item.
                ReturnValues = ReturnValues.AllOldAttributes,
            };
            Document document = await productCatalog.DeleteItemAsync(sampleBookId,
config);
            Console.WriteLine("DeleteBook: Printing deleted just deleted...");

            PrintDocument(document);
        }

        /// <summary>
        /// Prints the information for the supplied DynamoDB document.
        /// </summary>
        /// <param name="updatedDocument">A DynamoDB document object.</param>
        public static void PrintDocument(Document updatedDocument)
        {
            if (updatedDocument is null)
            {
                return;
            }

            foreach (var attribute in updatedDocument.GetAttributeNames())
            {
                string stringValue = null;
                var value = updatedDocument[attribute];

                if (value is null)
                {
                    continue;
                }

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
```

```
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                  select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}", attribute,
stringValue);
            }
        }
    }
```

Führen Sie Batch-Schreibvorgänge unter Verwendung eines Dokumentmodells durch.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to perform batch
    /// operations.
    /// </summary>
    public class MidLevelBatchWriteItem
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            await SingleTableBatchWrite(client);
            await MultiTableBatchWrite(client);
        }

        /// <summary>
        /// Perform a batch operation on a single DynamoDB table.
        /// </summary>
        /// <param name="client">An initialized DynamoDB object.</param>
        public static async Task SingleTableBatchWrite(IAmazonDynamoDB client)
        {
            Table productCatalog = Table.LoadTable(client, "ProductCatalog");
            var batchWrite = productCatalog.CreateBatchWrite();

            var book1 = new Document
```

```
            {
                ["Id"] = 902,
                ["Title"] = "My book1 in batch write using .NET helper classes",
                ["ISBN"] = "902-11-11-1111",
                ["Price"] = 10,
                ["ProductCategory"] = "Book",
                ["Authors"] = new List<string> { "Author 1", "Author 2", "Author
3" },
                ["Dimensions"] = "8.5x11x.5",
                ["InStock"] = new DynamoDBBool(true),
                ["QuantityOnHand"] = new DynamoDBNull(), // Quantity is unknown at
this time.
            };

            batchWrite.AddDocumentToPut(book1);

            // Specify delete item using overload that takes PK.
            batchWrite.AddKeyToDelete(12345);
            Console.WriteLine("Performing batch write in SingleTableBatchWrite()");
            await batchWrite.ExecuteAsync();
        }

        /// <summary>
        /// Perform a batch operation involving multiple DynamoDB tables.
        /// </summary>
        /// <param name="client">An initialized DynamoDB client object.</param>
        public static async Task MultiTableBatchWrite(IAmazonDynamoDB client)
        {
            // Specify item to add in the Forum table.
            Table forum = Table.LoadTable(client, "Forum");
            var forumBatchWrite = forum.CreateBatchWrite();

            var forum1 = new Document
            {
                ["Name"] = "Test BatchWrite Forum",
                ["Threads"] = 0,
            };
            forumBatchWrite.AddDocumentToPut(forum1);

            // Specify item to add in the Thread table.
            Table thread = Table.LoadTable(client, "Thread");
            var threadBatchWrite = thread.CreateBatchWrite();

            var thread1 = new Document
```

```
            {
                ["ForumName"] = "S3 forum",
                ["Subject"] = "My sample question",
                ["Message"] = "Message text",
                ["KeywordTags"] = new List<string> { "S3", "Bucket" },
            };
            threadBatchWrite.AddDocumentToPut(thread1);

            // Specify item to delete from the Thread table.
            threadBatchWrite.AddKeyToDelete("someForumName", "someSubject");

            // Create multi-table batch.
            var superBatch = new MultiTableDocumentBatchWrite();
            superBatch.AddBatch(forumBatchWrite);
            superBatch.AddBatch(threadBatchWrite);
            Console.WriteLine("Performing batch write in MultiTableBatchWrite()");

            // Execute the batch.
            await superBatch.ExecuteAsync();
        }
    }
```

Scannen Sie eine Tabelle unter Verwendung eines Dokumentmodells.

```
    /// <summary>
    /// Shows how to use mid-level Amazon DynamoDB API calls to scan a DynamoDB
    /// table for values.
    /// </summary>
    public class MidLevelScanOnly
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");

            await FindProductsWithNegativePrice(productCatalogTable);
            await FindProductsWithNegativePriceWithConfig(productCatalogTable);
        }
```

```
        /// <summary>
        /// Retrieves any products that have a negative price in a DynamoDB table.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePrice(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced < 0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            Search search = productCatalogTable.Scan(scanFilter);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePrice:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Finds any items in the ProductCatalog table using a DynamoDB
        /// configuration object.
        /// </summary>
        /// <param name="productCatalogTable">A DynamoDB table object.</param>
        public static async Task FindProductsWithNegativePriceWithConfig(
          Table productCatalogTable)
        {
            // Assume there is a price error. So we scan to find items priced < 0.
            var scanFilter = new ScanFilter();
            scanFilter.AddCondition("Price", ScanOperator.LessThan, 0);

            var config = new ScanOperationConfig()
            {
                Filter = scanFilter,
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string> { "Title", "Id" },
```

```
            };

            Search search = productCatalogTable.Scan(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindProductsWithNegativePriceWithConfig:
printing ............");

                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the details of the passed DynamoDB document object on the
        /// console.
        /// </summary>
        /// <param name="document">A DynamoDB document object.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];
                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
```

```
        }
    }
```

Verwenden Sie ein Dokumentmodell, um eine Tabelle abzufragen und zu scannen.

```
    /// <summary>
    /// Shows how to perform mid-level query procedures on an Amazon DynamoDB
    /// table.
    /// </summary>
    public class MidLevelQueryAndScan
    {
        public static async Task Main()
        {
            IAmazonDynamoDB client = new AmazonDynamoDBClient();

            // Query examples.
            Table replyTable = Table.LoadTable(client, "Reply");
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 2";

            await FindRepliesInLast15Days(replyTable);
            await FindRepliesInLast15DaysWithConfig(replyTable, forumName,
threadSubject);
            await FindRepliesPostedWithinTimePeriod(replyTable, forumName,
threadSubject);

            // Get Example.
            Table productCatalogTable = Table.LoadTable(client, "ProductCatalog");
            int productId = 101;

            await GetProduct(productCatalogTable, productId);
        }

        /// <summary>
        /// Retrieves information about a product from the DynamoDB table
        /// ProductCatalog based on the product ID and displays the information
        /// on the console.
        /// </summary>
        /// <param name="tableName">The name of the table from which to retrieve
        /// product information.</param>
```

```
        /// <param name="productId">The ID of the product to retrieve.</param>
        public static async Task GetProduct(Table tableName, int productId)
        {
            Console.WriteLine("*** Executing GetProduct() ***");
            Document productDocument = await tableName.GetItemAsync(productId);
            if (productDocument != null)
            {
                PrintDocument(productDocument);
            }
            else
            {
                Console.WriteLine("Error: product " + productId + " does not
exist");
            }
        }

        /// <summary>
        /// Retrieves replies from the passed DynamoDB table object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        public static async Task FindRepliesInLast15Days(
          Table table)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, "Id");
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            // Use Query overloads that take the minimum required query parameters.
            Search search = table.Query(filter);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15Days:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }
```

```
        /// <summary>
        /// Retrieve replies made during a specific time period.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadSubject">The subject of the thread, which we are
        /// searching for replies.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          Table table,
          string forumName,
          string threadSubject)
        {
            DateTime startDate = DateTime.UtcNow.Subtract(new TimeSpan(21, 0, 0,
0));
            DateTime endDate = DateTime.UtcNow.Subtract(new TimeSpan(1, 0, 0, 0));

            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadSubject);
            filter.AddCondition("ReplyDateTime", QueryOperator.Between, startDate,
endDate);

            var config = new QueryOperationConfig()
            {
                Limit = 2, // 2 items/page.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
        {
          "Message",
          "ReplyDateTime",
          "PostedBy",
        },
                ConsistentRead = true,
                Filter = filter,
            };

            Search search = table.Query(config);

            do
            {
                var documentList = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesPostedWithinTimePeriod: printing
replies posted within dates: {0} and {1} ............", startDate, endDate);
```

```
                foreach (var document in documentList)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Perform a query for replies made in the last 15 days using a DynamoDB
        /// QueryOperationConfig object.
        /// </summary>
        /// <param name="table">The table we want to query.</param>
        /// <param name="forumName">The name of the forum that we're interested
in.</param>
        /// <param name="threadName">The bane of the thread that we are searching
        /// for replies.</param>
        public static async Task FindRepliesInLast15DaysWithConfig(
          Table table,
          string forumName,
          string threadName)
        {
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);
            var filter = new QueryFilter("Id", QueryOperator.Equal, forumName + "#"
+ threadName);
            filter.AddCondition("ReplyDateTime", QueryOperator.GreaterThan,
twoWeeksAgoDate);

            var config = new QueryOperationConfig()
            {
                Filter = filter,

                // Optional parameters.
                Select = SelectValues.SpecificAttributes,
                AttributesToGet = new List<string>
                {
                  "Message",
                  "ReplyDateTime",
                  "PostedBy",
                },
                ConsistentRead = true,
            };
```

```
            Search search = table.Query(config);

            do
            {
                var documentSet = await search.GetNextSetAsync();
                Console.WriteLine("\nFindRepliesInLast15DaysWithConfig:
printing ............");

                foreach (var document in documentSet)
                {
                    PrintDocument(document);
                }
            }
            while (!search.IsDone);
        }

        /// <summary>
        /// Displays the contents of the passed DynamoDB document on the console.
        /// </summary>
        /// <param name="document">A DynamoDB document to display.</param>
        public static void PrintDocument(Document document)
        {
            Console.WriteLine();
            foreach (var attribute in document.GetAttributeNames())
            {
                string stringValue = null;
                var value = document[attribute];

                if (value is Primitive)
                {
                    stringValue = value.AsPrimitive().Value.ToString();
                }
                else if (value is PrimitiveList)
                {
                    stringValue = string.Join(",", (from primitive
                      in value.AsPrimitiveList().Entries
                                                    select
primitive.Value).ToArray());
                }

                Console.WriteLine($"{attribute} - {stringValue}");
            }
        }
    }
```

Verwenden eines übergeordneten Object-Persistence-Modells

Das folgende Codebeispiel zeigt, wie CREATE, READ, UPDATE, AND DELETE (CRUD) und Batch-Operationen mithilfe eines Objektpersistenzmodells für DynamoDB und eines SDK ausgeführt werden. AWS

Weitere Informationen finden Sie unter Object-Persistence-Modell.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie CRUD-Operationen unter Verwendung eines übergeordneten Object-Persistence-Modells durch.

```
/// <summary>
/// Shows how to perform high-level CRUD operations on an Amazon DynamoDB
/// table.
/// </summary>
public class HighLevelItemCrud
{
    public static async Task Main()
    {
        var client = new AmazonDynamoDBClient();
        DynamoDBContext context = new DynamoDBContext(client);
        await PerformCRUDOperations(context);
    }

    public static async Task PerformCRUDOperations(IDynamoDBContext context)
    {
        int bookId = 1001; // Some unique value.
        Book myBook = new Book
        {
```

```
                Id = bookId,
                Title = "object persistence-AWS SDK for.NET SDK-Book 1001",
                Isbn = "111-1111111001",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
            };

            // Save the book to the ProductCatalog table.
            await context.SaveAsync(myBook);

            // Retrieve the book from the ProductCatalog table.
            Book bookRetrieved = await context.LoadAsync<Book>(bookId);

            // Update some properties.
            bookRetrieved.Isbn = "222-2222221001";

            // Update existing authors list with the following values.
            bookRetrieved.BookAuthors = new List<string> { " Author 1", "Author
x" };
            await context.SaveAsync(bookRetrieved);

            // Retrieve the updated book. This time, add the optional
            // ConsistentRead parameter using DynamoDBContextConfig object.
            await context.LoadAsync<Book>(bookId, new DynamoDBContextConfig
            {
                ConsistentRead = true,
            });

            // Delete the book.
            await context.DeleteAsync<Book>(bookId);

            // Try to retrieve deleted book. It should return null.
            Book deletedBook = await context.LoadAsync<Book>(bookId, new
DynamoDBContextConfig
            {
                ConsistentRead = true,
            });

            if (deletedBook == null)
            {
                Console.WriteLine("Book is deleted");
            }
        }
    }
```

Führen Sie Batch-Schreibvorgänge unter Verwendung eines übergeordneten Object-Persistence-Modells durch.

```
/// <summary>
/// Performs high-level batch write operations to an Amazon DynamoDB table.
/// This example was written using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class HighLevelBatchWriteItem
{
    public static async Task SingleTableBatchWrite(IDynamoDBContext context)
    {
        Book book1 = new Book
        {
            Id = 902,
            InPublication = true,
            Isbn = "902-11-11-1111",
            PageCount = "100",
            Price = 10,
            ProductCategory = "Book",
            Title = "My book3 in batch write",
        };

        Book book2 = new Book
        {
            Id = 903,
            InPublication = true,
            Isbn = "903-11-11-1111",
            PageCount = "200",
            Price = 10,
            ProductCategory = "Book",
            Title = "My book4 in batch write",
        };

        var bookBatch = context.CreateBatchWrite<Book>();
        bookBatch.AddPutItems(new List<Book> { book1, book2 });

        Console.WriteLine("Adding two books to ProductCatalog table.");
        await bookBatch.ExecuteAsync();
    }
```

```
        public static async Task MultiTableBatchWrite(IDynamoDBContext context)
        {
            // New Forum item.
            Forum newForum = new Forum
            {
                Name = "Test BatchWrite Forum",
                Threads = 0,
            };
            var forumBatch = context.CreateBatchWrite<Forum>();
            forumBatch.AddPutItem(newForum);

            // New Thread item.
            Thread newThread = new Thread
            {
                ForumName = "S3 forum",
                Subject = "My sample question",
                KeywordTags = new List<string> { "S3", "Bucket" },
                Message = "Message text",
            };

            DynamoDBOperationConfig config = new DynamoDBOperationConfig();
            config.SkipVersionCheck = true;
            var threadBatch = context.CreateBatchWrite<Thread>(config);
            threadBatch.AddPutItem(newThread);
            threadBatch.AddDeleteKey("some partition key value", "some sort key
value");

            var superBatch = new MultiTableBatchWrite(forumBatch, threadBatch);

            Console.WriteLine("Performing batch write in MultiTableBatchWrite().");
            await superBatch.ExecuteAsync();
        }

        public static async Task Main()
        {
            AmazonDynamoDBClient client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);

            await SingleTableBatchWrite(context);
            await MultiTableBatchWrite(context);
        }
    }
```

Weisen Sie einer Tabelle unter Verwendung eines übergeordneten Object-Persistence-Modells beliebige Daten zu.

```
    /// <summary>
    /// Shows how to map arbitrary data to an Amazon DynamoDB table.
    /// </summary>
    public class HighLevelMappingArbitraryData
    {
        /// <summary>
        /// Creates a book, adds it to the DynamoDB ProductCatalog table, retrieves
        /// the new book from the table, updates the dimensions and writes the
        /// changed item back to the table.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to write and
        /// read data from the table.</param>
        public static async Task AddRetrieveUpdateBook(IDynamoDBContext context)
        {
            // Create a book.
            DimensionType myBookDimensions = new DimensionType()
            {
                Length = 8M,
                Height = 11M,
                Thickness = 0.5M,
            };

            Book myBook = new Book
            {
                Id = 501,
                Title = "AWS SDK for .NET Object Persistence Model Handling
 Arbitrary Data",
                Isbn = "999-9999999999",
                BookAuthors = new List<string> { "Author 1", "Author 2" },
                Dimensions = myBookDimensions,
            };

            // Add the book to the DynamoDB table ProductCatalog.
            await context.SaveAsync(myBook);

            // Retrieve the book.
            Book bookRetrieved = await context.LoadAsync<Book>(501);
```

```
            // Update the book dimensions property.
            bookRetrieved.Dimensions.Height += 1;
            bookRetrieved.Dimensions.Length += 1;
            bookRetrieved.Dimensions.Thickness += 0.2M;

            // Write the changed item to the table.
            await context.SaveAsync(bookRetrieved);
        }

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();
            DynamoDBContext context = new DynamoDBContext(client);
            await AddRetrieveUpdateBook(context);
        }
    }
```

Verwenden Sie ein übergeordnetes Object-Persistence-Modell, um eine Tabelle abzufragen und zu scannen.

```
    /// <summary>
    /// Shows how to perform high-level query and scan operations to Amazon
    /// DynamoDB tables.
    /// </summary>
    public class HighLevelQueryAndScan
    {
        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            DynamoDBContext context = new DynamoDBContext(client);

            // Get an item.
            await GetBook(context, 101);

            // Sample forum and thread to test queries.
            string forumName = "Amazon DynamoDB";
            string threadSubject = "DynamoDB Thread 1";
```

```
            // Sample queries.
            await FindRepliesInLast15Days(context, forumName, threadSubject);
            await FindRepliesPostedWithinTimePeriod(context, forumName,
 threadSubject);

            // Scan table.
            await FindProductsPricedLessThanZero(context);
        }

        public static async Task GetBook(IDynamoDBContext context, int productId)
        {
            Book bookItem = await context.LoadAsync<Book>(productId);

            Console.WriteLine("\nGetBook: Printing result.....");
            Console.WriteLine($"Title: {bookItem.Title} \n ISBN:{bookItem.Isbn} \n
 No. of pages: {bookItem.PageCount}");
        }

        /// <summary>
        /// Queries a DynamoDB table to find replies posted within the last 15 days.
        /// </summary>
        /// <param name="context">The DynamoDB context used to perform the query.</
param>
        /// <param name="forumName">The name of the forum that we're interested
 in.</param>
        /// <param name="threadSubject">The thread object containing the query
 parameters.</param>
        public static async Task FindRepliesInLast15Days(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string replyId = $"{forumName} #{threadSubject}";
            DateTime twoWeeksAgoDate = DateTime.UtcNow - TimeSpan.FromDays(15);

            List<object> times = new List<object>();
            times.Add(twoWeeksAgoDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("PostedBy", ScanOperator.GreaterThan,
 times.ToArray());
            scs.Add(sc);

            var cfg = new DynamoDBOperationConfig
```

```
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(replyId, cfg);
            IEnumerable<Reply> latestReplies = await response.GetRemainingAsync();

            Console.WriteLine("\nReplies in last 15 days:");

            foreach (Reply r in latestReplies)
            {

 Console.WriteLine($"{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries for replies posted within a specific time period.
        /// </summary>
        /// <param name="context">The DynamoDB context used to perform the query.</
param>
        /// <param name="forumName">The name of the forum that we're interested
 in.</param>
        /// <param name="threadSubject">Information about the subject that we're
        /// interested in.</param>
        public static async Task FindRepliesPostedWithinTimePeriod(
          IDynamoDBContext context,
          string forumName,
          string threadSubject)
        {
            string forumId = forumName + "#" + threadSubject;
            Console.WriteLine("\nReplies posted within time period:");

            DateTime startDate = DateTime.UtcNow - TimeSpan.FromDays(30);
            DateTime endDate = DateTime.UtcNow - TimeSpan.FromDays(1);

            List<object> times = new List<object>();
            times.Add(startDate);
            times.Add(endDate);

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc = new ScanCondition("LastPostedBy", ScanOperator.Between,
 times.ToArray());
            scs.Add(sc);
```

```
            var cfg = new DynamoDBOperationConfig
            {
                QueryFilter = scs,
            };

            AsyncSearch<Reply> response = context.QueryAsync<Reply>(forumId, cfg);
            IEnumerable<Reply> repliesInAPeriod = await
response.GetRemainingAsync();

            foreach (Reply r in repliesInAPeriod)
            {

Console.WriteLine("{r.Id}\t{r.PostedBy}\t{r.Message}\t{r.ReplyDateTime}");
            }
        }

        /// <summary>
        /// Queries the DynamoDB ProductCatalog table for products costing less
        /// than zero.
        /// </summary>
        /// <param name="context">The DynamoDB context object used to perform the
        /// query.</param>
        public static async Task FindProductsPricedLessThanZero(IDynamoDBContext
context)
        {
            int price = 0;

            List<ScanCondition> scs = new List<ScanCondition>();
            var sc1 = new ScanCondition("Price", ScanOperator.LessThan, price);
            var sc2 = new ScanCondition("ProductCategory", ScanOperator.Equal,
"Book");
            scs.Add(sc1);
            scs.Add(sc2);

            AsyncSearch<Book> response = context.ScanAsync<Book>(scs);

            IEnumerable<Book> itemsWithWrongPrice = await
response.GetRemainingAsync();

            Console.WriteLine("\nFindProductsPricedLessThanZero: Printing
result.....");

            foreach (Book r in itemsWithWrongPrice)
```

```
            {
                Console.WriteLine($"{r.Id}\t{r.Title}\t{r.Price}\t{r.Isbn}");
            }
        }
    }
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
```

```
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
```

```
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();

        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
```

```
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für .NET

Zeigt, wie eine Serverless-Anwendung, die aus einem API Gateway mit Lambda und DynamoDB besteht, mithilfe des .NET-SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Amazon EC2-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    public static ILogger<EC2Basics> _logger = null!;
    public static EC2Wrapper _ec2Wrapper = null!;
    public static SsmWrapper _ssmWrapper = null!;
    public static UiMethods _uiMethods = null!;
```

```
    public static string associationId = null!;
    public static string allocationId = null!;
    public static string instanceId = null!;
    public static string keyPairName = null!;
    public static string groupName = null!;
    public static string tempFileName = null!;
    public static string secGroupId = null!;
    public static bool isInteractive = true;

    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management (Amazon SSM) Service.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        SetUpServices(host);

        var uniqueName = Guid.NewGuid().ToString();
        keyPairName = "mvp-example-key-pair" + uniqueName;
        groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        try
        {
            // Start the scenario.
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(isInteractive);

            // Create the key pair.
```

```
            _uiMethods.DisplayTitle("Create RSA key pair");
            Console.Write("Let's create an RSA key pair that you can be use to ");
            Console.WriteLine("securely connect to your EC2 instance.");
            var keyPair = await _ec2Wrapper.CreateKeyPair(keyPairName);

            // Save key pair information to a temporary file.
            tempFileName = _ec2Wrapper.SaveKeyPair(keyPair);

            Console.WriteLine(
                $"Created the key pair: {keyPair.KeyName} and saved it to:
{tempFileName}");
            string? answer = "";
            if (isInteractive)
            {
                do
                {
                    Console.Write("Would you like to list your existing key pairs?
");
                    answer = Console.ReadLine();
                } while (answer!.ToLower() != "y" && answer.ToLower() != "n");
            }

            if (!isInteractive || answer == "y")
            {
                // List existing key pairs.
                _uiMethods.DisplayTitle("Existing key pairs");

                // Passing an empty string to the DescribeKeyPairs method will
return
                // a list of all existing key pairs.
                var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
                keyPairs.ForEach(kp =>
                {
                    Console.WriteLine(
                        $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
                });
            }

            _uiMethods.PressEnter(isInteractive);

            // Create the security group.
            Console.WriteLine(
                "Let's create a security group to manage access to your instance.");
```

```
            secGroupId = await _ec2Wrapper.CreateSecurityGroup(groupName,
groupDescription);
            Console.WriteLine(
                "Let's add rules to allow all HTTP and HTTPS inbound traffic and to
allow SSH only from your current IP address.");

            _uiMethods.DisplayTitle("Security group information");
            var secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);

            Console.WriteLine($"Created security group {groupName} in your default
VPC.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll authorize the security group we just created so that it
can");
            Console.WriteLine("access the EC2 instances you create.");
            await _ec2Wrapper.AuthorizeSecurityGroupIngress(groupName);

            secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);
            Console.WriteLine($"Now let's look at the permissions again.");
            secGroups.ForEach(group =>
            {
                _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
            });
            _uiMethods.PressEnter(isInteractive);

            // Get list of available Amazon Linux 2 Amazon Machine Images (AMIs).
            var parameters =
                await _ssmWrapper.GetParametersByPath(
                    "/aws/service/ami-amazon-linux-latest");

            List<string> imageIds = parameters.Select(param =>
param.Value).ToList();

            var images = await _ec2Wrapper.DescribeImages(imageIds);

            var i = 1;
            images.ForEach(image =>
            {
```

```
                Console.WriteLine($"\t{i++}\t{image.Description}");
            });

            int choice = 1;
            bool validNumber = false;
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an image: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedImage = images[choice - 1];

            // Display available instance types.
            _uiMethods.DisplayTitle("Instance Types");
            var instanceTypes =
                await _ec2Wrapper.DescribeInstanceTypes(selectedImage.Architecture);

            i = 1;
            instanceTypes.ForEach(instanceType =>
            {
                Console.WriteLine($"\t{i++}\t{instanceType.InstanceType}");
            });
            if (isInteractive)
            {
                do
                {
                    Console.Write("Please select an instance type: ");
                    var selImage = Console.ReadLine();
                    validNumber = int.TryParse(selImage, out choice);
                } while (!validNumber);
            }

            var selectedInstanceType = instanceTypes[choice - 1].InstanceType;

            // Create an EC2 instance.
            _uiMethods.DisplayTitle("Creating an EC2 Instance");
            instanceId = await _ec2Wrapper.RunInstances(selectedImage.ImageId,
                selectedInstanceType, keyPairName, secGroupId);
```

```
            _uiMethods.PressEnter(isInteractive);

            var instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine(
                "\nYou can use SSH to connect to your instance. For example:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we'll stop the instance and then start it again to see what's
changed.");

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nNotice the change in the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            _uiMethods.PressEnter(isInteractive);

            Console.WriteLine(
                "Now we will stop the instance again. Then we will create and
associate an");
            Console.WriteLine("Elastic IP address to use with our instance.");

            await _ec2Wrapper.StopInstances(instanceId);
            _uiMethods.PressEnter(isInteractive);

            _uiMethods.DisplayTitle("Allocate Elastic IP address");
            Console.WriteLine(
```

```
                "You can allocate an Elastic IP address and associate it with your
instance\nto keep a consistent IP address even when your instance restarts.");
            var allocationResponse = await _ec2Wrapper.AllocateAddress();
            allocationId = allocationResponse.AllocationId;
            Console.WriteLine(
                "Now we will associate the Elastic IP address with our instance.");
            associationId = await _ec2Wrapper.AssociateAddress(allocationId,
instanceId);

            // Start the instance again.
            Console.WriteLine("Now let's start the instance again.");
            await _ec2Wrapper.StartInstances(instanceId);

            Console.WriteLine("\nLet's see what changed.");

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("Instance information");
            _ec2Wrapper.DisplayInstanceInformation(instance);

            Console.WriteLine("\nHere is the SSH information:");
            Console.WriteLine(
                $"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

            Console.WriteLine("Let's stop and start the instance again.");
            _uiMethods.PressEnter(isInteractive);

            await _ec2Wrapper.StopInstances(instanceId);

            Console.WriteLine("\nThe instance has stopped.");

            Console.WriteLine("Now let's start it up again.");
            await _ec2Wrapper.StartInstances(instanceId);

            instance = await _ec2Wrapper.DescribeInstance(instanceId);
            _uiMethods.DisplayTitle("New Instance Information");
            _ec2Wrapper.DisplayInstanceInformation(instance);
            Console.WriteLine("Note that the IP address did not change this time.");
            _uiMethods.PressEnter(isInteractive);

            await Cleanup();
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, "There was a problem with the scenario, starting
cleanup.");
            await Cleanup();
        }

        _uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
        _uiMethods.PressEnter(isInteractive);
    }

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host"></param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<EC2Basics>(loggerFactory);

        // Now the client is available for injection.
        _ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        _ssmWrapper = host.Services.GetRequiredService<SsmWrapper>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Clean up any resources from the scenario.
    /// </summary>
    /// <returns></returns>
    public static async Task Cleanup()
    {
        _uiMethods.DisplayTitle("Clean up resources");
        Console.WriteLine("Now let's clean up the resources we created.");

        Console.WriteLine("Disassociate the Elastic IP address and release it.");
        // Disassociate the Elastic IP address.
        await _ec2Wrapper.DisassociateIp(associationId);

        // Delete the Elastic IP address.
        await _ec2Wrapper.ReleaseAddress(allocationId);

        // Terminate the instance.
```

```
        Console.WriteLine("Terminating the instance we created.");
        await _ec2Wrapper.TerminateInstances(instanceId);

        // Delete the security group.
        Console.WriteLine($"Deleting the Security Group: {groupName}.");
        await _ec2Wrapper.DeleteSecurityGroup(secGroupId);

        // Delete the RSA key pair.
        Console.WriteLine($"Deleting the key pair: {keyPairName}");
        await _ec2Wrapper.DeleteKeyPair(keyPairName);
        Console.WriteLine("Deleting the temporary file with the key information.");
        _ec2Wrapper.DeleteTempFile(tempFileName);
        _uiMethods.PressEnter(isInteractive);
    }
}
```

Definieren Sie eine Klasse, die EC2-Aktionen umschließt.

```
/// <summary>
/// Methods of this class perform Amazon Elastic Compute Cloud (Amazon EC2).
/// </summary>
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEC2;
    private readonly ILogger<EC2Wrapper> _logger;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EC2 client.</param>
    /// <param name="logger">The injected logger.</param>
    public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
    {
        _amazonEC2 = amazonService;
        _logger = logger;
    }

    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP address
    // constant even when you restart the associated instance.
    /// </summary>
```

```
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation ID
 {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-
addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
```

```
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
```

```
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
```

```
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
        return ipString.Trim();
    }

    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
 {keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }

    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
```

```
   public async Task<string> CreateSecurityGroup(string groupName, string
groupDescription)
   {
       try
       {
           var response = await _amazonEC2.CreateSecurityGroupAsync(
               new CreateSecurityGroupRequest(groupName, groupDescription));

           // Wait until the security group exists.
           int retries = 5;
           while (retries-- > 0)
           {
               var groups = await DescribeSecurityGroups(response.GroupId);
               if (groups.Any())
               {
                   return response.GroupId;
               }

               Thread.Sleep(5000);
               retries--;
           }
           _logger.LogError($"Unable to find newly created group {groupName}.");
           throw new DoesNotExistException("security group not found");
       }
       catch (AmazonEC2Exception ec2Exception)
       {
           if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
           {
               _logger.LogError(
                   $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
           }
           throw;
       }
       catch (Exception ex)
       {
           _logger.LogError(
               $"An error occurred while creating the security group.:
{ex.Message}");
           throw;
       }
   }
```

```
    /// <summary>
    /// Create a new Amazon EC2 VPC.
    /// </summary>
    /// <param name="cidrBlock">The CIDR block for the new security group.</param>
    /// <returns>The VPC Id of the new VPC.</returns>
    public async Task<string?> CreateVPC(string cidrBlock)
    {

        try
        {
            var response = await _amazonEC2.CreateVpcAsync(new CreateVpcRequest
            {
                CidrBlock = cidrBlock,
            });

            Vpc vpc = response.Vpc;
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
            return vpc.VpcId;
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"Couldn't create VPC because: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
```

```
                _logger.LogError($"KeyPair {keyPairName} does not exist and cannot
be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
```

```
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be deleted.
Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon EC2 VPC.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteVpc(string vpcId)
    {
        var request = new DeleteVpcRequest
        {
            VpcId = vpcId,
        };

        var response = await _amazonEC2.DeleteVpcAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Get information about existing Amazon EC2 images.
    /// </summary>
    /// <returns>A list of image information.</returns>
    public async Task<List<Image>> DescribeImages(List<string>? imageIds)
    {
        var request = new DescribeImagesRequest();
        if (imageIds is not null)
        {
            // If the imageIds list is not null, add the list
            // to the request object.
            request.ImageIds = imageIds;
        }
```

```
        var response = await _amazonEC2.DescribeImagesAsync(request);
        return response.Images;
    }

    /// <summary>
    /// Display the information returned by DescribeImages.
    /// </summary>
    /// <param name="images">The list of image information to display.</param>
    public void DisplayImageInfo(List<Image> images)
    {
        images.ForEach(image =>
        {
            Console.WriteLine($"{image.Name} Created on: {image.CreationDate}");
        });

    }

    /// <summary>
    /// Get information about an Amazon EC2 instance.
    /// </summary>
    /// <param name="instanceId">The instance Id of the EC2 instance.</param>
    /// <returns>An EC2 instance.</returns>
    public async Task<Instance> DescribeInstance(string instanceId)
    {
        var response = await _amazonEC2.DescribeInstancesAsync(
            new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
        return response.Reservations[0].Instances[0];
    }

    /// <summary>
    /// Display EC2 instance information.
    /// </summary>
    /// <param name="instance">The instance Id of the EC2 instance.</param>
    public void DisplayInstanceInformation(Instance instance)
    {
        Console.WriteLine($"ID: {instance.InstanceId}");
        Console.WriteLine($"Image ID: {instance.ImageId}");
        Console.WriteLine($"{instance.InstanceType}");
        Console.WriteLine($"Key Name: {instance.KeyName}");
        Console.WriteLine($"VPC ID: {instance.VpcId}");
        Console.WriteLine($"Public IP: {instance.PublicIpAddress}");
        Console.WriteLine($"State: {instance.State.Name}");
```

```
    }

    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
```

```
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size strings
conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }

            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
```

```
                _logger.LogError(
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();

            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;
```

```
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups. {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));
```

```
            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }


    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId = associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
```

```
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
```

```
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
 {ec2Exception.Message}");
            }
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance {ec2InstanceId}.:
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId = allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
 {ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
```

```
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StartInstancesAsync(request);

            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }
```

```
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };

            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                $"An error occurred while terminating the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AllocateAddress

- AssociateAddress
- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    // instance. By using an Elastic IP address, you can keep the public IP address
    // constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation ID
 {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-
addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
 exceeded. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
 {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie AllocateAddress in der AWS SDK für .NET API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Associates an Elastic IP address with an instance. When this association is
    /// created, the Elastic IP's public IP address is immediately used as the
 public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
 instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
 {ec2Exception.Message}");
            }
```

```
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie AssociateAddress in der AWS SDK für .NET API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Authorize the local computer ingress to EC2 instances associated
    /// with the virtual private cloud (VPC) security group.
    /// </summary>
    /// <param name="groupName">The name of the security group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
    {
        try
        {
            // Get the IP address for the local computer.
            var ipAddress = await GetIpAddress();
            Console.WriteLine($"Your IP address is: {ipAddress}");
            var ipRanges =
                new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
```

```
            var permission = new IpPermission
            {
                Ipv4Ranges = ipRanges,
                IpProtocol = "tcp",
                FromPort = 22,
                ToPort = 22
            };
            var permissions = new List<IpPermission> { permission };
            var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
                new AuthorizeSecurityGroupIngressRequest(groupName, permissions));
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
            {
                _logger.LogError(
                    $"The ingress rule already exists. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while authorizing ingress.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Authorize the local computer for ingress to
    /// the Amazon EC2 SecurityGroup.
    /// </summary>
    /// <returns>The IPv4 address of the computer running the scenario.</returns>
    private static async Task<string> GetIpAddress()
    {
        var httpClient = new HttpClient();
        var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

        // The IP address is returned with a new line
        // character on the end. Trim off the whitespace and
        // return the value to the caller.
```

```
        return ipString.Trim();
    }
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK für .NET API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon EC2 key pair with a specified name.
    /// </summary>
    /// <param name="keyPairName">The name for the new key pair.</param>
    /// <returns>The Amazon EC2 key pair created.</returns>
    public async Task<KeyPair?> CreateKeyPair(string keyPairName)
    {
        try
        {
            var request = new CreateKeyPairRequest { KeyName = keyPairName, };

            var response = await _amazonEC2.CreateKeyPairAsync(request);

            var kp = response.KeyPair;
            // Return the key pair so it can be saved if needed.

            // Wait until the key pair exists.
            int retries = 5;
            while (retries-- > 0)
            {
                Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
                var keyPairs = await DescribeKeyPairs(keyPairName);
```

```
                if (keyPairs.Any())
                {
                    return kp;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
            throw new DoesNotExistException("KeyPair not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
            {
                _logger.LogError(
                    $"A key pair called {keyPairName} already exists.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the key pair.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Save KeyPair information to a temporary file.
    /// </summary>
    /// <param name="keyPair">The name of the key pair.</param>
    /// <returns>The full path to the temporary file.</returns>
    public string SaveKeyPair(KeyPair keyPair)
    {
        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

        // Save the key pair to a file in a temporary folder.
        using var stream = new FileStream(pemFileName, FileMode.Create);
        using var writer = new StreamWriter(stream);
```

```
        writer.WriteLine(keyPair.KeyMaterial);

        return pemFileName;
    }
```

- API-Details finden Sie CreateKeyPair in der AWS SDK für .NET API-Referenz.

`CreateLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateLaunchTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
    /// the instance is started. This script installs the Python packages and starts
a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
            await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
                _instanceProfileName, instancePolicyPath);
```

```
            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while creating the template.:
 {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie CreateLaunchTemplate in der AWS SDK für .NET API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
 groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
```

```
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK für .NET API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyPair(string keyPairName)
    {
        try
        {
            await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError($"KeyPair {keyPairName} does not exist and cannot
be deleted. Please verify the key pair name and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Delete the temporary file where the key pair information was saved.
    /// </summary>
```

```
    /// <param name="tempFileName">The path to the temporary file.</param>
    public void DeleteTempFile(string tempFileName)
    {
        if (File.Exists(tempFileName))
        {
            File.Delete(tempFileName);
        }
    }
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK für .NET API-Referenz.

`DeleteLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLaunchTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
```

```
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DeleteLaunchTemplate in der AWS SDK für .NET API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteSecurityGroup(string groupId)
    {
        try
```

```
        {
            var response =
                await _amazonEC2.DeleteSecurityGroupAsync(
                    new DeleteSecurityGroupRequest { GroupId = groupId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"Security Group {groupId} does not exist and cannot be deleted.
Please verify the ID and try again.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK für .NET API-Referenz.

`DescribeAvailabilityZones`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAvailabilityZones`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability zones.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeAvailabilityZones in der AWS SDK für .NET API-Referenz.

`DescribeIamInstanceProfileAssociations`

Das folgende Codebeispiel zeigt die Verwendung`DescribeIamInstanceProfileAssociations`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the instance profile association data for an instance.
```

```
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeIamInstanceProfileAssociations in der AWS SDK für .NET API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        try
        {
            var request = new DescribeInstanceTypesRequest();

            var filters = new List<Filter>
            {
                new Filter("processor-info.supported-architecture",
                    new List<string> { architecture.ToString() })
            };
            filters.Add(new Filter("instance-type", new() { "*.micro",
"*.small" }));

            request.Filters = filters;
            var instanceTypes = new List<InstanceTypeInfo>();

            var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
            await foreach (var instanceType in paginator.InstanceTypes)
            {
                instanceTypes.Add(instanceType);
            }

            return instanceTypes;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
```

```
                _logger.LogError(
                    $"Parameters are invalid. Ensure architecture and size strings
conform to DescribeInstanceTypes API reference.");
            }

            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK für .NET API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about EC2 instances with a particular state.
    /// </summary>
    /// <param name="tagName">The name of the tag to filter on.</param>
    /// <param name="tagValue">The value of the tag to look for.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> GetInstancesWithState(string state)
    {
        try
        {
            // Filters the results of the instance list.
            var filters = new List<Filter>
```

```
            {
                new Filter
                {
                    Name = $"instance-state-name",
                    Values = new List<string> { state, },
                },
            };
            var request = new DescribeInstancesRequest { Filters = filters, };

            Console.WriteLine($"\nShowing instances with state {state}");
            var paginator = _amazonEC2.Paginators.DescribeInstances(request);

            await foreach (var response in paginator.Responses)
            {
                foreach (var reservation in response.Reservations)
                {
                    foreach (var instance in reservation.Instances)
                    {
                        Console.Write($"Instance ID: {instance.InstanceId} ");
                        Console.WriteLine($"\tCurrent State:
{instance.State.Name}");
                    }
                }
            }

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidParameterValue")
            {
                _logger.LogError(
                    $"Invalid parameter value for filtering instances.");
            }

            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list instances because: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DescribeInstances in der AWS SDK für .NET API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an Amazon EC2 key pair.
    /// </summary>
    /// <param name="keyPairName">The name of the key pair.</param>
    /// <returns>A list of key pair information.</returns>
    public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
    {
        try
        {
            var request = new DescribeKeyPairsRequest();
            if (!string.IsNullOrEmpty(keyPairName))
            {
                request = new DescribeKeyPairsRequest
                {
                    KeyNames = new List<string> { keyPairName }
                };
            }

            var response = await _amazonEC2.DescribeKeyPairsAsync(request);
            return response.KeyPairs.ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
            {
                _logger.LogError(
```

```
                    $"A key pair called {keyPairName} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while describing the key pair.: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK für .NET API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information for one or all Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        try
        {
            var securityGroups = new List<SecurityGroup>();
            var request = new DescribeSecurityGroupsRequest();
```

```
            if (!string.IsNullOrEmpty(groupId))
            {
                var groupIds = new List<string> { groupId };
                request.GroupIds = groupIds;
            }

            var paginatorForSecurityGroups =
                _amazonEC2.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            return securityGroups;

        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
            {
                _logger.LogError(
                    $"A security group {groupId} does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while listing security groups. {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Display the information returned by the call to
    /// DescribeSecurityGroupsAsync.
    /// </summary>
    /// <param name="securityGroup">A list of security group information.</param>
    public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
    {
        Console.WriteLine($"{securityGroup.GroupName}");
```

```
        Console.WriteLine("Ingress permissions:");
        securityGroup.IpPermissions.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
        Console.WriteLine("Egress permissions:");
        securityGroup.IpPermissionsEgress.ForEach(permission =>
        {
            Console.WriteLine($"\tFromPort: {permission.FromPort}");
            Console.WriteLine($"\tIpProtocol: {permission.IpProtocol}");

            Console.Write($"\tIpv4Ranges: ");
            permission.Ipv4Ranges.ForEach(range => { Console.Write($"{range.CidrIp}
"); });

            Console.WriteLine($"\n\tIpv6Ranges:");
            permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

            Console.Write($"\n\tPrefixListIds: ");
            permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

            Console.WriteLine($"\n\tTo Port: {permission.ToPort}");
        });
    }
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für .NET API-Referenz.

`DescribeSubnets`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubnets`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
```

```
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeSubnets in der AWS SDK für .NET API-Referenz.

`DescribeVpcs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVpcs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
```

```
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeVpcs in der AWS SDK für .NET API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId = associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DisassociateAddress in der AWS SDK für .NET API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie eine Instance anhand ihrer ID neu.

```
    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)
    {
        try
        {
            var request = new RebootInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.RebootInstancesAsync(request);

            // Wait for the instance to be running.
            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

            return true;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
            }
            return false;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while rebooting the instance {ec2InstanceId}.:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

Ersetzen Sie das Profil für eine Instance und starten Sie einen Webserver neu.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
```

```
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie RebootInstances in der AWS SDK für .NET API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Release an Elastic IP address. After the Elastic IP address is released,
    /// it can no longer be used.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId = allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
 {ec2Exception.Message}");
            }

            return false;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie ReleaseAddress in der AWS SDK für .NET API-Referenz.

`ReplaceIamInstanceProfileAssociation`

Das folgende Codebeispiel zeigt die Verwendung`ReplaceIamInstanceProfileAssociation`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
```

```
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
```

```
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ReplaceIamInstanceProfileAssociation in der AWS SDK für .NET API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.Write("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
```

```
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie RunInstances in der AWS SDK für .NET API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
    /// to start.</param>
    /// <returns>Async task.</returns>
    public async Task StartInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StartInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StartInstancesAsync(request);
```

```
            Console.Write("Waiting for instance to start. ");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while starting the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
```

```
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- API-Details finden Sie StartInstances in der AWS SDK für .NET API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Stop an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance to
    /// stop.</param>
    /// <returns>Async task.</returns>
    public async Task StopInstances(string ec2InstanceId)
    {
        try
        {
            var request = new StopInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId },
            };

            await _amazonEC2.StopInstancesAsync(request);
            Console.Write("Waiting for the instance to stop.");
```

```
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

            Console.WriteLine("\nThe instance has stopped.");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while stopping the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);
```

```
            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- API-Details finden Sie StopInstances in der AWS SDK für .NET API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Terminate an EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the EC2 instance
    /// to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
    {
        try
        {
            var request = new TerminateInstancesRequest
            {
                InstanceIds = new List<string> { ec2InstanceId }
            };
```

```
            var response = await _amazonEC2.TerminateInstancesAsync(request);
            Console.Write("Waiting for the instance to terminate.");
            await WaitForInstanceState(ec2InstanceId, InstanceStateName.Terminated);

            Console.WriteLine($"\nThe instance {ec2InstanceId} has been
terminated.");
            return response.TerminatingInstances;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while terminating the instance.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
```

```
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEC2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
```

- API-Details finden Sie TerminateInstances in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
```

```
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
```

```
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
```

```
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
```

```
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
_elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
protocol, port, defaultVpc.VpcId);
```

```
        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
```

```
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }
```

```
    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
 different parts of the system\n" +
                          "to create situations where the web service fails, and
 shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
 these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                          "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();
```

```
        Console.WriteLine("Instead of failing when the recommendation service fails,
the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
```

```
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy");
```

```
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
```

```
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
```

```
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
```

```
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
```

```
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
```

```
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
```

```
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
    /// the instance is started. This script installs the Python packages and starts
a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
```

```
            await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
 availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
```

```
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
{groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
```

```
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
```

```
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
```

```
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
```

```
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
```

```
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
```

```
   /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
   /// <returns>Async task.</returns>
   public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
   {
       try
       {
           await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
                   {
                       Name = credsProfileName
                   }
               });
           // Allow time before resetting.
           Thread.Sleep(25000);

           await _amazonEc2.RebootInstancesAsync(
               new RebootInstancesRequest(new List<string>() { instanceId }));
           Thread.Sleep(25000);
           var instanceReady = false;
           var retries = 5;
           while (retries-- > 0 && !instanceReady)
           {
               var instancesPaginator =
                   _amazonSsm.Paginators.DescribeInstanceInformation(
                       new DescribeInstanceInformationRequest());
               // Get the entire list using the paginator.
               await foreach (var instance in
instancesPaginator.InstanceInformationList)
               {
                   instanceReady = instance.InstanceId == instanceId;
                   if (instanceReady)
                   {
                       break;
                   }
               }
           }
           Console.WriteLine("Waiting for instance to be running.");
           await WaitForInstanceState(instanceId, InstanceStateName.Running);
           Console.WriteLine("Instance ready.");
```

```
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
```

```
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
```

```
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
```

```
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
```

```
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
```

```
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
```

```
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
```

```
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
```

```
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
 subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
 List<string> subnetIds, TargetGroup targetGroup)
    {
```

```
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
```

```
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }
```

```
        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
```

```
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;
```

```
    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
```

```
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
```

```
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
```

```
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }
```

```
    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Amazon ECS-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`ListClusters`

Das folgende Codebeispiel zeigt die Verwendung`ListClusters`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
```

```
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
            throw new Exception($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
        }
    }
```

- API-Details finden Sie ListClusters in der AWS SDK für .NET API-Referenz.

`ListServices`

Das folgende Codebeispiel zeigt die Verwendung`ListServices`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();

        var request = new ListServicesRequest
        {
            Cluster = clusterARN
```

```
        };
        // Call the ListServices API operation and get the list of service ARNs
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }
```

- API-Details finden Sie ListServices in der AWS SDK für .NET API-Referenz.

`ListTasks`

Das folgende Codebeispiel zeigt die Verwendung`ListTasks`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
```

```
    {
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
```

- API-Details finden Sie ListTasks in der AWS SDK für .NET API-Referenz.

Szenarien

Abrufen von ARN-Informationen für Cluster, Services und Aufgaben

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen einer Liste aller Cluster.
- Abrufen von Services für einen Cluster.
- Abrufen von Aufgaben für einen Cluster.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
using Amazon.ECS;
using ECSActions;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace ECSScenario;

public class ECSScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.


    This .NET example performs the following tasks:
        1. List ECS Cluster ARNs.
        2. List services in every cluster
        3. List Task ARNs in every cluster.
    */

    private static ILogger logger = null!;
    private static ECSWrapper _ecsWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
        .Build();

        ILoggerFactory loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ECSScenario>();

        var loggerECSWarpper = LoggerFactory.Create(builder =>
{ builder.AddConsole(); })
          .CreateLogger<ECSWrapper>();

        var amazonECSClient = new AmazonECSClient();

        _ecsWrapper = new ECSWrapper(amazonECSClient, loggerECSWarpper);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon ECS example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListClusterARNs();
            await ListServiceARNs();
            await ListTaskARNs();

        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// List ECS Cluster ARNs
    /// </summary>
    private static async Task ListClusterARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List Cluster ARNs from ECS.");
        var arns = await _ecsWrapper.GetClusterARNSAsync();
```

```
        foreach (var arn in arns)
        {
            Console.WriteLine($"Cluster arn: {arn}");
            Console.WriteLine($"Cluster name: {arn.Split("/").Last()}");
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List services in every cluster
    /// </summary>
    private static async Task ListServiceARNs()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List Service ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting services for cluster name:
{clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));


            var serviceARNs = await _ecsWrapper.GetServiceARNSAsync(clusterARN);

            foreach (var serviceARN in serviceARNs)
            {
                Console.WriteLine($"Service arn: {serviceARN}");
                Console.WriteLine($"Service name: {serviceARN.Split("/").Last()}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// List tasks in every cluster
    /// </summary>
    private static async Task ListTaskARNs()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. List Task ARNs in every cluster.");
        var clusterARNs = await _ecsWrapper.GetClusterARNSAsync();

        foreach (var clusterARN in clusterARNs)
        {
            Console.WriteLine($"Getting tasks for cluster name:
 {clusterARN.Split("/").Last()}");
            Console.WriteLine(new string('.', 5));

            var taskARNs = await _ecsWrapper.GetTaskARNsAsync(clusterARN);

            foreach (var taskARN in taskARNs)
            {
                Console.WriteLine($"Task arn: {taskARN}");
            }
        }
        Console.WriteLine(new string('-', 80));
    }
}
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Amazon-ECS-Aktionen zu verwalten.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.Logging;

namespace ECSActions;

public class ECSWrapper
{
    private readonly AmazonECSClient _ecsClient;
    private readonly ILogger<ECSWrapper> _logger;

    /// <summary>
    /// Constructor for the ECS wrapper.
    /// </summary>
    /// <param name="ecsClient">The injected ECS client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public ECSWrapper(AmazonECSClient ecsClient, ILogger<ECSWrapper> logger)
```

```
    {
        _logger = logger;
        _ecsClient = ecsClient;
    }

    /// <summary>
    /// List cluster ARNs available.
    /// </summary>
    /// <returns>The ARN list of clusters.</returns>
    public async Task<List<string>> GetClusterARNSAsync()
    {

        Console.WriteLine("Getting a list of all the clusters in your AWS
account...");
        List<string> clusterArnList = new List<string>();
        // Get a list of all the clusters in your AWS account
        try
        {

            var listClustersResponse = _ecsClient.Paginators.ListClusters(new
ListClustersRequest
            {
            });

            var clusterArns = listClustersResponse.ClusterArns;

            // Print the ARNs of the clusters
            await foreach (var clusterArn in clusterArns)
            {
                clusterArnList.Add(clusterArn);
            }

            if (clusterArnList.Count == 0)
            {
                _logger.LogWarning("No clusters found in your AWS account.");
            }
            return clusterArnList;
        }
        catch (Exception e)
        {
            _logger.LogError($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
```

```
            throw new Exception($"An error occurred while getting a list of all the
clusters in your AWS account. {e.InnerException}");
        }
    }

    /// <summary>
    /// List service ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of services in given cluster.</returns>
    public async Task<List<string>> GetServiceARNSAsync(string clusterARN)
    {
        List<string> serviceArns = new List<string>();

        var request = new ListServicesRequest
        {
            Cluster = clusterARN
        };
        // Call the ListServices API operation and get the list of service ARNs
        var serviceList = _ecsClient.Paginators.ListServices(request);

        await foreach (var serviceARN in serviceList.ServiceArns)
        {
            if (serviceARN is null)
                continue;

            serviceArns.Add(serviceARN);
        }

        if (serviceArns.Count == 0)
        {
            _logger.LogWarning($"No services found in cluster {clusterARN} .");
        }

        return serviceArns;
    }

    /// <summary>
    /// List task ARNs available.
    /// </summary>
    /// <param name="clusterARN">The arn of the ECS cluster.</param>
    /// <returns>The ARN list of tasks in given cluster.</returns>
    public async Task<List<string>> GetTaskARNsAsync(string clusterARN)
    {
```

```
        // Set up the request to describe the tasks in the service
        var listTasksRequest = new ListTasksRequest
        {
            Cluster = clusterARN
        };
        List<string> taskArns = new List<string>();

        // Call the ListTasks API operation and get the list of task ARNs
        var tasks = _ecsClient.Paginators.ListTasks(listTasksRequest);

        await foreach (var task in tasks.TaskArns)
        {
            if (task is null)
                continue;


            taskArns.Add(task);
        }

        if (taskArns.Count == 0)
        {
            _logger.LogWarning("No tasks found in cluster: " + clusterARN);
        }

        return taskArns;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - ListClusters
  - ListServices
  - ListTasks

Beispiele für Elastic Load Balancing — Version 2 mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateListener`

Das folgende Codebeispiel zeigt die Verwendung`CreateListener`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
```

```
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
```

```
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }
```

- API-Details finden Sie CreateListener in der AWS SDK für .NET API-Referenz.

`CreateLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`CreateLoadBalancer`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
```

```
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
```

```
        return createLbResponse.LoadBalancers[0];
    }
```

- API-Details finden Sie CreateLoadBalancer in der AWS SDK für .NET API-Referenz.

`CreateTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateTargetGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
```

```
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }
```

- API-Details finden Sie CreateTargetGroup in der AWS SDK für .NET API-Referenz.

`DeleteLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLoadBalancer`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
```

```
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }
```

- API-Details finden Sie DeleteLoadBalancer in der AWS SDK für .NET API-Referenz.

`DeleteTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTargetGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
```

```
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

- API-Details finden Sie DeleteTargetGroup in der AWS SDK für .NET API-Referenz.

`DescribeLoadBalancers`

Das folgende Codebeispiel zeigt die Verwendung`DescribeLoadBalancers`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }
```

- API-Details finden Sie DescribeLoadBalancers in der AWS SDK für .NET API-Referenz.

`DescribeTargetHealth`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTargetHealth`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
```

```
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }
```

- API-Details finden Sie DescribeTargetHealth in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);
```

```
            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
```

```
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
```

```
                            $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                            $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
```

```
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");
```

```
        Console.WriteLine($"\nNow, checking target health indicates that the
 instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
 the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
```

```
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
```

```
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
```

```
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
```

```
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
                                        "]" +
                                        "}," +
                                   "\"Action\": \"sts:AssumeRole\"" +
                                   "}]" +
                               "}";

       var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

       var policyArn = "";

       try
       {
           var createPolicyResult = await _amazonIam.CreatePolicyAsync(
               new CreatePolicyRequest
               {
                   PolicyName = policyName,
                   PolicyDocument = policyDocument
               });
```

```
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
```

```
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
```

```
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
```

```
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
```

```
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                  $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
```

```
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
```

```
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
```

```
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
```

```
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
```

```
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
 computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
 instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
 while running this example.
    /// If you do, be sure to remove public access when you're done.
```

```
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }
```

```
    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
```

```
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
```

```
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
```

```
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
```

```
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
```

```
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
```

```
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
```

```
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
```

```
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

EventBridge Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo EventBridge

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von EventBridge beginnen.

SDK für .NET

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.EventBridge;
using Amazon.EventBridge.Model;

namespace EventBridgeActions;
```

```
public static class HelloEventBridge
{
    static async Task Main(string[] args)
    {
        var eventBridgeClient = new AmazonEventBridgeClient();

        Console.WriteLine($"Hello Amazon EventBridge! Following are some of your
 EventBuses:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five event buses.
        var response = await eventBridgeClient.ListEventBusesAsync(
            new ListEventBusesRequest()
            {
                Limit = 5
            });

        foreach (var eventBus in response.EventBuses)
        {
            Console.WriteLine($"\tEventBus: {eventBus.Name}");
            Console.WriteLine($"\tArn: {eventBus.Arn}");
            Console.WriteLine($"\tPolicy: {eventBus.Policy}");
            Console.WriteLine();
        }
    }
}
```

- API-Details finden Sie ListEventBuses in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Regel und fügen Sie ihr ein Ziel hinzu.
- Aktivieren und deaktivieren Sie Regeln.
- Listen Sie Regeln und Ziele auf und aktualisieren Sie sie.
- Senden Sie Ereignisse und bereinigen Sie dann die Ressourcen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public class EventBridgeScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks with Amazon EventBridge:
    - Create a rule.
    - Add a target to a rule.
    - Enable and disable rules.
    - List rules and targets.
    - Update rules and targets.
    - Send events.
    - Delete the rule.
    */

    private static ILogger logger = null!;
    private static EventBridgeWrapper _eventBridgeWrapper = null!;
    private static IConfiguration _configuration = null!;

    private static IAmazonIdentityManagementService? _iamClient = null!;
    private static IAmazonSimpleNotificationService? _snsClient = null!;
    private static IAmazonS3 _s3Client = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonEventBridge>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddAWSService<IAmazonS3>()
            .AddAWSService<IAmazonSimpleNotificationService>()
            .AddTransient<EventBridgeWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<EventBridgeScenario>();

        ServicesSetup(host);

        string topicArn = "";
        string roleArn = "";

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            roleArn = await CreateRole();

            await CreateBucketWithEventBridgeEvents();

            await AddEventRule(roleArn);

            await ListEventRules();

            topicArn = await CreateSnsTopic();

            var email = await SubscribeToSnsTopic(topicArn);

            await AddSnsTarget(topicArn);
```

```
            await ListTargets();

            await ListRulesForTarget(topicArn);

            await UploadS3File(_s3Client);

            await ChangeRuleState(false);

            await GetRuleState();

            await UpdateSnsEventRule(topicArn);

            await ChangeRuleState(true);

            await UploadS3File(_s3Client);

            await UpdateToCustomRule(topicArn);

            await TriggerCustomRule(email);

            await CleanupResources(topicArn);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources(topicArn);
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon EventBridge example scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _eventBridgeWrapper =
host.Services.GetRequiredService<EventBridgeWrapper>();
        _snsClient =
host.Services.GetRequiredService<IAmazonSimpleNotificationService>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
```

```
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
    }

    /// <summary>
    /// Create a role to be used by EventBridge.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateRole()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating a role to use with EventBridge and attaching
managed policy AmazonEventBridgeFullAccess.");
        Console.WriteLine(new string('-', 80));

        var roleName = _configuration["roleName"];

        var assumeRolePolicy = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                   "\"Effect\": \"Allow\"," +
                                   "\"Principal\": {" +
                                   $"\"Service\": \"events.amazonaws.com\"" +
                                   "}," +
                                   "\"Action\": \"sts:AssumeRole\"" +
                                   "}]" +
                                   "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = roleName
            });

        await _iamClient.AttachRolePolicyAsync(
            new AttachRolePolicyRequest()
            {
                PolicyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess",
                RoleName = roleName
            });
        // Allow time for the role to be ready.
        Thread.Sleep(10000);
```

```
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Create an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
events enabled.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateBucketWithEventBridgeEvents()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an S3 bucket with EventBridge events enabled.");

        var testBucketName = _configuration["testBucketName"];

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            testBucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = testBucketName,
                UseClientRegion = true
            });
        }

        await _s3Client.PutBucketNotificationAsync(new
PutBucketNotificationRequest()
        {
            BucketName = testBucketName,
            EventBridgeConfiguration = new EventBridgeConfiguration()
        });

        Console.WriteLine($"\tAdded bucket {testBucketName} with EventBridge events
enabled.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and upload a file to an S3 bucket to trigger an event.
    /// </summary>
```

```
    /// <returns>Async task.</returns>
    private static async Task UploadS3File(IAmazonS3 s3Client)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Uploading a file to the test bucket. This will trigger a
subscription email.");

        var testBucketName = _configuration["testBucketName"];

        var fileName = $"example_upload_{DateTime.UtcNow.Ticks}.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for testing uploads.");
        }

        await s3Client.PutObjectAsync(new PutObjectRequest()
        {
            FilePath = fileName,
            BucketName = testBucketName
        });

        Console.WriteLine($"\tPress Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create an Amazon Simple Notification Service (Amazon SNS) topic to use as an
EventBridge target.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> CreateSnsTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "Creating an Amazon Simple Notification Service (Amazon SNS) topic for
email subscriptions.");

        var topicName = _configuration["topicName"];
```

```
        string topicPolicy = "{" +
                             "\"Version\": \"2012-10-17\"," +
                             "\"Statement\": [{" +
                             "\"Sid\": \"EventBridgePublishTopic\"," +
                             "\"Effect\": \"Allow\"," +
                             "\"Principal\": {" +
                             $"\"Service\": \"events.amazonaws.com\"" +
                             "}," +
                             "\"Resource\": \"*\"," +
                             "\"Action\": \"sns:Publish\"" +
                             "}]" +
                             "}";

        var topicAttributes = new Dictionary<string, string>()
        {
            { "Policy", topicPolicy }
        };

        var topicResponse = await _snsClient!.CreateTopicAsync(new
CreateTopicRequest()
        {
            Name = topicName,
            Attributes = topicAttributes

        });

        Console.WriteLine($"\tAdded topic {topicName} for email subscriptions.");

        Console.WriteLine(new string('-', 80));

        return topicResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a user email to an SNS topic.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>The user's email.</returns>
    private static async Task<string> SubscribeToSnsTopic(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
```

```
        string email = "";
        while (string.IsNullOrEmpty(email))
        {
            Console.WriteLine("Enter your email to subscribe to the Amazon SNS
topic:");
            email = Console.ReadLine()!;
        }

        var subscriptions = new List<string>();
        var paginatedSubscriptions =
_snsClient!.Paginators.ListSubscriptionsByTopic(
            new ListSubscriptionsByTopicRequest()
            {
                TopicArn = topicArn
            });

        // Get the entire list using the paginator.
        await foreach (var subscription in paginatedSubscriptions.Subscriptions)
        {
            subscriptions.Add(subscription.Endpoint);
        }

        if (subscriptions.Contains(email))
        {
            Console.WriteLine($"\tYour email is already subscribed.");
            Console.WriteLine(new string('-', 80));
            return email;
        }

        await _snsClient.SubscribeAsync(new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "email",
            Endpoint = email
        });

        Console.WriteLine($"Use the link in the email you received to confirm your
subscription, then press Enter to continue.");

        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
        return email;
    }
```

```
    /// <summary>
    /// Add a rule which triggers when a file is uploaded to an S3 bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role used by EventBridge.</param>
    /// <returns>Async task.</returns>
    private static async Task AddEventRule(string roleArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Creating an EventBridge event that sends an email when an
Amazon S3 object is created.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await _eventBridgeWrapper.PutS3UploadRule(roleArn, eventRuleName,
testBucketName);
        Console.WriteLine($"\tAdded event rule {eventRuleName} for bucket
{testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an SNS target to the rule.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task AddSnsTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Adding a target to the rule to that sends an email when
the rule is triggered.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];
        var topicName = _configuration["topicName"];
        await _eventBridgeWrapper.AddSnsTargetToRule(eventRuleName, topicArn);
        Console.WriteLine($"\tAdded event rule {eventRuleName} with Amazon SNS
target {topicName} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// List the event rules on the default event bus.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListEventRules()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Current event rules:");

        var rules = await _eventBridgeWrapper.ListAllRulesForEventBus();
        rules.ForEach(r => Console.WriteLine($"\tRule: {r.Name} Description:
{r.Description} State: {r.State}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the event target to use a transform.
    /// </summary>
    /// <param name="topicArn">The SNS topic ARN target to update.</param>
    /// <returns>Async task.</returns>
    private static async Task UpdateSnsEventRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Let's update the event target with a transform.");

        var eventRuleName = _configuration["eventRuleName"];
        var testBucketName = _configuration["testBucketName"];

        await
_eventBridgeWrapper.UpdateS3UploadRuleTargetWithTransform(eventRuleName, topicArn);
        Console.WriteLine($"\tUpdated event rule {eventRuleName} with Amazon SNS
target {topicArn} for bucket {testBucketName}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Update the rule to use a custom event pattern.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task UpdateToCustomRule(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("Updating the event pattern to be triggered by a custom
event instead.");

        var eventRuleName = _configuration["eventRuleName"];

        await _eventBridgeWrapper.UpdateCustomEventPattern(eventRuleName);

        Console.WriteLine($"\tUpdated event rule {eventRuleName} to custom
pattern.");
        await _eventBridgeWrapper.UpdateCustomRuleTargetWithTransform(eventRuleName,
            topicArn);

        Console.WriteLine($"\tUpdated event target {topicArn}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Send rule events for a custom rule using the user's email address.
    /// </summary>
    /// <param name="email">The email address to include.</param>
    /// <returns>Async task.</returns>
    private static async Task TriggerCustomRule(string email)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Sending an event to trigger the rule. This will trigger a
subscription email.");

        await _eventBridgeWrapper.PutCustomEmailEvent(email);

        Console.WriteLine($"\tEvents have been sent. Press Enter to continue.");
        Console.ReadLine();

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the targets for a rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListTargets()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the targets for a particular rule.");
```

```
        var eventRuleName = _configuration["eventRuleName"];
        var targets = await _eventBridgeWrapper.ListAllTargetsOnRule(eventRuleName);
        targets.ForEach(t => Console.WriteLine($"\tTarget: {t.Arn} Id: {t.Id} Input:
{t.Input}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List all of the rules for a particular target.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic.</param>
    /// <returns>Async task.</returns>
    private static async Task ListRulesForTarget(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("List all of the rules for a particular target.");

        var rules = await _eventBridgeWrapper.ListAllRuleNamesByTarget(topicArn);
        rules.ForEach(r => Console.WriteLine($"\tRule: {r}"));

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Enable or disable a particular rule.
    /// </summary>
    /// <param name="isEnabled">True to enable the rule, otherwise false.</param>
    /// <returns>Async task.</returns>
    private static async Task ChangeRuleState(bool isEnabled)
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        if (!isEnabled)
        {
            Console.WriteLine($"Disabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.DisableRuleByName(eventRuleName);
        }
        else
        {
            Console.WriteLine($"Enabling the rule: {eventRuleName}");
            await _eventBridgeWrapper.EnableRuleByName(eventRuleName);
```

```
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the current state of the rule.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetRuleState()
    {
        Console.WriteLine(new string('-', 80));
        var eventRuleName = _configuration["eventRuleName"];

        var state = await _eventBridgeWrapper.GetRuleStateByRuleName(eventRuleName);
        Console.WriteLine($"Rule {eventRuleName} is in current state {state}.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="topicArn">The ARN of the SNS topic to clean up.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources(string topicArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        var eventRuleName = _configuration["eventRuleName"];
        if (GetYesNoResponse($"\tDelete all targets and event rule {eventRuleName}?
(y/n)"))
        {
            Console.WriteLine($"\tRemoving all targets from the event rule.");
            await _eventBridgeWrapper.RemoveAllTargetsFromRule(eventRuleName);

            Console.WriteLine($"\tDeleting event rule.");
            await _eventBridgeWrapper.DeleteRuleByName(eventRuleName);
        }

        var topicName = _configuration["topicName"];
        if (GetYesNoResponse($"\tDelete Amazon SNS subscription topic {topicName}?
(y/n)"))
```

```
        {
            Console.WriteLine($"\tDeleting topic.");
            await _snsClient!.DeleteTopicAsync(new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        }

        var bucketName = _configuration["testBucketName"];
        if (GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.S3Objects
                    .Select(o => new KeyVersion { Key = o.Key }).ToList()
            });
            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
        }

        var roleName = _configuration["roleName"];
        if (GetYesNoResponse($"\tDelete role {roleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policy and deleting role.");

            await _iamClient!.DetachRolePolicyAsync(new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess",
            });

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
```

```
                RoleName = roleName
            });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Erstellen Sie eine Klasse, die EventBridge Operationen umschließt.

```
/// <summary>
/// Wrapper for Amazon EventBridge operations.
/// </summary>
public class EventBridgeWrapper
{
    private readonly IAmazonEventBridge _amazonEventBridge;
    private readonly ILogger<EventBridgeWrapper> _logger;

    /// <summary>
    /// Constructor for the EventBridge wrapper.
    /// </summary>
    /// <param name="amazonEventBridge">The injected EventBridge client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public EventBridgeWrapper(IAmazonEventBridge amazonEventBridge,
 ILogger<EventBridgeWrapper> logger)
```

```
    {
        _amazonEventBridge = amazonEventBridge;
        _logger = logger;
    }

    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }

    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty, uses
the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
```

```
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }

    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created in
 a bucket.
```

```
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]" +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }

    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
 event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string ruleName,
 string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();
```

```
        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Update a custom rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
event bus.</param>
```

```
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateCustomRuleTargetWithTransform(string ruleName,
string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
                InputTransformer = new InputTransformer()
                {
                    InputTemplate = "\"Notification: sample event was received.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }

    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
```

```
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }

    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });
```

```
        return response.RuleArn;
    }

    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string targetArn,
string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
```

```
        return targetID;
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }

    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteRuleByName(string ruleName)
    {
        var response = await _amazonEventBridge.DeleteRuleAsync(
            new DeleteRuleRequest()
            {
                Name = ruleName
            });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Aktionen

`DeleteRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Regel anhand ihres Namens.

```
/// <summary>
/// Delete an event rule by name.
/// </summary>
/// <param name="ruleName">The name of the event rule.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteRuleByName(string ruleName)
{
    var response = await _amazonEventBridge.DeleteRuleAsync(
        new DeleteRuleRequest()
        {
            Name = ruleName
        });

    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API-Details finden Sie DeleteRule in der AWS SDK für .NET API-Referenz.

`DescribeRule`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie den Status einer Regel anhand der Regelbeschreibung ab.

```
    /// <summary>
    /// Get the state for a rule by the rule name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="eventBusName">The optional name of the event bus. If empty,
uses the default event bus.</param>
    /// <returns>The state of the rule.</returns>
    public async Task<RuleState> GetRuleStateByRuleName(string ruleName, string?
eventBusName = null)
    {
        var ruleResponse = await _amazonEventBridge.DescribeRuleAsync(
            new DescribeRuleRequest()
            {
                Name = ruleName,
                EventBusName = eventBusName
            });
        return ruleResponse.State;
    }
```

- API-Details finden Sie DescribeRule in der AWS SDK für .NET API-Referenz.

`DisableRule`

Das folgende Codebeispiel zeigt die Verwendung`DisableRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Deaktivieren Sie eine Regel anhand ihres Regelnamens.

```
    /// <summary>
    /// Disable a particular rule on an event bus.
    /// </summary
```

```
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.DisableRuleAsync(
            new DisableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DisableRule in der AWS SDK für .NET API-Referenz.

`EnableRule`

Das folgende Codebeispiel zeigt die Verwendung`EnableRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
    /// <summary>
    /// Enable a particular rule on an event bus.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableRuleByName(string ruleName)
    {
        var ruleResponse = await _amazonEventBridge.EnableRuleAsync(
            new EnableRuleRequest()
            {
                Name = ruleName
            });
        return ruleResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- API-Details finden Sie EnableRule in der AWS SDK für .NET API-Referenz.

`ListRuleNamesByTarget`

Das folgende Codebeispiel zeigt die Verwendung`ListRuleNamesByTarget`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regelnamen mithilfe des Ziels auf.

```
    /// <summary>
    /// List names of all rules matching a target.
    /// </summary>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <returns>The list of rule names.</returns>
    public async Task<List<string>> ListAllRuleNamesByTarget(string targetArn)
    {
        var results = new List<string>();
        var request = new ListRuleNamesByTargetRequest()
        {
            TargetArn = targetArn
        };
        ListRuleNamesByTargetResponse response;
        do
        {
            response = await _amazonEventBridge.ListRuleNamesByTargetAsync(request);
            results.AddRange(response.RuleNames);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
    }
```

- API-Details finden Sie ListRuleNamesByTarget in der AWS SDK für .NET API-Referenz.

`ListRules`

Das folgende Codebeispiel zeigt die Verwendung`ListRules`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regeln für einen Event Bus auf.

```
    /// <summary>
    /// List the rules on an event bus.
    /// </summary>
    /// <param name="eventBusArn">The optional ARN of the event bus. If empty, uses
the default event bus.</param>
    /// <returns>The list of rules.</returns>
    public async Task<List<Rule>> ListAllRulesForEventBus(string? eventBusArn =
null)
    {
        var results = new List<Rule>();
        var request = new ListRulesRequest()
        {
            EventBusName = eventBusArn
        };
        // Get all of the pages of rules.
        ListRulesResponse response;
        do
        {
            response = await _amazonEventBridge.ListRulesAsync(request);
            results.AddRange(response.Rules);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);
```

```
        return results;
    }
```

- API-Details finden Sie ListRules in der AWS SDK für .NET API-Referenz.

`ListTargetsByRule`

Das folgende Codebeispiel zeigt die Verwendung`ListTargetsByRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Ziele für eine Regel mithilfe des Regelnamens auf.

```
    /// <summary>
    /// List all of the targets matching a rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>The list of targets.</returns>
    public async Task<List<Target>> ListAllTargetsOnRule(string ruleName)
    {
        var results = new List<Target>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse response;
        do
        {
            response = await _amazonEventBridge.ListTargetsByRuleAsync(request);
            results.AddRange(response.Targets);
            request.NextToken = response.NextToken;

        } while (response.NextToken is not null);

        return results;
```

```
    }
```

- API-Details finden Sie ListTargetsByRule in der AWS SDK für .NET API-Referenz.

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie ein Ereignis, das einem benutzerdefinierten Muster für eine Regel entspricht.

```
    /// <summary>
    /// Add an event to the event bus that includes an email, message, and time.
    /// </summary>
    /// <param name="email">The email to use in the event detail of the custom
event.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutCustomEmailEvent(string email)
    {
        var eventDetail = new
        {
            UserEmail = email,
            Message = "This event was generated by example code.",
            UtcTime = DateTime.UtcNow.ToString("g")
        };
        var response = await _amazonEventBridge.PutEventsAsync(
            new PutEventsRequest()
            {
                Entries = new List<PutEventsRequestEntry>()
                {
                    new PutEventsRequestEntry()
                    {
                        Source = "ExampleSource",
                        Detail = JsonSerializer.Serialize(eventDetail),
```

```
                        DetailType = "ExampleType"
                    }
                }
            });

        return response.FailedEntryCount == 0;
    }
```

- API-Details finden Sie PutEvents in der AWS SDK für .NET API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Bucket von Amazon Simple Storage Service hinzugefügt wird.

```
    /// <summary>
    /// Create a new event rule that triggers when an Amazon S3 object is created in
 a bucket.
    /// </summary>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="ruleName">The name to give the rule.</param>
    /// <param name="bucketName">The name of the bucket to trigger the event.</
param>
    /// <returns>The ARN of the new rule.</returns>
    public async Task<string> PutS3UploadRule(string roleArn, string ruleName,
 string bucketName)
    {
        string eventPattern = "{" +
                                "\"source\": [\"aws.s3\"]," +
                                    "\"detail-type\": [\"Object Created\"]," +
                                    "\"detail\": {" +
```

```
                                        "\"bucket\": {" +
                                            "\"name\": [\"" + bucketName + "\"]" +
                                        "}" +
                                    "}" +
                                "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Example S3 upload rule for EventBridge",
                RoleArn = roleArn,
                EventPattern = eventPattern
            });

        return response.RuleArn;
    }
```

Erstellen Sie eine Regel, die ein benutzerdefiniertes Muster verwendet.

```
    /// <summary>
    /// Update a rule to use a custom defined event pattern.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <returns>The ARN of the updated rule.</returns>
    public async Task<string> UpdateCustomEventPattern(string ruleName)
    {
        string customEventsPattern = "{" +
                                     "\"source\": [\"ExampleSource\"]," +
                                     "\"detail-type\": [\"ExampleType\"]" +
                                     "}";

        var response = await _amazonEventBridge.PutRuleAsync(
            new PutRuleRequest()
            {
                Name = ruleName,
                Description = "Custom test rule",
                EventPattern = customEventsPattern
            });

        return response.RuleArn;
    }
```

- API-Details finden Sie PutRule in der AWS SDK für .NET API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Amazon-SNS-Thema als Ziel für eine Regel hinzu.

```
    /// <summary>
    /// Add an Amazon SNS target topic to a rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule to update.</param>
    /// <param name="targetArn">The ARN of the Amazon SNS target.</param>
    /// <param name="eventBusArn">The optional event bus name, uses default if
empty.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> AddSnsTargetToRule(string ruleName, string targetArn,
string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        // Create the list of targets and add a new target.
        var targets = new List<Target>
        {
            new Target()
            {
                Arn = targetArn,
                Id = targetID
            }
        };

        // Add the targets to the rule.
```

```
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });

        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return targetID;
    }
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
    /// <summary>
    /// Update an Amazon S3 object created rule with a transform on the target.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="targetArn">The ARN of the target.</param>
    /// <param name="eventBusArn">Optional event bus ARN. If empty, uses the default
event bus.</param>
    /// <returns>The ID of the target.</returns>
    public async Task<string> UpdateS3UploadRuleTargetWithTransform(string ruleName,
string targetArn, string? eventBusArn = null)
    {
        var targetID = Guid.NewGuid().ToString();

        var targets = new List<Target>
        {
            new Target()
            {
                Id = targetID,
                Arn = targetArn,
```

```
                InputTransformer = new InputTransformer()
                {
                    InputPathsMap = new Dictionary<string, string>()
                    {
                        {"bucket", "$.detail.bucket.name"},
                        {"time", "$.time"}
                    },
                    InputTemplate = "\"Notification: an object was uploaded to
bucket <bucket> at <time>.\""
                }
            }
        };
        var response = await _amazonEventBridge.PutTargetsAsync(
            new PutTargetsRequest()
            {
                EventBusName = eventBusArn,
                Rule = ruleName,
                Targets = targets,
            });
        if (response.FailedEntryCount > 0)
        {
            response.FailedEntries.ForEach(e =>
            {
                _logger.LogError(
                    $"Failed to add target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }
        return targetID;
    }
```

- API-Details finden Sie PutTargets in der AWS SDK für .NET API-Referenz.

`RemoveTargets`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTargets`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Entfernen Sie alle Ziele für eine Regel mithilfe des Regelnamens.

```
    /// <summary>
    /// Delete an event rule by name.
    /// </summary>
    /// <param name="ruleName">The name of the event rule.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> RemoveAllTargetsFromRule(string ruleName)
    {
        var targetIds = new List<string>();
        var request = new ListTargetsByRuleRequest()
        {
            Rule = ruleName
        };
        ListTargetsByRuleResponse targetsResponse;
        do
        {
            targetsResponse = await
_amazonEventBridge.ListTargetsByRuleAsync(request);
            targetIds.AddRange(targetsResponse.Targets.Select(t => t.Id));
            request.NextToken = targetsResponse.NextToken;

        } while (targetsResponse.NextToken is not null);

        var removeResponse = await _amazonEventBridge.RemoveTargetsAsync(
            new RemoveTargetsRequest()
            {
                Rule = ruleName,
                Ids = targetIds
            });

        if (removeResponse.FailedEntryCount > 0)
        {
            removeResponse.FailedEntries.ForEach(e =>
            {
```

```
                _logger.LogError(
                    $"Failed to remove target {e.TargetId}: {e.ErrorMessage}, code
{e.ErrorCode}");
            });
        }

        return removeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie RemoveTargets in der AWS SDK für .NET API-Referenz.

EventBridge Scheduler-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von EventBridge Scheduler Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo Scheduler EventBridge

Das folgende Codebeispiel zeigt, wie Sie mit EventBridge Scheduler beginnen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static class HelloScheduler
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the EventBridge Scheduler service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
            ).Build();

        // Now the client is available for injection.
        var schedulerClient = host.Services.GetRequiredService<IAmazonScheduler>();

        // You can use await and any of the async methods to get a response, or a
 paginator to list schedules or groups.
        var results = new List<ScheduleSummary>();
        var paginateSchedules = schedulerClient.Paginators.ListSchedules(
            new ListSchedulesRequest());
        Console.WriteLine(
            $"Hello AWS Scheduler! Let's list schedules in your account.");
        // Get the entire list using the paginator.
        await foreach (var schedule in paginateSchedules.Schedules)
        {
            results.Add(schedule);
        }
        Console.WriteLine($"\tTotal of {results.Count} schedule(s) available.");
        results.ForEach(s => Console.WriteLine($"\tSchedule: {s.Name}"));
    }
}
```

- API-Details finden Sie ListSchedules in der AWS SDK für .NET API-Referenz.

Aktionen

`CreateSchedule`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchedule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines when
 the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which the
 schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the schedule
 after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible time
 window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false otherwise.</
returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
```

```
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule '{name}' due to a conflict.
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while creating schedule '{name}' "
+
                             $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie CreateSchedule in der AWS SDK für .NET API-Referenz.

`CreateScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateScheduleGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await _amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;
```

```
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
 conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie CreateScheduleGroup in der AWS SDK für .NET API-Referenz.

`DeleteSchedule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSchedule`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false otherwise.</
returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
```

```
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteSchedule in der AWS SDK für .NET API-Referenz.

`DeleteScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteScheduleGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource was
not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteScheduleGroup in der AWS SDK für .NET API-Referenz.

Szenarien

Geplante Ereignisse

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Stellen Sie einen CloudFormation Stack mit den erforderlichen Ressourcen bereit.
- Erstellen Sie eine EventBridge Scheduler-Zeitplangruppe.
- Erstellen Sie einen einmaligen EventBridge Scheduler-Zeitplan mit einem flexiblen Zeitfenster.
- Erstellen Sie einen wiederkehrenden EventBridge Scheduler-Zeitplan mit einer bestimmten Rate.
- Löschen Sie EventBridge Scheduler, den Zeitplan und die Zeitplangruppe.
- Bereinigen Sie die Ressourcen und löschen Sie den Stack.

SDK für .NET

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using SchedulerActions;
using Exception = System.Exception;

namespace SchedulerScenario;

public class SchedulerWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
```

```
    1. Prepare the Application:
       - Prompt the user for an email address to use for the subscription for the
 SNS topic subscription.
       - Prompt the user for a name for the Cloud Formation stack.
       - Deploy the Cloud Formation template in resources/cfn_template.yaml for
 resource creation.
       - Store the outputs of the stack into variables for use in the scenario.
       - Create a schedule group for all schedules.

    2. Create one-time Schedule:
       - Create a one-time schedule to send an initial event.
       - Use a Flexible Time Window and set the schedule to delete after completion.
       - Wait for the user to receive the event email from SNS.

    3. Create a time-based schedule:
       - Prompt the user for how many X times per Y hours a recurring event should
 be scheduled.
       - Create the scheduled event for X times per hour for Y hours.
       - Wait for the user to receive the event email from SNS.
       - Delete the schedule when the user is finished.

    4. Clean up:
       - Prompt the user for y/n answer if they want to destroy the stack and clean
 up all resources.
       - Delete the schedule group.
       - Destroy the Cloud Formation stack and wait until the stack has been
 removed.
    */

    public static ILogger<SchedulerWorkflow> _logger = null!;
    public static SchedulerWrapper _schedulerWrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _roleArn = null!;
    private static string _snsTopicArn = null!;

    public static bool _interactive = true;
    private static string _stackName = "default-scheduler-scenario-stack-name";
    private static string _scheduleGroupName = "scenario-schedules-group";
    private static string _stackResourcePath = "../../../../../../scenarios/
features/eventbridge_scheduler/resources/cfn_template.yaml";

    public static async Task Main(string[] args)
    {
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonScheduler>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<SchedulerWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<SchedulerWorkflow>();

            _schedulerWrapper =
host.Services.GetRequiredService<SchedulerWrapper>();
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon EventBridge Scheduler Scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await CreateOneTimeSchedule();
                Console.WriteLine(new string('-', 80));

                Console.WriteLine(new string('-', 80));
                await CreateRecurringSchedule();
                Console.WriteLine(new string('-', 80));
            }
```

```
            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("Amazon EventBridge Scheduler scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        try
        {
            // Prompt the user for an email address to use for the subscription.
            Console.WriteLine("\nThis example creates resources in a CloudFormation
stack, including an SNS topic" +
                              "\nthat will be subscribed to the EventBridge Scheduler
events. " +
                              "\n\nYou will need to confirm the subscription in order to
receive event emails. ");

            var emailAddress = PromptUserForEmail();

            // Prompt the user for a name for the CloudFormation stack
            _stackName = PromptUserForStackName();

            // Deploy the CloudFormation stack
            var deploySuccess = await DeployCloudFormationStack(_stackName,
emailAddress);

            if (deploySuccess)
            {
```

```
                // Create a schedule group for all schedules
                await
_schedulerWrapper.CreateScheduleGroupAsync(_scheduleGroupName);

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }
        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="email">The email to use for the subscription.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
string email)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = { Capability.CAPABILITY_NAMED_IAM }
            };

            // If an email is provided, set the parameter.
            if (!string.IsNullOrWhiteSpace(email))
            {
                request.Parameters = new List<Parameter>()
                {
                    new() { ParameterKey = "email", ParameterValue = email }
                };
```

```
            }

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                // Wait for the stack to be in CREATE_COMPLETE state
                bool stackCreated = await WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    // Retrieve the output values
                    var success = await GetStackOutputs(response.StackId);
                    return success;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already exists.
Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName, email);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
```

```
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="client">The CloudFormation client.</param>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds.

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation complete.");
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                        describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    Console.WriteLine("CloudFormation stack creation failed.");
                    return false;
                }
            }
```

```
            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Retrieves the output values from the CloudFormation stack.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    private static async Task<bool> GetStackOutputs(string stackId)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest { StackName =
stackId };

            var describeStacksResponse =
                await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                var stack = describeStacksResponse.Stacks[0];
                _roleArn = GetStackOutputValue(stack, "RoleARN");
                _snsTopicArn = GetStackOutputValue(stack, "SNStopicARN");
                return true;
            }
            else
            {
                _logger.LogError($"No stack found for stack outputs: {stackId}");
                return false;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(
                ex, $"Failed to retrieve CloudFormation stack outputs: {stackId}");
            return false;
```

```
        }
    }

    /// <summary>
    /// Get an output value by key from a CloudFormation stack.
    /// </summary>
    /// <param name="stack">The CloudFormation stack.</param>
    /// <param name="outputKey">The key of the output.</param>
    /// <returns>The value as a string.</returns>
    private static string GetStackOutputValue(Stack stack, string outputKey)
    {
        var output = stack.Outputs.First(o => o.OutputKey == outputKey);
        var outputValue = output.OutputValue;
        Console.WriteLine($"Stack output {outputKey}: {outputValue}");
        return outputValue;
    }

    /// <summary>
    /// Creates a one-time schedule to send an initial event.
    /// </summary>
    /// <returns>True if the one-time schedule was created successfully.</returns>
    public static async Task<bool> CreateOneTimeSchedule()
    {
        var scheduleName =
            PromptUserForResourceName("Enter a name for the one-time schedule:");

        Console.WriteLine($"Creating a one-time schedule named '{scheduleName}' " +
                          $"\nto send an initial event in 1 minute with a flexible
time window...");
        try
        {
            // Create a one-time schedule with a flexible time
            // window set to delete after completion.
            // You may also set a timezone instead of using UTC.
            var scheduledTime = DateTime.UtcNow.AddMinutes(1).ToString("s");

            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"at({scheduledTime})",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"One time scheduled event test from schedule {scheduleName}.",
                true,
```

```
                useFlexibleTimeWindow: true);

            Console.WriteLine($"Subscription email will receive an email from this
event.");
            Console.WriteLine($"You must confirm your subscription to receive event
emails.");

            Console.WriteLine($"One-time schedule '{scheduleName}' created
successfully.");
            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the one-time
schedule '{scheduleName}'.");
            return false;
        }
    }

    /// <summary>
    /// Create a recurring schedule to send events at a specified rate in minutes.
    /// </summary>
    /// <returns>True if the recurring schedule was created successfully.</returns>
    public static async Task<bool> CreateRecurringSchedule()
    {
        Console.WriteLine("Creating a recurring schedule to send events for one
hour...");

        try
        {
            // Prompt the user for a schedule name.
            var scheduleName =
                PromptUserForResourceName("Enter a name for the recurring schedule:
");

            // Prompt the user for the schedule rate (in minutes).
            var scheduleRateInMinutes =
```

```
                PromptUserForInteger("Enter the desired schedule rate (in minutes): 
");

            // Create the recurring schedule.
            var createSuccess = await _schedulerWrapper.CreateScheduleAsync(
                scheduleName,
                $"rate({scheduleRateInMinutes} minutes)",
                _scheduleGroupName,
                _snsTopicArn,
                _roleArn,
                $"Recurrent event test from schedule {scheduleName}.");

            Console.WriteLine($"Subscription email will receive an email from this 
event.");
            Console.WriteLine($"You must confirm your subscription to receive event 
emails.");

            // Delete the schedule when the user is finished.
            if (!_interactive || GetYesNoResponse($"Are you ready to delete the 
'{scheduleName}' schedule? (y/n)"))
            {
                await _schedulerWrapper.DeleteScheduleAsync(scheduleName, 
_scheduleGroupName);
            }

            return createSuccess;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(ex, $"The target with ARN '{_snsTopicArn}' was not 
found.");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while creating the recurring 
schedule.");
            return false;
        }
    }

    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
```

```
    /// <returns>True if the cleanup was successful.</returns>
    public static async Task<bool> Cleanup()
    {
        // Prompt the user to confirm cleanup.
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete all resources created by this scenario? (y/n) ");
        if (cleanup)
        {
            try
            {
                // Delete the schedule group.
                var groupDeleteSuccess = await
_schedulerWrapper.DeleteScheduleGroupAsync(_scheduleGroupName);

                // Destroy the CloudFormation stack and wait for it to be removed.
                var stackDeleteSuccess = await DeleteCloudFormationStack(_stackName,
false);

                return groupDeleteSuccess && stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex,
                    "An error occurred while cleaning up the resources.");
                return false;
            }
        }
        _logger.LogInformation("EventBridge Scheduler scenario is complete.");
        return true;
    }

    /// <summary>
    /// Delete the resources in the stack and wait for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName, bool
forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };
```

```
        if (forceDelete)
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{_stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(_stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{_stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{_stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Wait for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the stack.</param>
    /// <param name="forceDelete">True to force delete the stack.</param>
    /// <returns>True if successful.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 10;
        const int retryDelay = 30000; // 30 seconds

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
```

```
            };

            try
            {
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
describeStacksResponse.Stacks[0].StackStatus == StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }
                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_FAILED)
                {
                    // Try one time to force delete.
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                // Stack does not exist, so it has been successfully deleted.
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack '{stackName}'
to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
```

```
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompt the user for a valid email address.
    /// </summary>
    /// <returns>The valid email address.</returns>
    private static string PromptUserForEmail()
    {
        if (_interactive)
        {
            Console.WriteLine("Enter an email address to use for event
subscriptions: ");

            string email = Console.ReadLine()!;

            if (!IsValidEmail(email))
            {
                Console.WriteLine("Invalid email address. Please try again.");
                return PromptUserForEmail();
            }
            return email;
        }
        // Used when running without user prompts.
        return "";
    }

    /// <summary>
    /// Prompt the user for a non-empty stack name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForStackName()
    {
        Console.WriteLine("Enter a name for the AWS Cloud Formation Stack: ");
        if (_interactive)
        {
            string stackName = Console.ReadLine()!;
            var regex = "[a-zA-Z][-a-zA-Z0-9]|arn:[-a-zA-Z0-9:/._+]";
            if (!Regex.IsMatch(stackName, regex))
            {
```

```
                Console.WriteLine(
                    $"Invalid stack name. Please use a name that matches the pattern
{regex}.");
                return PromptUserForStackName();
            }

            return stackName;
        }
        // Used when running without user prompts.
        return _stackName;
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static string PromptUserForResourceName(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
            string resourceName = Console.ReadLine()!;
            var regex = "[0-9a-zA-Z-_.]+";
            if (!Regex.IsMatch(resourceName, regex))
            {
                Console.WriteLine($"Invalid resource name. Please use a name that
matches the pattern {regex}.");
                return PromptUserForResourceName(prompt);
            }
            return resourceName!;
        }
        // Used when running without user prompts.
        return "resource-" + Guid.NewGuid();
    }

    /// <summary>
    /// Prompt the user for a non-empty resource name.
    /// </summary>
    /// <returns>The valid stack name</returns>
    private static int PromptUserForInteger(string prompt)
    {
        if (_interactive)
        {
            Console.WriteLine(prompt);
```

```
            string stringResponse = Console.ReadLine()!;
            if (string.IsNullOrWhiteSpace(stringResponse) ||
                !Int32.TryParse(stringResponse, out var intResponse))
            {
                Console.WriteLine($"Invalid integer. ");
                return PromptUserForInteger(prompt);
            }
            return intResponse!;
        }
        // Used when running without user prompts.
        return 1;
    }

    /// <summary>
    /// Use System Mail to check for a valid email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email address.</returns>
    private static bool IsValidEmail(string email)
    {
        try
        {
            var mailAddress = new System.Net.Mail.MailAddress(email);
            return mailAddress.Address == email;
        }
        catch
        {
            // Invalid emails will cause an exception, return false.
            return false;
        }
    }
}
```

Wrapper für Service-Vorgänge.

```
using Amazon.Scheduler;
using Amazon.Scheduler.Model;
using Microsoft.Extensions.Logging;

namespace SchedulerActions;

/// <summary>
```

```
/// Wrapper class for Amazon EventBridge Scheduler operations.
/// </summary>
public class SchedulerWrapper
{
    private readonly IAmazonScheduler _amazonScheduler;
    private readonly ILogger<SchedulerWrapper> _logger;

    /// <summary>
    /// Constructor for the SchedulerWrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EventBridge Scheduler client.</
param>
    /// <param name="logger">The injected logger.</param>
    public SchedulerWrapper(IAmazonScheduler amazonScheduler,
 ILogger<SchedulerWrapper> logger)
    {
        _amazonScheduler = amazonScheduler;
        _logger = logger;
    }

    /// <summary>
    /// Creates a new schedule in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule.</param>
    /// <param name="scheduleExpression">The schedule expression that defines when
 the schedule should run.</param>
    /// <param name="scheduleGroupName">The name of the schedule group to which the
 schedule should be added.</param>
    /// <param name="deleteAfterCompletion">Indicates whether to delete the schedule
 after completion.</param>
    /// <param name="useFlexibleTimeWindow">Indicates whether to use a flexible time
 window for the schedule.</param>
    /// <param name="targetArn">ARN of the event target.</param>
    /// <param name="roleArn">Execution Role ARN.</param>
    /// <returns>True if the schedule was created successfully, false otherwise.</
returns>
    public async Task<bool> CreateScheduleAsync(
            string name,
            string scheduleExpression,
            string scheduleGroupName,
            string targetArn,
            string roleArn,
            string input,
            bool deleteAfterCompletion = false,
```

```
            bool useFlexibleTimeWindow = false)
    {
        try
        {
            int hoursToRun = 1;
            int flexibleTimeWindowMinutes = 10;

            var request = new CreateScheduleRequest
            {
                Name = name,
                ScheduleExpression = scheduleExpression,
                GroupName = scheduleGroupName,
                Target = new Target { Arn = targetArn, RoleArn = roleArn, Input =
input },
                ActionAfterCompletion = deleteAfterCompletion
                    ? ActionAfterCompletion.DELETE
                    : ActionAfterCompletion.NONE,
                StartDate = DateTime.UtcNow, // Ignored for one-time schedules.
                EndDate =
                    DateTime.UtcNow
                        .AddHours(hoursToRun) // Ignored for one-time schedules.
            };
            // Allow a flexible time window if the caller specifies it.
            request.FlexibleTimeWindow = new FlexibleTimeWindow
            {
                Mode = useFlexibleTimeWindow
                    ? FlexibleTimeWindowMode.FLEXIBLE
                    : FlexibleTimeWindowMode.OFF,
                MaximumWindowInMinutes = useFlexibleTimeWindow
                    ? flexibleTimeWindowMinutes
                    : null
            };

            var response = await _amazonScheduler.CreateScheduleAsync(request);

            Console.WriteLine($"Successfully created schedule '{name}' " +
                              $"in schedule group '{scheduleGroupName}':
{response.ScheduleArn}.");
            return true;
        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
```

```
            _logger.LogError($"Failed to create schedule '{name}' due to a conflict.
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating schedule '{name}' "
+
                              $"in schedule group '{scheduleGroupName}':
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Creates a new schedule group in Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group.</param>
    /// <returns>True if the schedule group was created successfully, false
otherwise.</returns>
    public async Task<bool> CreateScheduleGroupAsync(string name)
    {
        try
        {
            var request = new CreateScheduleGroupRequest { Name = name };

            var response = await _amazonScheduler.CreateScheduleGroupAsync(request);

            Console.WriteLine($"Successfully created schedule group '{name}':
{response.ScheduleGroupArn}.");
            return true;

        }
        catch (ConflictException ex)
        {
            // If the name is not unique, a ConflictException will be thrown.
            _logger.LogError($"Failed to create schedule group '{name}' due to a
conflict. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
```

```
                $"An error occurred while creating schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an existing schedule from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule to delete.</param>
    /// <param name="groupName">The group name of the schedule to delete.</param>
    /// <returns>True if the schedule was deleted successfully, false otherwise.</
returns>
    public async Task<bool> DeleteScheduleAsync(string name, string groupName)
    {
        try
        {
            var request = new DeleteScheduleRequest
            {
                Name = name,
                GroupName = groupName
            };

            await _amazonScheduler.DeleteScheduleAsync(request);

            Console.WriteLine($"Successfully deleted schedule with name '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule with ID '{name}' because the resource
 was not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule with ID '{name}':
 {ex.Message}");
            return false;
        }
    }
```

```
    /// <summary>
    /// Deletes an existing schedule group from Amazon EventBridge Scheduler.
    /// </summary>
    /// <param name="name">The name of the schedule group to delete.</param>
    /// <returns>True if the schedule group was deleted successfully, false
 otherwise.</returns>
    public async Task<bool> DeleteScheduleGroupAsync(string name)
    {
        try
        {
            var request = new DeleteScheduleGroupRequest { Name = name };

            await _amazonScheduler.DeleteScheduleGroupAsync(request);

            Console.WriteLine($"Successfully deleted schedule group '{name}'.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError(
                $"Failed to delete schedule group '{name}' because the resource was
 not found: {ex.Message}");
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while deleting schedule group '{name}':
 {ex.Message}");
            return false;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule

- DeleteScheduleGroups

Beispiele für Amazon Glacier mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Glacier Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Amazon Glacier

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Glacier beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
```

```
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

- API-Details finden Sie ListVaults in der AWS SDK für .NET API-Referenz.

Aktionen

`AddTagsToVault`

Das folgende Codebeispiel zeigt die Verwendung`AddTagsToVault`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add tags to the items in an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to add tags to.</param>
```

```
    /// <param name="key">The name of the object to tag.</param>
    /// <param name="value">The tag value to add.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddTagsToVaultAsync(string vaultName, string key, string
value)
    {
        var request = new AddTagsToVaultRequest
        {
            Tags = new Dictionary<string, string>
                {
                    { key, value },
                },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.AddTagsToVaultAsync(request);
        return response.HttpStatusCode == HttpStatusCode.NoContent;
    }
```

- API-Details finden Sie AddTagsToVault in der AWS SDK für .NET API-Referenz.

`CreateVault`

Das folgende Codebeispiel zeigt die Verwendung`CreateVault`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to create.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateVaultAsync(string vaultName)
    {
```

```
        var request = new CreateVaultRequest
        {
            // Setting the AccountId to "-" means that
            // the account associated with the current
            // account will be used.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.CreateVaultAsync(request);

        Console.WriteLine($"Created {vaultName} at: {response.Location}");

        return response.HttpStatusCode == HttpStatusCode.Created;
    }
```

- API-Details finden Sie CreateVault in der AWS SDK für .NET API-Referenz.

`DescribeVault`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVault`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to describe.</param>
    /// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
    public async Task<string> DescribeVaultAsync(string vaultName)
    {
        var request = new DescribeVaultRequest
        {
            AccountId = "-",
            VaultName = vaultName,
```

```
        };

        var response = await _glacierService.DescribeVaultAsync(request);

        // Display the information about the vault.
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber of Archives:
{response.NumberOfArchives}\tSize (in bytes): {response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

- API-Details finden Sie DescribeVault in der AWS SDK für .NET API-Referenz.

`InitiateJob`

Das folgende Codebeispiel zeigt die Verwendung`InitiateJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Archiv aus einem Tresor ab. In diesem Beispiel wird die ArchiveTransferManager Klasse verwendet. API-Details finden Sie unter ArchiveTransferManager.

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
```

```
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string vaultName,
string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then polling
SQS queue for the archive to be available.");
            Console.WriteLine("When the archive is available, downloading will
begin.");
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
options);

            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
```

```
    }
```

- API-Details finden Sie InitiateJob in der AWS SDK für .NET API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List Amazon S3 Glacier jobs.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list jobs for.</param>
    /// <returns>A list of Amazon S3 Glacier jobs.</returns>
    public async Task<List<GlacierJobDescription>> ListJobsAsync(string vaultName)
    {
        var request = new ListJobsRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the current account.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListJobsAsync(request);

        return response.JobList;
    }
```

- API-Details finden Sie ListJobs in der AWS SDK für .NET API-Referenz.

`ListTagsForVault`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForVault`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

- API-Details finden Sie ListTagsForVault in der AWS SDK für .NET API-Referenz.

`ListVaults`

Das folgende Codebeispiel zeigt die Verwendung`ListVaults`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// List the Amazon S3 Glacier vaults associated with the current account.
/// </summary>
/// <returns>A list containing information about each vault.</returns>
public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
{
    var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
        new ListVaultsRequest { AccountId = "-" });
    var vaultList = new List<DescribeVaultOutput>();

    await foreach (var vault in glacierVaultPaginator.VaultList)
    {
        vaultList.Add(vault);
    }

    return vaultList;
}
```

- API-Details finden Sie ListVaults in der AWS SDK für .NET API-Referenz.

`UploadArchive`

Das folgende Codebeispiel zeigt die Verwendung`UploadArchive`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Upload an object to an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
   /// the archive to.</param>
   /// <param name="archiveFilePath">The file path of the archive to upload to the
vault.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
string archiveFilePath)
   {
       try
       {
           var manager = new ArchiveTransferManager(_glacierService);

           // Upload an archive.
           var response = await manager.UploadAsync(vaultName, "upload archive
test", archiveFilePath);
           return response.ArchiveId;
       }
       catch (AmazonGlacierException ex)
       {
           Console.WriteLine(ex.Message);
           return string.Empty;
       }
   }
```

- API-Details finden Sie UploadArchive in der AWS SDK für .NET API-Referenz.

AWS Glue Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für .NET

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;
```

```
    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- API-Details finden Sie ListJobs in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die AWS Glue Funktionen umschließt, die im Szenario verwendet werden.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;
```

```
    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be executed.</
param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service (Amazon
 S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
```

```
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName, string
 bucketUrl, string jobName, string roleName, string description, string scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };
```

```
        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new DeleteCrawlerRequest
{ Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
```

```
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
 found.");
        return null;
    }


    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
```

```
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us behind
the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
```

```
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new ListJobsRequest
{ MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
```

```
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonGlue>()
            .AddTransient<GlueWrapper>()
            .AddTransient<UiWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
        .CreateLogger<GlueBasics>();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
```

```
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // These values are stored in settings.json
        // Once you have run the CDK script to deploy the resources,
        // edit the file to set "BucketName", "RoleName", and "ScriptURL"
        // to the appropriate values. Also set "CrawlerName" to the name
        // you want to give the crawler when it is created.
        string bucketName = _configuration["BucketName"]!;
        string bucketUrl = _configuration["BucketUrl"]!;
        string crawlerName = _configuration["CrawlerName"]!;
        string roleName = _configuration["RoleName"]!;
        string sourceData = _configuration["SourceData"]!;
        string dbName = _configuration["DbName"]!;
        string cron = _configuration["Cron"]!;
        string scriptUrl = _configuration["ScriptURL"]!;
        string jobName = _configuration["JobName"]!;

        var wrapper = host.Services.GetRequiredService<GlueWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UiWrapper>();

        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
```

```
            Console.WriteLine($"The crawler {crawlerName} is now ready for use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on {database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
```

```
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState == "STOPPED"
||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState == "TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
```

```
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
        {

 Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}
```

```
namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the URL
 to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
 tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
 demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
```

```
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun

- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be executed.</
param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service (Amazon
 S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
```

```
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie CreateCrawler in der AWS SDK für .NET API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName, string
 bucketUrl, string jobName, string roleName, string description, string scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
```

```
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie CreateJob in der AWS SDK für .NET API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new DeleteCrawlerRequest
{ Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteCrawler in der AWS SDK für .NET API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteDatabase in der AWS SDK für .NET API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteJob in der AWS SDK für .NET API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK für .NET API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }
```

- API-Details finden Sie GetCrawler in der AWS SDK für .NET API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- API-Details finden Sie GetDatabase in der AWS SDK für .NET API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- API-Details finden Sie GetJobRun in der AWS SDK für .NET API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
```

```
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };

        // No need to loop to get all the log groups--the SDK does it for us behind
the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- API-Details finden Sie GetJobRuns in der AWS SDK für .NET API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }
```

- API-Details finden Sie GetTables in der AWS SDK für .NET API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
```

```
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new ListJobsRequest
{ MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- API-Details finden Sie ListJobs in der AWS SDK für .NET API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };
```

```
        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie StartCrawler in der AWS SDK für .NET API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };
```

```
        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- API-Details finden Sie StartJobRun in der AWS SDK für .NET API-Referenz.

IAM-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
```

```
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
```

```
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
```

```
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName, string
 description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
 { UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
```

```
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new DeletePolicyRequest
{ PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
```

```
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
 DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</param>
```

```
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest { UserName
= userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
```

```
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>> ListAttachedRolePoliciesAsync(string
roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new ListAttachedRolePoliciesRequest
{ RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
```

```
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator = _IAMService.Paginators.ListRolePolicies(new
ListRolePoliciesRequest { RoleName = roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
```

```
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
```

```
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string roleName,
 string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM policy.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string policyName,
 string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
```

```
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId = accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for use.");

        // Define a role policy document that allows the new user
        // to assume the role.
```

```
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon S3
buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
```

```
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
```

```
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
 secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await s3Wrapper.AssumeS3RoleAsync("temporary-
session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
 {bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);
```

```
        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
```

```
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
```

```
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
```

```
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it full
access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries to
list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
```

```
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }
```

```
        PressEnter();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für .NET API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
```

```
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- API-Details finden Sie CreateAccessKey in der AWS SDK für .NET API-Referenz.

`CreateInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung`CreateInstanceProfile`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
```

```
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
```

```
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }
```

```
        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- API-Details finden Sie CreateInstanceProfile in der AWS SDK für .NET API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- API-Details finden Sie CreatePolicy in der AWS SDK für .NET API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- API-Details finden Sie CreateRole in der AWS SDK für .NET API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
```

```
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName, string
 description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK für .NET API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
 { UserName = userName });
        return response.User;
```

```
    }
```

- API-Details finden Sie CreateUser in der AWS SDK für .NET API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK für .NET API-Referenz.

`DeleteInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung`DeleteInstanceProfile`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
```

```
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }
```

- API-Details finden Sie DeleteInstanceProfile in der AWS SDK für .NET API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
```

```
        var response = await _IAMService.DeletePolicyAsync(new DeletePolicyRequest
{ PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeletePolicy in der AWS SDK für .NET API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteRole in der AWS SDK für .NET API-Referenz.

`DeleteRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRolePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteRolePolicy in der AWS SDK für .NET API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteUser in der AWS SDK für .NET API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an IAM user policy.
```

```
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
 userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
 DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteUserPolicy in der AWS SDK für .NET API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
```

```
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK für .NET API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK für .NET API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
{ PolicyArn = policyArn });
        return response.Policy;
    }
```

- API-Details finden Sie GetPolicy in der AWS SDK für .NET API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an IAM role.
```

```
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }
```

- API-Details finden Sie GetRole in der AWS SDK für .NET API-Referenz.

`GetUser`

Das folgende Codebeispiel zeigt die Verwendung`GetUser`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest { UserName
= userName });
        return response.User;
    }
```

- API-Details finden Sie GetUser in der AWS SDK für .NET API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>> ListAttachedRolePoliciesAsync(string
roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new ListAttachedRolePoliciesRequest
{ RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK für .NET API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- API-Details finden Sie ListGroups in der AWS SDK für .NET API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- API-Details finden Sie ListPolicies in der AWS SDK für .NET API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator = _IAMService.Paginators.ListRolePolicies(new
ListRolePoliciesRequest { RoleName = roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- API-Details finden Sie ListRolePolicies in der AWS SDK für .NET API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
```

```
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }
```

- API-Details finden Sie ListRoles in der AWS SDK für .NET API-Referenz.

Listet SAML-Anbieter auf

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für .NET -API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- API-Details finden Sie ListUsers in der AWS SDK für .NET API-Referenz.

`PutRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutRolePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string roleName,
 string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie PutRolePolicy in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper = host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several AWS
resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways to
make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide book,
movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each contain
a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across several
Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the Auto
Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs 'server_startup_script.sh'
when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the `server.py`
script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2 instances,
each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3, _autoScalerWrapper.GroupName,
zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or continue
with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGroupN
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.LoadB
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper.Lo
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port, ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);
```

```
            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for your
 default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management Console.
\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port, ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
 ipString);
                }
            }
            loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
 browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
```

```
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite of
these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
```

```
                            $"The table name is contained in a Systems Manager
 parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                            $"To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service fails,
 the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
 customer with a somewhat better experience than failure.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
 "static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
 endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
 checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
 _smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in the
 target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
```

```
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile that
contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns either
a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo, a
deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not for
Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the Auto
Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load balancer
can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");
```

```
        Console.WriteLine($"\nNow, checking target health indicates that the
 instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two for
 the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
 endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
 rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by an
 auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler start
 a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
 instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a successful
 recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the load
 balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance typically
 takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
 instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
 checks mean all instances report as unhealthy.");

        await
 _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter, "this-
is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
 continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
 return a static response rather than fail");
```

```
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account, we
can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBalanc
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGroup
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupName)
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
```

```
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
 incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
```

```
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with a
specified name.
    /// An instance's associated profile defines a role that is assumed by the
```

```
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to the
role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
```

```
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
```

```
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
```

```
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
    /// The launch template specifies a Bash script in its user data field that runs
after
```

```
   /// the instance is started. This script installs the Python packages and starts
a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to create
and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
                   }
```

```
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                  $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"An error occurred while listing availability zones.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with size
 {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
    }

    /// <summary>
```

```
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
```

```
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId} does
not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the subnets.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name {_launchTemplateName}
was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
```

```
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await _amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
```

```
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile is
replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate with
the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association for
the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
```

```
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance {instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance {instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not found");
            }

            throw;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while replacing the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for {instanceId}.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
```

```
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
```

```
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the calling
computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP address
while running this example.
    /// If you do, be sure to remove public access when you're done.
```

```
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be open
to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }
```

```
    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName, string
 targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
```

```
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened times
 and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer exists.</
param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
```

```
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
```

```
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
```

```
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
```

```
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn = targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
```

```
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books, movies,
 and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
```

```
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
```

```
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await File.ReadAllTextAsync(recommendationsPath);
        var records =
            JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
```

```
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-architecture-
table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
```

```
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Beispiele für Amazon Keyspaces unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Keyspaces Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Keyspaces

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von Amazon Keyspaces beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace KeyspacesActions;

public class HelloKeyspaces
{
    private static ILogger logger = null!;
```

```
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient = host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- API-Details finden Sie ListKeyspaces in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die zeitpunktbezogene Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.

- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using System.Security.Cryptography.X509Certificates;
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;


namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
```

```
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonKeyspaces>()
            .AddTransient<KeyspacesWrapper>()
            .AddTransient<CassandraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<KeyspacesBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var keyspacesWrapper = host.Services.GetRequiredService<KeyspacesWrapper>();
        var uiMethods = new UiMethods();

        var keyspaceName = configuration["KeyspaceName"];
        var tableName = configuration["TableName"];

        bool success; // Used to track the results of some operations.

        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the keyspace.
        var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

        // Wait for the keyspace to be available. GetKeyspace results in a
        // resource not found error until it is ready for use.
        try
        {
            var getKeyspaceArn = "";
            Console.Write($"Created {keyspaceName}. Waiting for it to become
available. ");
            do
            {
```

```
                getKeyspaceArn = await keyspacesWrapper.GetKeyspace(keyspaceName);
                Console.Write(". ");
            } while (getKeyspaceArn != keyspaceArn);
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine("Waiting for keyspace to be created.");
        }

        Console.WriteLine($"\nThe keyspace {keyspaceName} is ready for use.");

        uiMethods.PressEnter();

        // Create the table.
        // First define the schema.
        var allColumns = new List<ColumnDefinition>
        {
            new ColumnDefinition { Name = "title", Type = "text" },
            new ColumnDefinition { Name = "year", Type = "int" },
            new ColumnDefinition { Name = "release_date", Type = "timestamp" },
            new ColumnDefinition { Name = "plot", Type = "text" },
        };

        var partitionKeys = new List<PartitionKey>
        {
            new PartitionKey { Name = "year", },
            new PartitionKey { Name = "title" },
        };

        var tableSchema = new SchemaDefinition
        {
            AllColumns = allColumns,
            PartitionKeys = partitionKeys,
        };

        var tableArn = await keyspacesWrapper.CreateTable(keyspaceName, tableSchema,
tableName);

        // Wait for the table to be active.
        try
        {
            var resp = new GetTableResponse();
            Console.Write("Waiting for the new table to be active. ");
            do
```

```
            {
                try
                {
                    resp = await keyspacesWrapper.GetTable(keyspaceName, tableName);
                    Console.Write(".");
                }
                catch (ResourceNotFoundException)
                {
                    Console.Write(".");
                }
            } while (resp.Status != TableStatus.ACTIVE);

            // Display the table's schema.
            Console.WriteLine($"\nTable {tableName} has been created in
{keyspaceName}");
            Console.WriteLine("Let's take a look at the schema.");
            uiMethods.DisplayTitle("All columns");
            resp.SchemaDefinition.AllColumns.ForEach(column =>
            {
                Console.WriteLine($"{column.Name,-40}\t{column.Type,-20}");
            });

            uiMethods.DisplayTitle("Cluster keys");
            resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
            {

Console.WriteLine($"{clusterKey.Name,-40}\t{clusterKey.OrderBy,-20}");
            });

            uiMethods.DisplayTitle("Partition keys");
            resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
            {
                Console.WriteLine($"{partitionKey.Name}");
            });

            uiMethods.PressEnter();
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        // Access Apache Cassandra using the Cassandra drive for C#.
        var cassandraWrapper = host.Services.GetRequiredService<CassandraWrapper>();
```

```
        var movieFilePath = configuration["MovieFile"];

        Console.WriteLine("Let's add some movies to the table we created.");
        var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);

        uiMethods.PressEnter();

        Console.WriteLine("Added the following movies to the table:");
        var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
        uiMethods.DisplayTitle("All Movies");

        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            var plot = row.GetValue<string>("plot");
            var release_date = row.GetValue<DateTime>("release_date");
            Console.WriteLine($"{release_date}\t{title}\t{year}\n{plot}");
            Console.WriteLine(uiMethods.SepBar);
        }

        // Update the table schema
        uiMethods.DisplayTitle("Update table schema");
        Console.WriteLine("Now we will update the table to add a boolean field
called watched.");

        // First save the current time as a UTC Date so the original
        // table can be restored later.
        var timeChanged = DateTime.UtcNow;

        // Now update the schema.
        var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
tableName);
        uiMethods.PressEnter();

        Console.WriteLine("Now let's mark some of the movies as watched.");

        // Pick some files to mark as watched.
        var movieToWatch = rows[2].GetValue<string>("title");
        var watchedMovieYear = rows[2].GetValue<int>("year");
        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);
```

```
        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();

        Console.WriteLine("We can restore the table to its previous state but that
can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")
        {
            var restoredTableName = $"{tableName}_restored";
            var restoredTableArn = await keyspacesWrapper.RestoreTable(
```

```
                keyspaceName,
                tableName,
                restoredTableName,
                timeChanged);
            // Loop and call GetTable until the table is gone. Once it has been
            // deleted completely, GetTable will raise a ResourceNotFoundException.
            bool wasRestored = false;

            try
            {
                do
                {
                    var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                    wasRestored = (resp.Status == TableStatus.ACTIVE);
                } while (!wasRestored);
            }
            catch (ResourceNotFoundException)
            {
                // If the restored table raised an error, it isn't
                // ready yet.
                Console.Write(".");
            }
        }

        uiMethods.DisplayTitle("Clean up resources.");

        // Delete the table.
        success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);

        Console.WriteLine($"Table {tableName} successfully deleted from
{keyspaceName}.");
        Console.WriteLine("Waiting for the table to be removed completely. ");

        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a ResourceNotFoundException.
        bool wasDeleted = false;

        try
        {
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName, tableName);
            } while (!wasDeleted);
```

```
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
            Console.WriteLine($"{ex.Message} indicates that the table has been
 deleted.");
        }

        // Delete the keyspace.
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
 complete.");
    }
}
```

```
namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
```

```
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be created.</
param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
 schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
 PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }


    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }


    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new ListTablesRequest
{ KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }


    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
```

```
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }


    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type = "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }

}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;

/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
 database.
///
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/repository/
sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;

    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();

        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/{_certFileName}");

        //var httpClient = new HttpClient();
```

```
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
        _amazoncert = new X509Certificate2($"{_localPathToFile}/{_certFileName}");

        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>
    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport =
 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
```

```
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
        {
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
 movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;

        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
 (title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
 '{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
```

```
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }

    /// <summary>
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
{keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
tableName, string title, int year)
    {
        var session = _cluster.Connect();
```

```
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
 WHERE title = $${title}$$ AND year = {year};";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }

    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
 tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT title,
 year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW FILTERING"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateKeyspace
  - CreateTable

- DeleteKeyspace
- DeleteTable
- GetKeyspace
- GetTable
- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyspace`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Create a new keyspace.
/// </summary>
/// <param name="keyspaceName">The name for the new keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
public async Task<string> CreateKeyspace(string keyspaceName)
{
    var response =
        await _amazonKeyspaces.CreateKeyspaceAsync(
            new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- API-Details finden Sie CreateKeyspace in der AWS SDK für .NET API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be created.</
param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
 schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
 PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }
```

- API-Details finden Sie CreateTable in der AWS SDK für .NET API-Referenz.

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete an existing keyspace.
/// </summary>
/// <param name="keyspaceName"></param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
        new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API-Details finden Sie DeleteKeyspace in der AWS SDK für .NET API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK für .NET API-Referenz.

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyspace`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }
```

- API-Details finden Sie GetKeyspace in der AWS SDK für .NET API-Referenz.

GetTable

Das folgende Codebeispiel zeigt die VerwendungGetTable.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

- API-Details finden Sie GetTable in der AWS SDK für .NET API-Referenz.

ListKeyspaces

Das folgende Codebeispiel zeigt die VerwendungListKeyspaces.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }
```

- API-Details finden Sie ListKeyspaces in der AWS SDK für .NET API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new ListTablesRequest
 { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

 Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

- API-Details finden Sie ListTables in der AWS SDK für .NET API-Referenz.

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung`RestoreTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
 param>
```

```
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }
```

- API-Details finden Sie RestoreTable in der AWS SDK für .NET API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
```

```
        var newColumn = new ColumnDefinition { Name = "watched", Type = "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

- API-Details finden Sie UpdateTable in der AWS SDK für .NET API-Referenz.

Kinesis-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Kinesis Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`AddTagsToStream`

Das folgende Codebeispiel zeigt die Verwendung`AddTagsToStream`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream.
/// </summary>
public class TagStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        var tags = new Dictionary<string, string>
        {
            { "Project", "Sample Kinesis Project" },
            { "Application", "Sample Kinesis App" },
        };

        var success = await ApplyTagsToStreamAsync(client, streamName, tags);

        if (success)
        {
            Console.WriteLine($"Taggs successfully added to {streamName}.");
        }
        else
        {
            Console.WriteLine("Tags were not added to the stream.");
        }
    }
```

```
        /// <summary>
        /// Applies the set of tags to the named Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client.</param>
        /// <param name="streamName">The name of the Kinesis stream to which
        /// the tags will be attached.</param>
        /// <param name="tags">A sictionary containing key/value pairs which
        /// will be used to create the Kinesis tags.</param>
        /// <returns>A Boolean value which represents the success or failure
        /// of AddTagsToStreamAsync.</returns>
        public static async Task<bool> ApplyTagsToStreamAsync(
            IAmazonKinesis client,
            string streamName,
            Dictionary<string, string> tags)
        {
            var request = new AddTagsToStreamRequest
            {
                StreamName = streamName,
                Tags = tags,
            };

            var response = await client.AddTagsToStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- API-Details finden Sie AddTagsToStream in der AWS SDK für .NET API-Referenz.

`CreateStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateStream`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// This example shows how to create a new Amazon Kinesis stream.
    /// </summary>
    public class CreateStream
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamName = "AmazonKinesisStream";
            int shardCount = 1;

            var success = await CreateNewStreamAsync(client, streamName,
 shardCount);
            if (success)
            {
                Console.WriteLine($"The stream, {streamName} successfully
 created.");
            }
        }

        /// <summary>
        /// Creates a new Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client.</param>
        /// <param name="streamName">The name for the new stream.</param>
        /// <param name="shardCount">The number of shards the new stream will
        /// use. The throughput of the stream is a function of the number of
        /// shards; more shards are required for greater provisioned
        /// throughput.</param>
        /// <returns>A Boolean value indicating whether the stream was created.</
returns>
        public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis client,
 string streamName, int shardCount)
        {
            var request = new CreateStreamRequest
            {
                StreamName = streamName,
```

```
                ShardCount = shardCount,
            };

            var response = await client.CreateStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- API-Details finden Sie CreateStream in der AWS SDK für .NET API-Referenz.

`DeleteStream`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStream`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to delete an Amazon Kinesis stream.
/// </summary>
public class DeleteStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        var success = await DeleteStreamAsync(client, streamName);
```

```
            if (success)
            {
                Console.WriteLine($"Stream, {streamName} successfully deleted.");
            }
            else
            {
                Console.WriteLine("Stream not deleted.");
            }
        }

        /// <summary>
        /// Deletes a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the string to delete.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client,
 string streamName)
        {
            // If EnforceConsumerDeletion is true, any consumers
            // of this stream will also be deleted. If it is set
            // to false and this stream has any consumers, the
            // call will fail with a ResourceInUseException.
            var request = new DeleteStreamRequest
            {
                StreamName = streamName,
                EnforceConsumerDeletion = true,
            };

            var response = await client.DeleteStreamAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- API-Details finden Sie DeleteStream in der AWS SDK für .NET API-Referenz.

`DeregisterStreamConsumer`

Das folgende Codebeispiel zeigt die Verwendung`DeregisterStreamConsumer`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Shows how to deregister a consumer from an Amazon Kinesis stream.
    /// </summary>
    public class DeregisterConsumer
    {
        public static async Task Main(string[] args)
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream";
            string consumerName = "CONSUMER_NAME";
            string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";

            var success = await DeregisterConsumerAsync(client, streamARN,
 consumerARN, consumerName);

            if (success)
            {
                Console.WriteLine($"{consumerName} successfully deregistered.");
            }
            else
            {
                Console.WriteLine($"{consumerName} was not successfully
 deregistered.");
            }
        }
```

```
        /// <summary>
        /// Deregisters a consumer from a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of a Kinesis stream.</param>
        /// <param name="consumerARN">The ARN of the consumer.</param>
        /// <param name="consumerName">The name of the consumer.</param>
        /// <returns>A Boolean value representing the success of the operation.</
returns>
        public static async Task<bool> DeregisterConsumerAsync(
            IAmazonKinesis client,
            string streamARN,
            string consumerARN,
            string consumerName)
        {
            var request = new DeregisterStreamConsumerRequest
            {
                StreamARN = streamARN,
                ConsumerARN = consumerARN,
                ConsumerName = consumerName,
            };

            var response = await client.DeregisterStreamConsumerAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- API-Details finden Sie DeregisterStreamConsumer in der AWS SDK für .NET API-Referenz.

`ListStreamConsumers`

Das folgende Codebeispiel zeigt die Verwendung`ListStreamConsumers`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// List the consumers of an Amazon Kinesis stream.
    /// </summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;

            var consumers = await ListConsumersAsync(client, streamARN, maxResults);

            if (consumers.Count > 0)
            {
                consumers
                    .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
 {c.ConsumerARN}"));
            }
            else
            {
                Console.WriteLine("No consumers found.");
            }
        }

        /// <summary>
        /// Retrieve a list of the consumers for a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamARN">The ARN of the stream for which we want to
        /// retrieve a list of clients.</param>
        /// <param name="maxResults">The maximum number of results to return.</
param>
        /// <returns>A list of Consumer objects.</returns>
```

```
        public static async Task<List<Consumer>> ListConsumersAsync(IAmazonKinesis
client, string streamARN, int maxResults)
        {
            var request = new ListStreamConsumersRequest
            {
                StreamARN = streamARN,
                MaxResults = maxResults,
            };

            var response = await client.ListStreamConsumersAsync(request);

            return response.Consumers;
        }
    }
```

- API-Details finden Sie ListStreamConsumers in der AWS SDK für .NET API-Referenz.

`ListStreams`

Das folgende Codebeispiel zeigt die Verwendung`ListStreams`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// Retrieves and displays a list of existing Amazon Kinesis streams.
    /// </summary>
    public class ListStreams
    {
```

```
        public static async Task Main(string[] args)
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            var response = await client.ListStreamsAsync(new ListStreamsRequest());

            List<string> streamNames = response.StreamNames;

            if (streamNames.Count > 0)
            {
                streamNames
                    .ForEach(s => Console.WriteLine($"Stream name: {s}"));
            }
            else
            {
                Console.WriteLine("No streams were found.");
            }
        }
    }
```

- API-Details finden Sie ListStreams in der AWS SDK für .NET API-Referenz.

`ListTagsForStream`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForStream`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
```

```
    /// Shows how to list the tags that have been attached to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class ListTags
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string streamName = "AmazonKinesisStream";

            await ListTagsAsync(client, streamName);
        }

        /// <summary>
        /// List the tags attached to a Kinesis stream.
        /// </summary>
        /// <param name="client">An initialized Kinesis client object.</param>
        /// <param name="streamName">The name of the Kinesis stream for which you
        /// wish to display tags.</param>
        public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
        {
            var request = new ListTagsForStreamRequest
            {
                StreamName = streamName,
                Limit = 10,
            };

            var response = await client.ListTagsForStreamAsync(request);
            DisplayTags(response.Tags);

            while (response.HasMoreTags)
            {
                request.ExclusiveStartTagKey = response.Tags[response.Tags.Count -
1].Key;
                response = await client.ListTagsForStreamAsync(request);
            }
        }

        /// <summary>
        /// Displays the items in a list of Kinesis tags.
        /// </summary>
        /// <param name="tags">A list of the Tag objects to be displayed.</param>
        public static void DisplayTags(List<Tag> tags)
```

```
        {
            tags
                .ForEach(t => Console.WriteLine($"Key: {t.Key} Value: {t.Value}"));
        }
    }
```

- API-Details finden Sie ListTagsForStream in der AWS SDK für .NET API-Referenz.

`RegisterStreamConsumer`

Das folgende Codebeispiel zeigt die Verwendung`RegisterStreamConsumer`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Kinesis;
    using Amazon.Kinesis.Model;

    /// <summary>
    /// This example shows how to register a consumer to an Amazon Kinesis
    /// stream.
    /// </summary>
    public class RegisterConsumer
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();
            string consumerName = "NEW_CONSUMER_NAME";
            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";

            var consumer = await RegisterConsumerAsync(client, consumerName,
 streamARN);
```

```
            if (consumer is not null)
            {
                Console.WriteLine($"{consumer.ConsumerName}");
            }
        }

        /// <summary>
        /// Registers the consumer to a Kinesis stream.
        /// </summary>
        /// <param name="client">The initialized Kinesis client object.</param>
        /// <param name="consumerName">A string representing the consumer.</param>
        /// <param name="streamARN">The ARN of the stream.</param>
        /// <returns>A Consumer object that contains information about the
consumer.</returns>
        public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis
client, string consumerName, string streamARN)
        {
            var request = new RegisterStreamConsumerRequest
            {
                ConsumerName = consumerName,
                StreamARN = streamARN,
            };

            var response = await client.RegisterStreamConsumerAsync(request);
            return response.Consumer;
        }
    }
```

- API-Details finden Sie RegisterStreamConsumer in der AWS SDK für .NET API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
```

```
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
```

```
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure { ItemIdentifier =
 record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

AWS KMS Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren AWS KMS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
```

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The alias name must start with alias/ and can be
            // up to 256 alphanumeric characters long.
            var aliasName = "alias/ExampleAlias";

            // The value supplied as the TargetKeyId can be either
            // the key ID or key Amazon Resource Name (ARN) of the
            // AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

            var request = new CreateAliasRequest
            {
                AliasName = aliasName,
                TargetKeyId = keyId,
            };

            var response = await client.CreateAliasAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Alias, {aliasName}, successfully created.");
            }
            else
            {
                Console.WriteLine($"Could not create alias.");
            }
        }
    }
```

- API-Details finden Sie CreateAlias in der AWS SDK für .NET API-Referenz.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identity that is given permission to perform the operations
            // specified in the grant.
            var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

            // The identifier of the AWS KMS key to which the grant applies. You
            // can use the key ID or the Amazon Resource Name (ARN) of the KMS key.
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

            var request = new CreateGrantRequest
            {
                GranteePrincipal = grantee,
                KeyId = keyId,

                // A list of operations that the grant allows.
                Operations = new List<string>
                {
                    "Encrypt",
                    "Decrypt",
                },
            };

            var response = await client.CreateGrantAsync(request);

            string grantId = response.GrantId; // The unique identifier of the
grant.
            string grantToken = response.GrantToken; // The grant token.

            Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
        }
    }
```

- API-Details finden Sie CreateGrant in der AWS SDK für .NET API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// Shows how to create a new AWS Key Management Service (AWS KMS)
    /// key.
    /// </summary>
    public class CreateKey
    {
        public static async Task Main()
        {
            // Note that if you need to create a Key in an AWS Region
            // other than the Region defined for the default user, you need to
            // pass the Region to the client constructor.
            var client = new AmazonKeyManagementServiceClient();

            // The call to CreateKeyAsync will create a symmetrical AWS KMS
            // key. For more information about symmetrical and asymmetrical
            // keys, see:
            //
            // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
```

```
            var response = await client.CreateKeyAsync(new CreateKeyRequest());

            // The KeyMetadata object contains information about the new AWS KMS
key.
            KeyMetadata keyMetadata = response.KeyMetadata;

            if (keyMetadata is not null)
            {
                Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
created.");
            }
            else
            {
                Console.WriteLine("Could not create KMS Key.");
            }
        }
    }
```

- API-Details finden Sie CreateKey in der AWS SDK für .NET API-Referenz.

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// Retrieve information about an AWS Key Management Service (AWS KMS) key.
    /// You can supply either the key Id or the key Amazon Resource Name (ARN)
```

```
    /// to the DescribeKeyRequest KeyId property.
    /// </summary>
    public class DescribeKey
    {
        public static async Task Main()
        {
            var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
            var request = new DescribeKeyRequest
            {
                KeyId = keyId,
            };

            var client = new AmazonKeyManagementServiceClient();

            var response = await client.DescribeKeyAsync(request);
            var metadata = response.KeyMetadata;

            Console.WriteLine($"{metadata.KeyId} created on:
{metadata.CreationDate}");
            Console.WriteLine($"State: {metadata.KeyState}");
            Console.WriteLine($"{metadata.Description}");
        }
    }
```

- API-Details finden Sie DescribeKey in der AWS SDK für .NET API-Referenz.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
```

```
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been disabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
            {
                KeyId = keyId,
            });
            Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
        }
    }
}
```

- API-Details finden Sie DisableKey in der AWS SDK für .NET API-Referenz.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// Enable an AWS Key Management Service (AWS KMS) key.
    /// </summary>
    public class EnableKey
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();

            // The identifier of the AWS KMS key to enable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

            var request = new EnableKeyRequest
            {
                KeyId = keyId,
            };

            var response = await client.EnableKeyAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                // Retrieve information about the key to show that it has now
                // been enabled.
                var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
                {
```

```
                    KeyId = keyId,
                });
                Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
            }
        }
    }
```

- API-Details finden Sie EnableKey in der AWS SDK für .NET API-Referenz.

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAliases`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
    /// List the AWS Key Management Service (AWS KMS) aliases that have been defined
for
    /// the keys in the same AWS Region as the default user. If you want to list
    /// the aliases in a different Region, pass the Region to the client
    /// constructor.
    /// </summary>
    public class ListAliases
    {
        public static async Task Main()
        {
            var client = new AmazonKeyManagementServiceClient();
```

```
            var request = new ListAliasesRequest();
            var response = new ListAliasesResponse();

            do
            {
                response = await client.ListAliasesAsync(request);

                response.Aliases.ForEach(alias =>
                {
                    Console.WriteLine($"Created: {alias.CreationDate} Last Update:
{alias.LastUpdatedDate} Name: {alias.AliasName}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- API-Details finden Sie ListAliases in der AWS SDK für .NET API-Referenz.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.KeyManagementService;
    using Amazon.KeyManagementService.Model;

    /// <summary>
```

```
    /// List the AWS Key Management Service (AWS KMS) grants that are associated
with
    /// a specific key.
    /// </summary>
    public class ListGrants
    {
        public static async Task Main()
        {
            // The identifier of the AWS KMS key to disable. You can use the
            // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
            var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
            var client = new AmazonKeyManagementServiceClient();
            var request = new ListGrantsRequest
            {
                KeyId = keyId,
            };

            var response = new ListGrantsResponse();

            do
            {
                response = await client.ListGrantsAsync(request);

                response.Grants.ForEach(grant =>
                {
                    Console.WriteLine($"{grant.GrantId}");
                });

                request.Marker = response.NextMarker;
            }
            while (response.Truncated);
        }
    }
```

- API-Details finden Sie ListGrants in der AWS SDK für .NET API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
/// of the default user. To list keys in another AWS Region, supply the Region
/// as a parameter to the client constructor.
/// </summary>
public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();

        do
        {
            response = await client.ListKeysAsync(request);

            response.Keys.ForEach(key =>
            {
                Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
            });

            // Set the Marker property when response.Truncated is true
            // in order to get the next keys.
            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- API-Details finden Sie ListKeys in der AWS SDK für .NET API-Referenz.

Lambda-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren AWS Teams erstellt wurden und verwaltet werden. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace LambdaActions;

using Amazon.Lambda;

public class HelloLambda
{
    static async Task Main(string[] args)
    {
        var lambdaClient = new AmazonLambdaClient();

        Console.WriteLine("Hello AWS Lambda");
        Console.WriteLine("Let's get started with AWS Lambda by listing your
 existing Lambda functions:");

        var response = await lambdaClient.ListFunctionsAsync();
        response.Functions.ForEach(function =>
        {
            Console.WriteLine($"{function.FunctionName}\t{function.Description}");
        });
    }
}
```

- API-Details finden Sie ListFunctions in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Methoden, die Lambda-Aktionen ausführen.

```
namespace LambdaActions;

using Amazon.Lambda;
using Amazon.Lambda.Model;

/// <summary>
/// A class that implements AWS Lambda methods.
/// </summary>
public class LambdaWrapper
{
    private readonly IAmazonLambda _lambdaService;

    /// <summary>
    /// Constructor for the LambdaWrapper class.
```

```
    /// </summary>
    /// <param name="lambdaService">An initialized Lambda service client.</param>
    public LambdaWrapper(IAmazonLambda lambdaService)
    {
        _lambdaService = lambdaService;
    }

    /// <summary>
    /// Creates a new Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function.</param>
    /// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
    /// bucket where the zip file containing the code is located.</param>
    /// <param name="s3Key">The Amazon S3 key of the zip file.</param>
    /// <param name="role">The Amazon Resource Name (ARN) of a role with the
    /// appropriate Lambda permissions.</param>
    /// <param name="handler">The name of the handler function.</param>
    /// <returns>The Amazon Resource Name (ARN) of the newly created
    /// Lambda function.</returns>
    public async Task<string> CreateLambdaFunctionAsync(
        string functionName,
        string s3Bucket,
        string s3Key,
        string role,
        string handler)
    {
        // Defines the location for the function code.
        // S3Bucket - The S3 bucket where the file containing
        //            the source code is stored.
        // S3Key    - The name of the file containing the code.
        var functionCode = new FunctionCode
        {
            S3Bucket = s3Bucket,
            S3Key = s3Key,
        };

        var createFunctionRequest = new CreateFunctionRequest
        {
            FunctionName = functionName,
            Description = "Created by the Lambda .NET API",
            Code = functionCode,
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
```

```
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }


    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }


    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
```

```
        return response.Configuration;
    }


    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }


    /// <summary>
    /// Get a list of Lambda functions.
    /// </summary>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
    {
        var functionList = new List<FunctionConfiguration>();

        var functionPaginator =
            _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
        await foreach (var function in functionPaginator.Functions)
        {
            functionList.Add(function);
        }
```

```
        return functionList;
    }


    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }


    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment variables.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
```

```
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
        };

        var response = await
 _lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


}
```

Erstellen Sie eine Funktion, die das Szenario ausführt.

```
global using System.Threading.Tasks;
global using Amazon.IdentityManagement;
global using Amazon.Lambda;
global using LambdaActions;
global using LambdaScenarioCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


using Amazon.Lambda.Model;
using Microsoft.Extensions.Configuration;

namespace LambdaBasics;
```

```
public class LambdaBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonLambda>()
            .AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<LambdaWrapper>()
            .AddTransient<LambdaRoleWrapper>()
            .AddTransient<UIWrapper>()
        )
        .Build();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();


        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<LambdaBasics>();

        var lambdaWrapper = host.Services.GetRequiredService<LambdaWrapper>();
        var lambdaRoleWrapper =
 host.Services.GetRequiredService<LambdaRoleWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        string functionName = configuration["FunctionName"]!;
        string roleName = configuration["RoleName"]!;
        string policyDocument = "{" +
            " \"Version\": \"2012-10-17\"," +
            " \"Statement\": [ " +
```

```
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Principal\": {" +
            "            \"Service\": \"lambda.amazonaws.com\" " +
            "    }," +
            "        \"Action\": \"sts:AssumeRole\" " +
            "    }" +
            "]" +
        "}";

        var incrementHandler = configuration["IncrementHandler"];
        var calculatorHandler = configuration["CalculatorHandler"];
        var bucketName = configuration["BucketName"];
        var incrementKey = configuration["IncrementKey"];
        var calculatorKey = configuration["CalculatorKey"];
        var policyArn = configuration["PolicyArn"];

        uiWrapper.DisplayLambdaBasicsOverview();

        // Create the policy to use with the AWS Lambda functions and then attach
the
        // policy to a new role.
        var roleArn = await lambdaRoleWrapper.CreateLambdaRoleAsync(roleName,
policyDocument);

        Console.WriteLine("Waiting for role to become active.");
        uiWrapper.WaitABit(15, "Wait until the role is active before trying to use
it.");

        // Attach the appropriate AWS Identity and Access Management (IAM) role
policy to the new role.
        var success = await lambdaRoleWrapper.AttachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(10, "Allow time for the IAM policy to be attached to the
role.");

        // Create the Lambda function using a zip file stored in an Amazon Simple
Storage Service
        // (Amazon S3) bucket.
        uiWrapper.DisplayTitle("Create Lambda Function");
        Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
        var lambdaArn = await lambdaWrapper.CreateLambdaFunctionAsync(
            functionName,
            bucketName,
```

```
            incrementKey,
            roleArn,
            incrementHandler);

        Console.WriteLine("Waiting for the new function to be available.");
        Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");

        // Get the Lambda function.
        Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
        FunctionConfiguration config;
        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.State != State.Active);

        Console.WriteLine($"\nThe function, {functionName} has been created.");
        Console.WriteLine($"The runtime of this Lambda function is
{config.Runtime}.");

        uiWrapper.PressEnter();

        // List the Lambda functions.
        uiWrapper.DisplayTitle("Listing all Lambda functions.");
        var functions = await lambdaWrapper.ListFunctionsAsync();
        DisplayFunctionList(functions);

        uiWrapper.DisplayTitle("Invoke increment function");
        Console.WriteLine("Now that it has been created, invoke the Lambda increment
function.");
        string? value;
        do
        {
            Console.Write("Enter a value to increment: ");
            value = Console.ReadLine();
        }
        while (string.IsNullOrEmpty(value));

        string functionParameters = "{" +
            "\"action\": \"increment\", " +
            "\"x\": \"" + value + "\"" +
        "}";
```

```
        var answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
        Console.WriteLine($"{value} + 1 = {answer}.");

        uiWrapper.DisplayTitle("Update function");
        Console.WriteLine("Now update the Lambda function code.");
        await lambdaWrapper.UpdateFunctionCodeAsync(functionName, bucketName,
calculatorKey);

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        await lambdaWrapper.UpdateFunctionConfigurationAsync(
            functionName,
            calculatorHandler,
            new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });

        do
        {
            config = await lambdaWrapper.GetFunctionAsync(functionName);
            Console.Write(".");
        }
        while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

        uiWrapper.DisplayTitle("Call updated function");
        Console.WriteLine("Now call the updated function...");

        bool done = false;

        do
        {
            string? opSelected;

            Console.WriteLine("Select the operation to perform:");
            Console.WriteLine("\t1. add");
            Console.WriteLine("\t2. subtract");
            Console.WriteLine("\t3. multiply");
            Console.WriteLine("\t4. divide");
            Console.WriteLine("\tOr enter \"q\" to quit.");
```

```
            Console.WriteLine("Enter the number (1, 2, 3, 4, or q) of the operation
you want to perform: ");
            do
            {
                Console.Write("Your choice? ");
                opSelected = Console.ReadLine();
            }
            while (opSelected == string.Empty);

            var operation = (opSelected) switch
            {
                "1" => "add",
                "2" => "subtract",
                "3" => "multiply",
                "4" => "divide",
                "q" => "quit",
                _ => "add",
            };

            if (operation == "quit")
            {
                done = true;
            }
            else
            {
                // Get two numbers and an action from the user.
                value = string.Empty;
                do
                {
                    Console.Write("Enter the first value: ");
                    value = Console.ReadLine();
                }
                while (value == string.Empty);

                string? value2;
                do
                {
                    Console.Write("Enter a second value: ");
                    value2 = Console.ReadLine();
                }
                while (value2 == string.Empty);

                functionParameters = "{" +
                    "\"action\": \"" + operation + "\", " +
```

```
                    "\"x\": \"" + value + "\"," +
                    "\"y\": \"" + value2 + "\"" +
                "}";

                answer = await lambdaWrapper.InvokeFunctionAsync(functionName,
functionParameters);
                Console.WriteLine($"The answer when we {operation} the two numbers
is: {answer}.");
            }

            uiWrapper.PressEnter();
        } while (!done);

        // Delete the function created earlier.

        uiWrapper.DisplayTitle("Clean up resources");
        // Detach the IAM policy from the IAM role.
        Console.WriteLine("First detach the IAM policy from the role.");
        success = await lambdaRoleWrapper.DetachLambdaRolePolicyAsync(policyArn,
roleName);
        uiWrapper.WaitABit(15, "Let's wait for the policy to be fully detached from
the role.");

        Console.WriteLine("Delete the AWS Lambda function.");
        success = await lambdaWrapper.DeleteFunctionAsync(functionName);
        if (success)
        {
            Console.WriteLine($"The {functionName} function was deleted.");
        }
        else
        {
            Console.WriteLine($"Could not remove the function {functionName}");
        }

        // Now delete the IAM role created for use with the functions
        // created by the application.
        Console.WriteLine("Now we can delete the role that we created.");
        success = await lambdaRoleWrapper.DeleteLambdaRoleAsync(roleName);
        if (success)
        {
            Console.WriteLine("The role has been successfully removed.");
        }
        else
        {
```

```
            Console.WriteLine("Couldn't delete the role.");
        }

        Console.WriteLine("The Lambda Scenario is now complete.");
        uiWrapper.PressEnter();

        // Displays a formatted list of existing functions returned by the
        // LambdaMethods.ListFunctions.
        void DisplayFunctionList(List<FunctionConfiguration> functions)
        {
            functions.ForEach(functionConfig =>
            {

 Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
            });
        }
    }
}


namespace LambdaActions;

using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class LambdaRoleWrapper
{
    private readonly IAmazonIdentityManagementService _lambdaRoleService;

    public LambdaRoleWrapper(IAmazonIdentityManagementService lambdaRoleService)
    {
        _lambdaRoleService = lambdaRoleService;
    }

    /// <summary>
    /// Attach an AWS Identity and Access Management (IAM) role policy to the
    /// IAM role to be assumed by the AWS Lambda functions created for the scenario.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM policy.</
param>
    /// <param name="roleName">The name of the IAM role to attach the IAM policy
 to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> AttachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to create.</param>
    /// <param name="policyDocument">The policy document for the new IAM role.</
param>
    /// <returns>A string representing the ARN for newly created role.</returns>
    public async Task<string> CreateLambdaRoleAsync(string roleName, string
 policyDocument)
    {
        var request = new CreateRoleRequest
        {
            AssumeRolePolicyDocument = policyDocument,
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.CreateRoleAsync(request);
        return response.Role.Arn;
    }

    /// <summary>
    /// Deletes an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteLambdaRoleAsync(string roleName)
    {
        var request = new DeleteRoleRequest
        {
            RoleName = roleName,
        };

        var response = await _lambdaRoleService.DeleteRoleAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    public async Task<bool> DetachLambdaRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _lambdaRoleService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest { PolicyArn = policyArn, RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}


namespace LambdaScenarioCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the AWS Lambda Basics scenario.
    /// </summary>
    public void DisplayLambdaBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to AWS Lambda Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an AWS Identity and Access Management (IAM)
 role that will be assumed by the functions we create.");
        Console.WriteLine("\t2. Attaches an IAM role policy that has Lambda
 permissions.");
        Console.WriteLine("\t3. Creates a Lambda function that increments the value
 passed to it.");
        Console.WriteLine("\t4. Calls the increment function and passes a value.");
        Console.WriteLine("\t5. Updates the code so that the function is a simple
 calculator.");
        Console.WriteLine("\t6. Calls the calculator function with the values
 entered.");
        Console.WriteLine("\t7. Deletes the Lambda function.");
        Console.WriteLine("\t7. Detaches the IAM role policy.");
        Console.WriteLine("\t8. Deletes the IAM role.");
        PressEnter();
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
```

```
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
```

```
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaIncrement;

public class Function
{

    /// <summary>
    /// A simple function increments the integer parameter.
    /// </summary>
    /// <param name="input">A JSON string containing an action, which must be
    /// "increment" and a string representing the value to increment.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the incremented value of the parameter.</
returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        if (input["action"] == "increment")
        {
            int inputValue = Convert.ToInt32(input["x"]);
            return inputValue + 1;
        }
        else
        {
            return 0;
```

```
        }
    }
}
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaCalculator;

public class Function
{

    /// <summary>
    /// A simple function that takes two number in string format and performs
    /// the requested arithmetic function.
    /// </summary>
    /// <param name="input">JSON data containing an action, and x and y values.
    /// Valid actions include: add, subtract, multiply, and divide.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the results of the calculation.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
 context)
    {
        var action = input["action"];
        int x = Convert.ToInt32(input["x"]);
        int y = Convert.ToInt32(input["y"]);
        int result;
        switch (action)
        {
            case "add":
                result = x + y;
                break;
            case "subtract":
                result = x - y;
```

```
                break;
            case "multiply":
                result = x * y;
                break;
            case "divide":
                if (y == 0)
                {
                    Console.Error.WriteLine("Divide by zero error.");
                    result = 0;
                }
                else
                    result = x / y;
                break;
            default:
                Console.Error.WriteLine($"{action} is not a valid operation.");
                result = 0;
                break;
        }
        return result;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Creates a new Lambda function.
/// </summary>
/// <param name="functionName">The name of the function.</param>
/// <param name="s3Bucket">The Amazon Simple Storage Service (Amazon S3)
/// bucket where the zip file containing the code is located.</param>
/// <param name="s3Key">The Amazon S3 key of the zip file.</param>
/// <param name="role">The Amazon Resource Name (ARN) of a role with the
/// appropriate Lambda permissions.</param>
/// <param name="handler">The name of the handler function.</param>
/// <returns>The Amazon Resource Name (ARN) of the newly created
/// Lambda function.</returns>
public async Task<string> CreateLambdaFunctionAsync(
    string functionName,
    string s3Bucket,
    string s3Key,
    string role,
    string handler)
{
    // Defines the location for the function code.
    // S3Bucket - The S3 bucket where the file containing
    //            the source code is stored.
    // S3Key    - The name of the file containing the code.
    var functionCode = new FunctionCode
    {
        S3Bucket = s3Bucket,
        S3Key = s3Key,
    };

    var createFunctionRequest = new CreateFunctionRequest
    {
        FunctionName = functionName,
        Description = "Created by the Lambda .NET API",
        Code = functionCode,
```

```
            Handler = handler,
            Runtime = Runtime.Dotnet6,
            Role = role,
        };

        var reponse = await
_lambdaService.CreateFunctionAsync(createFunctionRequest);
        return reponse.FunctionArn;
    }
```

- API-Details finden Sie CreateFunction in der AWS SDK für .NET API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an AWS Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// delete.</param>
    /// <returns>A Boolean value that indicates the success of the action.</returns>
    public async Task<bool> DeleteFunctionAsync(string functionName)
    {
        var request = new DeleteFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.DeleteFunctionAsync(request);

        // A return value of NoContent means that the request was processed.
```

```
        // In this case, the function was deleted, and the return value
        // is intentionally blank.
        return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
    }
```

- API-Details finden Sie DeleteFunction in der AWS SDK für .NET API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets information about a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function for
    /// which to retrieve information.</param>
    /// <returns>Async Task.</returns>
    public async Task<FunctionConfiguration> GetFunctionAsync(string functionName)
    {
        var functionRequest = new GetFunctionRequest
        {
            FunctionName = functionName,
        };

        var response = await _lambdaService.GetFunctionAsync(functionRequest);
        return response.Configuration;
    }
```

- API-Details finden Sie GetFunction in der AWS SDK für .NET API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Invoke a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to
    /// invoke.</param
    /// <param name="parameters">The parameter values that will be passed to the
function.</param>
    /// <returns>A System Threading Task.</returns>
    public async Task<string> InvokeFunctionAsync(
        string functionName,
        string parameters)
    {
        var payload = parameters;
        var request = new InvokeRequest
        {
            FunctionName = functionName,
            Payload = payload,
        };

        var response = await _lambdaService.InvokeAsync(request);
        MemoryStream stream = response.Payload;
        string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
        return returnValue;
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für .NET -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get a list of Lambda functions.
/// </summary>
/// <returns>A list of FunctionConfiguration objects.</returns>
public async Task<List<FunctionConfiguration>> ListFunctionsAsync()
{
    var functionList = new List<FunctionConfiguration>();

    var functionPaginator =
        _lambdaService.Paginators.ListFunctions(new ListFunctionsRequest());
    await foreach (var function in functionPaginator.Functions)
    {
        functionList.Add(function);
    }

    return functionList;
}
```

- API-Details finden Sie ListFunctions in der AWS SDK für .NET API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update an existing Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the Lambda function to update.</
param>
    /// <param name="bucketName">The bucket where the zip file containing
    /// the Lambda function code is stored.</param>
    /// <param name="key">The key name of the source code file.</param>
    /// <returns>Async Task.</returns>
    public async Task UpdateFunctionCodeAsync(
        string functionName,
        string bucketName,
        string key)
    {
        var functionCodeRequest = new UpdateFunctionCodeRequest
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await
 _lambdaService.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
 {response.LastModified}.");
    }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für .NET API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the code of a Lambda function.
    /// </summary>
    /// <param name="functionName">The name of the function to update.</param>
    /// <param name="functionHandler">The code that performs the function's
 actions.</param>
    /// <param name="environmentVariables">A dictionary of environment variables.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateFunctionConfigurationAsync(
        string functionName,
        string functionHandler,
        Dictionary<string, string> environmentVariables)
    {
        var request = new UpdateFunctionConfigurationRequest
        {
            Handler = functionHandler,
            FunctionName = functionName,
            Environment = new Amazon.Lambda.Model.Environment { Variables =
 environmentVariables },
        };

        var response = await
 _lambdaService.UpdateFunctionConfigurationAsync(request);

        Console.WriteLine(response.LastModified);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Transformieren von Daten mit S3 Object Lambda

Das folgende Codebeispiel zeigt, wie Sie Daten für Ihre Anwendung mit S3 Object Lambda transformieren können.

SDK für .NET

Zeigt, wie Sie benutzerdefinierten Code zu standardmäßigen S3-GET-Anforderungen hinzufügen, um das angeforderte Objekt, das von S3 abgerufen wurde, so zu ändern, dass das Objekt den Anforderungen des anfordernden Clients oder der anfordernden Anwendung entspricht.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda

- Amazon S3

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion mithilfe von .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace aws_rds;

public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
```

```
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                 $"Pwd={authToken};";



        try
        {
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";

            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
```

```
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The sum
 is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server error" })
        };
    }
}
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegrationSampleCode;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task FunctionHandler(KinesisEvent evnt, ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return;
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
```

```
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                throw;
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public void FunctionHandler(DynamoDBEvent dynamoEvent, ILambdaContext context)
    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");

        foreach (var record in dynamoEvent.Records)
        {
            context.Logger.LogInformation($"Event ID: {record.EventID}");
            context.Logger.LogInformation($"Event Name: {record.EventName}");

            context.Logger.LogInformation(JsonSerializer.Serialize(record));
        }

        context.Logger.LogInformation("Stream processing complete.");
    }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon-DocumentDB-Ereignisses mit Lambda unter Verwendung von .NET.

```
using Amazon.Lambda.Core;
using System.Text.Json;
using System;
using System.Collections.Generic;
using System.Text.Json.Serialization;
//Assembly attribute to enable the Lambda function's JSON input to be converted into
 a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace LambdaDocDb;

public class Function
{

     /// <summary>
    /// Lambda function entry point to process Amazon DocumentDB events.
    /// </summary>
    /// <param name="event">The Amazon DocumentDB event.</param>
    /// <param name="context">The Lambda context object.</param>
    /// <returns>A string to indicate successful processing.</returns>
    public string FunctionHandler(Event evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Events)
        {
            ProcessDocumentDBEvent(record, context);
        }

        return "OK";
    }
```

```
    private void ProcessDocumentDBEvent(DocumentDBEventRecord record,
ILambdaContext context)
   {

       var eventData = record.Event;
       var operationType = eventData.OperationType;
       var databaseName = eventData.Ns.Db;
       var collectionName = eventData.Ns.Coll;
       var fullDocument = JsonSerializer.Serialize(eventData.FullDocument, new
JsonSerializerOptions { WriteIndented = true });

       context.Logger.LogLine($"Operation type: {operationType}");
       context.Logger.LogLine($"Database: {databaseName}");
       context.Logger.LogLine($"Collection: {collectionName}");
       context.Logger.LogLine($"Full document:\n{fullDocument}");
   }



   public class Event
   {
       [JsonPropertyName("eventSourceArn")]
       public string EventSourceArn { get; set; }

       [JsonPropertyName("events")]
       public List<DocumentDBEventRecord> Events { get; set; }

       [JsonPropertyName("eventSource")]
       public string EventSource { get; set; }
   }

   public class DocumentDBEventRecord
   {
       [JsonPropertyName("event")]
       public EventData Event { get; set; }
   }

   public class EventData
   {
       [JsonPropertyName("_id")]
       public IdData Id { get; set; }

       [JsonPropertyName("clusterTime")]
       public ClusterTime ClusterTime { get; set; }
```

```
        [JsonPropertyName("documentKey")]
        public DocumentKey DocumentKey { get; set; }

        [JsonPropertyName("fullDocument")]
        public Dictionary<string, object> FullDocument { get; set; }

        [JsonPropertyName("ns")]
        public Namespace Ns { get; set; }

        [JsonPropertyName("operationType")]
        public string OperationType { get; set; }
    }

    public class IdData
    {
        [JsonPropertyName("_data")]
        public string Data { get; set; }
    }

    public class ClusterTime
    {
        [JsonPropertyName("$timestamp")]
        public Timestamp Timestamp { get; set; }
    }

    public class Timestamp
    {
        [JsonPropertyName("t")]
        public long T { get; set; }

        [JsonPropertyName("i")]
        public int I { get; set; }
    }

    public class DocumentKey
    {
        [JsonPropertyName("_id")]
        public Id Id { get; set; }
    }

    public class Id
    {
        [JsonPropertyName("$oid")]
```

```
        public string Oid { get; set; }
    }

    public class Namespace
    {
        [JsonPropertyName("db")]
        public string Db { get; set; }

        [JsonPropertyName("coll")]
        public string Coll { get; set; }
    }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace MSKLambda;
```

```
public class Function
{


    /// <param name="input">The event for the Lambda function handler to process.</
param>
    /// <param name="context">The ILambdaContext that provides methods for logging
 and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von .NET

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
```

```
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request - {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record, ILambdaContext
 context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
```

```
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }
```

```
    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
using System.Text;
using System.Text.Json.Serialization;
using Amazon.Lambda.Core;
using Amazon.Lambda.KinesisEvents;
using AWS.Lambda.Powertools.Logging;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace KinesisIntegration;

public class Function
{
    // Powertools Logger requires an environment variables against your function
    // POWERTOOLS_SERVICE_NAME
    [Logging(LogEvent = true)]
    public async Task<StreamsEventResponse> FunctionHandler(KinesisEvent evnt,
 ILambdaContext context)
    {
        if (evnt.Records.Count == 0)
        {
            Logger.LogInformation("Empty Kinesis Event received");
            return new StreamsEventResponse();
        }

        foreach (var record in evnt.Records)
        {
            try
            {
                Logger.LogInformation($"Processed Event with EventId:
 {record.EventId}");
                string data = await GetRecordDataAsync(record.Kinesis, context);
                Logger.LogInformation($"Data: {data}");
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex)
            {
                Logger.LogError($"An error occurred {ex.Message}");
                /* Since we are working with streams, we can return the failed item
 immediately.
```

```
                    Lambda will immediately begin to retry processing from this
 failed item onwards. */
                return new StreamsEventResponse
                {
                    BatchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>
                    {
                        new StreamsEventResponse.BatchItemFailure { ItemIdentifier =
 record.Kinesis.SequenceNumber }
                    }
                };
            }
        }
        Logger.LogInformation($"Successfully processed {evnt.Records.Count}
 records.");
        return new StreamsEventResponse();
    }

    private async Task<string> GetRecordDataAsync(KinesisEvent.Record record,
 ILambdaContext context)
    {
        byte[] bytes = record.Data.ToArray();
        string data = Encoding.UTF8.GetString(bytes);
        await Task.CompletedTask; //Placeholder for actual async work
        return data;
    }
}

public class StreamsEventResponse
{
    [JsonPropertyName("batchItemFailures")]
    public IList<BatchItemFailure> BatchItemFailures { get; set; }
    public class BatchItemFailure
    {
        [JsonPropertyName("itemIdentifier")]
        public string ItemIdentifier { get; set; }
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Text.Json;
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.DynamoDBEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace AWSLambda_DDB;

public class Function
{
    public StreamsEventResponse FunctionHandler(DynamoDBEvent dynamoEvent,
 ILambdaContext context)

    {
        context.Logger.LogInformation($"Beginning to process
 {dynamoEvent.Records.Count} records...");
        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 List<StreamsEventResponse.BatchItemFailure>();
        StreamsEventResponse streamsEventResponse = new StreamsEventResponse();
```

```
        foreach (var record in dynamoEvent.Records)
        {
            try
            {
                var sequenceNumber = record.Dynamodb.SequenceNumber;
                context.Logger.LogInformation(sequenceNumber);
            }
            catch (Exception ex)
            {
                context.Logger.LogError(ex.Message);
                batchItemFailures.Add(new StreamsEventResponse.BatchItemFailure()
 { ItemIdentifier = record.Dynamodb.SequenceNumber });
            }
        }

        if (batchItemFailures.Count > 0)
        {
            streamsEventResponse.BatchItemFailures = batchItemFailures;
        }

        context.Logger.LogInformation("Stream processing complete.");
        return streamsEventResponse;
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
```

```
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für .NET

Zeigt, wie eine Serverless-Anwendung, die aus einem API Gateway mit Lambda und DynamoDB besteht, mithilfe des .NET-SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

MediaConvert Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren MediaConvert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo MediaConvert

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Elemental MediaConvert beginnen.

SDK für .NET

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.MediaConvert;
using Amazon.MediaConvert.Model;

namespace MediaConvertActions;

public static class HelloMediaConvert
{
    static async Task Main(string[] args)
    {
        // Create the client using the default profile.
        var mediaConvertClient = new AmazonMediaConvertClient();

        Console.WriteLine($"Hello AWS Elemental MediaConvert! Your MediaConvert Jobs
 are:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get some MediaConvert jobs.
        var response = await mediaConvertClient.ListJobsAsync(
            new ListJobsRequest()
            {
                MaxResults = 10
            }
```

```
            );

        foreach (var job in response.Jobs)
        {
            Console.WriteLine($"\tJob: {job.Id} status {job.Status}");
            Console.WriteLine();
        }
    }
}
```

- API-Details finden Sie DescribeEndpoints in der AWS SDK für .NET API-Referenz.

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();
```

```
        var wrapper = new MediaConvertWrapper(mcClient);
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Creating job for input file {fileInput}.");
        var jobId = await wrapper.CreateJob(mediaConvertRole!, fileInput!,
 fileOutput!);
        Console.WriteLine($"Created job with Job ID: {jobId}");
        Console.WriteLine(new string('-', 80));
```

Erstellen Sie den Auftrag mit der Wrapper-Methode und geben Sie die Auftrags-ID zurück.

```
    /// <summary>
    /// Create a job to convert a media file.
    /// </summary>
    /// <param name="mediaConvertRole">The Amazon Resource Name (ARN) of the media
 convert role, as specified here:
    /// https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-in-
mediaconvert-configured.html</param>
    /// <param name="fileInput">The Amazon Simple Storage Service (Amazon S3)
 location of the input media file.</param>
    /// <param name="fileOutput">The Amazon S3 location for the output media file.</
param>
    /// <returns>The ID of the new job.</returns>
    public async Task<string> CreateJob(string mediaConvertRole, string fileInput,
        string fileOutput)
    {
        CreateJobRequest createJobRequest = new CreateJobRequest
        {
            Role = mediaConvertRole
        };

        createJobRequest.UserMetadata.Add("Customer", "Amazon");

        JobSettings jobSettings = new JobSettings
        {
            AdAvailOffset = 0,
            TimecodeConfig = new TimecodeConfig
            {
                Source = TimecodeSource.EMBEDDED
```

```
            }
        };
        createJobRequest.Settings = jobSettings;

        #region OutputGroup

        OutputGroup ofg = new OutputGroup
        {
            Name = "File Group",
            OutputGroupSettings = new OutputGroupSettings
            {
                Type = OutputGroupType.FILE_GROUP_SETTINGS,
                FileGroupSettings = new FileGroupSettings
                {
                    Destination = fileOutput
                }
            }
        };

        Output output = new Output
        {
            NameModifier = "_1"
        };

        #region VideoDescription

        VideoDescription vdes = new VideoDescription
        {
            ScalingBehavior = ScalingBehavior.DEFAULT,
            TimecodeInsertion = VideoTimecodeInsertion.DISABLED,
            AntiAlias = AntiAlias.ENABLED,
            Sharpness = 50,
            AfdSignaling = AfdSignaling.NONE,
            DropFrameTimecode = DropFrameTimecode.ENABLED,
            RespondToAfd = RespondToAfd.NONE,
            ColorMetadata = ColorMetadata.INSERT,
            CodecSettings = new VideoCodecSettings
            {
                Codec = VideoCodec.H_264
            }
        };
        output.VideoDescription = vdes;

        H264Settings h264 = new H264Settings
```

```
        {
            InterlaceMode = H264InterlaceMode.PROGRESSIVE,
            NumberReferenceFrames = 3,
            Syntax = H264Syntax.DEFAULT,
            Softness = 0,
            GopClosedCadence = 1,
            GopSize = 90,
            Slices = 1,
            GopBReference = H264GopBReference.DISABLED,
            SlowPal = H264SlowPal.DISABLED,
            SpatialAdaptiveQuantization = H264SpatialAdaptiveQuantization.ENABLED,
            TemporalAdaptiveQuantization = H264TemporalAdaptiveQuantization.ENABLED,
            FlickerAdaptiveQuantization = H264FlickerAdaptiveQuantization.DISABLED,
            EntropyEncoding = H264EntropyEncoding.CABAC,
            Bitrate = 5000000,
            FramerateControl = H264FramerateControl.SPECIFIED,
            RateControlMode = H264RateControlMode.CBR,
            CodecProfile = H264CodecProfile.MAIN,
            Telecine = H264Telecine.NONE,
            MinIInterval = 0,
            AdaptiveQuantization = H264AdaptiveQuantization.HIGH,
            CodecLevel = H264CodecLevel.AUTO,
            FieldEncoding = H264FieldEncoding.PAFF,
            SceneChangeDetect = H264SceneChangeDetect.ENABLED,
            QualityTuningLevel = H264QualityTuningLevel.SINGLE_PASS,
            FramerateConversionAlgorithm =
                H264FramerateConversionAlgorithm.DUPLICATE_DROP,
            UnregisteredSeiTimecode = H264UnregisteredSeiTimecode.DISABLED,
            GopSizeUnits = H264GopSizeUnits.FRAMES,
            ParControl = H264ParControl.SPECIFIED,
            NumberBFramesBetweenReferenceFrames = 2,
            RepeatPps = H264RepeatPps.DISABLED,
            FramerateNumerator = 30,
            FramerateDenominator = 1,
            ParNumerator = 1,
            ParDenominator = 1
        };
        output.VideoDescription.CodecSettings.H264Settings = h264;

        #endregion VideoDescription

        #region AudioDescription

        AudioDescription ades = new AudioDescription
```

```
        {
            LanguageCodeControl = AudioLanguageCodeControl.FOLLOW_INPUT,
            // This name matches one specified in the following Inputs.
            AudioSourceName = "Audio Selector 1",
            CodecSettings = new AudioCodecSettings
            {
                Codec = AudioCodec.AAC
            }
        };

        AacSettings aac = new AacSettings
        {
            AudioDescriptionBroadcasterMix =
AacAudioDescriptionBroadcasterMix.NORMAL,
            RateControlMode = AacRateControlMode.CBR,
            CodecProfile = AacCodecProfile.LC,
            CodingMode = AacCodingMode.CODING_MODE_2_0,
            RawFormat = AacRawFormat.NONE,
            SampleRate = 48000,
            Specification = AacSpecification.MPEG4,
            Bitrate = 64000
        };
        ades.CodecSettings.AacSettings = aac;
        output.AudioDescriptions.Add(ades);

        #endregion AudioDescription

        #region Mp4 Container

        output.ContainerSettings = new ContainerSettings
        {
            Container = ContainerType.MP4
        };
        Mp4Settings mp4 = new Mp4Settings
        {
            CslgAtom = Mp4CslgAtom.INCLUDE,
            FreeSpaceBox = Mp4FreeSpaceBox.EXCLUDE,
            MoovPlacement = Mp4MoovPlacement.PROGRESSIVE_DOWNLOAD
        };
        output.ContainerSettings.Mp4Settings = mp4;

        #endregion Mp4 Container

        ofg.Outputs.Add(output);
```

```
        createJobRequest.Settings.OutputGroups.Add(ofg);

        #endregion OutputGroup

        #region Input

        Input input = new Input
        {
            FilterEnable = InputFilterEnable.AUTO,
            PsiControl = InputPsiControl.USE_PSI,
            FilterStrength = 0,
            DeblockFilter = InputDeblockFilter.DISABLED,
            DenoiseFilter = InputDenoiseFilter.DISABLED,
            TimecodeSource = InputTimecodeSource.EMBEDDED,
            FileInput = fileInput
        };

        AudioSelector audsel = new AudioSelector
        {
            Offset = 0,
            DefaultSelection = AudioDefaultSelection.NOT_DEFAULT,
            ProgramSelection = 1,
            SelectorType = AudioSelectorType.TRACK
        };
        audsel.Tracks.Add(1);
        input.AudioSelectors.Add("Audio Selector 1", audsel);

        input.VideoSelector = new VideoSelector
        {
            ColorSpace = ColorSpace.FOLLOW
        };

        createJobRequest.Settings.Inputs.Add(input);

        #endregion Input

        CreateJobResponse createJobResponse =
            await _amazonMediaConvert.CreateJobAsync(createJobRequest);

        var jobId = createJobResponse.Job.Id;

        return jobId;
    }
```

- API-Details finden Sie CreateJob in der AWS SDK für .NET API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Rufen Sie einen Job anhand seiner ID ab.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting job information for Job ID {jobId}");
        var job = await wrapper.GetJobById(jobId);
```

```
        Console.WriteLine($"Job {job.Id} created on {job.CreatedAt:d} has status
{job.Status}.");
        Console.WriteLine(new string('-', 80));
```

```
    /// <summary>
    /// Get the job information for a job by its ID.
    /// </summary>
    /// <param name="jobId">The ID of the job.</param>
    /// <returns>The Job object.</returns>
    public async Task<Job> GetJobById(string jobId)
    {
        var jobResponse = await _amazonMediaConvert.GetJobAsync(
                new GetJobRequest
                {
                    Id = jobId
                });

        return jobResponse.Job;
    }
```

- API-Details finden Sie GetJob in der AWS SDK für .NET API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die Dateispeicherorte, den Client und den Wrapper ein.

```
        // MediaConvert role Amazon Resource Name (ARN).
        // For information on creating this role, see
```

```
        // https://docs.aws.amazon.com/mediaconvert/latest/ug/creating-the-iam-role-
in-mediaconvert-configured.html.
        var mediaConvertRole = _configuration["mediaConvertRoleARN"];

        // Include the file input and output locations in settings.json or
 settings.local.json.
        var fileInput = _configuration["fileInput"];
        var fileOutput = _configuration["fileOutput"];

        AmazonMediaConvertClient mcClient = new AmazonMediaConvertClient();

        var wrapper = new MediaConvertWrapper(mcClient);
```

Listen Sie die Jobs mit einem bestimmten Status auf.

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Listing all complete jobs.");
        var completeJobs = await wrapper.ListAllJobsByStatus(JobStatus.COMPLETE);
        completeJobs.ForEach(j =>
        {
            Console.WriteLine($"Job {j.Id} created on {j.CreatedAt:d} has status
 {j.Status}.");
        });
```

Listen Sie die Jobs mithilfe eines Paginators auf.

```
    /// <summary>
    /// List all of the jobs with a particular status using a paginator.
    /// </summary>
    /// <param name="status">The status to use when listing jobs.</param>
    /// <returns>The list of jobs matching the status.</returns>
    public async Task<List<Job>> ListAllJobsByStatus(JobStatus? status = null)
    {
        var returnedJobs = new List<Job>();

        var paginatedJobs = _amazonMediaConvert.Paginators.ListJobs(
                new ListJobsRequest
                {
                    Status = status
                });
```

```
        // Get the entire list using the paginator.
        await foreach (var job in paginatedJobs.Jobs)
        {
            returnedJobs.Add(job);
        }

        return returnedJobs;
    }
```

- API-Details finden Sie ListJobs in der AWS SDK für .NET API-Referenz.

Amazon MSK-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von .NET.

```
using System.Text;
using Amazon.Lambda.Core;
using Amazon.Lambda.KafkaEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace MSKLambda;

public class Function
{


    /// <param name="input">The event for the Lambda function handler to process.</
param>
    /// <param name="context">The ILambdaContext that provides methods for logging
 and describing the Lambda environment.</param>
    /// <returns></returns>
    public void FunctionHandler(KafkaEvent evnt, ILambdaContext context)
    {

        foreach (var record in evnt.Records)
        {
            Console.WriteLine("Key:" + record.Key);
            foreach (var eventRecord in record.Value)
            {
                var valueBytes = eventRecord.Value.ToArray();
                var valueText = Encoding.UTF8.GetString(valueBytes);

                Console.WriteLine("Message:" + valueText);
            }
        }
    }


}
```

Beispiele für Organisationen mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET with Organizations Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachPolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
```

```
    {
        /// <summary>
        /// Initializes the Organizations client object and then calls the
        /// AttachPolicyAsync method to attach the policy to the root
        /// organization.
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new AttachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.AttachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- API-Details finden Sie AttachPolicy in der AWS SDK für .NET API-Referenz.

`CreateAccount`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccount`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations account.
/// </summary>
public class CreateAccount
{
    /// <summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
        };

        var response = await client.CreateAccountAsync(request);
        var status = response.CreateAccountStatus;

        Console.WriteLine($"The staus of {status.AccountName} is
{status.State}.");
    }
}
```

- API-Details finden Sie CreateAccount in der AWS SDK für .NET API-Referenz.

`CreateOrganization`

Das folgende Codebeispiel zeigt die Verwendung`CreateOrganization`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;
```

```
            Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
{newOrg.MasterAccountId}");
        }
    }
```

- API-Details finden Sie CreateOrganization in der AWS SDK für .NET API-Referenz.

`CreateOrganizationalUnit`

Das folgende Codebeispiel zeigt die Verwendung`CreateOrganizationalUnit`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var orgUnitName = "ProductDevelopmentUnit";

            var request = new CreateOrganizationalUnitRequest
            {
                Name = orgUnitName,
                ParentId = "r-0000",
            };

            var response = await client.CreateOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created organizational unit:
{orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
{response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- API-Details finden Sie CreateOrganizationalUnit in der AWS SDK für .NET API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
```

```
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations Policy.
/// </summary>
public class CreatePolicy
{
    /// <summary>
    /// Initializes the AWS Organizations client object, uses it to
    /// create a new Organizations Policy, and then displays information
    /// about the newly created Policy.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyContent = "{" +
            "   \"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:*\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        try
        {
            var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
            {
                Content = policyContent,
                Description = "Enables admins of attached accounts to delegate
all Amazon S3 permissions",
                Name = "AllowAllS3Actions",
                Type = "SERVICE_CONTROL_POLICY",
            });

            Policy policy = response.Policy;
            Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
        }
        catch (Exception ex)
        {
```

```
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- API-Details finden Sie CreatePolicy in der AWS SDK für .NET API-Referenz.

`DeleteOrganization`

Das folgende Codebeispiel zeigt die Verwendung`DeleteOrganization`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Successfully deleted organization.");
            }
            else
            {
                Console.WriteLine("Could not delete organization.");
            }
        }
    }
```

- API-Details finden Sie DeleteOrganization in der AWS SDK für .NET API-Referenz.

`DeleteOrganizationalUnit`

Das folgende Codebeispiel zeigt die Verwendung`DeleteOrganizationalUnit`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Shows how to delete an existing AWS Organizations organizational unit.
    /// </summary>
    public class DeleteOrganizationalUnit
    {
        /// <summary>
```

```
        /// Initializes the Organizations client object and calls
        /// DeleteOrganizationalUnitAsync to delete the organizational unit
        /// with the selected ID.
        /// </summary>
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var orgUnitId = "ou-0000-00000000";

            var request = new DeleteOrganizationalUnitRequest
            {
                OrganizationalUnitId = orgUnitId,
            };

            var response = await client.DeleteOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- API-Details finden Sie DeleteOrganizationalUnit in der AWS SDK für .NET API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
        else
        {
```

```
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- API-Details finden Sie DeletePolicy in der AWS SDK für .NET API-Referenz.

`DetachPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachPolicy`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";
```

```
            var targetId = "r-0000";

            var request = new DetachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.DetachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- API-Details finden Sie DetachPolicy in der AWS SDK für .NET API-Referenz.

`ListAccounts`

Das folgende Codebeispiel zeigt die Verwendung`ListAccounts`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;
```

```
/// <summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account.
/// </summary>
public class ListAccounts
{
    /// <summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var request = new ListAccountsRequest
        {
            MaxResults = 5,
        };

        var response = new ListAccountsResponse();
        try
        {
            do
            {
                response = await client.ListAccountsAsync(request);
                response.Accounts.ForEach(a => DisplayAccounts(a));
                if (response.NextToken is not null)
                {
                    request.NextToken = response.NextToken;
                }
            }
            while (response.NextToken is not null);
        }
        catch (AWSOrganizationsNotInUseException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    /// <summary>
    /// Displays information about an Organizations account.
    /// </summary>
```

```
        /// <param name="account">An Organizations account for which to display
        /// information on the console.</param>
        private static void DisplayAccounts(Account account)
        {
            string accountInfo = $"{account.Id} {account.Name}\t{account.Status}";

            Console.WriteLine(accountInfo);
        }
    }
```

- API-Details finden Sie ListAccounts in der AWS SDK für .NET API-Referenz.

`ListOrganizationalUnitsForParent`

Das folgende Codebeispiel zeigt die Verwendung`ListOrganizationalUnitsForParent`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Lists the AWS Organizations organizational units that belong to an
    /// organization.
    /// </summary>
    public class ListOrganizationalUnitsForParent
    {
        /// <summary>
        /// Initializes the Organizations client object and then uses it to
        /// call the ListOrganizationalUnitsForParentAsync method to retrieve
        /// the list of organizational units.
        /// </summary>
```

```
        public static async Task Main()
        {
            // Create the client object using the default account.
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var parentId = "r-0000";

            var request = new ListOrganizationalUnitsForParentRequest
            {
                ParentId = parentId,
                MaxResults = 5,
            };

            var response = new ListOrganizationalUnitsForParentResponse();
            try
            {
                do
                {
                    response = await
client.ListOrganizationalUnitsForParentAsync(request);
                    response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations organizational unit.
        /// </summary>
        /// <param name="unit">The OrganizationalUnit for which to display
        /// information.</param>
        public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
        {
            string accountInfo = $"{unit.Id} {unit.Name}\t{unit.Arn}";
```

```
            Console.WriteLine(accountInfo);
        }
    }
```

- API-Details finden Sie ListOrganizationalUnitsForParent in der AWS SDK für .NET API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            // The value for the Filter parameter is required and must must be
            // one of the following:
            //     AISERVICES_OPT_OUT_POLICY
            //     BACKUP_POLICY
            //     SERVICE_CONTROL_POLICY
            //     TAG_POLICY
            var request = new ListPoliciesRequest
            {
                Filter = "SERVICE_CONTROL_POLICY",
                MaxResults = 5,
            };

            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id} {policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
```

```
    }
```

- API-Details finden Sie ListPolicies in der AWS SDK für .NET API-Referenz.

Partner Central-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Partner Central Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`CreateOpportunity`.

SDK für .NET

Erstellen Sie eine Opportunity.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;
```

```
namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new CreateOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Origin = "Partner Referral",
                    Customer = new Customer
                    {
                        Account = new Account
                        {
                            Address = new Address
                            {
                                CountryCode = "US",
                                PostalCode = "99502",
                                StateOrRegion = "Alaska"
                            },
                            CompanyName = "TestCompanyName",
                            Duns = "123456789",
                            WebsiteUrl = "www.test.io",
                            Industry = "Automotive"
                        },
                        Contacts = new List<Contact>
                        {
                            new Contact
                            {
                                Email = "test@test.io",
                                FirstName = "John  ",
                                LastName = "Doe",
                                Phone = "+14444444444",
```

```
                                BusinessTitle = "test title"
                            }
                        }
                    },
                    LifeCycle = new LifeCycle
                    {
                        ReviewStatus = "Submitted",
                        TargetCloseDate = "2024-12-30"
                    },
                    Marketing = new Marketing
                    {
                        Source = "None"
                    },
                    OpportunityType = "Net New Business",
                    PrimaryNeedsFromAws = new List<string> { "Co-Sell -
Architectural Validation" },
                    Project = new Project
                    {
                        Title = "Moin Test UUID",
                        CustomerBusinessProblem = "Sandbox is not working as
expected",
                        CustomerUseCase = "AI Machine Learning and Analytics",
                        DeliveryModels = new List<string> { "SaaS or PaaS" },
                        ExpectedCustomerSpend = new List<ExpectedCustomerSpend>
                        {
                            new ExpectedCustomerSpend
                            {
                                Amount = "2000.0",
                                CurrencyCode = "USD",
                                Frequency = "Monthly",
                                TargetCompany = "Ibexlabs"
                            }
                        },
                        SalesActivities = new List<string> { "Initialized
discussions with customer" }
                    }
                };

                try
                {
                    var response = await client.CreateOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
Formatting.Indented);
```

```
                    Console.WriteLine(formattedJson);
                }
                catch (ValidationException ex)
                {
                    Console.WriteLine("Validation error: " + ex.Message);
                }
                catch (AmazonPartnerCentralSellingException e)
                {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- API-Details finden Sie CreateOpportunity in der AWS SDK für .NET API-Referenz.

`GetOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`GetOpportunity`.

SDK für .NET

Rufen Sie eine Opportunity ab.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
```

```
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static readonly string identifier = "O1111111";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new GetOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Identifier = identifier
                };

                try {
                    var response = await client.GetOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
```

```
}
```

- API-Details finden Sie GetOpportunity in der AWS SDK für .NET API-Referenz.

`ListOpportunities`

Das folgende Codebeispiel zeigt die Verwendung`ListOpportunities`.

SDK für .NET

Listen Sie Opportunitys auf.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "Sandbox";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                //var config = new AmazonPartnerCentralSellingConfig()
                //{
                //    ServiceURL = "https://partnercentral-selling.us-
east-1.api.aws",
                //};
```

```
                //var client = new AmazonPartnerCentralSellingClient(awsCredentials,
 config);
                var client = new AmazonPartnerCentralSellingClient(awsCredentials);
                var request = new ListOpportunitiesRequest
                {
                    Catalog = catalogToUse,
                    MaxResults = 2
                };

                try {
                    var response = await client.ListOpportunitiesAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    foreach (var opportunity in response.OpportunitySummaries)
                    {
                        Console.WriteLine("Opportunity id: " + opportunity.Id);
                    }
                    string formattedJson =
 JsonConvert.SerializeObject(response.OpportunitySummaries, Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- API-Details finden Sie ListOpportunities in der AWS SDK für .NET API-Referenz.

Amazon Pinpoint-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
using Amazon;
using Amazon.Pinpoint;
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendEmailMessage;

public class SendEmailMainClass
{
```

```
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load test settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally load local settings.
        .Build();

        // The AWS Region that you want to use to send the email. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The "From" address. This address has to be verified in Amazon Pinpoint
        // in the region you're using to send email.
        string senderAddress = configuration["SenderAddress"]!;

        // The address on the "To" line. If your Amazon Pinpoint account is in
        // the sandbox, this address also has to be verified.
        string toAddress = configuration["ToAddress"]!;

        // The Amazon Pinpoint project/application ID to use when you send this
message.
        // Make sure that the SMS channel is enabled for the project or application
        // that you choose.
        string appId = configuration["AppId"]!;

        try
        {
            await SendEmailMessage(region, appId, toAddress, senderAddress);
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
ex.Message);
        }
    }

    public static async Task<MessageResponse> SendEmailMessage(
        string region, string appId, string toAddress, string senderAddress)
    {
        var client = new
AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));
```

```
        // The subject line of the email.
        string subject = "Amazon Pinpoint Email test";

        // The body of the email for recipients whose email clients don't
        // support HTML content.
        string textBody = @"Amazon Pinpoint Email Test (.NET)"
                          + "\n----------------------------------"
                          + "\nThis email was sent using the Amazon Pinpoint API
 using the AWS SDK for .NET.";

        // The body of the email for recipients whose email clients support
        // HTML content.
        string htmlBody = @"<html>"
                          + "\n<head></head>"
                          + "\n<body>"
                          + "\n  <h1>Amazon Pinpoint Email Test (AWS SDK for .NET)</
h1>"
                          + "\n  <p>This email was sent using the "
                          + "\n    <a href='https://aws.amazon.com/pinpoint/'>Amazon
 Pinpoint</a> API "
                          + "\n    using the <a href='https://aws.amazon.com/sdk-
for-net/'>AWS SDK for .NET</a>"
                          + "\n  </p>"
                          + "\n</body>"
                          + "\n</html>";

        // The character encoding the you want to use for the subject line and
        // message body of the email.
        string charset = "UTF-8";

        var sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
            {
                Addresses = new Dictionary<string, AddressConfiguration>
                {
                    {
                        toAddress,
                        new AddressConfiguration
                        {
                            ChannelType = ChannelType.EMAIL
                        }
```

```
                    }
                },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    EmailMessage = new EmailMessage
                    {
                        FromAddress = senderAddress,
                        SimpleEmail = new SimpleEmail
                        {
                            HtmlPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = htmlBody
                            },
                            TextPart = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = textBody
                            },
                            Subject = new SimpleEmailPart
                            {
                                Charset = charset,
                                Data = subject
                            }
                        }
                    }
                }
            }
        };
        Console.WriteLine("Sending message...");
        SendMessagesResponse response = await client.SendMessagesAsync(sendRequest);
        Console.WriteLine("Message sent!");
        return response.MessageResponse;
    }
}
```

Senden Sie eine SMS-Nachricht.

```
using Amazon;
using Amazon.Pinpoint;
```

```
using Amazon.Pinpoint.Model;
using Microsoft.Extensions.Configuration;

namespace SendSmsMessage;

public class SendSmsMessageMainClass
{
    public static async Task Main(string[] args)
    {
        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // The AWS Region that you want to use to send the message. For a list of
        // AWS Regions where the Amazon Pinpoint API is available, see
        // https://docs.aws.amazon.com/pinpoint/latest/apireference/
        string region = "us-east-1";

        // The phone number or short code to send the message from. The phone number
        // or short code that you specify has to be associated with your Amazon
 Pinpoint
        // account. For best results, specify long codes in E.164 format.
        string originationNumber = configuration["OriginationNumber"]!;

        // The recipient's phone number.  For best results, you should specify the
        // phone number in E.164 format.
        string destinationNumber = configuration["DestinationNumber"]!;

        // The Pinpoint project/ application ID to use when you send this message.
        // Make sure that the SMS channel is enabled for the project or application
        // that you choose.
        string appId = configuration["AppId"]!;

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        MessageType messageType = MessageType.TRANSACTIONAL;

        // The registered keyword associated with the originating short code.
        string? registeredKeyword = configuration["RegisteredKeyword"];
```

```
        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        string? senderId = configuration["SenderId"];

        try
        {
            var response = await SendSmsMessage(region, appId, destinationNumber,
                originationNumber, registeredKeyword, senderId, messageType);
            Console.WriteLine($"Message sent to
 {response.MessageResponse.Result.Count} recipient(s).");
            foreach (var messageResultValue in
                     response.MessageResponse.Result.Select(r => r.Value))
            {
                Console.WriteLine($"{messageResultValue.MessageId} Status:
 {messageResultValue.DeliveryStatus}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("The message wasn't sent. Error message: " +
 ex.Message);
        }
    }

    public static async Task<SendMessagesResponse> SendSmsMessage(
        string region, string appId, string destinationNumber, string
 originationNumber,
        string? keyword, string? senderId, MessageType messageType)
    {

        // The content of the SMS message.
        string message = "This message was sent through Amazon Pinpoint using" +
                         " the AWS SDK for .NET. Reply STOP to opt out.";


        var client = new
 AmazonPinpointClient(RegionEndpoint.GetBySystemName(region));

        SendMessagesRequest sendRequest = new SendMessagesRequest
        {
            ApplicationId = appId,
            MessageRequest = new MessageRequest
```

```
            {
                Addresses =
                    new Dictionary<string, AddressConfiguration>
                    {
                        {
                            destinationNumber,
                            new AddressConfiguration { ChannelType =
 ChannelType.SMS }
                        }
                    },
                MessageConfiguration = new DirectMessageConfiguration
                {
                    SMSMessage = new SMSMessage
                    {
                        Body = message,
                        MessageType = MessageType.TRANSACTIONAL,
                        OriginationNumber = originationNumber,
                        SenderId = senderId,
                        Keyword = keyword
                    }
                }
            }
        };
        SendMessagesResponse response = await client.SendMessagesAsync(sendRequest);
        return response;
    }
}
```

- API-Details finden Sie SendMessages in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Polly mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DeleteLexicon`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLexicon`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Deletes an existing Amazon Polly lexicon using the AWS SDK for .NET.
/// </summary>
public class DeleteLexicon
{
    public static async Task Main()
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        var success = await DeletePollyLexiconAsync(client, lexiconName);

        if (success)
```

```
            {
                Console.WriteLine($"Successfully deleted {lexiconName}.");
            }
            else
            {
                Console.WriteLine($"Could not delete {lexiconName}.");
            }
        }

        /// <summary>
        /// Deletes the named Amazon Polly lexicon.
        /// </summary>
        /// <param name="client">The initialized Amazon Polly client object.</param>
        /// <param name="lexiconName">The name of the Amazon Polly lexicon to
        /// delete.</param>
        /// <returns>A Boolean value indicating the success of the operation.</
returns>
        public static async Task<bool> DeletePollyLexiconAsync(
            AmazonPollyClient client,
            string lexiconName)
        {
            var deleteLexiconRequest = new DeleteLexiconRequest()
            {
                Name = lexiconName,
            };

            var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
```

- API-Details finden Sie DeleteLexicon in der AWS SDK für .NET API-Referenz.

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
            while (nextToken is not null);
```

```
                do
                {
                    var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                    nextToken = enUsVoicesResponse.NextToken;
                    enUsVoicesRequest.NextToken = nextToken;

                    Console.WriteLine("\nen-US voices: ");
                    enUsVoicesResponse.Voices.ForEach(voice =>
                    {
                        DisplayVoiceInfo(voice);
                    });
                }
                while (nextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }

        public static void DisplayVoiceInfo(Voice voice)
        {
            Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
        }
    }
```

- API-Details finden Sie DescribeVoices in der AWS SDK für .NET API-Referenz.

`GetLexicon`

Das folgende Codebeispiel zeigt die Verwendung`GetLexicon`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Retrieves information about a specific Amazon Polly lexicon.
/// </summary>
public class GetLexicon
{
    public static async Task Main(string[] args)
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        await GetPollyLexiconAsync(client, lexiconName);
    }

    public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
string lexiconName)
    {
        var getLexiconRequest = new GetLexiconRequest()
        {
            Name = lexiconName,
        };

        try
        {
            var response = await client.GetLexiconAsync(getLexiconRequest);
            Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
            Console.WriteLine($"Content: {response.Lexicon.Content}");
        }
        catch (Exception ex)
```

```
            {
                Console.WriteLine("Error: " + ex.Message);
            }
        }
    }
```

- API-Details finden Sie GetLexicon in der AWS SDK für .NET API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

/// <summary>
/// Lists the Amazon Polly lexicons that have been defined. By default,
/// lists the lexicons that are defined in the same AWS Region as the default
/// user. To view Amazon Polly lexicons that are defined in a different AWS
/// Region, supply it as a parameter to the Amazon Polly constructor.
/// </summary>
public class ListLexicons
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        var request = new ListLexiconsRequest();

        try
```

```
            {
                Console.WriteLine("All voices: ");

                do
                {
                    var response = await client.ListLexiconsAsync(request);
                    request.NextToken = response.NextToken;

                    response.Lexicons.ForEach(lexicon =>
                    {
                        var attributes = lexicon.Attributes;
                        Console.WriteLine($"Name: {lexicon.Name}");
                        Console.WriteLine($"\tAlphabet: {attributes.Alphabet}");
                        Console.WriteLine($"\tLanguageCode:
{attributes.LanguageCode}");
                        Console.WriteLine($"\tLastModified:
{attributes.LastModified}");
                        Console.WriteLine($"\tLexemesCount:
{attributes.LexemesCount}");
                        Console.WriteLine($"\tLexiconArn: {attributes.LexiconArn}");
                        Console.WriteLine($"\tSize: {attributes.Size}");
                    });
                }
                while (request.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- API-Details finden Sie ListLexicons in der AWS SDK für .NET API-Referenz.

`PutLexicon`

Das folgende Codebeispiel zeigt die Verwendung`PutLexicon`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    /// <summary>
    /// Creates a new Amazon Polly lexicon using the AWS SDK for .NET.
    /// </summary>
    public class PutLexicon
    {
        public static async Task Main()
        {
            string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
                "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/
pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
                "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-
lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
                "alphabet=\"ipa\" xml:lang=\"en-US\">" +
                "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>" +
                "</lexicon>";
            string lexiconName = "SampleLexicon";

            var client = new AmazonPollyClient();
            var putLexiconRequest = new PutLexiconRequest()
            {
                Name = lexiconName,
                Content = lexiconContent,
            };

            try
            {
                var response = await client.PutLexiconAsync(putLexiconRequest);
                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
```

```
                    Console.WriteLine($"Successfully created Lexicon:
{lexiconName}.");
                }
                else
                {
                    Console.WriteLine($"Could not create Lexicon: {lexiconName}.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Exception caught: " + ex.Message);
            }
        }
    }
```

- API-Details finden Sie PutLexicon in der AWS SDK für .NET API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
```

```
            string text = "Twas brillig, and the slithy toves did gyre and gimbol in
the wabe";

            var client = new AmazonPollyClient();
            var response = await PollySynthesizeSpeech(client, text);

            WriteSpeechToStream(response.AudioStream, outputFileName);
        }

        /// <summary>
        /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
        /// to speech.
        /// </summary>
        /// <param name="client">The Amazon Polly client object used to connect
        /// to the Amazon Polly service.</param>
        /// <param name="text">The text to convert to speech.</param>
        /// <returns>A SynthesizeSpeechResponse object that includes an AudioStream
        /// object with the converted text.</returns>
        private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
        {
            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Mp3,
                VoiceId = VoiceId.Joanna,
                Text = text,
            };

            var synthesizeSpeechResponse =
                await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

            return synthesizeSpeechResponse;
        }

        /// <summary>
        /// Writes the AudioStream returned from the call to
        /// SynthesizeSpeechAsync to a file in MP3 format.
        /// </summary>
        /// <param name="audioStream">The AudioStream returned from the
        /// call to the SynthesizeSpeechAsync method.</param>
        /// <param name="outputFileName">The full path to the file in which to
        /// save the audio stream.</param>
        private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
```

```
        {
            var outputStream = new FileStream(
                outputFileName,
                FileMode.Create,
                FileAccess.Write);
            byte[] buffer = new byte[2 * 1024];
            int readBytes;

            while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
            {
                outputStream.Write(buffer, 0, readBytes);
            }

            // Flushes the buffer to avoid losing the last second or so of
            // the synthesized text.
            outputStream.Flush();
            Console.WriteLine($"Saved {outputFileName} to disk.");
        }
    }
```

Synthetisieren Sie mit Amazon Polly mithilfe eines SDK Sprache aus Text mithilfe von Sprachzeichen. AWS

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Polly;
    using Amazon.Polly.Model;

    public class SynthesizeSpeechMarks
    {
        public static async Task Main()
        {
            var client = new AmazonPollyClient();
            string outputFileName = "speechMarks.json";

            var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
            {
                OutputFormat = OutputFormat.Json,
                SpeechMarkTypes = new List<string>
```

```
                {
                    SpeechMarkType.Viseme,
                    SpeechMarkType.Word,
                },
                VoiceId = VoiceId.Joanna,
                Text = "This is a sample text to be synthesized.",
            };

            try
            {
                using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
                {
                    var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                    var buffer = new byte[2 * 1024];
                    int readBytes;

                    var inputStream = synthesizeSpeechResponse.AudioStream;
                    while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) > 0)
                    {
                        outputStream.Write(buffer, 0, readBytes);
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- API-Details finden Sie SynthesizeSpeech in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon RDS unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.RDS;
using Amazon.RDS.Model;

namespace RDSActions;
```

```
public static class HelloRds
{
    static async Task Main(string[] args)
    {
        var rdsClient = new AmazonRDSClient();

        Console.WriteLine($"Hello Amazon RDS! Following are some of your DB
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first twenty DB instances.
        var response = await rdsClient.DescribeDBInstancesAsync(
            new DescribeDBInstancesRequest()
            {
                MaxRecords = 20 // Must be between 20 and 100.
            });

        foreach (var instance in response.DBInstances)
        {
            Console.WriteLine($"\tDB name: {instance.DBName}");
            Console.WriteLine($"\tArn: {instance.DBInstanceArn}");
            Console.WriteLine($"\tIdentifier: {instance.DBInstanceIdentifier}");
            Console.WriteLine();
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für .NET - API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.

- Löschen Sie die Instance und die Parametergruppe.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSInstanceScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Returns a list of the available DB engine families using the
 DescribeDBEngineVersionsAsync method.
    2.  Selects an engine family and creates a custom DB parameter group using the
 CreateDBParameterGroupAsync method.
    3.  Gets the parameter groups using the DescribeDBParameterGroupsAsync method.
    4.  Gets parameters in the group using the DescribeDBParameters method.
    5.  Parses and displays parameters in the group.
    6.  Modifies both the auto_increment_offset and auto_increment_increment
 parameters
        using the ModifyDBParameterGroupAsync method.
    7.  Gets and displays the updated parameters using the DescribeDBParameters
 method with a source of "user".
    8.  Gets a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Displays and selects from a list of micro instance classes available for the
 selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
 parameter group
        using the CreateDBInstanceAsync method.
```

```
   11. Waits for DB instance to be ready using the DescribeDBInstancesAsync method.
   12. Prints out the connection endpoint string for the new DB instance.
   13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
method.
   14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
   15. Deletes the DB instance using the DeleteDBInstanceAsync method.
   16. Waits for DB instance to be deleted using the DescribeDbInstances method.
   17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static RDSWrapper rdsWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon RDS service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<RDSWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<RDSInstanceScenario>();

       rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

       Console.WriteLine(sepBar);
       Console.WriteLine(
           "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
       Console.WriteLine(sepBar);

       try
       {
```

```
            var parameterGroupFamily = await ChooseParameterGroupFamily();

            var parameterGroup = await CreateDbParameterGroup(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParameters(parameterGroup.DBParameterGroupName, parameters);

            await DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

            var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

            var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
                instanceChoice.DBInstanceClass, newInstanceIdentifier);
            if (newInstance != null)
            {
                DisplayConnectionString(newInstance);

                await CreateSnapshot(newInstance);

                await DeleteRdsInstance(newInstance);
            }

            await DeleteParameterGroup(parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }
```

```
    /// <summary>
    /// Choose the RDS DB parameter group family from a list of available options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

        Console.WriteLine("1. The following is a list of available DB parameter
group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new
DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");

        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
```

```
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        foreach (var p in parameters)
        {
            if (p.IsModifiable && p.DataType == "integer")
            {
                int newValue = 0;
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    Int32.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

        Console.WriteLine(sepBar);
```

```
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");


        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }


    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family
{dbParameterGroupFamily}:");
```

```
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. Engine: {version.Engine} Version
 {version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption> ChooseDbInstanceClass(string
 engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
 engineVersion);

        Console.WriteLine($"8. Available micro DB instance classes for engine
 {engine} and version {engineVersion}:");
        int i = 1;

        // Filter to micro instances for this example.
```

```
        allowedInstances = allowedInstances
            .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("9. Select an available DB instance class by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new RDS DB instance.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB instance.</
param>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
 instanceIdentifier)
    {
        Console.WriteLine(sepBar);
```

```
        Console.WriteLine($"10. Create a new DB instance with identifier
{instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await rdsWrapper.DescribeDBInstances();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
        }
        else
        {
            Console.WriteLine("Please enter an admin user name:");
            var username = Console.ReadLine();

            Console.WriteLine("Please enter an admin password:");
            var password = Console.ReadLine();

            newInstance = await rdsWrapper.CreateDBInstance(
                "ExampleInstance",
                instanceIdentifier,
                parameterGroup.DBParameterGroupName,
                engineName,
                engineVersion,
                instanceClass,
                20,
                username,
                password
            );

            // 11. Wait for the DB instance to be ready.

            Console.WriteLine("11. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
```

```
                newInstance = instances.First();
                Thread.Sleep(30000);
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("12. New DB instance connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {instance.Endpoint.Address} -P {instance.Endpoint.Port}
 "
            + $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an RDS DB instance.
    /// </summary>
    /// <param name="instance">DB instance to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"13. Creating snapshot from DB instance
 {instance.DBInstanceIdentifier}.");
        var snapshot = await
 rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
 DateTime.Now.Ticks);

        // Wait for the snapshot to be available
        bool isSnapshotReady = false;
```

```
        Console.WriteLine($"14. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
            Thread.Sleep(30000);
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Delete an RDS DB instance.
    /// </summary>
    /// <param name="instance">The DB instance to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteRdsInstance(DBInstance newInstance)
    {
        Console.WriteLine(sepBar);
        // Delete the DB instance.
        Console.WriteLine($"15. Delete the DB instance
{newInstance.DBInstanceIdentifier}.");
        await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

        // Wait for the DB instance to delete.
        Console.WriteLine($"16. Waiting for the DB instance to delete...");
        bool isInstanceDeleted = false;

        while (!isInstanceDeleted)
        {
            var instance = await rdsWrapper.DescribeDBInstances();
            isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
            Thread.Sleep(30000);
        }

        Console.WriteLine("DB instance deleted.");
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Delete a DB parameter group.
    /// </summary>
    /// <param name="parameterGroup">The parameter group to delete.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteParameterGroup(DBParameterGroup parameterGroup)
    {
        Console.WriteLine(sepBar);
        // Delete the parameter group.
        Console.WriteLine($"17. Delete the DB parameter group
 {parameterGroup.DBParameterGroupName}.");
        await
 rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

        Console.WriteLine(sepBar);
    }
```

Wrapper-Methoden, die vom Szenario für DB-Instance-Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with DB
 instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }


    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family name.</
param>
```

```
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }



    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }



    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
```

```
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }



    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

Wrapper-Methoden, die vom Szenario für DB-Parametergruppen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name
 = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }



    /// <summary>
    /// Create a new DB parameter group. Use the action
 DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
```

```
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }



    /// <summary>
    /// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }



    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
 group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
```

```
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter group.</
param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

Wrapper-Methoden, die vom Szenario für DB-Snapshot-Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
 snapshots.
/// </summary>
```

```
public partial class RDSWrapper
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
 string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }



    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
 dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
```

```
        return results;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
    {
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBName = dbName,
                DBInstanceIdentifier = dbInstanceIdentifier,
                DBParameterGroupName = parameterGroupName,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass,
                AllocatedStorage = allocatedStorage,
                MasterUsername = adminName,
                MasterUserPassword = adminPassword
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für .NET -API-Referenz.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
   /// to determine when the DB parameter group is ready to use.
   /// </summary>
   /// <param name="name">Name of the DB parameter group.</param>
   /// <param name="family">Family of the DB parameter group.</param>
   /// <param name="description">Description of the DB parameter group.</param>
   /// <returns>The new DB parameter group.</returns>
   public async Task<DBParameterGroup> CreateDBParameterGroup(
       string name, string family, string description)
   {
       var response = await _amazonRDS.CreateDBParameterGroupAsync(
           new CreateDBParameterGroupRequest()
           {
               DBParameterGroupName = name,
               DBParameterGroupFamily = family,
               Description = description
           });
       return response.DBParameterGroup;
   }
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK für .NET API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,
string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
            {
                DBSnapshotIdentifier = snapshotIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        return response.DBSnapshot;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für .NET -API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier,
            SkipFinalSnapshot = true,
            DeleteAutomatedBackups = true
        });

    return response.DBInstance;
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für .NET -API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter
group
    /// or be associated with any DB instances.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK für .NET API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für .NET API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
```

```
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für .NET - API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
```

```
    /// <param name="name">Optional name of the DB parameter group to describe.</
param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name
 = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK für .NET API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB parameters from a specific parameter group.
    /// </summary>
    /// <param name="dbParameterGroupName">Name of a specific DB parameter group.</
param>
    /// <param name="source">Optional source for selecting parameters.</param>
    /// <returns>List of parameter values.</returns>
    public async Task<List<Parameter>> DescribeDBParameters(string
 dbParameterGroupName, string source = null)
    {
        var results = new List<Parameter>();
```

```
        var paginateParameters = _amazonRDS.Paginators.DescribeDBParameters(
            new DescribeDBParametersRequest()
            {
                DBParameterGroupName = dbParameterGroupName,
                Source = source
            });
        // Get the entire list using the paginator.
        await foreach (var parameters in paginateParameters.Parameters)
        {
            results.Add(parameters);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für .NET - API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
```

```
        var snapshotsPaginator = _amazonRDS.Paginators.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });

        // Get the entire list using the paginator.
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
        {
            results.Add(snapshots);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für .NET - API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
```

```
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für .NET API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
```

```
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="parameters">List of parameters. Maximum of 20 per request.</
param>
    /// <returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie Sie den verwenden AWS SDK für .NET , um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend erstellt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion mithilfe von .NET.

```
using System.Data;
using System.Text.Json;
using Amazon.Lambda.APIGatewayEvents;
using Amazon.Lambda.Core;
using MySql.Data.MySqlClient;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace aws_rds;
```

```
public class InputModel
{
    public string key1 { get; set; }
    public string key2 { get; set; }
}

public class Function
{
    /// <summary>
    // Handles the Lambda function execution for connecting to RDS using IAM
 authentication.
    /// </summary>
    /// <param name="input">The input event data passed to the Lambda function</
param>
    /// <param name="context">The Lambda execution context that provides runtime
 information</param>
    /// <returns>A response object containing the execution result</returns>

    public async Task<APIGatewayProxyResponse>
 FunctionHandler(APIGatewayProxyRequest request, ILambdaContext context)
    {
        // Sample Input: {"body": "{\"key1\":\"20\", \"key2\":\"25\"}"}
        var input = JsonSerializer.Deserialize<InputModel>(request.Body);

        /// Obtain authentication token
        var authToken = RDSAuthTokenGenerator.GenerateAuthToken(
            Environment.GetEnvironmentVariable("RDS_ENDPOINT"),
            Convert.ToInt32(Environment.GetEnvironmentVariable("RDS_PORT")),
            Environment.GetEnvironmentVariable("RDS_USERNAME")
        );

        /// Build the Connection String with the Token
        string connectionString =
 $"Server={Environment.GetEnvironmentVariable("RDS_ENDPOINT")};" +

 $"Port={Environment.GetEnvironmentVariable("RDS_PORT")};" +

 $"Uid={Environment.GetEnvironmentVariable("RDS_USERNAME")};" +
                                  $"Pwd={authToken};";


        try
        {
```

```
            await using var connection = new MySqlConnection(connectionString);
            await connection.OpenAsync();

            const string sql = "SELECT @param1 + @param2 AS Sum";

            await using var command = new MySqlCommand(sql, connection);
            command.Parameters.AddWithValue("@param1", int.Parse(input.key1 ??
 "0"));
            command.Parameters.AddWithValue("@param2", int.Parse(input.key2 ??
 "0"));

            await using var reader = await command.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                int result = reader.GetInt32("Sum");

                //Sample Response: {"statusCode":200,"body":"{\"message\":\"The sum
 is: 45\"}","isBase64Encoded":false}
                return new APIGatewayProxyResponse
                {
                    StatusCode = 200,
                    Body = JsonSerializer.Serialize(new { message = $"The sum is:
 {result}" })
                };
            }

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = 500,
            Body = JsonSerializer.Serialize(new { error = "Internal server error" })
        };
    }
}
```

Beispiele für Amazon RDS Data Service unter SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für .NET Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für .NET In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend erstellt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Rekognition mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to compare faces in two images.
    /// </summary>
    public class CompareFaces
    {
        public static async Task Main()
        {
            float similarityThreshold = 70F;
            string sourceImage = "source.jpg";
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
```

```
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top} matches
with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
        }
    }
```

- API-Details finden Sie CompareFaces in der AWS SDK für .NET API-Referenz.

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateCollection`.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
```

```
        }
    }
```

- API-Details finden Sie CreateCollection in der AWS SDK für .NET API-Referenz.

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCollection`.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
```

```
            };

            var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
            Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
        }
    }
```

- API-Details finden Sie DeleteCollection in der AWS SDK für .NET API-Referenz.

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFaces`.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
```

```
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
                Console.WriteLine($"FaceID: {face}");
            });
        }
    }
```

- API-Details finden Sie DeleteFaces in der AWS SDK für .NET API-Referenz.

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCollection`.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
```

```
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to describe the contents of a
    /// collection.
    /// </summary>
    public class DescribeCollection
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine($"Describing collection: {collectionId}");

            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
{describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- API-Details finden Sie DescribeCollection in der AWS SDK für .NET API-Referenz.

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung`DetectFaces`.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect faces within an image
    /// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DetectFaces
    {
        public static async Task Main()
        {
            string photo = "input.jpg";
            string bucket = "amzn-s3-demo-bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectFacesRequest = new DetectFacesRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },

                // Attributes can be "ALL" or "DEFAULT".
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
```

```
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
{face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
{face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Zeigt Informationen zum Begrenzungsrahmen für alle Gesichter in einem Bild an.

```
    using System;
    using System.Collections.Generic;
    using System.Drawing;
    using System.IO;
    using System.Threading.Tasks;
```

```
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to display the details of the
    /// bounding boxes around the faces detected in an image.
    /// </summary>
    public class ImageOrientationBoundingBox
    {
        public static async Task Main()
        {
            string photo = @"D:\Development\AWS-Examples\Rekognition\target.jpg"; //
"photo.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            int height;
            int width;

            // Used to extract original photo width/height
            using (var imageBitmap = new Bitmap(photo))
            {
                height = imageBitmap.Height;
                width = imageBitmap.Width;
            }

            Console.WriteLine("Image Information:");
            Console.WriteLine(photo);
```

```
            Console.WriteLine("Image Height: " + height);
            Console.WriteLine("Image Width: " + width);

            try
            {
                var detectFacesRequest = new DetectFacesRequest()
                {
                    Image = image,
                    Attributes = new List<string>() { "ALL" },
                };

                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                detectFacesResponse.FaceDetails.ForEach(face =>
                {
                    Console.WriteLine("Face:");
                    ShowBoundingBoxPositions(
                        height,
                        width,
                        face.BoundingBox,
                        detectFacesResponse.OrientationCorrection);

                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"The detected face is estimated to be between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Display the bounding box information for an image.
        /// </summary>
        /// <param name="imageHeight">The height of the image.</param>
        /// <param name="imageWidth">The width of the image.</param>
        /// <param name="box">The bounding box for a face found within the image.</
param>
        /// <param name="rotation">The rotation of the face's bounding box.</param>
```

```
        public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
BoundingBox box, string rotation)
        {
            float left;
            float top;

            if (rotation == null)
            {
                Console.WriteLine("No estimated orientation. Check Exif data.");
                return;
            }

            // Calculate face position based on image orientation.
            switch (rotation)
            {
                case "ROTATE_0":
                    left = imageWidth * box.Left;
                    top = imageHeight * box.Top;
                    break;
                case "ROTATE_90":
                    left = imageHeight * (1 - (box.Top + box.Height));
                    top = imageWidth * box.Left;
                    break;
                case "ROTATE_180":
                    left = imageWidth - (imageWidth * (box.Left + box.Width));
                    top = imageHeight * (1 - (box.Top + box.Height));
                    break;
                case "ROTATE_270":
                    left = imageHeight * box.Top;
                    top = imageWidth * (1 - box.Left - box.Width);
                    break;
                default:
                    Console.WriteLine("No estimated orientation information. Check
Exif data.");
                    return;
            }

            // Display face location information.
            Console.WriteLine($"Left: {left}");
            Console.WriteLine($"Top: {top}");
            Console.WriteLine($"Face Width: {imageWidth * box.Width}");
            Console.WriteLine($"Face Height: {imageHeight * box.Height}");
        }
    }
```

- API-Details finden Sie DetectFaces in der AWS SDK für .NET API-Referenz.

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MaxLabels = 10,
                MinConfidence = 75F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Erkennen Sie Labels in einer Bilddatei, die auf Ihrem Computer gespeichert ist.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect labels within an image
    /// stored locally.
    /// </summary>
    public class DetectLabelsLocalFile
    {
```

```
        public static async Task Main()
        {
            string photo = "input.jpg";

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = image,
                MaxLabels = 10,
                MinConfidence = 77F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine($"Detected labels for {photo}");
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"{label.Name}: {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
```

```
    }
```

- API-Details finden Sie DetectLabels in der AWS SDK für .NET API-Referenz.

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectModerationLabels`.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new DetectModerationLabelsRequest()
        {
            Image = new Image()
```

```
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MinConfidence = 60F,
            };

            try
            {
                var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- API-Details finden Sie DetectModerationLabels in der AWS SDK für .NET API-Referenz.

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung`DetectText`.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "amzn-s3-demo-bucket"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
        };

        try
        {
```

```
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
```

- API-Details finden Sie DetectText in der AWS SDK für .NET API-Referenz.

`GetCelebrityInfo`

Das folgende Codebeispiel zeigt die Verwendung`GetCelebrityInfo`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
```

```
    /// Shows how to use Amazon Rekognition to retrieve information about the
    /// celebrity identified by the supplied celebrity Id.
    /// </summary>
    public class CelebrityInfo
    {
        public static async Task Main()
        {
            string celebId = "nnnnnnnn";

            var rekognitionClient = new AmazonRekognitionClient();

            var celebrityInfoRequest = new GetCelebrityInfoRequest
            {
                Id = celebId,
            };

            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- API-Details finden Sie GetCelebrityInfo in der AWS SDK für .NET API-Referenz.

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung`IndexFaces`.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
public class AddFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Image
        {
            S3Object = new S3Object
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var indexFacesRequest = new IndexFacesRequest
        {
            Image = image,
            CollectionId = collectionId,
```

```
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- API-Details finden Sie IndexFaces in der AWS SDK für .NET API-Referenz.

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung`ListCollections`.

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
```

```
    /// current account.
    /// </summary>
    public class ListCollections
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            Console.WriteLine("Listing collections");
            int limit = 10;

            var listCollectionsRequest = new ListCollectionsRequest
            {
                MaxResults = limit,
            };

            var listCollectionsResponse = new ListCollectionsResponse();

            do
            {
                if (listCollectionsResponse is not null)
                {
                    listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
                }

                listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- API-Details finden Sie ListCollections in der AWS SDK für .NET API-Referenz.

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung`ListFaces`.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
```

```
                listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
                listFacesResponse.Faces.ForEach(face =>
                {
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- API-Details finden Sie ListFaces in der AWS SDK für .NET API-Referenz.

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung`RecognizeCelebrities`.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Shows how to use Amazon Rekognition to identify celebrities in a photo.
    /// </summary>
    public class CelebritiesInImage
    {
```

```
        public static async Task Main(string[] args)
        {
            string photo = "moviestars.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

            var img = new Amazon.Rekognition.Model.Image();
            byte[] data = null;
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load file {photo}");
                return;
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);

            Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
```

```
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count} face(s)
were unrecognized.");
        }
    }
```

- API-Details finden Sie RecognizeCelebrities in der AWS SDK für .NET API-Referenz.

`SearchFaces`

Das folgende Codebeispiel zeigt die Verwendung`SearchFaces`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to find faces in an image that
    /// match the face Id provided in the method request.
    /// </summary>
    public class SearchFacesMatchingId
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
```

```
            var rekognitionClient = new AmazonRekognitionClient();

            // Search collection for faces matching the face id.
            var searchFacesRequest = new SearchFacesRequest
            {
                CollectionId = collectionId,
                FaceId = faceId,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

            Console.WriteLine("Face matching faceId " + faceId);

            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
            });
        }
    }
```

- API-Details finden Sie SearchFaces in der AWS SDK für .NET API-Referenz.

`SearchFacesByImage`

Das folgende Codebeispiel zeigt die Verwendung`SearchFacesByImage`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to search for images matching those
    /// in a collection.
    /// </summary>
    public class SearchFacesMatchingImage
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " + photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
```

```
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- API-Details finden Sie SearchFacesByImage in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für die Domainregistrierung von Route 53 unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Route 53-Domainregistrierung Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen

Erste Schritte

Hallo Route-53-Domainregistrierung

Das folgende Codebeispiel zeigt, wie Sie mit der Route 53-Domainregistrierung beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static class HelloRoute53Domains
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon Route 53 domain registration service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRoute53Domains>()
            ).Build();

        // Now the client is available for injection.
        var route53Client =
 host.Services.GetRequiredService<IAmazonRoute53Domains>();

        // You can use await and any of the async methods to get a response.
        var response = await route53Client.ListPricesAsync(new ListPricesRequest
 { Tld = "com" });
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices
 for .com domain operations:");
        var comPrices = response.Prices.FirstOrDefault();
        if (comPrices != null)
```

```
        {
            Console.WriteLine($"\tRegistration: {comPrices.RegistrationPrice?.Price}
 {comPrices.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {comPrices.RenewalPrice?.Price}
 {comPrices.RenewalPrice?.Currency}");
        }
    }
}
```

- API-Details finden Sie ListPrices in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Auflisten der aktuellen Domains und der Vorgänge des letzten Jahres
- Anzeigen der Abrechnung für das vergangene Jahr und der Preise für Domaintypen
- Abrufen von Domainvorschlägen
- Überprüfen der Verfügbarkeit und Übertragbarkeit von Domains
- Optional: Anfordern einer Domainregistrierung
- Abrufen eines Vorgangsdetails
- Optional: Abrufen eines Domaindetails

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public static class Route53DomainScenario
```

```
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. List current domains.
        2. List operations in the past year.
        3. View billing for the account in the past year.
        4. View prices for domain types.
        5. Get domain suggestions.
        6. Check domain availability.
        7. Check domain transferability.
        8. Optionally, request a domain registration.
        9. Get an operation detail.
       10. Optionally, get a domain detail.
   */

    private static Route53Wrapper _route53Wrapper = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
                    .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonRoute53Domains>()
                .AddTransient<Route53Wrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        var logger = LoggerFactory.Create(builder =>
```

```
        {
            builder.AddConsole();
        }).CreateLogger(typeof(Route53DomainScenario));

        _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Route 53 domains example
scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            await ListDomains();
            await ListOperations();
            await ListBillingRecords();
            await ListPrices();
            await ListDomainSuggestions();
            await CheckDomainAvailability();
            await CheckDomainTransferability();
            var operationId = await RequestDomainRegistration();
            await GetOperationalDetail(operationId);
            await GetDomainDetails();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The Amazon Route 53 domains example scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List account registered domains.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomains()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List account domains.");
        var domains = await _route53Wrapper.ListDomains();
```

```
        for (int i = 0; i < domains.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {domains[i].DomainName}");
        }

        if (!domains.Any())
        {
            Console.WriteLine("\tNo domains found in this account.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListOperations()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List account domain operations in the past year.");
        var operations = await _route53Wrapper.ListOperations(
            DateTime.Today.AddYears(-1));
        for (int i = 0; i < operations.Count; i++)
        {
            Console.WriteLine($"\tOperation Id: {operations[i].OperationId}");
            Console.WriteLine($"\tStatus: {operations[i].Status}");
            Console.WriteLine($"\tDate: {operations[i].SubmittedDate}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListBillingRecords()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. View billing for the account in the past year.");
        var billingRecords = await _route53Wrapper.ViewBilling(
            DateTime.Today.AddYears(-1),
            DateTime.Today);
        for (int i = 0; i < billingRecords.Count; i++)
```

```
        {
            Console.WriteLine($"\tBill Date:
{billingRecords[i].BillDate.ToShortDateString()}");
            Console.WriteLine($"\tOperation: {billingRecords[i].Operation}");
            Console.WriteLine($"\tPrice: {billingRecords[i].Price}");
        }
        if (!billingRecords.Any())
        {
            Console.WriteLine("\tNo billing records found in this account for the
past year.");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List prices for a few domain types.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListPrices()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. View prices for domain types.");
        var domainTypes = new List<string> { "net", "com", "org", "co" };

        var prices = await _route53Wrapper.ListPrices(domainTypes);
        foreach (var pr in prices)
        {
            Console.WriteLine($"\tName: {pr.Name}");
            Console.WriteLine($"\tRegistration: {pr.RegistrationPrice?.Price}
{pr.RegistrationPrice?.Currency}");
            Console.WriteLine($"\tRenewal: {pr.RenewalPrice?.Price}
{pr.RenewalPrice?.Currency}");
            Console.WriteLine($"\tTransfer: {pr.TransferPrice?.Price}
{pr.TransferPrice?.Currency}");
            Console.WriteLine($"\tChange Ownership: {pr.ChangeOwnershipPrice?.Price}
{pr.ChangeOwnershipPrice?.Currency}");
            Console.WriteLine($"\tRestoration: {pr.RestorationPrice?.Price}
{pr.RestorationPrice?.Currency}");
            Console.WriteLine();
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// List domain suggestions for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomainSuggestions()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Get domain suggestions.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to get available domain
suggestions.");
            domainName = Console.ReadLine();
        }

        var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName,
true, 5);
        foreach (var suggestion in suggestions)
        {
            Console.WriteLine($"\tSuggestion Name: {suggestion.DomainName}");
            Console.WriteLine($"\tAvailability: {suggestion.Availability}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check availability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainAvailability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Check domain availability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain availability.");
            domainName = Console.ReadLine();
        }

        var availability = await
_route53Wrapper.CheckDomainAvailability(domainName);
        Console.WriteLine($"\tAvailability: {availability}");
        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckDomainTransferability()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Check domain transferability.");
        string? domainName = null;
        while (domainName == null || string.IsNullOrWhiteSpace(domainName))
        {
            Console.WriteLine($"Enter a domain name to check domain
 transferability.");
            domainName = Console.ReadLine();
        }

        var transferability = await
 _route53Wrapper.CheckDomainTransferability(domainName);
        Console.WriteLine($"\tTransferability: {transferability}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check transferability for a domain name.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> RequestDomainRegistration()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Optionally, request a domain registration.");

        Console.WriteLine($"\tNote: This example uses domain request settings in
 settings.json.");
        Console.WriteLine($"\tTo change the domain registration settings, set the
 values in that file.");
        Console.WriteLine($"\tRemember, registering an actual domain will incur an
 account billing cost.");
        Console.WriteLine($"\tWould you like to begin a domain registration? (y/
n)");
        var ynResponse = Console.ReadLine();
```

```
        if (ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase))
        {
            string domainName = _configuration["DomainName"];
            ContactDetail contact = new ContactDetail();
            contact.CountryCode =
CountryCode.FindValue(_configuration["Contact:CountryCode"]);
            contact.ContactType =
ContactType.FindValue(_configuration["Contact:ContactType"]);

            _configuration.GetSection("Contact").Bind(contact);

            var operationId = await _route53Wrapper.RegisterDomain(
                domainName,
                Convert.ToBoolean(_configuration["AutoRenew"]),
                Convert.ToInt32(_configuration["DurationInYears"]),
                contact);
            if (operationId != null)
            {
                Console.WriteLine(
                    $"\tRegistration requested. Operation Id: {operationId}");
            }

            return operationId;
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }

    /// <summary>
    /// Get details for an operation.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetOperationalDetail(string? operationId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Get an operation detail.");

        var operationDetails =
            await _route53Wrapper.GetOperationDetail(operationId);

        Console.WriteLine(operationDetails);
```

```
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Optionally, get details for a registered domain.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> GetDomainDetails()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Get details on a domain.");

        Console.WriteLine($"\tNote: you must have a registered domain to get
 details.");
        Console.WriteLine($"\tWould you like to get domain details? (y/n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string? domainName = null;
            while (domainName == null)
            {
                Console.WriteLine($"\tEnter a domain name to get details.");
                domainName = Console.ReadLine();
            }

            var domainDetails = await _route53Wrapper.GetDomainDetail(domainName);
            Console.WriteLine(domainDetails);
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }
}
```

Wrapper-Methoden, die vom Szenario für Route-53-Domainregistrierungsaktionen verwendet werden.

```
public class Route53Wrapper
{
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;
```

```
    private readonly ILogger<Route53Wrapper> _logger;
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,
ILogger<Route53Wrapper> logger)
    {
        _amazonRoute53Domains = amazonRoute53Domains;
        _logger = logger;
    }


    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }


    /// <summary>
    /// Check the availability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for availability.</param>
    /// <returns>An availability result string.</returns>
    public async Task<string> CheckDomainAvailability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
            new CheckDomainAvailabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Availability.Value;
    }
```

```
    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
                DomainName = domain
            }
        );
        return result.Transferability.Transferable.Value;
    }


    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</param>
    /// <param name="suggestionCount">The number of suggestions to return. Defaults
 to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
 bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }


    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
```

```
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
 {operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }


    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain registration.</
param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int
 duration, ContactDetail contact)
    {
```

```
        // This example uses the same contact information for admin, registrant, and
tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
{domainName}");
            return null;
        }
    }


    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime
endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
```

```
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }


    /// <summary>
    /// List the domains for the account.
    /// </summary>
    /// <returns>A collection of domain summary records.</returns>
    public async Task<List<DomainSummary>> ListDomains()
    {
        var results = new List<DomainSummary>();
        var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
            new ListDomainsRequest());

        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }


    /// <summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
 submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });
```

```
        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }


    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on {result.CreationDate.ToShortDateString()}.
\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CheckDomainAvailability
  - CheckDomainTransferability

- GetDomainDetail
- GetDomainSuggestions
- GetOperationDetail
- ListDomains
- ListOperations
- ListPrices
- RegisterDomain
- ViewBilling

Aktionen

`CheckDomainAvailability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainAvailability`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Availability.Value;
```

```
    }
```

- API-Details finden Sie CheckDomainAvailability in der AWS SDK für .NET API-Referenz.

`CheckDomainTransferability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainTransferability`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Check the transferability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for transferability.</param>
/// <returns>A transferability result string.</returns>
public async Task<string> CheckDomainTransferability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
        new CheckDomainTransferabilityRequest
        {
            DomainName = domain
        }
    );
    return result.Transferability.Transferable.Value;
}
```

- API-Details finden Sie CheckDomainTransferability in der AWS SDK für .NET API-Referenz.

`GetDomainDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainDetail`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"\tDomain {domainName}:\n" +
                          $"\tCreated on {result.CreationDate.ToShortDateString()}.
\n" +
                          $"\tAdmin contact is {result.AdminContact.Email}.\n" +
                          $"\tAuto-renew is {result.AutoRenew}.\n";

            return details;
        }
        catch (InvalidInputException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
```

- API-Details finden Sie GetDomainDetail in der AWS SDK für .NET API-Referenz.

`GetDomainSuggestions`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainSuggestions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</param>
    /// <param name="suggestionCount">The number of suggestions to return. Defaults
to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }
```

- API-Details finden Sie GetDomainSuggestions in der AWS SDK für .NET API-Referenz.

`GetOperationDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetOperationDetail`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details for a domain action operation.
    /// </summary>
    /// <param name="operationId">The operational Id.</param>
    /// <returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\tOperation {operationId}:\n" +
                          $"\tFor domain {operationDetails.DomainName} on
{operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                          $"\tMessage is {operationDetails.Message}.\n" +
                          $"\tStatus is {operationDetails.Status}.\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }
```

- API-Details finden Sie GetOperationDetail in der AWS SDK für .NET API-Referenz.

`ListDomains`

Das folgende Codebeispiel zeigt die Verwendung`ListDomains`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
    var paginateDomains = _amazonRoute53Domains.Paginators.ListDomains(
        new ListDomainsRequest());

    // Get the entire list using the paginator.
    await foreach (var domain in paginateDomains.Domains)
    {
        results.Add(domain);
    }
    return results;
}
```

- API-Details finden Sie ListDomains in der AWS SDK für .NET API-Referenz.

`ListOperations`

Das folgende Codebeispiel zeigt die Verwendung`ListOperations`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginators.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }
```

- API-Details finden Sie ListOperations in der AWS SDK für .NET API-Referenz.

`ListPrices`

Das folgende Codebeispiel zeigt die Verwendung`ListPrices`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginators.ListPrices(new
ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var prices in paginatePrices.Prices)
        {
            results.Add(prices);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }
```

- API-Details finden Sie ListPrices in der AWS SDK für .NET API-Referenz.

`RegisterDomain`

Das folgende Codebeispiel zeigt die Verwendung`RegisterDomain`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Initiate a domain registration request.
    /// </summary>
    /// <param name="contact">Contact details.</param>
    /// <param name="domainName">The domain name to register.</param>
    /// <param name="autoRenew">True if the domain should automatically renew.</
param>
    /// <param name="duration">The duration in years for the domain registration.</
param>
    /// <returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int
 duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant, and
 tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidInputException)
        {
            _logger.LogInformation($"Unable to request registration for domain
 {domainName}");
            return null;
        }
    }
```

- API-Details finden Sie RegisterDomain in der AWS SDK für .NET API-Referenz.

`ViewBilling`

Das folgende Codebeispiel zeigt die Verwendung`ViewBilling`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// View billing records for the account between a start and end date.
    /// </summary>
    /// <param name="startDate">The start date for billing results.</param>
    /// <param name="endDate">The end date for billing results.</param>
    /// <returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime
endDate)
    {
        var results = new List<BillingRecord>();
        var paginateBilling = _amazonRoute53Domains.Paginators.ViewBilling(
            new ViewBillingRequest()
            {
                Start = startDate,
                End = endDate
            });

        // Get the entire list using the paginator.
        await foreach (var billingRecords in paginateBilling.BillingRecords)
        {
            results.Add(billingRecords);
        }
        return results;
    }
```

- API-Details finden Sie ViewBilling in der AWS SDK für .NET API-Referenz.

Amazon S3-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```
    /// <summary>
```

```
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string destKey,
 string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
```

```
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key {destKey}
in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- API-Details finden Sie CopyObject in der AWS SDK für .NET API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the bucket.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie CreateBucket in der AWS SDK für .NET API-Referenz.

`DeleteBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketCors`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- API-Details finden Sie DeleteBucketCors in der AWS SDK für .NET API-Referenz.

`DeleteBucketLifecycle`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketLifecycle`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client, string
bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- API-Details finden Sie DeleteBucketLifecycle in der AWS SDK für .NET API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt in einem nicht versionierten S3-Bucket.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
```

```
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
 client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from {bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

Löschen Sie ein Objekt in einem versionierten S3-Bucket.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
```

```
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3 client,
string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName, keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
```

```
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- API-Details finden Sie DeleteObject in der AWS SDK für .NET API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte in einem nicht versionierten S3-Bucket.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete multiple objects from an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    /// <summary>
    /// The Main method initializes the Amazon S3 client and the name of
    /// the bucket and then passes those values to MultiObjectDeleteAsync.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";

        // If the Amazon S3 bucket from which you wish to delete objects is not
        // located in the same AWS Region as the default user, define the
        // AWS Region for the Amazon S3 bucket as a parameter to the client
        // constructor.
        IAmazonS3 s3Client = new AmazonS3Client();

        await MultiObjectDeleteAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method uses the passed Amazon S3 client to first create and then
    /// delete three files from the named bucket.
    /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);
```

```
            Console.WriteLine($"Successfully deleted
 {errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call PutObjectAsync.</
param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3 client,
 int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
```

```
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

Löschen Sie mehrere Objekte in einem versionierten S3-Bucket.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
```

```
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>> DeleteObjectsAsync(IAmazonS3
client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker and
```

```
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client, string
bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
```

```
        private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion> keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to delete
the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
```

```
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were deleted.</
param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client, string
 bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
 bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
```

```
        /// Create temporary Amazon S3 objects to show how object deletion wors in
an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- API-Details finden Sie DeleteObjects in der AWS SDK für .NET API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList> GetACLForBucketAsync(IAmazonS3
 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- API-Details finden Sie GetBucketAcl in der AWS SDK für .NET API-Referenz.

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketCors`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- API-Details finden Sie GetBucketCors in der AWS SDK für .NET API-Referenz.

`GetBucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketEncryption`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get and print the encryption settings of a bucket.
    /// </summary>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task GetEncryptionSettings(string bucketName)
    {
        // Check and print the bucket encryption settings.
        Console.WriteLine($"Getting encryption settings for bucket {bucketName}.");

        try
        {
            var settings =
                await _s3Client.GetBucketEncryptionAsync(
                    new GetBucketEncryptionRequest() { BucketName = bucketName });

            foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
            {
                Console.WriteLine(
                    $"\tAlgorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
                Console.WriteLine(
                    $"\tKey:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServiceK
            }
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
```

```
    }
```

- API-Details finden Sie GetBucketEncryption in der AWS SDK für .NET API-Referenz.

`GetBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketLifecycleConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- API-Details finden Sie GetBucketLifecycleConfiguration in der AWS SDK für .NET API-Referenz.

`GetBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketWebsite`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- API-Details finden Sie GetBucketWebsite in der AWS SDK für .NET API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```
    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
```

```
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read successful. Here are the first
20 bytes of the object:\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }
```

- API-Details finden Sie GetObject in der AWS SDK für .NET API-Referenz.

GetObjectLegalHold

Das folgende Codebeispiel zeigt die VerwendungGetObjectLegalHold.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in {bucketName}:
" +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- API-Details finden Sie GetObjectLegalHold in der AWS SDK für .NET API-Referenz.

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLockConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Get the object lock configuration details for an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to get details.</param>
   /// <returns>The bucket's object lock configuration details.</returns>
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await _amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
```

```
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS SDK für .NET API-Referenz.

`GetObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectRetention`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in {bucketName}:
" +
```

```
                                $"\n\t{response.Retention.Mode} until
 {response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- API-Details finden Sie GetObjectRetention in der AWS SDK für .NET API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
```

```
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für .NET API-Referenz.

`ListObjectVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectVersions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
```

```
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3 client,
 string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size: {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
```

```
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- API-Details finden Sie ListObjectVersions in der AWS SDK für .NET API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte mit einem Paginator auf.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "amzn-s3-demo-bucket";

        public static async Task Main()
```

```
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für .NET API-Referenz.

`PutBucketAccelerateConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAccelerateConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which the
    /// method will be enabling acceleration.</param>
```

```
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- API-Details finden Sie PutBucketAccelerateConfiguration in der AWS SDK für .NET API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be created.</
param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
 client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
 System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
```

```
            }

            return false;
        }
```

- API-Details finden Sie PutBucketAcl in der AWS SDK für .NET API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client client,
CORSConfiguration configuration)
        {
            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- API-Details finden Sie PutBucketCors in der AWS SDK für .NET API-Referenz.

`PutBucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketEncryption`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Set the bucket server side encryption to use AWSKMS with a customer-managed
key id.
   /// </summary>
   /// <param name="bucketName">Name of the bucket.</param>
   /// <param name="kmsKeyId">The Id of the KMS Key.</param>
   /// <returns>True if successful.</returns>
   public static async Task<bool> SetBucketServerSideEncryption(string bucketName,
string kmsKeyId)
   {
       var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
       {
           ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
           {
               new ServerSideEncryptionRule
               {
                   ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                   {
                       ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                       ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                   }
               }
           }
       };
       try
       {
```

```
            var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration = serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
        }
        return false;
    }
```

- API-Details finden Sie PutBucketEncryption in der AWS SDK für .NET API-Referenz.

`PutBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketLifecycleConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
```

```
        /// <param name="bucketName">A string representing the S3 bucket to
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- API-Details finden Sie PutBucketLifecycleConfiguration in der AWS SDK für .NET API-Referenz.

`PutBucketLogging`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketLogging`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;
    using Microsoft.Extensions.Configuration;

    /// <summary>
    /// This example shows how to enable logging on an Amazon Simple Storage
```

```
    /// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
    /// this example. The first is the bucket for which you wish to enable
    /// logging, and the second is the location where you want to store the
    /// logs.
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of logs
to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the source
bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" + logPrefix +
@"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"": ""arn:aws:s3:::" +
sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
```

```
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be used
        /// to configure and apply logging to the selected Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
```

```
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load local
settings.
                .Build();
        }
    }
```

- API-Details finden Sie PutBucketLogging in der AWS SDK für .NET API-Referenz.

`PutBucketNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketNotificationConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "amzn-s3-demo-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic, sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
```

```
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType> { EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- API-Details finden Sie PutBucketNotificationConfiguration in der AWS SDK für .NET API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- API-Details finden Sie PutBucketWebsite in der AWS SDK für .NET API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie ein Objekt mit serverseitiger Verschlüsselung hoch.

```
    using System;
    using System.Threading.Tasks;
```

```
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256,
                };
```

```
                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
 response.ServerSideEncryptionMethod;

                Console.WriteLine($"Encryption method used: {0}",
 objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an object");
            }
        }
    }
```

Fügen Sie ein Objekt mithilfe einer bedingten Anforderung ein.

```
    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
    /// <returns>The ETag if the conditional write is successful, empty otherwise.</
returns>
    public async Task<string> PutObjectConditional(string objectKey, string bucket,
 string content)
    {
        try
        {
```

```
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
```

- API-Details finden Sie PutObject in der AWS SDK für .NET API-Referenz.

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLegalHold`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie PutObjectLegalHold in der AWS SDK für .NET API-Referenz.

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLockConfiguration`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
```

```
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in days
or years but not both.
                        }
                    }
                }
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS SDK für .NET API-Referenz.

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectRetention`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie PutObjectRetention in der AWS SDK für .NET API-Referenz.

`RestoreObject`

Das folgende Codebeispiel zeigt die Verwendung`RestoreObject`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
```

```
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };
```

```
            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" : "finished
or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- API-Details finden Sie RestoreObject in der AWS SDK für .NET API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine Amazon-S3-Aktion ausführen kann.

```
    using System;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class GenPresignedUrl
    {
        public static void Main()
        {
            const string bucketName = "amzn-s3-demo-bucket";
            const string objectKey = "sample.txt";
```

```
            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName, objectKey,
 timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
 bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
```

```
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

Generieren Sie eine vorsignierte URL und führen Sie ein Upload mit dieser URL durch.

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
```

```
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL passed
 in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
```

```
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet

dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erste Schritte mit der Verschlüsselung

Das folgende Codebeispiel veranschaulicht die ersten Schritte mit der Verschlüsselung von Amazon-S3-Objekten.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
```

```
            string keyName = "exampleobject.txt";
            string copyTargetKeyName = "examplecopy.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await UploadObjectAsync(client,
bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what you
uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
```

```
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by DownloadObjectAsync.</
returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
 client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
```

```
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                }
                else
```

```
                {
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
```

```
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CopyObject
  - GetObject
  - GetObjectMetadata

Erste Schritte mit Tags

Das folgende Codebeispiel zeigt die ersten Schritte mit Tags für Amazon-S3-Objekte.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
```

```
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
```

```
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
```

```
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- API-Details finden Sie GetObjectTagging in der AWS SDK für .NET API-Referenz.

Sperren von Amazon-S3-Objekten

Das folgende Codebeispiel zeigt, wie Sie mit S3-Object-Lock-Features arbeiten.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
```

```
   Before running this .NET code example, set up your development environment,
including your credentials.

   This .NET example performs the following tasks:
       1. Create test Amazon Simple Storage Service (S3) buckets with different
lock policies.
       2. Upload sample objects to each bucket.
       3. Set some Legal Hold and Retention Periods on objects and buckets.
       4. Investigate lock policies by viewing settings or attempting to delete or
overwrite objects.
       5. Clean up objects and buckets.
  */

   public static S3ActionsWrapper _s3ActionsWrapper = null!;
   public static IConfiguration _configuration = null!;
   private static string _resourcePrefix = null!;
   private static string noLockBucketName = null!;
   private static string lockEnabledBucketName = null!;
   private static string retentionAfterCreationBucketName = null!;
   private static List<string> bucketNames = new List<string>();
   private static List<string> fileNames = new List<string>();

   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonS3>()
                   .AddTransient<S3ActionsWrapper>()
           )
           .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();
```

```
        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
```

```
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create several
 S3\n" +
            "buckets and files to demonstrate working with S3 locking features.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without object
 lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName, false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
 true);
        await
 _s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
 false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();
```

```
        Console.WriteLine("\nA bucket can be configured to use object locking with a
default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName) +
i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
```

```
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on individual
files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for 1
day.
```

```
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                        }
                    }
                }
        }
        Console.WriteLine(new string('-', 80));
        return true;
    }

    // <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
```

```
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the scenario."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the following
choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
```

```
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
```

```
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null, allFiles.Select(f
=> f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files and
buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
```

```
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
                await _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName,
fileInfo.Key, hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
```

```
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
using System.Net;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the bucket.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);
```

```
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
```

```
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in days
or years but not both.
                        }
                    }
                }
```

```
            };

            var response = await _amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in {bucketName}:
" +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
```

```
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
```

```
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in {bucketName}:
" +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await _amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");
```

```
            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to {bucketName}.");
            return false;
        }
    }
```

```
    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
```

```
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
 {bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - GetObjectLegalHold

- GetObjectLockConfiguration
- GetObjectRetention
- PutObjectLegalHold
- PutObjectLockConfiguration
- PutObjectRetention

Senden von bedingten Anforderungen

Das folgende Codebeispiel zeigt, wie Sie Vorbedingungen zu Amazon-S3-Anforderungen hinzufügen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der bedingten Amazon-S3-Anforderung zeigt.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This example demonstrates the use of conditional requests for S3 operations.
```

```
   You can use conditional requests to add preconditions to S3 read requests to
return or copy
   an object based on its Entity tag (ETag), or last modified date.
   You can use a conditional write requests to prevent overwrites by ensuring
   there is no existing object with the same key.
  */

   public static S3ActionsWrapper _s3ActionsWrapper = null!;
   public static IConfiguration _configuration = null!;
   public static string _resourcePrefix = null!;
   public static string _sourceBucketName = null!;
   public static string _destinationBucketName = null!;
   public static string _sampleObjectKey = null!;
   public static string _sampleObjectEtag = null!;
   public static bool _interactive = true;


   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonS3>()
                   .AddTransient<S3ActionsWrapper>()
           )
           .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       ServicesSetup(host);

       try
       {
           Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            ConfigurationSetup();
            _sampleObjectEtag = await Setup(_sourceBucketName,
_destinationBucketName, _sampleObjectKey);

            await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
_sampleObjectKey, _sampleObjectEtag, 0);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await CleanupScenario(_sourceBucketName, _destinationBucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";
```

```
        _sourceBucketName = _resourcePrefix + "-source";
        _destinationBucketName = _resourcePrefix + "-dest";
        _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
 uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
 bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string destBucket,
 string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create several
 S3\n" +
            "buckets and files to demonstrate working with S3 conditional requests.
\n" +
            "This example demonstrates the use of conditional requests for S3
 operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
 requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r\n"
 +
            "You can use a conditional write requests to prevent overwrites by
 ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.\r
\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
 example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
```

```
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
 sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task CleanupScenario(string sourceBucket, string destBucket)
    {
        await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
        await _s3ActionsWrapper.CleanupBucketByName(destBucket);
    }

    /// <summary>
    /// Displays a list of the objects in the test buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task DisplayBuckets(string sourceBucket, string destBucket)
    {
        await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
        await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
    }

    /// <summary>
    /// Displays the menu of conditional request options for the user.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The key of the test object in the source bucket.</
param>
    /// <param name="etag">The ETag of the test object in the source bucket.</param>
    public static async Task DisplayDemoChoices(string sourceBucket, string
 destBucket, string objectKey, string etag, int defaultChoice)
    {
        var actions = new[]
        {
```

```
            "Print a list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit."
        };

        var conditions = new[]
        {
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
        };

        var conditionTypes = new[]
        {
            S3ConditionType.IfMatch,
            S3ConditionType.IfNoneMatch,
            S3ConditionType.IfModifiedSince,
            S3ConditionType.IfUnmodifiedSince,
        };

        var yesterdayDate = DateTime.UtcNow.AddDays(-1);

        int choice;
        while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) !=
4)
        {
            switch (choice)
            {
                case 0:
                    Console.WriteLine("Listing the objects and buckets.");
                    await DisplayBuckets(sourceBucket, destBucket);
                    break;
                case 1:
                    int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                    if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                    {
```

```
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                    }
                    else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                    {
                        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                    }
                    break;
                case 2:
                    int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
                    string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
                    if (copyConditionTypeIndex == 0 || copyConditionTypeIndex == 1)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
                    }
                    else if (copyConditionTypeIndex == 2 || copyConditionTypeIndex
== 3)
                    {
                        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
                    }
                    break;
                case 3:
                    Console.WriteLine("Perform a conditional write using IfNoneMatch
condition on the object key.");
                    Console.WriteLine("If the key is a duplicate, the write will
fail.");
                    string newObjectKey = GetStringResponse("Enter an object key:",
"newObjectKey");
                    await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
                    break;
            }

            if (!_interactive)
            {
                break;
            }
```

```
        }

        Console.WriteLine("Proceeding to cleanup.");
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files and
buckets? (y/n) "))
        {
            await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
            await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you might
incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
```

```
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
 defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not interactive.</
param>
    /// <returns>The string response.</returns>
```

```
    public static string GetStringResponse(string? question, string defaultAnswer)
    {
        string? answer = "";
        if (_interactive)
        {
            do
            {
                Console.WriteLine(question);
                answer = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(answer));
        }
        else
        {
            answer = defaultAnswer;
        }

        return answer;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
```

```
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
 sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
```

```
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read successful. Here are the first
20 bytes of the object:\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional read failed: Precondition failed");
            }
            else if (e.ErrorCode == "NotModified")
            {
                _logger.LogError("Conditional read failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
    /// </summary>
    /// <param name="objectKey">The key of the object to upload.</param>
    /// <param name="bucket">The source bucket of the object.</param>
    /// <param name="content">The content to upload as a string.</param>
```

```
    /// <returns>The ETag if the conditional write is successful, empty otherwise.</
returns>
    public async Task<string> PutObjectConditional(string objectKey, string bucket,
 string content)
    {
        try
        {
            var putObjectRequest = new PutObjectRequest
            {
                BucketName = bucket,
                Key = objectKey,
                ContentBody = content,
                IfNoneMatch = "*"
            };

            var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
            _logger.LogInformation($"Conditional write successful for key
 {objectKey} in bucket {bucket}.");
            return putResult.ETag;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }

    /// <summary>
    /// Copies an object from one Amazon S3 bucket to another with a conditional
 request.
    /// </summary>
    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
```

```
    /// <param name="conditionType">The type of condition to apply, e.g.
 'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
 'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
 dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
 etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string destKey,
 string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null, string?
 etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
 conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
 conditionType, null);
            }
```

```
            await _amazonS3.CopyObjectAsync(copyObjectRequest);
            _logger.LogInformation($"Conditional copy successful for key {destKey}
in bucket {destBucket}.");
            return true;
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional copy failed: Precondition failed");
            }
            else if (e.ErrorCode == "304")
            {
                _logger.LogError("Conditional copy failed: Object not modified");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return false;
        }
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithName(string bucketName)
    {
        Console.WriteLine($"\tCreating bucket {bucketName}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            };

            await _amazonS3.PutBucketAsync(request);
            var bucketReady = false;
```

```
            var retries = 5;
            while (!bucketReady && retries > 0)
            {
                Thread.Sleep(5000);
                bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
                retries--;
            }

            return bucketReady;
        }
        catch (BucketAlreadyExistsException ex)
        {
            Console.WriteLine($"Bucket already exists: '{ex.Message}'");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Cleans up objects and deletes the bucket by name.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task CleanupBucketByName(string bucketName)
    {
        try
        {
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
            }
            await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest { BucketName =
bucketName });
            Console.WriteLine($"Cleaned up bucket: {bucketName}.");
        }
        catch (AmazonS3Exception e)
```

```
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }

    /// <summary>
    /// List the contents of the bucket with their ETag.
    /// </summary>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
    {
        var results = new List<S3Object>();
        try
        {
            Console.WriteLine($"\t Items in bucket {bucketName}");
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            if (listObjectsResponse.S3Objects.Count == 0)
            {
                Console.WriteLine("\t\tNo objects found.");
            }
            else
            {
                foreach (var obj in listObjectsResponse.S3Objects)
                {
                    Console.WriteLine($"\t\t object: {obj.Key} ETag {obj.ETag}");
                }
            }
            results = listObjectsResponse.S3Objects;

        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
```

```
            {
                _logger.LogError($"Bucket {bucketName} does not exist.");
            }
            else
            {
                _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
                throw;
            }
        }

        return results;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket by deleting the objects and then the bucket itself.
```

```
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CleanUpBucketByName(string bucketName)
    {
        try
        {
            var allFiles = await ListBucketContentsByName(bucketName);

            foreach (var fileInfo in allFiles)
            {
                await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
            }

            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CopyObject
  - GetObject
  - PutObject

Verwalten von Zugriffssteuerungslisten (ACL)

Das folgende Codebeispiel zeigt, wie Sie eine neue Zugriffssteuerungsliste (ACL) für Amazon-S3-Buckets verwalten.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to manage Amazon Simple Storage Service
    /// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
    /// access.
    /// </summary>
    public class ManageACLs
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket1";
            string newBucketName = "amzn-s3-demo-bucket2";
            string keyName = "sample-object.txt";
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3 bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }
```

```
        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }
```

```
        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to get
 the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
```

```
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);
```

```
            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/s3/
LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl

- PutObjectAcl

Erstellen einer mehrteiligen Kopie

Das folgende Codebeispiel zeigt, wie Sie eine mehrteilige Kopie eines Amazon-S3-Objekts erstellen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
```

```
        /// This method uses the passed client object to perform a multipart
        /// copy operation.
        /// </summary>
        /// <param name="client">An Amazon S3 client object that will be used
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
```

```
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

Transformieren von Daten mit S3 Object Lambda

Das folgende Codebeispiel zeigt, wie Sie Daten für Ihre Anwendung mit S3 Object Lambda transformieren können.

SDK für .NET

Zeigt, wie Sie benutzerdefinierten Code zu standardmäßigen S3-GET-Anforderungen hinzufügen, um das angeforderte Objekt, das von S3 abgerufen wurde, so zu ändern, dass das Objekt den Anforderungen des anfordernden Clients oder der anfordernden Anwendung entspricht.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Amazon S3

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Funktionen auf, die mithilfe von Amazon S3 Dateien zu und von einem S3-Bucket übertragen TransferUtility.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";
```

```
DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil, bucketName,
 fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil, bucketName,
 keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
```

```
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil, bucketName,
 s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();
```

```
// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to an
\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth - textToCenter.Length) /
 2));
```

```
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

Laden Sie eine einzelne Datei hoch.

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</param>
        /// <returns>A boolean value indicating the success of the action.</returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from {localPath}
because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
```

```
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
```

```
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
            }
        }
```

Laden Sie eine einzelne Datei herunter.

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
```

```
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

Laden Sie den Inhalt eines S3 Buckets herunter.

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to download.</
param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
```

```
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

Verfolgen Sie den Fortschritt eines Uploads mit dem TransferUtility.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Transfer;

    /// <summary>
    /// This example shows how to track the progress of a multipart upload
    /// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
    /// upload to an Amazon S3 bucket.
    /// </summary>
    public class TrackMPUUsingHighLevelAPI
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string keyName = "sample_pic.png";
            string path = "filepath/directory/";
```

```
            string filePath = $"{path}{keyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            await TrackMPUAsync(client, bucketName, filePath, keyName);
        }

        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
```

```
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

Laden Sie ein Objekt mit Verschlüsselung hoch.

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
```

```
    {
        public static async Task Main()
        {
            string existingBucketName = "amzn-s3-demo-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();

            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
```

```
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
```

```
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");

                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von .NET

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
```

```
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request - {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

SageMaker KI-Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von SageMaker KI Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo SageMaker KI

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von SageMaker KI beginnen können.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
```

```
namespace SageMakerActions;

public static class HelloSageMaker
{
    static async Task Main(string[] args)
    {
        var sageMakerClient = new AmazonSageMakerClient();

        Console.WriteLine($"Hello Amazon SageMaker! Let's list some of your notebook
 instances:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five notebook instances.
        var response = await sageMakerClient.ListNotebookInstancesAsync(
            new ListNotebookInstancesRequest()
            {
                MaxResults = 5
            });

        if (!response.NotebookInstances.Any())
        {
            Console.WriteLine($"No notebook instances found.");
            Console.WriteLine("See https://docs.aws.amazon.com/sagemaker/latest/dg/
howitworks-create-ws.html to create one.");
        }

        foreach (var notebookInstance in response.NotebookInstances)
        {
            Console.WriteLine($"\tInstance:
 {notebookInstance.NotebookInstanceName}");
            Console.WriteLine($"\tArn: {notebookInstance.NotebookInstanceArn}");
            Console.WriteLine($"\tCreation Date:
 {notebookInstance.CreationTime.ToShortDateString()}");
            Console.WriteLine();
        }
    }
}
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK für .NET API-Referenz.

Aktionen

`CreatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`CreatePipeline`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
```

```
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }
```

- API-Details finden Sie CreatePipeline in der AWS SDK für .NET API-Referenz.

`DeletePipeline`

Das folgende Codebeispiel zeigt die Verwendung`DeletePipeline`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a SageMaker pipeline by name.
    /// </summary>
    /// <param name="pipelineName">The name of the pipeline to delete.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public async Task<string> DeletePipelineByName(string pipelineName)
    {
        var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
            new DeletePipelineRequest()
            {
                PipelineName = pipelineName
            });

        return deleteResponse.PipelineArn;
    }
```

- API-Details finden Sie DeletePipeline in der AWS SDK für .NET API-Referenz.

`DescribePipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribePipelineExecution`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
    public async Task<PipelineExecutionStatus> CheckPipelineExecutionStatus(string
pipelineExecutionArn)
    {
        var describeResponse = await
_amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }
```

- API-Details finden Sie DescribePipelineExecution in der AWS SDK für .NET API-Referenz.

`StartPipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartPipelineExecution`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple Storage
Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
```

```
                S3Uri = outputLocationUrl
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value =
 JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value =
 JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value =
 JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }
```

- API-Details finden Sie StartPipelineExecution in der AWS SDK für .NET API-Referenz.

`UpdatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`UpdatePipeline`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
```

```
                });

            return createResponse.PipelineArn;
        }
    }
```

- API-Details finden Sie UpdatePipeline in der AWS SDK für .NET API-Referenz.

Szenarien

Erste Schritte mit Geodatenaufträgen und Pipelines

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter Erstellen und Ausführen von SageMaker Pipelines mithilfe von AWS SDKs. Community.aws

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die SageMaker KI-Operationen umschließt.

```
using System.Text.Json;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
```

```
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;

namespace SageMakerActions;

/// <summary>
/// Wrapper class for Amazon SageMaker actions and logic.
/// </summary>
public class SageMakerWrapper
{
    private readonly IAmazonSageMaker _amazonSageMaker;
    public SageMakerWrapper(IAmazonSageMaker amazonSageMaker)
    {
        _amazonSageMaker = amazonSageMaker;
    }

    /// <summary>
    /// Create a pipeline from a JSON definition, or update it if the pipeline
 already exists.
    /// </summary>
    /// <returns>The Amazon Resource Name (ARN) of the pipeline.</returns>
    public async Task<string> SetupPipeline(string pipelineJson, string roleArn,
 string name, string description, string displayName)
    {
        try
        {
            var updateResponse = await _amazonSageMaker.UpdatePipelineAsync(
                new UpdatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });
            return updateResponse.PipelineArn;
        }
        catch (Amazon.SageMaker.Model.ResourceNotFoundException)
        {
            var createResponse = await _amazonSageMaker.CreatePipelineAsync(
                new CreatePipelineRequest()
                {
                    PipelineDefinition = pipelineJson,
                    PipelineDescription = description,
```

```
                    PipelineDisplayName = displayName,
                    PipelineName = name,
                    RoleArn = roleArn
                });

            return createResponse.PipelineArn;
        }
    }

    /// <summary>
    /// Run a pipeline with input and output file locations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue to use for pipeline
callbacks.</param>
    /// <param name="inputLocationUrl">The input location in Amazon Simple Storage
Service (Amazon S3).</param>
    /// <param name="outputLocationUrl">The output location in Amazon S3.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="executionRoleArn">The ARN of the role.</param>
    /// <returns>The ARN of the pipeline run.</returns>
    public async Task<string> ExecutePipeline(
        string queueUrl,
        string inputLocationUrl,
        string outputLocationUrl,
        string pipelineName,
        string executionRoleArn)
    {
        var inputConfig = new VectorEnrichmentJobInputConfig()
        {
            DataSourceConfig = new()
            {
                S3Data = new VectorEnrichmentJobS3Data()
                {
                    S3Uri = inputLocationUrl
                }
            },
            DocumentType = VectorEnrichmentJobDocumentType.CSV
        };

        var exportConfig = new ExportVectorEnrichmentJobOutputConfig()
        {
            S3Data = new VectorEnrichmentJobS3Data()
            {
                S3Uri = outputLocationUrl
```

```
            }
        };

        var jobConfig = new VectorEnrichmentJobConfig()
        {
            ReverseGeocodingConfig = new ReverseGeocodingConfig()
            {
                XAttributeName = "Longitude",
                YAttributeName = "Latitude"
            }
        };

#pragma warning disable SageMaker1002 // Property value does not match required
 pattern is allowed here to match the pipeline definition.
        var startExecutionResponse = await
 _amazonSageMaker.StartPipelineExecutionAsync(
            new StartPipelineExecutionRequest()
            {
                PipelineName = pipelineName,
                PipelineExecutionDisplayName = pipelineName + "-example-execution",
                PipelineParameters = new List<Parameter>()
                {
                    new Parameter() { Name = "parameter_execution_role", Value =
 executionRoleArn },
                    new Parameter() { Name = "parameter_queue_url", Value =
 queueUrl },
                    new Parameter() { Name = "parameter_vej_input_config", Value =
 JsonSerializer.Serialize(inputConfig) },
                    new Parameter() { Name = "parameter_vej_export_config", Value =
 JsonSerializer.Serialize(exportConfig) },
                    new Parameter() { Name = "parameter_step_1_vej_config", Value =
 JsonSerializer.Serialize(jobConfig) }
                }
            });
#pragma warning restore SageMaker1002
        return startExecutionResponse.PipelineExecutionArn;
    }

    /// <summary>
    /// Check the status of a run.
    /// </summary>
    /// <param name="pipelineExecutionArn">The ARN.</param>
    /// <returns>The status of the pipeline.</returns>
```

```
    public async Task<PipelineExecutionStatus> CheckPipelineExecutionStatus(string
 pipelineExecutionArn)
    {
        var describeResponse = await
 _amazonSageMaker.DescribePipelineExecutionAsync(
            new DescribePipelineExecutionRequest()
            {
                PipelineExecutionArn = pipelineExecutionArn
            });

        return describeResponse.PipelineExecutionStatus;
    }

    /// <summary>
    /// Delete a SageMaker pipeline by name.
    /// </summary>
    /// <param name="pipelineName">The name of the pipeline to delete.</param>
    /// <returns>The ARN of the pipeline.</returns>
    public async Task<string> DeletePipelineByName(string pipelineName)
    {
        var deleteResponse = await _amazonSageMaker.DeletePipelineAsync(
            new DeletePipelineRequest()
            {
                PipelineName = pipelineName
            });

        return deleteResponse.PipelineArn;
    }
}
```

Erstellen Sie eine Funktion, die Rückrufe aus der SageMaker AI-Pipeline verarbeitet.

```
using System.Text.Json;
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;
using Amazon.SageMaker;
using Amazon.SageMaker.Model;
using Amazon.SageMakerGeospatial;
using Amazon.SageMakerGeospatial.Model;
```

```
// Assembly attribute to enable the AWS Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SageMakerLambda;

/// <summary>
/// The AWS Lambda function handler for the Amazon SageMaker pipeline.
/// </summary>
public class SageMakerLambdaFunction
{
    /// <summary>
    /// Default constructor. This constructor is used by AWS Lambda to construct the
 instance. When invoked in a Lambda environment
    /// the AWS credentials will come from the AWS Identity and Access Management
 (IAM) role associated with the function. The AWS Region will be set to the
    /// Region that the Lambda function is running in.
    /// </summary>
    public SageMakerLambdaFunction()
    {
    }

    /// <summary>
    /// The AWS Lambda function handler that processes events from the SageMaker
 pipeline and starts a job or export.
    /// </summary>
    /// <param name="request">The custom SageMaker pipeline request object.</param>
    /// <param name="context">The Lambda context.</param>
    /// <returns>The dictionary of output parameters.</returns>
    public async Task<Dictionary<string, string>> FunctionHandler(PipelineRequest
 request, ILambdaContext context)
    {
        var geoSpatialClient = new AmazonSageMakerGeospatialClient();
        var sageMakerClient = new AmazonSageMakerClient();
        var responseDictionary = new Dictionary<string, string>();
        context.Logger.LogInformation("Function handler started with request: " +
 JsonSerializer.Serialize(request));
        if (request.Records != null && request.Records.Any())
        {
            context.Logger.LogInformation("Records found, this is a queue event.
 Processing the queue records.");
            foreach (var message in request.Records)
            {
```

```
                await ProcessMessageAsync(message, context, geoSpatialClient,
sageMakerClient);
            }
        }
        else if (!string.IsNullOrEmpty(request.vej_export_config))
        {
            context.Logger.LogInformation("Export configuration found, this is an
export. Start the Vector Enrichment Job (VEJ) export.");

            var outputConfig =
                JsonSerializer.Deserialize<ExportVectorEnrichmentJobOutputConfig>(
                    request.vej_export_config);

            var exportResponse = await
geoSpatialClient.ExportVectorEnrichmentJobAsync(
                new ExportVectorEnrichmentJobRequest()
                {
                    Arn = request.vej_arn,
                    ExecutionRoleArn = request.Role,
                    OutputConfig = outputConfig
                });
            context.Logger.LogInformation($"Export response:
{JsonSerializer.Serialize(exportResponse)}");
            responseDictionary = new Dictionary<string, string>
            {
                { "export_eoj_status", exportResponse.ExportStatus.ToString() },
                { "vej_arn", exportResponse.Arn }
            };
        }
        else if (!string.IsNullOrEmpty(request.vej_name))
        {
            context.Logger.LogInformation("Vector Enrichment Job name found,
starting the job.");
            var inputConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobInputConfig>(
                    request.vej_input_config);

            var jobConfig =
                JsonSerializer.Deserialize<VectorEnrichmentJobConfig>(
                    request.vej_config);

            var jobResponse = await geoSpatialClient.StartVectorEnrichmentJobAsync(
                new StartVectorEnrichmentJobRequest()
                {
```

```
                    ExecutionRoleArn = request.Role,
                    InputConfig = inputConfig,
                    Name = request.vej_name,
                    JobConfig = jobConfig

                });
            context.Logger.LogInformation("Job response: " +
JsonSerializer.Serialize(jobResponse));
            responseDictionary = new Dictionary<string, string>
            {
                { "vej_arn", jobResponse.Arn },
                { "statusCode", jobResponse.HttpStatusCode.ToString() }
            };
        }
        return responseDictionary;
    }

    /// <summary>
    /// Process a queue message and check the status of a SageMaker job.
    /// </summary>
    /// <param name="message">The queue message.</param>
    /// <param name="context">The Lambda context.</param>
    /// <param name="geoClient">The SageMaker GeoSpatial client.</param>
    /// <param name="sageMakerClient">The SageMaker client.</param>
    /// <returns>Async task.</returns>
    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
ILambdaContext context,
        AmazonSageMakerGeospatialClient geoClient, AmazonSageMakerClient
sageMakerClient)
    {
        context.Logger.LogInformation($"Processed message {message.Body}");

        // Get information about the SageMaker job.
        var payload = JsonSerializer.Deserialize<QueuePayload>(message.Body);
        context.Logger.LogInformation($"Payload token {payload!.token}");
        var token = payload.token;

        if (payload.arguments.ContainsKey("vej_arn"))
        {
            // Use the job ARN and the token to get the job status.
            var job_arn = payload.arguments["vej_arn"];
            context.Logger.LogInformation($"Token: {token}, arn {job_arn}");

            var jobInfo = geoClient.GetVectorEnrichmentJobAsync(
```

```
                new GetVectorEnrichmentJobRequest()
                {
                    Arn = job_arn
                });
            context.Logger.LogInformation("Job info: " +
 JsonSerializer.Serialize(jobInfo));
            if (jobInfo.Result.Status == VectorEnrichmentJobStatus.COMPLETED)
            {
                context.Logger.LogInformation($"Status completed, resuming
 pipeline...");
                await sageMakerClient.SendPipelineExecutionStepSuccessAsync(
                    new SendPipelineExecutionStepSuccessRequest()
                    {
                        CallbackToken = token,
                        OutputParameters = new List<OutputParameter>()
                        {
                            new OutputParameter()
                                { Name = "export_status", Value =
 jobInfo.Result.Status }
                        }
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.FAILED)
            {
                context.Logger.LogInformation($"Status failed, stopping
 pipeline...");
                await sageMakerClient.SendPipelineExecutionStepFailureAsync(
                    new SendPipelineExecutionStepFailureRequest()
                    {
                        CallbackToken = token,
                        FailureReason = jobInfo.Result.ErrorDetails.ErrorMessage
                    });
            }
            else if (jobInfo.Result.Status == VectorEnrichmentJobStatus.IN_PROGRESS)
            {
                // Put this message back in the queue to reprocess later.
                context.Logger.LogInformation(
                    $"Status still in progress, check back later.");
                throw new("Job still running.");
            }
        }
    }
}
```

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public static class PipelineWorkflow
{
    public static IAmazonIdentityManagementService _iamClient = null!;
    public static SageMakerWrapper _sageMakerWrapper = null!;
    public static IAmazonSQS _sqsClient = null!;
    public static IAmazonS3 _s3Client = null!;
    public static IAmazonLambda _lambdaClient = null!;
    public static IConfiguration _configuration = null!;

    public static string lambdaFunctionName = "SageMakerExampleFunction";
    public static string sageMakerRoleName = "SageMakerExampleRole";
    public static string lambdaRoleName = "SageMakerExampleLambdaRole";

    private static string[] lambdaRolePolicies = null!;
    private static string[] sageMakerRolePolicies = null!;

    static async Task Main(string[] args)
    {
        var options = new AWSOptions() { Region = RegionEndpoint.USWest2 };
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>(options)
                    .AddAWSService<IAmazonEC2>(options)
                    .AddAWSService<IAmazonSageMaker>(options)
                    .AddAWSService<IAmazonSageMakerGeospatial>(options)
                    .AddAWSService<IAmazonSQS>(options)
                    .AddAWSService<IAmazonS3>(options)
                    .AddAWSService<IAmazonLambda>(options)
                    .AddTransient<SageMakerWrapper>()
        )
        .Build();

        _configuration = new ConfigurationBuilder()
```

```
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ServicesSetup(host);
        string queueUrl = "";
        string queueName = _configuration["queueName"];
        string bucketName = _configuration["bucketName"];
        var pipelineName = _configuration["pipelineName"];

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "Welcome to the Amazon SageMaker pipeline example scenario.");
            Console.WriteLine(
                "\nThis example scenario will guide you through setting up and
running an" +
                "\nAmazon SageMaker pipeline. The pipeline uses an AWS Lambda
function and an" +
                "\nAmazon SQS Queue. It runs a vector enrichment reverse geocode job
to" +
                "\nreverse geocode addresses in an input file and store the results
in an export file.");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine(
                "First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.");
            Console.WriteLine(new string('-', 80));

            var lambdaRoleArn = await CreateLambdaRole();
            var sageMakerRoleArn = await CreateSageMakerRole();
            var functionArn = await SetupLambda(lambdaRoleArn, true);
            queueUrl = await SetupQueue(queueName);
            await SetupBucket(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Now we can create and run our pipeline.");
            Console.WriteLine(new string('-', 80));
```

```
            await SetupPipeline(sageMakerRoleArn, functionArn, pipelineName);
            var executionArn = await ExecutePipeline(queueUrl, sageMakerRoleArn,
 pipelineName, bucketName);
            await WaitForPipelineExecution(executionArn);

            await GetOutputResults(bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("The pipeline has completed. To view the pipeline and
 runs " +
                              "in SageMaker Studio, follow these instructions:" +
                              "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/
pipelines-studio.html");
            Console.WriteLine(new string('-', 80));

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await CleanupResources(true, queueUrl, pipelineName, bucketName);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("SageMaker pipeline scenario is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
 {ex.Message}");
            await CleanupResources(true, queueUrl, pipelineName, bucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sageMakerWrapper = host.Services.GetRequiredService<SageMakerWrapper>();
        _iamClient =
 host.Services.GetRequiredService<IAmazonIdentityManagementService>();
```

```
        _sqsClient = host.Services.GetRequiredService<IAmazonSQS>();
        _s3Client = host.Services.GetRequiredService<IAmazonS3>();
        _lambdaClient = host.Services.GetRequiredService<IAmazonLambda>();
    }

    /// <summary>
    /// Set up AWS Lambda, either by updating an existing function or creating a new
function.
    /// </summary>
    /// <param name="roleArn">The role Amazon Resource Name (ARN) to use for the
Lambda function.</param>
    /// <param name="askUser">True to ask the user before updating.</param>
    /// <returns>The ARN of the function.</returns>
    public static async Task<string> SetupLambda(string roleArn, bool askUser)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Setting up the Lambda function for the pipeline.");
        var handlerName =
"SageMakerLambda::SageMakerLambda.SageMakerLambdaFunction::FunctionHandler";
        var functionArn = "";
        try
        {
            var functionInfo = await _lambdaClient.GetFunctionAsync(new
GetFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });

            var updateFunction = true;
            if (askUser)
            {
                updateFunction = GetYesNoResponse(
                    $"\tThe Lambda function {lambdaFunctionName} already exists, do
you want to update it?");
            }

            if (updateFunction)
            {
                // Update the Lambda function.
                using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
                await _lambdaClient.UpdateFunctionCodeAsync(
                    new UpdateFunctionCodeRequest()
                    {
```

```
                        FunctionName = lambdaFunctionName,
                        ZipFile = zipMemoryStream,
                    });
            }

            functionArn = functionInfo.Configuration.FunctionArn;
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"\tThe Lambda function {lambdaFunctionName} was not
found, creating the new function.");

            // Create the function if it does not already exist.
            using var zipMemoryStream = new MemoryStream(await
File.ReadAllBytesAsync("SageMakerLambda.zip"));
            var createResult = await _lambdaClient.CreateFunctionAsync(
                new CreateFunctionRequest()
                {
                    FunctionName = lambdaFunctionName,
                    Runtime = Runtime.Dotnet6,
                    Description = "SageMaker example function.",
                    Code = new FunctionCode()
                    {
                        ZipFile = zipMemoryStream
                    },
                    Handler = handlerName,
                    Role = roleArn,
                    Timeout = 30
                });

            functionArn = createResult.FunctionArn;
        }

        Console.WriteLine($"\tLambda ready with ARN {functionArn}.");
        Console.WriteLine(new string('-', 80));
        return functionArn;
    }

    /// <summary>
    /// Create a role to be used by AWS Lambda. Does not create the role if it
already exists.
    /// </summary>
    /// <returns>The role ARN.</returns>
    public static async Task<string> CreateLambdaRole()
```

```
    {
        Console.WriteLine(new string('-', 80));

        lambdaRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSQSFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess",
            "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy",
            "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
        };

        var roleArn = await GetRoleArnIfExists(lambdaRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to for AWS Lambda to use.");

        var assumeRolePolicy = "{" +
                               "\"Version\": \"2012-10-17\"," +
                               "\"Statement\": [{" +
                                    "\"Effect\": \"Allow\"," +
                                    "\"Principal\": {" +
                                        $"\"Service\": [" +
                                            "\"sagemaker.amazonaws.com\"," +
                                            "\"sagemaker-geospatial.amazonaws.com
\"," +
                                            "\"lambda.amazonaws.com\"," +
                                            "\"s3.amazonaws.com\"" +
                                        "]" +
                                    "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                               "}]" +
                           "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
```

```
                RoleName = lambdaRoleName
            });
        foreach (var policy in lambdaRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = lambdaRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));

        return roleResult.Role.Arn;
    }


    /// <summary>
    /// Create a role to be used by SageMaker.
    /// </summary>
    /// <returns>The role Amazon Resource Name (ARN).</returns>
    public static async Task<string> CreateSageMakerRole()
    {
        Console.WriteLine(new string('-', 80));

        sageMakerRolePolicies = new string[]{
            "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess",
            "arn:aws:iam::aws:policy/AmazonSageMakerGeospatialFullAccess",
        };

        var roleArn = await GetRoleArnIfExists(sageMakerRoleName);
        if (!string.IsNullOrEmpty(roleArn))
        {
            return roleArn;
        }

        Console.WriteLine("\tCreating a role to use with SageMaker.");

        var assumeRolePolicy = "{" +
                                        "\"Version\": \"2012-10-17\"," +
```

```
                                        "\"Statement\": [{" +
                                            "\"Effect\": \"Allow\"," +
                                            "\"Principal\": {" +
                                                $"\"Service\": [" +
                                                    "\"sagemaker.amazonaws.com\"," +
                                                    "\"sagemaker-
geospatial.amazonaws.com\"," +
                                                    "\"lambda.amazonaws.com\"," +
                                                    "\"s3.amazonaws.com\"" +
                                                "]" +
                                            "}," +
                                            "\"Action\": \"sts:AssumeRole\"" +
                                        "}]" +
                                    "}";

        var roleResult = await _iamClient!.CreateRoleAsync(
            new CreateRoleRequest()
            {
                AssumeRolePolicyDocument = assumeRolePolicy,
                Path = "/",
                RoleName = sageMakerRoleName
            });

        foreach (var policy in sageMakerRolePolicies)
        {
            await _iamClient.AttachRolePolicyAsync(
                new AttachRolePolicyRequest()
                {
                    PolicyArn = policy,
                    RoleName = sageMakerRoleName
                });
        }

        // Allow time for the role to be ready.
        Thread.Sleep(10000);
        Console.WriteLine($"\tRole ready with ARN {roleResult.Role.Arn}.");
        Console.WriteLine(new string('-', 80));
        return roleResult.Role.Arn;
    }

    /// <summary>
    /// Set up the SQS queue to use with the pipeline.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
```

```
    /// <returns>The URL for the queue.</returns>
    public static async Task<string> SetupQueue(string queueName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up queue {queueName}.");

        try
        {
            var queueInfo = await _sqsClient.GetQueueUrlAsync(new
GetQueueUrlRequest()
            { QueueName = queueName });
            return queueInfo.QueueUrl;
        }
        catch (QueueDoesNotExistException)
        {
            var attrs = new Dictionary<string, string>
            {
                {
                    QueueAttributeName.DelaySeconds,
                    "5"
                },
                {
                    QueueAttributeName.ReceiveMessageWaitTimeSeconds,
                    "5"
                },
                {
                    QueueAttributeName.VisibilityTimeout,
                    "300"
                },
            };

            var request = new CreateQueueRequest
            {
                Attributes = attrs,
                QueueName = queueName,
            };

            var response = await _sqsClient.CreateQueueAsync(request);
            Thread.Sleep(10000);
            await ConnectLambda(response.QueueUrl);
            Console.WriteLine($"\tQueue ready with Url {response.QueueUrl}.");
            Console.WriteLine(new string('-', 80));
            return response.QueueUrl;
        }
```

```
    }

    /// <summary>
    /// Connect the queue to the Lambda function as an event source.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue.</param>
    /// <returns>Async task.</returns>
    public static async Task ConnectLambda(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Connecting the Lambda function and queue for the
pipeline.");

        var queueAttributes = await _sqsClient.GetQueueAttributesAsync(
            new GetQueueAttributesRequest() { QueueUrl = queueUrl, AttributeNames =
new List<string>() { "All" } });
        var queueArn = queueAttributes.QueueARN;

        var eventSource = await _lambdaClient.ListEventSourceMappingsAsync(
             new ListEventSourceMappingsRequest()
             {
                 FunctionName = lambdaFunctionName
             });

        if (!eventSource.EventSourceMappings.Any())
        {
            // Only add the event source mapping if it does not already exist.
            await _lambdaClient.CreateEventSourceMappingAsync(
                new CreateEventSourceMappingRequest()
                {
                    EventSourceArn = queueArn,
                    FunctionName = lambdaFunctionName,
                    Enabled = true
                });
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the bucket to use for pipeline input and output.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
```

```
    public static async Task SetupBucket(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up bucket {bucketName}.");

        var bucketExists = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_s3Client,
            bucketName);

        if (!bucketExists)
        {
            await _s3Client.PutBucketAsync(new PutBucketRequest()
            {
                BucketName = bucketName,
                BucketRegion = S3Region.USWest2
            });

            Thread.Sleep(5000);

            await _s3Client.PutObjectAsync(new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = "samplefiles/latlongtest.csv",
                FilePath = "latlongtest.csv"
            });
        }

        Console.WriteLine($"\tBucket {bucketName} ready.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Display some results from the output directory.
    /// </summary>
    /// <param name="bucketName">The name for the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<string> GetOutputResults(string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Getting output results {bucketName}.");
        string outputKey = "";
        Thread.Sleep(15000);
        var outputFiles = await _s3Client.ListObjectsAsync(
              new ListObjectsRequest()
```

```
                {
                    BucketName = bucketName,
                    Prefix = "outputfiles/"
                });

        if (outputFiles.S3Objects.Any())
        {
            var sampleOutput = outputFiles.S3Objects.OrderBy(s =>
 s.LastModified).Last();
            Console.WriteLine($"\tOutput file: {sampleOutput.Key}");
            var outputSampleResponse = await _s3Client.GetObjectAsync(
                new GetObjectRequest()
                {
                    BucketName = bucketName,
                    Key = sampleOutput.Key
                });
            outputKey = sampleOutput.Key;
            StreamReader reader = new
 StreamReader(outputSampleResponse.ResponseStream);
            await reader.ReadLineAsync();
            Console.WriteLine("\tOutput file contents: \n");
            for (int i = 0; i < 10; i++)
            {
                if (!reader.EndOfStream)
                {
                    Console.WriteLine("\t" + await reader.ReadLineAsync());
                }
            }
        }

        Console.WriteLine(new string('-', 80));
        return outputKey;
    }

    /// <summary>
    /// Create a pipeline from the example pipeline JSON
    /// that includes the Lambda, callback, processing, and export jobs.
    /// </summary>
    /// <param name="roleArn">The ARN of the role for the pipeline.</param>
    /// <param name="functionArn">The ARN of the Lambda function for the pipeline.</
param>
    /// <param name="pipelineName">The name for the pipeline.</param>
    /// <returns>The ARN of the pipeline.</returns>
```

```
    public static async Task<string> SetupPipeline(string roleArn, string
functionArn, string pipelineName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Setting up the pipeline.");

        var pipelineJson = await File.ReadAllTextAsync("GeoSpatialPipeline.json");

        // Add the correct function ARN instead of the placeholder.
        pipelineJson = pipelineJson.Replace("*FUNCTION_ARN*", functionArn);

        var pipelineArn = await _sageMakerWrapper.SetupPipeline(pipelineJson,
roleArn, pipelineName,
            "sdk example pipeline", pipelineName);

        Console.WriteLine($"\tPipeline set up with ARN {pipelineArn}.");
        Console.WriteLine(new string('-', 80));

        return pipelineArn;
    }

    /// <summary>
    /// Start a pipeline run with job configurations.
    /// </summary>
    /// <param name="queueUrl">The URL for the queue used in the pipeline.</param>
    /// <param name="roleArn">The ARN of the role.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>The pipeline run ARN.</returns>
    public static async Task<string> ExecutePipeline(
        string queueUrl,
        string roleArn,
        string pipelineName,
        string bucketName)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Starting pipeline execution.");

        var input = $"s3://{bucketName}/samplefiles/latlongtest.csv";
        var output = $"s3://{bucketName}/outputfiles/";

        var executionARN =
            await _sageMakerWrapper.ExecutePipeline(queueUrl, input, output,
                pipelineName, roleArn);
```

```
        Console.WriteLine($"\tRun started with ARN {executionARN}.");
        Console.WriteLine(new string('-', 80));

        return executionARN;
    }

    /// <summary>
    /// Wait for a pipeline run to complete.
    /// </summary>
    /// <param name="executionArn">The pipeline run ARN.</param>
    /// <returns>Async task.</returns>
    public static async Task WaitForPipelineExecution(string executionArn)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Waiting for pipeline to finish.");

        PipelineExecutionStatus status;
        do
        {
            status = await
_sageMakerWrapper.CheckPipelineExecutionStatus(executionArn);
            Thread.Sleep(30000);
            Console.WriteLine($"\tStatus is {status}.");
        } while (status == PipelineExecutionStatus.Executing);

        Console.WriteLine($"\tPipeline finished with status {status}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="askUser">True to ask the user for cleanup.</param>
    /// <param name="queueUrl">The URL of the queue to clean up.</param>
    /// <param name="pipelineName">The name of the pipeline.</param>
    /// <param name="bucketName">The name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> CleanupResources(
        bool askUser,
        string queueUrl,
        string pipelineName,
        string bucketName)
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (!askUser || GetYesNoResponse($"\tDelete pipeline {pipelineName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting pipeline.");
            // Delete the pipeline.
            await _sageMakerWrapper.DeletePipelineByName(pipelineName);
        }

        if (!string.IsNullOrEmpty(queueUrl) && (!askUser ||
 GetYesNoResponse($"\tDelete queue {queueUrl}? (y/n)")))
        {
            Console.WriteLine($"\tDeleting queue.");
            // Delete the queue.
            await _sqsClient.DeleteQueueAsync(new DeleteQueueRequest(queueUrl));
        }

        if (!askUser || GetYesNoResponse($"\tDelete Amazon S3 bucket {bucketName}?
 (y/n)"))
        {
            Console.WriteLine($"\tDeleting bucket.");
            // Delete all objects in the bucket.
            var deleteList = await _s3Client.ListObjectsV2Async(new
 ListObjectsV2Request()
            {
                BucketName = bucketName
            });
            if (deleteList.KeyCount > 0)
            {
                await _s3Client.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.S3Objects
                        .Select(o => new KeyVersion { Key = o.Key }).ToList()
                });
            }

            // Now delete the bucket.
            await _s3Client.DeleteBucketAsync(new DeleteBucketRequest()
            {
                BucketName = bucketName
            });
```

```
        }

        if (!askUser || GetYesNoResponse($"\tDelete lambda {lambdaFunctionName}? (y/
n)"))
        {
            Console.WriteLine($"\tDeleting lambda function.");

            await _lambdaClient.DeleteFunctionAsync(new DeleteFunctionRequest()
            {
                FunctionName = lambdaFunctionName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {lambdaRoleName}? (y/n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in lambdaRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = lambdaRoleName,
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = lambdaRoleName
            });
        }

        if (!askUser || GetYesNoResponse($"\tDelete role {sageMakerRoleName}? (y/
n)"))
        {
            Console.WriteLine($"\tDetaching policies and deleting role.");

            foreach (var policy in sageMakerRolePolicies)
            {
                await _iamClient!.DetachRolePolicyAsync(new
 DetachRolePolicyRequest()
                {
                    RoleName = sageMakerRoleName,
```

```
                    PolicyArn = policy
                });
            }

            await _iamClient!.DeleteRoleAsync(new DeleteRoleRequest()
            {
                RoleName = sageMakerRoleName
            });
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a role's ARN if it already exists.
    /// </summary>
    /// <param name="roleName">The name of the AWS Identity and Access Management
(IAM) Role to look for.</param>
    /// <returns>The role ARN if it exists, otherwise an empty string.</returns>
    private static async Task<string> GetRoleArnIfExists(string roleName)
    {
        Console.WriteLine($"Checking for role named {roleName}.");

        try
        {
            var existingRole = await _iamClient.GetRoleAsync(new GetRoleRequest()
            {
                RoleName = lambdaRoleName
            });
            return existingRole.Role.Arn;
        }
        catch (NoSuchEntityException)
        {
            return string.Empty;
        }
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
```

```
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Secrets Manager-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Secrets Manager Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.SecretsManager;
using Amazon.SecretsManager.Model;

/// <summary>
/// This example uses the Amazon Web Service Secrets Manager to retrieve
/// the secret value for the provided secret name.
/// </summary>
public class GetSecretValue
{
    /// <summary>
    /// The main method initializes the necessary values and then calls
    /// the GetSecretAsync and DecodeString methods to get the decoded
    /// secret value for the secret named in secretName.
    /// </summary>
    public static async Task Main()
    {
        string secretName = "<<{{MySecretName}}>>";
        string secret;

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
            secret = DecodeString(response);

            if (!string.IsNullOrEmpty(secret))
            {
                Console.WriteLine($"The decoded secret value is: {secret}.");
            }
```

```
                else
                {
                    Console.WriteLine("No secret value was returned.");
                }
            }
        }

        /// <summary>
        /// Retrieves the secret value given the name of the secret to
        /// retrieve.
        /// </summary>
        /// <param name="client">The client object used to retrieve the secret
        /// value for the given secret name.</param>
        /// <param name="secretName">The name of the secret value to retrieve.</
param>
        /// <returns>The GetSecretValueReponse object returned by
        /// GetSecretValueAsync.</returns>
        public static async Task<GetSecretValueResponse> GetSecretAsync(
            IAmazonSecretsManager client,
            string secretName)
        {
            GetSecretValueRequest request = new GetSecretValueRequest()
            {
                SecretId = secretName,
                VersionStage = "AWSCURRENT", // VersionStage defaults to AWSCURRENT
 if unspecified.
            };

            GetSecretValueResponse response = null;

            // For the sake of simplicity, this example handles only the most
            // general SecretsManager exception.
            try
            {
                response = await client.GetSecretValueAsync(request);
            }
            catch (AmazonSecretsManagerException e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }

            return response;
        }
```

```
        /// <summary>
        /// Decodes the secret returned by the call to GetSecretValueAsync and
        /// returns it to the calling program.
        /// </summary>
        /// <param name="response">A GetSecretValueResponse object containing
        /// the requested secret value returned by GetSecretValueAsync.</param>
        /// <returns>A string representing the decoded secret value.</returns>
        public static string DecodeString(GetSecretValueResponse response)
        {
            // Decrypts secret using the associated AWS Key Management Service
            // Customer Master Key (CMK.) Depending on whether the secret is a
            // string or binary value, one of these fields will be populated.
            if (response.SecretString is not null)
            {
                var secret = response.SecretString;
                return secret;
            }
            else if (response.SecretBinary is not null)
            {
                var memoryStream = response.SecretBinary;
                StreamReader reader = new StreamReader(memoryStream);
                string decodedBinarySecret =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
                return decodedBinarySecret;
            }
            else
            {
                return string.Empty;
            }
        }
    }
```

- API-Details finden Sie GetSecretValue in der AWS SDK für .NET API-Referenz.

Amazon SES-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an email template.
    /// </summary>
    /// <param name="name">Name of the template.</param>
    /// <param name="subject">Email subject.</param>
    /// <param name="text">Email body text.</param>
    /// <param name="html">Email HTML body text.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
 string text,
```

```
        string html)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.CreateTemplateAsync(
                new CreateTemplateRequest
                {
                    Template = new Template
                    {
                        TemplateName = name,
                        SubjectPart = subject,
                        TextPart = text,
                        HtmlPart = html
                    }
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("CreateEmailTemplateAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- API-Details finden Sie CreateTemplate in der AWS SDK für .NET API-Referenz.

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an email identity.
    /// </summary>
    /// <param name="identityEmail">The identity email to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteIdentityAsync(string identityEmail)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
                new DeleteIdentityRequest
                {
                    Identity = identityEmail
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- API-Details finden Sie DeleteIdentity in der AWS SDK für .NET API-Referenz.

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an email template.
    /// </summary>
    /// <param name="templateName">Name of the template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
                new DeleteTemplateRequest
                {
                    TemplateName = templateName
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteEmailTemplateAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

- API-Details finden Sie DeleteTemplate in der AWS SDK für .NET API-Referenz.

`GetIdentityVerificationAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetIdentityVerificationAttributes`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get identity verification status for an email.
    /// </summary>
    /// <returns>The verification status of the email.</returns>
    public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
    {
        var result = VerificationStatus.TemporaryFailure;
        try
        {
            var response =
                await
_amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
                    new GetIdentityVerificationAttributesRequest
                    {
                        Identities = new List<string> { email }
                    });

            if (response.VerificationAttributes.ContainsKey(email))
                result = response.VerificationAttributes[email].VerificationStatus;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- API-Details finden Sie GetIdentityVerificationAttributes in der AWS SDK für .NET API-Referenz.

`GetSendQuota`

Das folgende Codebeispiel zeigt die Verwendung`GetSendQuota`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information on the current account's send quota.
    /// </summary>
    /// <returns>The send quota response data.</returns>
    public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
    {
        var result = new GetSendQuotaResponse();
        try
        {
            var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
                new GetSendQuotaRequest());
            result = response;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetSendQuotaAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- API-Details finden Sie GetSendQuota in der AWS SDK für .NET API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the identities of a specified type for the current account.
    /// </summary>
    /// <param name="identityType">IdentityType to list.</param>
    /// <returns>The list of identities.</returns>
    public async Task<List<string>> ListIdentitiesAsync(IdentityType identityType)
    {
        var result = new List<string>();
        try
        {
            var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
                new ListIdentitiesRequest
                {
                    IdentityType = identityType
                });
            result = response.Identities;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListIdentitiesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- API-Details finden Sie ListIdentities in der AWS SDK für .NET API-Referenz.

`ListTemplates`

Das folgende Codebeispiel zeigt die Verwendung`ListTemplates`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List email templates for the current account.
    /// </summary>
    /// <returns>A list of template metadata.</returns>
    public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
    {
        var result = new List<TemplateMetadata>();
        try
        {
            var response = await _amazonSimpleEmailService.ListTemplatesAsync(
                new ListTemplatesRequest());
            result = response.TemplatesMetadata;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

- API-Details finden Sie ListTemplates in der AWS SDK für .NET API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
///  Send an email by using Amazon SES.
/// </summary>
/// <param name="toAddresses">List of recipients.</param>
/// <param name="ccAddresses">List of cc recipients.</param>
/// <param name="bccAddresses">List of bcc recipients.</param>
/// <param name="bodyHtml">Body of the email in HTML.</param>
/// <param name="bodyText">Body of the email in plain text.</param>
/// <param name="subject">Subject line of the email.</param>
/// <param name="senderAddress">From address.</param>
/// <returns>The messageId of the email.</returns>
public async Task<string> SendEmailAsync(List<string> toAddresses,
    List<string> ccAddresses, List<string> bccAddresses,
    string bodyHtml, string bodyText, string subject, string senderAddress)
{
    var messageId = "";
    try
    {
        var response = await _amazonSimpleEmailService.SendEmailAsync(
            new SendEmailRequest
            {
                Destination = new Destination
                {
                    BccAddresses = bccAddresses,
                    CcAddresses = ccAddresses,
                    ToAddresses = toAddresses
                },
                Message = new Message
                {
                    Body = new Body
                    {
                        Html = new Content
                        {
```

```
                                Charset = "UTF-8",
                                Data = bodyHtml
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyText
                            }
                        },
                        Subject = new Content
                        {
                            Charset = "UTF-8",
                            Data = subject
                        }
                    },
                    Source = senderAddress
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendEmailAsync failed with exception: " +
ex.Message);
        }

        return messageId;
    }
```

- API-Details finden Sie SendEmail in der AWS SDK für .NET API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Send an email using a template.
    /// </summary>
    /// <param name="sender">Address of the sender.</param>
    /// <param name="recipients">Addresses of the recipients.</param>
    /// <param name="templateName">Name of the email template.</param>
    /// <param name="templateDataObject">Data for the email template.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
        string templateName, object templateDataObject)
    {
        var messageId = "";
        try
        {
            // Template data should be serialized JSON from either a class or a
dynamic object.
            var templateData = JsonSerializer.Serialize(templateDataObject);

            var response = await _amazonSimpleEmailService.SendTemplatedEmailAsync(
                new SendTemplatedEmailRequest
                {
                    Source = sender,
                    Destination = new Destination
                    {
                        ToAddresses = recipients
                    },
                    Template = templateName,
                    TemplateData = templateData
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
ex.Message);
        }

        return messageId;
    }
```

- API-Details finden Sie SendTemplatedEmail in der AWS SDK für .NET API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Starts verification of an email identity. This request sends an email
    /// from Amazon SES to the specified email address. To complete
    /// verification, follow the instructions in the email.
    /// </summary>
    /// <param name="recipientEmailAddress">Email address to verify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyEmailIdentityAsync(string recipientEmailAddress)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.VerifyEmailIdentityAsync(
                new VerifyEmailIdentityRequest
                {
                    EmailAddress = recipientEmailAddress
                });

            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("VerifyEmailIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
```

```
    }
```

- API-Details finden Sie VerifyEmailIdentity in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie man die Amazon-DynamoDB-.NET-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für .NET

Zeigt, wie Sie die verwenden AWS SDK für .NET , um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful.NET-Backend erstellt wurde.

- Integrieren Sie eine React-Webanwendung in Dienste. AWS
- Auflisten, Hinzufügen, Aktualisieren und Löschen von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für Amazon SES API v2 unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon SES-API v2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateContact`

Das folgende Codebeispiel zeigt die Verwendung`CreateContact`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Creates a contact and adds it to the specified contact list.
/// </summary>
/// <param name="emailAddress">The email address of the contact.</param>
/// <param name="contactListName">The name of the contact list.</param>
```

```
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }
```

- API-Details finden Sie CreateContact in der AWS SDK für .NET API-Referenz.

`CreateContactList`

Das folgende Codebeispiel zeigt die Verwendung`CreateContactList`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact list:
{ex.Message}");
        }
        return false;
    }
```

- API-Details finden Sie CreateContactList in der AWS SDK für .NET API-Referenz.

`CreateEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailIdentity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse> CreateEmailIdentityAsync(string
emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };
```

```
        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email identity:
{ex.Message}");
            throw;
        }
```

```
    }
```

- API-Details finden Sie CreateEmailIdentity in der AWS SDK für .NET API-Referenz.

`CreateEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
 subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
```

```
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email template:
{ex.Message}");
        }

        return false;
    }
```

- API-Details finden Sie CreateEmailTemplate in der AWS SDK für .NET API-Referenz.

`DeleteContactList`

Das folgende Codebeispiel zeigt die Verwendung`DeleteContactList`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being modified
 by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact list:
 {ex.Message}");
        }

        return false;
```

```
    }
```

- API-Details finden Sie DeleteContactList in der AWS SDK für .NET API-Referenz.

`DeleteEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailIdentity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
```

```
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email identity:
{ex.Message}");
        }

        return false;
    }
```

- API-Details finden Sie DeleteEmailIdentity in der AWS SDK für .NET API-Referenz.

`DeleteEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailTemplate`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
```

```
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email template:
{ex.Message}");
        }

        return false;
    }
```

- API-Details finden Sie DeleteEmailTemplate in der AWS SDK für .NET API-Referenz.

`ListContacts`

Das folgende Codebeispiel zeigt die Verwendung`ListContacts`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
 {ex.Message}");
        }

        return new List<Contact>();
    }
```

- API-Details finden Sie ListContacts in der AWS SDK für .NET API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email from.</
param>
    /// <param name="toEmailAddresses">The email addresses to send the email to.</
param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
 (optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
 template (optional).</param>
    /// <param name="contactListName">The name of the contact list for unsubscribe
 functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
    public async Task<string> SendEmailAsync(string fromEmailAddress, List<string>
 toEmailAddresses, string? subject,
        string? htmlContent, string? textContent, string? templateName = null,
 string? templateData = null, string? contactListName = null)
    {
        var request = new SendEmailRequest
        {
            FromEmailAddress = fromEmailAddress
        };

        if (toEmailAddresses.Any())
        {
            request.Destination = new Destination { ToAddresses =
 toEmailAddresses };
```

```
        }

        if (!string.IsNullOrEmpty(templateName))
        {
            request.Content = new EmailContent()
            {
                Template = new Template
                {
                    TemplateName = templateName,
                    TemplateData = templateData
                }
            };
        }
        else
        {
            request.Content = new EmailContent
            {
                Simple = new Message
                {
                    Subject = new Content { Data = subject },
                    Body = new Body
                    {
                        Html = new Content { Data = htmlContent },
                        Text = new Content { Data = textContent }
                    }
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
```

```
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
        }

        return string.Empty;
    }
```

- API-Details finden Sie SendEmail in der AWS SDK für .NET API-Referenz.

Szenarien

Newsletter-Szenario

Das folgende Codebeispiel zeigt, wie Sie das Newsletter-Szenario von Amazon SES API v2 ausführen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace Sesv2Scenario;

public static class NewsletterWorkflow
{
    /*
      This scenario demonstrates how to use the Amazon Simple Email Service (SES) v2
 to send a coupon newsletter to a list of subscribers.
      The scenario performs the following tasks:

      1. Prepare the application:
         - Create a verified email identity for sending and replying to emails.
         - Create a contact list to store the subscribers' email addresses.
         - Create an email template for the coupon newsletter.

      2. Gather subscriber email addresses:
         - Prompt the user for a base email address.
```

```
        - Create 3 variants of the email address using subaddress extensions (e.g.,
user+ses-weekly-newsletter-1@example.com).
        - Add each variant as a contact to the contact list.
        - Send a welcome email to each new contact.

     3. Send the coupon newsletter:
        - Retrieve the list of contacts from the contact list.
        - Send the coupon newsletter using the email template to each contact.

     4. Monitor and review:
        - Provide instructions for the user to review the sending activity and
metrics in the AWS console.

     5. Clean up resources:
        - Delete the contact list (which also deletes all contacts within it).
        - Delete the email template.
        - Optionally delete the verified email identity.

   */

   public static SESv2Wrapper _sesv2Wrapper;
   public static string? _baseEmailAddress = null;
   public static string? _verifiedEmail = null;
   private static string _contactListName = "weekly-coupons-newsletter";
   private static string _templateName = "weekly-coupons";
   private static string _subject = "Weekly Coupons Newsletter";
   private static string _htmlContentFile = "coupon-newsletter.html";
   private static string _textContentFile = "coupon-newsletter.txt";
   private static string _htmlWelcomeFile = "welcome.html";
   private static string _textWelcomeFile = "welcome.txt";
   private static string _couponsDataFile = "sample_coupons.json";

   // Relative location of the resources folder.
   private static string _resourcesFilePathLocation = "../../../../resources/";

   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSimpleEmailServiceV2>()
                    .AddTransient<SESv2Wrapper>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon SES v2 Coupon Newsletter
Scenario.");
            Console.WriteLine("This scenario demonstrates how to use the Amazon
Simple Email Service (SES) v2 " +
                              "\r\nto send a coupon newsletter to a list of
subscribers.");

            // Prepare the application.
            var emailIdentity = await PrepareApplication();

            // Gather subscriber email addresses.
            await GatherSubscriberEmailAddresses(emailIdentity);

            // Send the coupon newsletter.
            await SendCouponNewsletter(emailIdentity);

            // Monitor and review.
            MonitorAndReview(true);

            // Clean up resources.
            await Cleanup(emailIdentity, true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon SES v2 Coupon Newsletter scenario is
complete.");
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
```

```
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sesv2Wrapper = host.Services.GetRequiredService<SESv2Wrapper>();
    }

    /// <summary>
    /// Set up the resources for the scenario.
    /// </summary>
    /// <returns>The email address of the verified identity.</returns>
    public static async Task<string?> PrepareApplication()
    {
        var htmlContent = await File.ReadAllTextAsync(_resourcesFilePathLocation +
_htmlContentFile);
        var textContent = await File.ReadAllTextAsync(_resourcesFilePathLocation +
_textContentFile);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("1. In this step, we will prepare the application:" +
                          "\r\n  - Create a verified email identity for sending and
replying to emails." +
                          "\r\n  - Create a contact list to store the subscribers'
email addresses." +
                          "\r\n  - Create an email template for the coupon
newsletter.\r\n");

        // Prompt the user for a verified email address.
        while (!IsEmail(_verifiedEmail))
        {
            Console.Write("Enter a verified email address or an email to verify: ");
            _verifiedEmail = Console.ReadLine();
        }

        try
        {
            // Create an email identity and start the verification process.
            await _sesv2Wrapper.CreateEmailIdentityAsync(_verifiedEmail);
            Console.WriteLine($"Identity {_verifiedEmail} created.");
        }
```

```
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Identity {_verifiedEmail} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email identity: {ex.Message}");
        }

        // Create a contact list.
        try
        {
            await _sesv2Wrapper.CreateContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Contact list {_contactListName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating contact list: {ex.Message}");
        }

        // Create an email template.
        try
        {
            await _sesv2Wrapper.CreateEmailTemplateAsync(_templateName, _subject,
htmlContent, textContent);
            Console.WriteLine($"Email template {_templateName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Email template {_templateName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email template: {ex.Message}");
        }

        return _verifiedEmail;
    }

    /// <summary>
```

```
    /// Generate subscriber addresses and send welcome emails.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
 PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> GatherSubscriberEmailAddresses(string
 fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("2. In Step 2, we will gather subscriber email addresses:"
 +
                          "\r\n  - Prompt the user for a base email address." +
                          "\r\n  - Create 3 variants of the email address using
 subaddress extensions (e.g., user+ses-weekly-newsletter-1@example.com)." +
                          "\r\n  - Add each variant as a contact to the contact
 list." +
                          "\r\n  - Send a welcome email to each new contact.\r\n");

        // Prompt the user for a base email address.
        while (!IsEmail(_baseEmailAddress))
        {
            Console.Write("Enter a base email address (e.g., user@example.com): ");
            _baseEmailAddress = Console.ReadLine();
        }

        // Create 3 variants of the email address using +ses-weekly-newsletter-1,
 +ses-weekly-newsletter-2, etc.
        var baseEmailAddressParts = _baseEmailAddress!.Split("@");
        for (int i = 1; i <= 3; i++)
        {
            string emailAddress = $"{baseEmailAddressParts[0]}+ses-weekly-
newsletter-{i}@{baseEmailAddressParts[1]}";

            try
            {
                // Create a contact with the email address in the contact list.
                await _sesv2Wrapper.CreateContactAsync(emailAddress,
 _contactListName);
                Console.WriteLine($"Contact {emailAddress} added to the
 {_contactListName} contact list.");
            }
            catch (AlreadyExistsException)
            {
```

```
                Console.WriteLine($"Contact {emailAddress} already exists in the
{_contactListName} contact list.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error creating contact {emailAddress}:
{ex.Message}");
                return false;
            }

            // Send a welcome email to the new contact.
            try
            {
                string subject = "Welcome to the Weekly Coupons Newsletter";
                string htmlContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _htmlWelcomeFile);
                string textContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _textWelcomeFile);

                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress, new
List<string> { emailAddress }, subject, htmlContent, textContent);
                Console.WriteLine($"Welcome email sent to {emailAddress}.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending welcome email to {emailAddress}:
{ex.Message}");
                return false;
            }

            // Wait 2 seconds before sending the next email (if the account is in
the SES Sandbox).
            await Task.Delay(2000);
        }

        return true;
    }

    /// <summary>
    ///  Send the coupon newsletter to the subscribers in the contact list.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
```

```
    public static async Task<bool> SendCouponNewsletter(string fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("3. In this step, we will send the coupon newsletter:" +
                          "\r\n  - Retrieve the list of contacts from the contact
list." +
                          "\r\n  - Send the coupon newsletter using the email
template to each contact.\r\n");


        // Retrieve the list of contacts from the contact list.
        var contacts = await _sesv2Wrapper.ListContactsAsync(_contactListName);
        if (!contacts.Any())
        {
            Console.WriteLine($"No contacts found in the {_contactListName} contact
list.");
            return false;
        }

        // Load the coupon data from the sample_coupons.json file.
        string couponsData = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _couponsDataFile);

        // Send the coupon newsletter to each contact using the email template.
        try
        {
            foreach (var contact in contacts)
            {
                // To use the Contact List for list management, send to only one
address at a time.
                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress,
                    new List<string> { contact.EmailAddress },
                    null, null, null, _templateName, couponsData, _contactListName);
            }

            Console.WriteLine($"Coupon newsletter sent to contact list
{_contactListName}.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending coupon newsletter to contact list
{_contactListName}: {ex.Message}");
            return false;
        }
```

```
        return true;
    }

    /// <summary>
    /// Provide instructions for monitoring sending activity and metrics.
    /// </summary>
    /// <param name="interactive">True to run in interactive mode.</param>
    /// <returns>True if successful.</returns>
    public static bool MonitorAndReview(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("4. In step 4, we will monitor and review:" +
                          "\r\n  - Provide instructions for the user to review the
sending activity and metrics in the AWS console.\r\n");

        Console.WriteLine("Review your sending activity using the SES Homepage in
the AWS console.");
        Console.WriteLine("Press Enter to open the SES Homepage in your default
browser...");
        if (interactive)
        {
            Console.ReadLine();
            try
            {
                // Open the SES Homepage in the default browser.
                Process.Start(new ProcessStartInfo
                {
                    FileName = "https://console.aws.amazon.com/ses/home",
                    UseShellExecute = true
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error opening the SES Homepage: {ex.Message}");
                return false;
            }
        }

        Console.WriteLine("Review the sending activity and email metrics, then press
Enter to continue...");
        if (interactive)
            Console.ReadLine();
        return true;
```

```
    }

    /// <summary>
    /// Clean up the resources used in the scenario.
    /// </summary>
    /// <param name="verifiedEmailAddress">The verified email address from
 PrepareApplication.</param>
    /// <param name="interactive">True if interactive.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Cleanup(string verifiedEmailAddress, bool
 interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("5. Finally, we clean up resources:" +
                          "\r\n  - Delete the contact list (which also deletes all
 contacts within it)." +
                          "\r\n  - Delete the email template." +
                          "\r\n  - Optionally delete the verified email identity.\r
\n");

        Console.WriteLine("Cleaning up resources...");

        // Delete the contact list (this also deletes all contacts in the list).
        try
        {
            await _sesv2Wrapper.DeleteContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Contact list {_contactListName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting contact list {_contactListName}:
 {ex.Message}");
            return false;
        }

        // Delete the email template.
        try
        {
            await _sesv2Wrapper.DeleteEmailTemplateAsync(_templateName);
            Console.WriteLine($"Email template {_templateName} deleted.");
```

```
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Email template {_templateName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting email template {_templateName}:
{ex.Message}");
            return false;
        }

        // Ask the user if they want to delete the email identity.
        var deleteIdentity = !interactive ||
            GetYesNoResponse(
                $"Do you want to delete the email identity {verifiedEmailAddress}?
(y/n) ");
        if (deleteIdentity)
        {
            try
            {
                await _sesv2Wrapper.DeleteEmailIdentityAsync(verifiedEmailAddress);
                Console.WriteLine($"Email identity {verifiedEmailAddress}
deleted.");
            }
            catch (NotFoundException)
            {
                Console.WriteLine(
                    $"Email identity {verifiedEmailAddress} not found.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"Error deleting email identity {verifiedEmailAddress}:
{ex.Message}");
                return false;
            }
        }
        else
        {
            Console.WriteLine(
                $"Skipping deletion of email identity {verifiedEmailAddress}.");
        }
```

```
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Simple check to verify a string is an email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email.</returns>
    private static bool IsEmail(string? email)
    {
        if (string.IsNullOrEmpty(email))
            return false;
        return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$",
 RegexOptions.IgnoreCase);
    }
}
```

Wrapper für Service-Vorgänge.

```
using System.Net;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;

namespace Sesv2Scenario;

/// <summary>
/// Wrapper class for Amazon Simple Email Service (SES) v2 operations.
/// </summary>
```

```
public class SESv2Wrapper
{

    private readonly IAmazonSimpleEmailServiceV2 _sesClient;

    /// <summary>
    /// Constructor for the SESv2Wrapper.
    /// </summary>
    /// <param name="sesClient">The injected SES v2 client.</param>
    public SESv2Wrapper(IAmazonSimpleEmailServiceV2 sesClient)
    {
        _sesClient = sesClient;
    }

    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
 contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
 exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
```

```
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
```

```
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact list:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse> CreateEmailIdentityAsync(string
emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
```

```
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email identity:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
```

```
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email template:
{ex.Message}");
        }

        return false;
    }

    /// <summary>
```

```
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being modified
 by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact list:
 {ex.Message}");
        }

        return false;
    }
```

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being modified
by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email identity:
{ex.Message}");
        }

        return false;
    }
```

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email template:
 {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
```

```
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
 {ex.Message}");
        }

        return new List<Contact>();
    }

    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email from.</
param>
    /// <param name="toEmailAddresses">The email addresses to send the email to.</
param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
 (optional).</param>
```

```
   /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
   /// <param name="contactListName">The name of the contact list for unsubscribe
functionality (optional).</param>
   /// <returns>The MessageId response from the SendEmail operation.</returns>
   public async Task<string> SendEmailAsync(string fromEmailAddress, List<string>
toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
```

```
                }
            };
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
 {ex.Message}");
        }

        return string.Empty;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Amazon SNS-Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SNS beginnen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
```

```
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK für .NET API-Referenz.

Aktionen

`CheckIfPhoneNumberIsOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`CheckIfPhoneNumberIsOptedOut`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" : "not
opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- API-Details finden Sie CheckIfPhoneNumberIsOptedOut in der AWS SDK für .NET API-Referenz.

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema mit einem bestimmten Namen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
```

```
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Erstellen Sie ein neues Thema mit einem Namen und spezifischen FIFO- und Deduplizierungsattributen.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
```

```
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- API-Details finden Sie CreateTopic in der AWS SDK für .NET API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Thema mit seinem Themen-ARN.

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
```

```
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteTopic in der AWS SDK für .NET API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-west-2:000000000000:ExampleSNSTopic";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK für .NET API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example will retrieve a list of the existing Amazon Simple
    /// Notification Service (Amazon SNS) subscriptions.
    /// </summary>
    public class ListSubscriptions
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            Console.WriteLine("Enter a topic ARN to list subscriptions for a
specific topic, " +
                              "or press Enter to list subscriptions for all
topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client, topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
```

```
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults to
 null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string topicArn
 = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
 client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
 paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }
```

```
        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- API-Details finden Sie ListSubscriptions in der AWS SDK für .NET API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
```

```
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// Lists the Amazon Simple Notification Service (Amazon SNS)
    /// topics for the current account.
    /// </summary>
    public class ListSNSTopics
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await GetTopicListAsync(client);
        }

        /// <summary>
        /// Retrieves the list of Amazon SNS topics in groups of up to 100
        /// topics.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the list of topics.</param>
        public static async Task GetTopicListAsync(IAmazonSimpleNotificationService
client)
        {
            // If there are more than 100 Amazon SNS topics, the call to
            // ListTopicsAsync will return a value to pass to the
            // method to retrieve the next 100 (or less) topics.
            string nextToken = string.Empty;

            do
            {
                var response = await client.ListTopicsAsync(nextToken);
                DisplayTopicsList(response.Topics);
                nextToken = response.NextToken;
            }
            while (!string.IsNullOrEmpty(nextToken));
        }

        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
```

```
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- API-Details finden Sie ListTopics in der AWS SDK für .NET API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen einer Nachricht für ein Thema.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
ExampleSNSTopic.";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
```

```
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);
```

```
                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }
```

Wenden Sie die Benutzerauswahl auf die Veröffentlichungsaktion an.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
```

```
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für .NET -API-Referenz.

`Abonnieren`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
        /// <summary>
        /// Creates a new subscription to a topic.
```

```
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
```

```
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für .NET -API-Referenz.

`Abonnement beenden`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem Abonnement-ARN von einem Thema ab.

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
```

```
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Weitere API-Informationen finden Sie unter Abmelden in der AWS SDK für .NET -API-Referenz.

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
```

```
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für .NET -API-Referenz.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
```

```
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();
```

```
            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for configuring
the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the results
in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic, deduplication
 is supported." +
                              $"\r\nDeduplication IDs are either set in the message
 or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an SNS
 FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication interval,
 is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/dg/
fifo-message-dedup.html.");
```

```
            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName, _useFifoTopic,
_useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS queue:
", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);
```

```
                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
```

```
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
```

```
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" : "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }
```

```
                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl, 10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
```

```
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the queue
at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message> messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
```

```
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer = true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
```

```
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;
```

```
    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
```

```
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
```

```
                                        "\"Principal\": {" +
                                            $"\"Service\": " +
                                                "\"sns.amazonaws.com\"" +
                                               "}," +
                                        "\"Action\": \"sqs:SendMessage\"," +
                                        $"\"Resource\": \"{queueArn}\"," +
                                         "\"Condition\": {" +
                                              "\"ArnEquals\": {" +
                                                   $"\"aws:SourceArn\": \"{topicArn}\""
 +
                                               "}" +
                                           "}" +
                                    "}]" +
                                "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
```

```
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
}
```

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
```

```
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }
```

```
        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
```

```
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateQueue

- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;
```

```
// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record, ILambdaContext
 context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Amazon SQS-Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.SQS;
using Amazon.SQS.Model;

namespace SQSActions;

public static class HelloSQS
{
    static async Task Main(string[] args)
    {
```

```
        var sqsClient = new AmazonSQSClient();

        Console.WriteLine($"Hello Amazon SQS! Following are some of your queues:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get the first five queues.
        var response = await sqsClient.ListQueuesAsync(
            new ListQueuesRequest()
            {
                MaxResults = 5
            });

        foreach (var queue in response.QueueUrls)
        {
            Console.WriteLine($"\tQueue Url: {queue}");
            Console.WriteLine();
        }
    }
}
```

- API-Details finden Sie ListQueues in der AWS SDK für .NET API-Referenz.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Warteschlage mit einem bestimmten Namen erstellen

```
    /// <summary>
    /// Create a queue with a specific name.
```

```
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }
```

Erstellen Sie eine Amazon-SQS-Warteschlange und senden Sie eine Nachricht an diese.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon;
   using Amazon.SQS;
   using Amazon.SQS.Model;

   public class CreateSendExample
   {
       // Specify your AWS Region (an example Region is shown).
       private static readonly string QueueName = "Example_Queue";
       private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
       private static IAmazonSQS client;

       public static async Task Main()
       {
           client = new AmazonSQSClient(ServiceRegion);
           var createQueueResponse = await CreateQueue(client, QueueName);

           string queueUrl = createQueueResponse.QueueUrl;

           Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
           {
               { "Title",   new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
               { "Author",  new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
               { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
           };

           string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";

           var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
messageAttributes);
       }

       /// <summary>
       /// Creates a new Amazon SQS queue using the queue name passed to it
```

```
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
 string queueName)
        {
            var request = new CreateQueueRequest
            {
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
```

```
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");

            return response;
        }
    }
```

- API-Details finden Sie CreateQueue in der AWS SDK für .NET API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht aus einer Amazon-SQS-Warteschlange und löschen Sie sie anschließend.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
```

```
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
```

```
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- API-Details finden Sie DeleteMessage in der AWS SDK für .NET API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK für .NET API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Warteschlange unter Verwendung ihrer URL löschen

```
/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- API-Details finden Sie DeleteQueue in der AWS SDK für .NET API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get the ARN for a queue from its URL.
/// </summary>
/// <param name="queueUrl">The URL of the queue.</param>
/// <returns>The ARN of the queue.</returns>
public async Task<string> GetQueueArnByUrl(string queueUrl)
{
    var getAttributesRequest = new GetQueueAttributesRequest()
    {
```

```
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }
```

- API-Details finden Sie GetQueueAttributes in der AWS SDK für .NET API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class GetQueueUrl
    {
        /// <summary>
        /// Initializes the Amazon SQS client object and then calls the
        /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
        /// queue.
        /// </summary>
        public static async Task Main()
        {
            // If the Amazon SQS message queue is not in the same AWS Region as your
            // default user, you need to provide the AWS Region as a parameter to
the
```

```
            // client constructor.
            var client = new AmazonSQSClient();

            string queueName = "New-Example-Queue";

            try
            {
                var response = await client.GetQueueUrlAsync(queueName);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
                }
            }
            catch (QueueDoesNotExistException ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine($"The queue {queueName} was not found.");
            }
        }
    }
```

- API-Details finden Sie GetQueueUrl in der AWS SDK für .NET API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Nachrichten aus einer Warteschlange unter Verwendung ihrer URL empfangen

```
    /// <summary>
```

```
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }
```

Empfangen Sie eine Nachricht aus einer Amazon-SQS-Warteschlange und löschen Sie sie anschließend.

```
        public static async Task Main()
        {
            // If the AWS Region you want to use is different from
            // the AWS Region defined for the default user, supply
            // the specify your AWS Region to the client constructor.
            var client = new AmazonSQSClient();
            string queueName = "Example_Queue";

            var queueUrl = await GetQueueUrl(client, queueName);
            Console.WriteLine($"The SQS queue's URL is {queueUrl}");

            var response = await ReceiveAndDeleteMessage(client, queueUrl);

            Console.WriteLine($"Message: {response.Messages[0]}");
        }

        /// <summary>
        /// Retrieve the queue URL for the queue named in the queueName
```

```
        /// property using the client object.
        /// </summary>
        /// <param name="client">The Amazon SQS client used to retrieve the
        /// queue URL.</param>
        /// <param name="queueName">A string representing  name of the queue
        /// for which to retrieve the URL.</param>
        /// <returns>The URL of the queue.</returns>
        public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
        {
            var request = new GetQueueUrlRequest
            {
                QueueName = queueName,
            };

            GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
            return response.QueueUrl;
        }

        /// <summary>
        /// Retrieves the message from the quque at the URL passed in the
        /// queueURL parameters using the client.
        /// </summary>
        /// <param name="client">The SQS client used to retrieve a message.</param>
        /// <param name="queueUrl">The URL of the queue from which to retrieve
        /// a message.</param>
        /// <returns>The response from the call to ReceiveMessageAsync.</returns>
        public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
        {
            // Receive a single message from the queue.
            var receiveMessageRequest = new ReceiveMessageRequest
            {
                AttributeNames = { "SentTimestamp" },
                MaxNumberOfMessages = 1,
                MessageAttributeNames = { "All" },
                QueueUrl = queueUrl,
                VisibilityTimeout = 0,
                WaitTimeSeconds = 0,
            };

            var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);
```

```
            // Delete the received message from the queue.
            var deleteMessageRequest = new DeleteMessageRequest
            {
                QueueUrl = queueUrl,
                ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
            };

            await client.DeleteMessageAsync(deleteMessageRequest);

            return receiveMessageResponse;
        }
    }
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für .NET API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Amazon-SQS-Warteschlange und senden Sie eine Nachricht an diese.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SQS;
    using Amazon.SQS.Model;

    public class CreateSendExample
    {
        // Specify your AWS Region (an example Region is shown).
        private static readonly string QueueName = "Example_Queue";
```

```
        private static readonly RegionEndpoint ServiceRegion =
 RegionEndpoint.USWest2;
        private static IAmazonSQS client;

        public static async Task Main()
        {
            client = new AmazonSQSClient(ServiceRegion);
            var createQueueResponse = await CreateQueue(client, QueueName);

            string queueUrl = createQueueResponse.QueueUrl;

            Dictionary<string, MessageAttributeValue> messageAttributes = new
 Dictionary<string, MessageAttributeValue>
            {
                { "Title",   new MessageAttributeValue { DataType = "String",
 StringValue = "The Whistler" } },
                { "Author",  new MessageAttributeValue { DataType = "String",
 StringValue = "John Grisham" } },
                { "WeeksOn", new MessageAttributeValue { DataType = "Number",
 StringValue = "6" } },
            };

            string messageBody = "Information about current NY Times fiction
 bestseller for week of 12/11/2016.";

            var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
 messageAttributes);
        }

        /// <summary>
        /// Creates a new Amazon SQS queue using the queue name passed to it
        /// in queueName.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueName">A string representing the name of the queue
        /// to create.</param>
        /// <returns>A CreateQueueResponse that contains information about the
        /// newly created queue.</returns>
        public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
 string queueName)
        {
            var request = new CreateQueueRequest
            {
```

```
                QueueName = queueName,
                Attributes = new Dictionary<string, string>
                {
                    { "DelaySeconds", "60" },
                    { "MessageRetentionPeriod", "86400" },
                },
            };

            var response = await client.CreateQueueAsync(request);
            Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

            return response;
        }

        /// <summary>
        /// Sends a message to an SQS queue.
        /// </summary>
        /// <param name="client">An SQS client object used to send the message.</
param>
        /// <param name="queueUrl">The URL of the queue to which to send the
        /// message.</param>
        /// <param name="messageBody">A string representing the body of the
        /// message to be sent to the queue.</param>
        /// <param name="messageAttributes">Attributes for the message to be
        /// sent to the queue.</param>
        /// <returns>A SendMessageResponse object that contains information
        /// about the message that was sent.</returns>
        public static async Task<SendMessageResponse> SendMessage(
            IAmazonSQS client,
            string queueUrl,
            string messageBody,
            Dictionary<string, MessageAttributeValue> messageAttributes)
        {
            var sendMessageRequest = new SendMessageRequest
            {
                DelaySeconds = 10,
                MessageAttributes = messageAttributes,
                MessageBody = messageBody,
                QueueUrl = queueUrl,
            };

            var response = await client.SendMessageAsync(sendMessageRequest);
            Console.WriteLine($"Sent a message with id : {response.MessageId}");
```

```
            return response;
        }
    }
```

- API-Details finden Sie SendMessage in der AWS SDK für .NET API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richtlinienattribut einer Warteschlange für ein Thema festlegen

```
    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
```

```
                                        $"\"Resource\": \"{queueArn}\"," +
                                         "\"Condition\": {" +
                                              "\"ArnEquals\": {" +
                                                   $"\"aws:SourceArn\": \"{topicArn}\""
+
                                                    "}" +
                                            "}" +
                                 "}]" +
                              "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK für .NET API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Console application to run a feature scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues scenario is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
```

```
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this scenario, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for configuring
 the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the results
 in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
```

```
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic, deduplication
 is supported." +
                              $"\r\nDeduplication IDs are either set in the message
 or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an SNS
 FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication interval,
 is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/dg/
fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName, _useFifoTopic,
 _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS queue:
", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue, enabling it to receive\r\n" +
```

```
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
 queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }
```

```
            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
```

```
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" : "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is a
 sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must set
 a message group ID." +
                                  "\r\nAll messages within the same group will be
 received in the order " +
                                  "they were published.");
```

```
                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID for
this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID for
this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?", false);
        }
    }

    /// <summary>
```

```
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl, 10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the queue
at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    public static async Task DeleteMessages(string queueUrl, List<Message> messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
```

```
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer = true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
```

```
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SQS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
 useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
```

```
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
```

```
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string topicArn,
string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                             "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
                                                $"\"aws:SourceArn\": \"{topicArn}\""
+
                                            "}" +
                                        "}" +
                                "}]" +
                            "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
```

```
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl, List<Message>
 messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
```

```
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Erstellen Sie eine Klasse, die Amazon–SNS-Operationen einschließt.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
```

```
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
 subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
 filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
 { { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</param>
    /// <param name="attributeValue">The optional attribute value for the message.</
param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
```

```
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Benutze den AWS Framework zur Nachrichtenverarbeitung für.NET mit Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mithilfe des AWS Message Processing Framework für.NET Anwendungen erstellen, die Amazon SQS-Nachrichten veröffentlichen und empfangen.

SDK für .NET

Stellt ein Tutorial für das AWS Message Processing Framework für .NET bereit. Das Tutorial erstellt eine Webanwendung, mit der der Benutzer Amazon-SQS-Nachricht veröffentlichen kann, sowie eine Befehlszeilenanwendung, die die Nachricht empfängt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Tutorial im AWS SDK für .NET Entwicklerhandbuch und im Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SQS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für .NET

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
```

```
namespace SqsIntegrationSampleCode
{
    public async Task FunctionHandler(SQSEvent evnt, ILambdaContext context)
    {
        foreach (var message in evnt.Records)
        {
            await ProcessMessageAsync(message, context);
        }

        context.Logger.LogInformation("done");
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed message {message.Body}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }

    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SQSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeriali
namespace sqsSample;

public class Function
{
    public async Task<SQSBatchResponse> FunctionHandler(SQSEvent evnt,
 ILambdaContext context)
    {
        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 List<SQSBatchResponse.BatchItemFailure>();
        foreach(var message in evnt.Records)
        {
            try
            {
                //process your message
                await ProcessMessageAsync(message, context);
            }
            catch (System.Exception)
            {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.Add(new
 SQSBatchResponse.BatchItemFailure{ItemIdentifier=message.MessageId});
            }
        }
        return new SQSBatchResponse(batchItemFailures);
```

```
    }

    private async Task ProcessMessageAsync(SQSEvent.SQSMessage message,
 ILambdaContext context)
    {
        if (String.IsNullOrEmpty(message.Body))
        {
            throw new Exception("No Body in SQS Message.");
        }
        context.Logger.LogInformation($"Processed message {message.Body}");
        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
}
```

Beispiele für Step Functions mit SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Step Functions

Das folgende Codebeispiel zeigt, wie Sie mit Step Functions beginnen.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

public class HelloStepFunctions
{
    static async Task Main()
    {
        var stepFunctionsClient = new AmazonStepFunctionsClient();

        Console.Clear();
        Console.WriteLine("Welcome to AWS Step Functions");
        Console.WriteLine("Let's list up to 10 of your state machines:");
        var stateMachineListRequest = new ListStateMachinesRequest { MaxResults =
 10 };

        // Get information for up to 10 Step Functions state machines.
        var response = await
 stepFunctionsClient.ListStateMachinesAsync(stateMachineListRequest);

        if (response.StateMachines.Count > 0)
        {
            response.StateMachines.ForEach(stateMachine =>
            {
                Console.WriteLine($"State Machine Name: {stateMachine.Name}\tAmazon
 Resource Name (ARN): {stateMachine.StateMachineArn}");
            });
        }
        else
        {
            Console.WriteLine("\tNo state machines were found.");
        }
```

```
    }
}
```

- API-Details finden Sie ListStateMachines in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
global using System.Text.Json;
global using Amazon.StepFunctions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using StepFunctionsActions;
```

```
global using LogLevel = Microsoft.Extensions.Logging.LogLevel;



using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.StepFunctions.Model;

namespace StepFunctionsBasics;

public class StepFunctionsBasics
{
    private static ILogger _logger = null!;
    private static IConfigurationRoot _configuration = null!;
    private static IAmazonIdentityManagementService _iamService = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Step Functions.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonStepFunctions>()
                    .AddAWSService<IAmazonIdentityManagementService>()
                    .AddTransient<StepFunctionsWrapper>()
            )
            .Build();

        _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<StepFunctionsBasics>();

        // Load configuration settings.
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var activityName = _configuration["ActivityName"];
```

```
        var stateMachineName = _configuration["StateMachineName"];

        var roleName = _configuration["RoleName"];
        var repoBaseDir = _configuration["RepoBaseDir"];
        var jsonFilePath = _configuration["JsonFilePath"];
        var jsonFileName = _configuration["JsonFileName"];

        var uiMethods = new UiMethods();
        var stepFunctionsWrapper =
 host.Services.GetRequiredService<StepFunctionsWrapper>();

        _iamService =
 host.Services.GetRequiredService<IAmazonIdentityManagementService>();

        // Load definition for the state machine from a JSON file.
        var stateDefinitionJson = File.ReadAllText($"{repoBaseDir}{jsonFilePath}
{jsonFileName}");

        Console.Clear();
        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("Create activity");
        Console.WriteLine("Let's start by creating an activity.");
        string activityArn;
        string stateMachineArn;

        // Check to see if the activity already exists.
        var activityList = await stepFunctionsWrapper.ListActivitiesAsync();
        var existingActivity = activityList.FirstOrDefault(activity => activity.Name
 == activityName);
        if (existingActivity is not null)
        {
            activityArn = existingActivity.ActivityArn;
            Console.WriteLine($"Activity, {activityName}, already exists.");
        }
        else
        {
            activityArn = await stepFunctionsWrapper.CreateActivity(activityName);
        }

        // Swap the placeholder in the JSON file with the Amazon Resource Name (ARN)
        // of the recently created activity.
```

```
        var stateDefinition =
stateDefinitionJson.Replace("{{DOC_EXAMPLE_ACTIVITY_ARN}}", activityArn);

        uiMethods.DisplayTitle("Create state machine");
        Console.WriteLine("Now we'll create a state machine.");

        // Find or create an IAM role that can be assumed by Step Functions.
        var role = await GetOrCreateStateMachineRole(roleName);

        // See if the state machine already exists.
        var stateMachineList = await stepFunctionsWrapper.ListStateMachinesAsync();
        var existingStateMachine =
            stateMachineList.FirstOrDefault(stateMachine => stateMachine.Name ==
stateMachineName);
        if (existingStateMachine is not null)
        {
            Console.WriteLine($"State machine, {stateMachineName}, already
exists.");
            stateMachineArn = existingStateMachine.StateMachineArn;
        }
        else
        {
            // Create the state machine.
            stateMachineArn =
                await stepFunctionsWrapper.CreateStateMachine(stateMachineName,
stateDefinition, role.Arn);
            uiMethods.PressEnter();
        }

        Console.WriteLine("The state machine has been created.");
        var describeStateMachineResponse = await
stepFunctionsWrapper.DescribeStateMachineAsync(stateMachineArn);


Console.WriteLine($"{describeStateMachineResponse.Name}\t{describeStateMachineResponse.Stat
        Console.WriteLine($"Current status: {describeStateMachineResponse.Status}");
        Console.WriteLine($"Amazon Resource Name (ARN) of the role assumed by the
state machine: {describeStateMachineResponse.RoleArn}");

        var userName = string.Empty;
        Console.Write("Before we start the state machine, tell me what should
ChatSFN call you? ");
        userName = Console.ReadLine();
```

```
        // Keep asking until the user enters a string value.
        while (string.IsNullOrEmpty(userName))
        {
            Console.Write("Enter your name: ");
            userName = Console.ReadLine();
        }

        var executionJson = @"{""name"": """ + userName + @"""}";

        // Start the state machine execution.
        Console.WriteLine("Now we'll start execution of the state machine.");
        var executionArn = await
stepFunctionsWrapper.StartExecutionAsync(executionJson, stateMachineArn);
        Console.WriteLine("State machine started.");

        Console.WriteLine($"Thank you, {userName}. Now let's get started...");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("ChatSFN");

        var isDone = false;
        var response = new GetActivityTaskResponse();
        var taskToken = string.Empty;
        var userChoice = string.Empty;

        while (!isDone)
        {
            response = await stepFunctionsWrapper.GetActivityTaskAsync(activityArn,
"MvpWorker");
            taskToken = response.TaskToken;

            // Parse the returned JSON string.
            var taskJsonResponse = JsonDocument.Parse(response.Input);
            var taskJsonObject = taskJsonResponse.RootElement;
            var message = taskJsonObject.GetProperty("message").GetString();
            var actions =
taskJsonObject.GetProperty("actions").EnumerateArray().Select(x =>
x.ToString()).ToList();
            Console.WriteLine($"\n{message}\n");

            // Prompt the user for another choice.
            Console.WriteLine("ChatSFN: What would you like me to do?");
            actions.ForEach(action => Console.WriteLine($"\t{action}"));
            Console.Write($"\n{userName}, tell me your choice: ");
```

```
            userChoice = Console.ReadLine();
            if (userChoice?.ToLower() == "done")
            {
                isDone = true;
            }

            Console.WriteLine($"You have selected: {userChoice}");
            var jsonResponse = @"{""action"": """ + userChoice + @"""}";

            await stepFunctionsWrapper.SendTaskSuccessAsync(taskToken,
jsonResponse);
        }

        await stepFunctionsWrapper.StopExecution(executionArn);
        Console.WriteLine("Now we will wait for the execution to stop.");
        DescribeExecutionResponse executionResponse;
        do
        {
            executionResponse = await
stepFunctionsWrapper.DescribeExecutionAsync(executionArn);
        } while (executionResponse.Status == ExecutionStatus.RUNNING);

        Console.WriteLine("State machine stopped.");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("State machine executions");
        Console.WriteLine("Now let's take a look at the execution values for the
state machine.");

        // List the executions.
        var executions = await
stepFunctionsWrapper.ListExecutionsAsync(stateMachineArn);

        uiMethods.DisplayTitle("Step function execution values");
        executions.ForEach(execution =>
        {
            Console.WriteLine($"{execution.Name}\t{execution.StartDate} to
{execution.StopDate}");
        });

        uiMethods.PressEnter();

        // Now delete the state machine and the activity.
        uiMethods.DisplayTitle("Clean up resources");
```

```
        Console.WriteLine("Deleting the state machine...");

        await stepFunctionsWrapper.DeleteStateMachine(stateMachineArn);
        Console.WriteLine("State machine deleted.");

        Console.WriteLine("Deleting the activity...");
        await stepFunctionsWrapper.DeleteActivity(activityArn);
        Console.WriteLine("Activity deleted.");

        Console.WriteLine("The Amazon Step Functions scenario is now complete.");
    }

    static async Task<Role> GetOrCreateStateMachineRole(string roleName)
    {
        // Define the policy document for the role.
        var stateMachineRolePolicy = @"{
         ""Version"": ""2012-10-17"",
        ""Statement"": [{
            ""Sid"": """",
            ""Effect"": ""Allow"",
            ""Principal"": {
                ""Service"": ""states.amazonaws.com""},
            ""Action"": ""sts:AssumeRole""}]}";

        var role = new Role();
        var roleExists = false;

        try
        {
            var getRoleResponse = await _iamService.GetRoleAsync(new GetRoleRequest
{ RoleName = roleName });
            roleExists = true;
            role = getRoleResponse.Role;
        }
        catch (NoSuchEntityException)
        {
            // The role doesn't exist. Create it.
            Console.WriteLine($"Role, {roleName} doesn't exist. Creating it...");
        }

        if (!roleExists)
        {
            var request = new CreateRoleRequest
            {
```

```
                RoleName = roleName,
                AssumeRolePolicyDocument = stateMachineRolePolicy,
            };

            var createRoleResponse = await _iamService.CreateRoleAsync(request);
            role = createRoleResponse.Role;
        }

        return role;
    }
}


namespace StepFunctionsBasics;

/// <summary>
/// Some useful methods to make screen display easier.
/// </summary>
public class UiMethods
{
    private readonly string _sepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Welcome to the AWS Step Functions Demo");

        Console.WriteLine("This example application will do the following:");
        Console.WriteLine("\t 1. Create an activity.");
        Console.WriteLine("\t 2. Create a state machine.");
        Console.WriteLine("\t 3. Start an execution.");
        Console.WriteLine("\t 4. Run the worker, then stop it.");
        Console.WriteLine("\t 5. List executions.");
        Console.WriteLine("\t 6. Clean up the resources created for the example.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
```

```
        Console.Write("\nPress <Enter> to continue.");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter"></param>
    /// <returns></returns>
    private string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(_sepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(_sepBar);
    }
}
```

Definieren Sie eine Klasse, die Zustandsautomaten und Aktivitätsaktionen umschließt.

```
namespace StepFunctionsActions;

using Amazon.StepFunctions;
using Amazon.StepFunctions.Model;

/// <summary>
/// Wrapper that performs AWS Step Functions actions.
/// </summary>
public class StepFunctionsWrapper
```

```
{
    private readonly IAmazonStepFunctions _amazonStepFunctions;

    /// <summary>
    /// The constructor for the StepFunctionsWrapper. Initializes the
    /// client object passed to it.
    /// </summary>
    /// <param name="amazonStepFunctions">An initialized Step Functions client
 object.</param>
    public StepFunctionsWrapper(IAmazonStepFunctions amazonStepFunctions)
    {
        _amazonStepFunctions = amazonStepFunctions;
    }

    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
    /// <param name="activityName">The name for the new Step Functions activity.</
param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
 CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }


    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
 definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
            Definition = definition,
```

```
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }


    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
```

```
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
 executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
 DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }


    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse> DescribeStateMachineAsync(string
 StateMachineArn)
    {
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
 DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }


    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
 activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
 GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }


    /// <summary>
```

```
    /// List the Step Functions activities for the current account.
    /// </summary>
    /// <returns>A list of ActivityListItems.</returns>
    public async Task<List<ActivityListItem>> ListActivitiesAsync()
    {
        var request = new ListActivitiesRequest();
        var activities = new List<ActivityListItem>();

        do
        {
            var response = await _amazonStepFunctions.ListActivitiesAsync(request);

            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }

            activities.AddRange(response.Activities);
        }
        while (request.NextToken is not null);

        return activities;
    }


    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
```

```
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }


    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
 ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }


    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the task.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
 taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
 SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
_amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }


    /// <summary>
    /// Stop execution of a Step Functions workflow.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of
    /// the Step Functions execution to stop.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StopExecution(string executionArn)
    {
        var response =
            await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
{ ExecutionArn = executionArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt die Verwendung`CreateActivity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a Step Functions activity using the supplied name.
    /// </summary>
```

```
    /// <param name="activityName">The name for the new Step Functions activity.</
param>
    /// <returns>The Amazon Resource Name (ARN) for the new activity.</returns>
    public async Task<string> CreateActivity(string activityName)
    {
        var response = await _amazonStepFunctions.CreateActivityAsync(new
 CreateActivityRequest { Name = activityName });
        return response.ActivityArn;
    }
```

- API-Details finden Sie CreateActivity in der AWS SDK für .NET API-Referenz.

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`CreateStateMachine`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineName">Name for the new Step Functions state
    /// machine.</param>
    /// <param name="definition">A JSON string that defines the Step Functions
    /// state machine.</param>
    /// <param name="roleArn">The Amazon Resource Name (ARN) of the role.</param>
    /// <returns></returns>
    public async Task<string> CreateStateMachine(string stateMachineName, string
 definition, string roleArn)
    {
        var request = new CreateStateMachineRequest
        {
            Name = stateMachineName,
```

```
            Definition = definition,
            RoleArn = roleArn
        };

        var response =
            await _amazonStepFunctions.CreateStateMachineAsync(request);
        return response.StateMachineArn;
    }
```

- API-Details finden Sie CreateStateMachine in der AWS SDK für .NET API-Referenz.

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteActivity`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a Step Machine activity.
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the activity.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteActivity(string activityArn)
    {
        var response = await _amazonStepFunctions.DeleteActivityAsync(new
DeleteActivityRequest { ActivityArn = activityArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteActivity in der AWS SDK für .NET API-Referenz.

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStateMachine`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a Step Functions state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// state machine.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteStateMachine(string stateMachineArn)
    {
        var response = await _amazonStepFunctions.DeleteStateMachineAsync(new
DeleteStateMachineRequest
        { StateMachineArn = stateMachineArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteStateMachine in der AWS SDK für .NET API-Referenz.

`DescribeExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribeExecution`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions execution.
    /// </summary>
    /// <param name="executionArn">The Amazon Resource Name (ARN) of the
    /// Step Functions execution.</param>
    /// <returns>The API response returned by the API.</returns>
    public async Task<DescribeExecutionResponse> DescribeExecutionAsync(string
 executionArn)
    {
        var response = await _amazonStepFunctions.DescribeExecutionAsync(new
 DescribeExecutionRequest { ExecutionArn = executionArn });
        return response;
    }
```

- API-Details finden Sie DescribeExecution in der AWS SDK für .NET API-Referenz.

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStateMachine`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about the specified Step Functions state machine.
    /// </summary>
    /// <param name="StateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine to retrieve.</param>
    /// <returns>Information about the specified Step Functions state machine.</
returns>
    public async Task<DescribeStateMachineResponse> DescribeStateMachineAsync(string
 StateMachineArn)
    {
```

```
        var response = await _amazonStepFunctions.DescribeStateMachineAsync(new
DescribeStateMachineRequest { StateMachineArn = StateMachineArn });
        return response;
    }
```

- API-Details finden Sie DescribeStateMachine in der AWS SDK für .NET API-Referenz.

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung`GetActivityTask`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a task with the specified Step Functions activity
    /// with the specified Amazon Resource Name (ARN).
    /// </summary>
    /// <param name="activityArn">The Amazon Resource Name (ARN) of
    /// the Step Functions activity.</param>
    /// <param name="workerName">The name of the Step Functions worker.</param>
    /// <returns>The response from the Step Functions activity.</returns>
    public async Task<GetActivityTaskResponse> GetActivityTaskAsync(string
activityArn, string workerName)
    {
        var response = await _amazonStepFunctions.GetActivityTaskAsync(new
GetActivityTaskRequest
        { ActivityArn = activityArn, WorkerName = workerName });
        return response;
    }
```

- API-Details finden Sie GetActivityTask in der AWS SDK für .NET API-Referenz.

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung`ListActivities`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// List the Step Functions activities for the current account.
/// </summary>
/// <returns>A list of ActivityListItems.</returns>
public async Task<List<ActivityListItem>> ListActivitiesAsync()
{
    var request = new ListActivitiesRequest();
    var activities = new List<ActivityListItem>();

    do
    {
        var response = await _amazonStepFunctions.ListActivitiesAsync(request);

        if (response.NextToken is not null)
        {
            request.NextToken = response.NextToken;
        }

        activities.AddRange(response.Activities);
    }
    while (request.NextToken is not null);

    return activities;
}
```

- API-Details finden Sie ListActivities in der AWS SDK für .NET API-Referenz.

`ListExecutions`

Das folgende Codebeispiel zeigt die Verwendung`ListExecutions`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve information about executions of a Step Functions
    /// state machine.
    /// </summary>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>A list of ExecutionListItem objects.</returns>
    public async Task<List<ExecutionListItem>> ListExecutionsAsync(string
stateMachineArn)
    {
        var executions = new List<ExecutionListItem>();
        ListExecutionsResponse response;
        var request = new ListExecutionsRequest { StateMachineArn =
stateMachineArn };

        do
        {
            response = await _amazonStepFunctions.ListExecutionsAsync(request);
            executions.AddRange(response.Executions);
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);

        return executions;
    }
```

- API-Details finden Sie ListExecutions in der AWS SDK für .NET API-Referenz.

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung`ListStateMachines`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a list of Step Functions state machines.
    /// </summary>
    /// <returns>A list of StateMachineListItem objects.</returns>
    public async Task<List<StateMachineListItem>> ListStateMachinesAsync()
    {
        var stateMachines = new List<StateMachineListItem>();
        var listStateMachinesPaginator =
            _amazonStepFunctions.Paginators.ListStateMachines(new
ListStateMachinesRequest());

        await foreach (var response in listStateMachinesPaginator.Responses)
        {
            stateMachines.AddRange(response.StateMachines);
        }

        return stateMachines;
    }
```

- API-Details finden Sie ListStateMachines in der AWS SDK für .NET API-Referenz.

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung`SendTaskSuccess`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Indicate that the Step Functions task, indicated by the
    /// task token, has completed successfully.
    /// </summary>
    /// <param name="taskToken">Identifies the task.</param>
    /// <param name="taskResponse">The response received from executing the task.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SendTaskSuccessAsync(string taskToken, string
 taskResponse)
    {
        var response = await _amazonStepFunctions.SendTaskSuccessAsync(new
 SendTaskSuccessRequest
        { TaskToken = taskToken, Output = taskResponse });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie SendTaskSuccess in der AWS SDK für .NET API-Referenz.

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start execution of an AWS Step Functions state machine.
    /// </summary>
    /// <param name="executionName">The name to use for the execution.</param>
    /// <param name="executionJson">The JSON string to pass for execution.</param>
    /// <param name="stateMachineArn">The Amazon Resource Name (ARN) of the
    /// Step Functions state machine.</param>
    /// <returns>The Amazon Resource Name (ARN) of the AWS Step Functions
    /// execution.</returns>
    public async Task<string> StartExecutionAsync(string executionJson, string
stateMachineArn)
    {
        var executionRequest = new StartExecutionRequest
        {
            Input = executionJson,
            StateMachineArn = stateMachineArn
        };

        var response = await
_amazonStepFunctions.StartExecutionAsync(executionRequest);
        return response.ExecutionArn;
    }
```

- API-Details finden Sie StartExecution in der AWS SDK für .NET API-Referenz.

AWS STS Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do not
 specify a region,
        // or to use a specific region.
```

```
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- API-Details finden Sie AssumeRole in der AWS SDK für .NET API-Referenz.

Support Beispiele unter Verwendung SDK für .NET

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für .NET with Aktionen ausführen und gängige Szenarien implementieren Support.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für .NET

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
```

```
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        // You must have one of the following AWS Support plans: Business,
 Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            ).Build();

        // Now the client is available for injection.
        var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\tHello AWS Support! There are {response.Services.Count}
 services available.");
    }
}
```

- API-Details finden Sie DescribeServices in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für .NET

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
/// <summary>
/// Hello AWS Support example.
/// </summary>
public static class SupportCaseScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    To use the AWS Support API, you must have one of the following AWS Support
 plans: Business, Enterprise On-Ramp, or Enterprise.

    This .NET example performs the following tasks:
    1.  Get and display services. Select a service from the list.
    2.  Select a category from the selected service.
    3.  Get and display severity levels and select a severity level from the list.
    4.  Create a support case using the selected service, category, and severity
 level.
    5.  Get and display a list of open support cases for the current day.
    6.  Create an attachment set with a sample text file to add to the case.
    7.  Add a communication with the attachment to the support case.
    8.  List the communications of the support case.
    9.  Describe the attachment set.
    10. Resolve the support case.
    11. Get a list of resolved cases for the current day.
   */

    private static SupportWrapper _supportWrapper = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonAWSSupport>(new AWSOptions() { Profile
= "default" })
                    .AddTransient<SupportWrapper>()
            )
            .Build();

        var logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger(typeof(SupportCaseScenario));

        _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS Support case example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var apiSupported = await _supportWrapper.VerifySubscription();
            if (!apiSupported)
            {
                logger.LogError("You must have a Business, Enterprise On-Ramp, or
Enterprise Support " +
                                "plan to use the AWS Support API. \n\tPlease
upgrade your subscription to run these examples.");
                return;
            }

            var service = await DisplayAndSelectServices();

            var category = DisplayAndSelectCategories(service);

            var severityLevel = await DisplayAndSelectSeverity();

            var caseId = await CreateSupportCase(service, category, severityLevel);
```

```
            await DescribeTodayOpenCases();

            var attachmentSetId = await CreateAttachmentSet();

            await AddCommunicationToCase(attachmentSetId, caseId);

            var attachmentId = await ListCommunicationsForCase(caseId);

            await DescribeCaseAttachment(attachmentId);

            await ResolveCase(caseId);

            await DescribeTodayResolvedCases();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("AWS Support case example scenario complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// List some available services from AWS Support, and select a service for the
example.
    /// </summary>
    /// <returns>The selected service.</returns>
    private static async Task<Service> DisplayAndSelectServices()
    {
        Console.WriteLine(new string('-', 80));
        var services = await _supportWrapper.DescribeServices();
        Console.WriteLine($"AWS Support client returned {services.Count}
services.");

        Console.WriteLine($"1. Displaying first 10 services:");
        for (int i = 0; i < 10 && i < services.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {services[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > services.Count)
```

```
        {
            Console.WriteLine(
                "Select an example support service by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return services[choiceNumber - 1];
    }

    /// <summary>
    /// List the available categories for a service and select a category for the
example.
    /// </summary>
    /// <param name="service">Service to use for displaying categories.</param>
    /// <returns>The selected category.</returns>
    private static Category DisplayAndSelectCategories(Service service)
    {
        Console.WriteLine(new string('-', 80));

        Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
        for (int i = 0; i < service.Categories.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {service.Categories[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
        {
            Console.WriteLine(
                "Select an example support category by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        Console.WriteLine(new string('-', 80));

        return service.Categories[choiceNumber - 1];
    }
```

```
    /// <summary>
    /// List available severity levels from AWS Support, and select a level for the
example.
    /// </summary>
    /// <returns>The selected severity level.</returns>
    private static async Task<SeverityLevel> DisplayAndSelectSeverity()
    {
        Console.WriteLine(new string('-', 80));
        var severityLevels = await _supportWrapper.DescribeSeverityLevels();

        Console.WriteLine($"3. Get and display available severity levels:");
        for (int i = 0; i < 10 && i < severityLevels.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {severityLevels[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
        {
            Console.WriteLine(
                "Select an example severity level by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return severityLevels[choiceNumber - 1];
    }

    /// <summary>
    /// Create an example support case.
    /// </summary>
    /// <param name="service">Service to use for the new case.</param>
    /// <param name="category">Category to use for the new case.</param>
    /// <param name="severity">Severity to use for the new case.</param>
    /// <returns>The caseId of the new support case.</returns>
    private static async Task<string> CreateSupportCase(Service service,
        Category category, SeverityLevel severity)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Create an example support case" +
                          $" with the following settings:" +
```

```
                            $" \n\tService: {service.Name}, Category: {category.Name}
" +
                            $"and Severity Level: {severity.Name}.");
        var caseId = await _supportWrapper.CreateCase(service.Code, category.Code,
severity.Code,
            "Example case for testing, ignore.", "This is my example support
case.");

        Console.WriteLine($"\tNew case created with ID {caseId}");

        Console.WriteLine(new string('-', 80));

        return caseId;
    }

    /// <summary>
    /// List open cases for the current day.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeTodayOpenCases()
    {
        Console.WriteLine($"5. List the open support cases for the current day.");
        // Describe the cases. If it is empty, try again and allow time for the new
case to appear.
        List<CaseDetails> currentOpenCases = null!;
        while (currentOpenCases == null || currentOpenCases.Count == 0)
        {
            Thread.Sleep(1000);
            currentOpenCases = await _supportWrapper.DescribeCases(
                new List<string>(),
                null,
                false,
                false,
                DateTime.UtcNow.Date,
                DateTime.UtcNow);
        }

        foreach (var openCase in currentOpenCases)
        {
            Console.WriteLine($"\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
        }

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Create an attachment set for a support case.
    /// </summary>
    /// <returns>The attachment set id.</returns>
    private static async Task<string> CreateAttachmentSet()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Create an attachment set for a support case.");
        var fileName = "example_attachment.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for attachment to a support case.");
        }

        await using var ms = new MemoryStream(await
 File.ReadAllBytesAsync(fileName));

        var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
            ms,
            fileName);

        Console.WriteLine($"\tNew attachment set created with id: \n
\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    /// <summary>
    /// Add an attachment set and communication to a case.
    /// </summary>
    /// <param name="attachmentSetId">Id of the attachment set.</param>
    /// <param name="caseId">Id of the case to receive the attachment set.</param>
    /// <returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId, string
 caseId)
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Add attachment set and communication to {caseId}.");

        await _supportWrapper.AddCommunicationToCase(
            caseId,
            "This is an example communication added to a support case.",
            attachmentSetId);

        Console.WriteLine($"\tNew attachment set and communication added to
{caseId}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List the communications for a case.
    /// </summary>
    /// <param name="caseId">Id of the case to describe.</param>
    /// <returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. List communications for case {caseId}.");

        var communications = await _supportWrapper.DescribeCommunications(caseId);
        var attachmentId = "";
        foreach (var communication in communications)
        {
            Console.WriteLine(
                $"\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
            if (communication.AttachmentSet.Any())
            {
                attachmentId = communication.AttachmentSet.First().AttachmentId;
            }
        }

        Console.WriteLine(new string('-', 80));
        return attachmentId;
    }

    /// <summary>
    /// Describe an attachment by id.
    /// </summary>
```

```
    /// <param name="attachmentId">Id of the attachment to describe.</param>
    /// <returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Describe the attachment set.");

        var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
        var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
        Console.WriteLine($"\tAttachment includes {attachment.FileName} with data:
\n\t{data}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Resolve the support case.
    /// </summary>
    /// <param name="caseId">Id of the case to resolve.</param>
    /// <returns>Async task.</returns>
    private static async Task ResolveCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Resolve case {caseId}.");

        var status = await _supportWrapper.ResolveCase(caseId);
        Console.WriteLine($"\tCase {caseId} has final status {status}");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List resolved cases for the current day.
    /// </summary>
    /// <returns>Async Task.</returns>
    private static async Task DescribeTodayResolvedCases()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. List the resolved support cases for the current
day.");
        var currentCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
```

```
            true,
            DateTime.UtcNow.Date,
            DateTime.UtcNow);

        foreach (var currentCase in currentCases)
        {
            if (currentCase.Status == "resolved")
            {
                Console.WriteLine(
                    $"\tCase: {currentCase.CaseId}: status {currentCase.Status}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
}
```

Wrapper-Methoden, die vom Szenario für Support Aktionen verwendet werden.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
public class SupportWrapper
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }


    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
```

```
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }



    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }



    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
```

```
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }



    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does not
exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates a
new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
```

```
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }



    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }



    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
```

```
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }



    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }



    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
```

```
    /// <param name="includeCommunication">True to include communication. Defaults
 to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases. Defaults
 to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?
 displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }



    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
```

```
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
            {
                CaseId = caseId
            });
        return response.FinalCaseStatus;
    }


    /// <summary>
    /// Verify the support level for AWS Support API access.
    /// </summary>
    /// <returns>True if the subscription level supports API access.</returns>
    public async Task<bool> VerifySubscription()
    {
        try
        {
            var response = await _amazonSupport.DescribeServicesAsync(
                new DescribeServicesRequest()
                {
                    Language = "en"
                });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Amazon.AWSSupport.AmazonAWSSupportException ex)
        {
            if (ex.ErrorCode == "SubscriptionRequiredException")
            {
                return false;
            }
            else throw;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AddAttachmentsToSet

- AddCommunicationToCase
- CreateCase
- DescribeAttachment
- DescribeCases
- DescribeCommunications
- DescribeServices
- DescribeSeverityLevels
- ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt die Verwendung`AddAttachmentsToSet`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Add an attachment to a set, or create a new attachment set if one does not
exist.
   /// </summary>
   /// <param name="data">The data for the attachment.</param>
   /// <param name="fileName">The file name for the attachment.</param>
   /// <param name="attachmentSetId">Optional setId for the attachment. Creates a
new attachment set if empty.</param>
   /// <returns>The setId of the attachment.</returns>
   public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
string? attachmentSetId = null)
   {
       var response = await _amazonSupport.AddAttachmentsToSetAsync(
           new AddAttachmentsToSetRequest
```

```
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }
```

- API-Details finden Sie AddAttachmentsToSet in der AWS SDK für .NET API-Referenz.

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`AddCommunicationToCase`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add communication to a case, including optional attachment set ID and CC
email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
```

```
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }
```

- API-Details finden Sie AddCommunicationToCase in der AWS SDK für .NET API-Referenz.

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung`CreateCase`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
```

```
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }
```

- API-Details finden Sie CreateCase in der AWS SDK für .NET API-Referenz.

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAttachment`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }
```

- API-Details finden Sie DescribeAttachment in der AWS SDK für .NET API-Referenz.

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCases`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication. Defaults
to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases. Defaults
to false.</param>
```

```
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <param name="language">Optional language support for your case.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
 are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?
 displayId = null, bool includeCommunication = true,
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
 beforeTime = null,
        string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginators.DescribeCases(
            new DescribeCasesRequest()
            {
                CaseIdList = caseIds,
                DisplayId = displayId,
                IncludeCommunications = includeCommunication,
                IncludeResolvedCases = includeResolvedCases,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s"),
                Language = language
            });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }
```

- API-Details finden Sie DescribeCases in der AWS SDK für .NET API-Referenz.

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCommunications`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</
param>
    /// <param name="beforeTime">The optional end date for a filtered search.</
param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
 DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
 _amazonSupport.Paginators.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("s"),
                BeforeTime = beforeTime?.ToString("s")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }
```

- API-Details finden Sie DescribeCommunications in der AWS SDK für .NET API-Referenz.

`DescribeServices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeServices`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }
```

- API-Details finden Sie DescribeServices in der AWS SDK für .NET API-Referenz.

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSeverityLevels`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Get the descriptions of support severity levels.
   /// </summary>
   /// <param name="name">Optional language for severity levels.
   /// Currently Chinese (“zh”), English ("en"), Japanese ("ja") and Korean (“ko”)
are supported.</param>
   /// <returns>The list of support severity levels.</returns>
   public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
   {
       var response = await _amazonSupport.DescribeSeverityLevelsAsync(
           new DescribeSeverityLevelsRequest()
           {
               Language = language
           });
       return response.SeverityLevels;
   }
```

- API-Details finden Sie DescribeSeverityLevels in der AWS SDK für .NET API-Referenz.

`ResolveCase`

Das folgende Codebeispiel zeigt die Verwendung`ResolveCase`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Resolve a support case by caseId.
/// </summary>
/// <param name="caseId">Id for the support case.</param>
/// <returns>The final status of the case after resolving.</returns>
public async Task<string> ResolveCase(string caseId)
{
    var response = await _amazonSupport.ResolveCaseAsync(
        new ResolveCaseRequest()
        {
            CaseId = caseId
        });
    return response.FinalCaseStatus;
}
```

- API-Details finden Sie ResolveCase in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Textract mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Textract Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHub Projekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Amazon Transcribe Beispiele mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `CreateVocabulary`

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a custom vocabulary using a list of phrases. Custom vocabularies
    /// improve transcription accuracy for one or more specific words.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.CreateVocabularyAsync(
            new CreateVocabularyRequest
            {
```

```
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- API-Details finden Sie CreateVocabulary in der AWS SDK für .NET API-Referenz.

`DeleteMedicalTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMedicalTranscriptionJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a medical transcription job. Also deletes the transcript associated
 with the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
 _amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteMedicalTranscriptionJob in der AWS SDK für .NET API-Referenz.

`DeleteTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTranscriptionJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with the
job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteTranscriptionJob in der AWS SDK für .NET API-Referenz.

`DeleteVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVocabulary`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete an existing custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(
        new DeleteVocabularyRequest
        {
            VocabularyName = vocabularyName
        });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- API-Details finden Sie DeleteVocabulary in der AWS SDK für .NET API-Referenz.

`GetTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`GetTranscriptionJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get details about a transcription job.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
    public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
            new GetTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.TranscriptionJob;
    }
```

- API-Details finden Sie GetTranscriptionJob in der AWS SDK für .NET API-Referenz.

`GetVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`GetVocabulary`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get information about a custom vocabulary.
    /// </summary>
    /// <param name="vocabularyName">Name of the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
    {
```

```
        var response = await _amazonTranscribeService.GetVocabularyAsync(
            new GetVocabularyRequest()
            {
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- API-Details finden Sie GetVocabulary in der AWS SDK für .NET API-Referenz.

`ListMedicalTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListMedicalTranscriptionJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List medical transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
    /// <returns>A list of summaries about medical transcription jobs.</returns>
    public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
        string? jobNameContains = null)
    {
        var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
            new ListMedicalTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
```

```
        return response.MedicalTranscriptionJobSummaries;
    }
```

- API-Details finden Sie ListMedicalTranscriptionJobs in der AWS SDK für .NET API-Referenz.

`ListTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTranscriptionJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
    /// <returns>A list of transcription job summaries.</returns>
    public async Task<List<TranscriptionJobSummary>> ListTranscriptionJobs(string?
jobNameContains = null)
    {
        var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
            new ListTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.TranscriptionJobSummaries;
    }
```

- API-Details finden Sie ListTranscriptionJobs in der AWS SDK für .NET API-Referenz.

`ListVocabularies`

Das folgende Codebeispiel zeigt die Verwendung`ListVocabularies`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must
contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?
nameContains = null,
        VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- API-Details finden Sie ListVocabularies in der AWS SDK für .NET API-Referenz.

`StartMedicalTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`StartMedicalTranscriptionJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start a medical transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the medical transcription job.</
param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon S3
 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="outputBucketName">Location for the output, typically an Amazon
 S3 location.</param>
    /// <param name="transcriptionType">Conversation or dictation transcription
 type.</param>
    /// <returns>A MedicalTransactionJob instance with information on the new job.</
returns>
    public async Task<MedicalTranscriptionJob> StartMedicalTranscriptionJob(
        string jobName, string mediaFileUri,
        MediaFormat mediaFormat, string outputBucketName,
 Amazon.TranscribeService.Type transcriptionType)
    {
        var response = await
 _amazonTranscribeService.StartMedicalTranscriptionJobAsync(
            new StartMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode =
                    LanguageCode
```

```
                        .EnUS, // The value must be en-US for medical
 transcriptions.
                OutputBucketName = outputBucketName,
                OutputKey =
                    jobName, // The value is a key used to fetch the output of the
 transcription.
                Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must be
 set.
                Type = transcriptionType
            });
        return response.MedicalTranscriptionJob;
    }
```

- API-Details finden Sie StartMedicalTranscriptionJob in der AWS SDK für .NET API-Referenz.

StartTranscriptionJob

Das folgende Codebeispiel zeigt die VerwendungStartTranscriptionJob.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Start a transcription job for a media file. This method returns
    /// as soon as the job is started.
    /// </summary>
    /// <param name="jobName">A unique name for the transcription job.</param>
    /// <param name="mediaFileUri">The URI of the media file, typically an Amazon S3
 location.</param>
    /// <param name="mediaFormat">The format of the media file.</param>
    /// <param name="languageCode">The language code of the media file, such as en-
US.</param>
    /// <param name="vocabularyName">Optional name of a custom vocabulary.</param>
```

```
    /// <returns>A TranscriptionJob instance with information on the new job.</
returns>
    public async Task<TranscriptionJob> StartTranscriptionJob(string jobName, string
 mediaFileUri,
        MediaFormat mediaFormat, LanguageCode languageCode, string? vocabularyName)
    {
        var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
            new StartTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName,
                Media = new Media()
                {
                    MediaFileUri = mediaFileUri
                },
                MediaFormat = mediaFormat,
                LanguageCode = languageCode,
                Settings = vocabularyName != null ? new Settings()
                {
                    VocabularyName = vocabularyName
                } : null
            });
        return response.TranscriptionJob;
    }
```

- API-Details finden Sie StartTranscriptionJob in der AWS SDK für .NET API-Referenz.

`UpdateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`UpdateVocabulary`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
```

```
    /// Update a custom vocabulary with new values. Update overwrites all existing
information.
    /// </summary>
    /// <param name="languageCode">The language code of the vocabulary.</param>
    /// <param name="phrases">Phrases to use in the vocabulary.</param>
    /// <param name="vocabularyName">Name for the vocabulary.</param>
    /// <returns>The state of the custom vocabulary.</returns>
    public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
        List<string> phrases, string vocabularyName)
    {
        var response = await _amazonTranscribeService.UpdateVocabularyAsync(
            new UpdateVocabularyRequest()
            {
                LanguageCode = languageCode,
                Phrases = phrases,
                VocabularyName = vocabularyName
            });
        return response.VocabularyState;
    }
```

- API-Details finden Sie UpdateVocabulary in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Translate mit SDK für .NET

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Translate Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für .NET

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

- Szenarien

Aktionen

`DescribeTextTranslationJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTextTranslationJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// The following example shows how to retrieve the details of
/// a text translation job using Amazon Translate.
/// </summary>
public class DescribeTextTranslation
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();

        // The Job Id is generated when the text translation job is started
        // with a call to the StartTextTranslationJob method.
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new DescribeTextTranslationJobRequest
        {
            JobId = jobId,
        };

        var jobProperties = await DescribeTranslationJobAsync(client, request);
```

```
            DisplayTranslationJobDetails(jobProperties);
        }

        /// <summary>
        /// Retrieve information about an Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client object.</
param>
        /// <param name="request">The DescribeTextTranslationJobRequest object.</
param>
        /// <returns>The TextTranslationJobProperties object containing
        /// information about the text translation job..</returns>
        public static async Task<TextTranslationJobProperties>
 DescribeTranslationJobAsync(
            AmazonTranslateClient client,
            DescribeTextTranslationJobRequest request)
        {
            var response = await client.DescribeTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                return response.TextTranslationJobProperties;
            }
            else
            {
                return null;
            }
        }

        /// <summary>
        /// Displays the properties of the text translation job.
        /// </summary>
        /// <param name="jobProperties">The properties of the text translation
        /// job returned by the call to DescribeTextTranslationJobAsync.</param>
        public static void DisplayTranslationJobDetails(TextTranslationJobProperties
 jobProperties)
        {
            if (jobProperties is null)
            {
                Console.WriteLine("No text translation job properties found.");
                return;
            }

            // Display the details of the text translation job.
            Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
```

```
        }
    }
```

- API-Details finden Sie DescribeTextTranslationJob in der AWS SDK für .NET API-Referenz.

`ListTextTranslationJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTextTranslationJobs`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// List Amazon Translate translation jobs, along with details about each job.
/// </summary>
public class ListTranslationJobs
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var filter = new TextTranslationJobFilter
        {
            JobStatus = "COMPLETED",
        };

        var request = new ListTextTranslationJobsRequest
        {
            MaxResults = 10,
            Filter = filter,
```

```
            };

            await ListJobsAsync(client, request);
        }

        /// <summary>
        /// List Amazon Translate text translation jobs.
        /// </summary>
        /// <param name="client">The initialized Amazon Translate client object.</
param>
        /// <param name="request">An Amazon Translate
        /// ListTextTranslationJobsRequest object detailing which text
        /// translation jobs are of interest.</param>
        public static async Task ListJobsAsync(
            AmazonTranslateClient client,
            ListTextTranslationJobsRequest request)
        {
            ListTextTranslationJobsResponse response;

            do
            {
                response = await client.ListTextTranslationJobsAsync(request);

 ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

                request.NextToken = response.NextToken;
            }
            while (response.NextToken is not null);
        }

        /// <summary>
        /// List existing translation job details.
        /// </summary>
        /// <param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
 ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
        {
            properties.ForEach(prop =>
            {
                Console.WriteLine($"{prop.JobId}: {prop.JobName}");
                Console.WriteLine($"Status: {prop.JobStatus}");
                Console.WriteLine($"Submitted time: {prop.SubmittedTime}");
            });
```

```
        }
    }
```

- API-Details finden Sie ListTextTranslationJobs in der AWS SDK für .NET API-Referenz.

StartTextTranslationJob

Das folgende Codebeispiel zeigt die VerwendungStartTextTranslationJob.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// This example shows how to use Amazon Translate to process the files in
/// an Amazon Simple Storage Service (Amazon S3) bucket. The translated results
/// will also be stored in an Amazon S3 bucket.
/// </summary>
public class BatchTranslate
{
    public static async Task Main()
    {
        var contentType = "text/plain";

        // Set this variable to an S3 bucket location with a folder."
        // Input files must be in a folder and not at the bucket root."
        var s3InputUri = "s3://amzn-s3-demo-bucket1/FOLDER/";
        var s3OutputUri = "s3://amzn-s3-demo-bucket2/";
```

```
            // This role must have permissions to read the source bucket and to read
 and
            // write to the destination bucket where the translated text will be
 stored.
            var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

            var client = new AmazonTranslateClient();

            var inputConfig = new InputDataConfig
            {
                ContentType = contentType,
                S3Uri = s3InputUri,
            };

            var outputConfig = new OutputDataConfig
            {
                S3Uri = s3OutputUri,
            };

            var request = new StartTextTranslationJobRequest
            {
                JobName = "ExampleTranslationJob",
                DataAccessRoleArn = dataAccessRoleArn,
                InputDataConfig = inputConfig,
                OutputDataConfig = outputConfig,
                SourceLanguageCode = "en",
                TargetLanguageCodes = new List<string> { "fr" },
            };

            var response = await StartTextTranslationAsync(client, request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId}: {response.JobStatus}");
            }
        }

        /// <summary>
        /// Start the Amazon Translate text translation job.
        /// </summary>
        /// <param name="client">The initialized AmazonTranslateClient object.</
param>
        /// <param name="request">The request object that includes details such
```

```
        /// as source and destination bucket names and the IAM Role that will
        /// be used to access the buckets.</param>
        /// <returns>The StartTextTranslationResponse object that includes the
        /// details of the request response.</returns>
        public static async Task<StartTextTranslationJobResponse>
StartTextTranslationAsync(AmazonTranslateClient client,
StartTextTranslationJobRequest request)
        {
            var response = await client.StartTextTranslationJobAsync(request);
            return response;
        }
    }
```

- API-Details finden Sie StartTextTranslationJob in der AWS SDK für .NET API-Referenz.

StopTextTranslationJob

Das folgende Codebeispiel zeigt die Verwendung`StopTextTranslationJob`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Translate;
    using Amazon.Translate.Model;

    /// <summary>
    /// Shows how to stop a running Amazon Translation Service text translation
    /// job.
    /// </summary>
    public class StopTextTranslationJob
    {
        public static async Task Main()
```

```
        {
            var client = new AmazonTranslateClient();
            var jobId = "1234567890abcdef01234567890abcde";

            var request = new StopTextTranslationJobRequest
            {
                JobId = jobId,
            };

            await StopTranslationJobAsync(client, request);
        }

        /// <summary>
        /// Sends a request to stop a text translation job.
        /// </summary>
        /// <param name="client">Initialized AmazonTrnslateClient object.</param>
        /// <param name="request">The request object to be passed to the
        /// StopTextJobAsync method.</param>
        public static async Task StopTranslationJobAsync(
            AmazonTranslateClient client,
            StopTextTranslationJobRequest request)
        {
            var response = await client.StopTextTranslationJobAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
            }
        }
    }
```

- API-Details finden Sie StopTextTranslationJob in der AWS SDK für .NET API-Referenz.

`TranslateText`

Das folgende Codebeispiel zeigt die Verwendung`TranslateText`.

SDK für .NET

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;
using Amazon.Translate;
using Amazon.Translate.Model;

/// <summary>
/// Take text from a file stored a Amazon Simple Storage Service (Amazon S3)
/// object and translate it using the Amazon Transfer Service.
/// </summary>
public class TranslateText
{
    public static async Task Main()
    {
        // If the region you want to use is different from the region
        // defined for the default user, supply it as a parameter to the
        // Amazon Translate client object constructor.
        var client = new AmazonTranslateClient();

        // Set the source language to "auto" to request Amazon Translate to
        // automatically detect te language of the source text.

        // You can get a list of the languages supposed by Amazon Translate
        // in the Amazon Translate Developer's Guide here:
        //      https://docs.aws.amazon.com/translate/latest/dg/what-is.html
        string srcLang = "en"; // English.
        string destLang = "fr"; // French.

        // The Amazon Simple Storage Service (Amazon S3) bucket where the
        // source text file is stored.
        string srcBucket = "amzn-s3-demo-bucket";
        string srcTextFile = "source.txt";
```

```
            var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
            var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

            ShowText(srcText, destText);
        }

        /// <summary>
        /// Use the Amazon S3 TransferUtility to retrieve the text to translate
        /// from an object in an S3 bucket.
        /// </summary>
        /// <param name="srcBucket">The name of the S3 bucket where the
        /// text is stored.
        /// </param>
        /// <param name="srcTextFile">The key of the S3 object that
        /// contains the text to translate.</param>
        /// <returns>A string representing the source text.</returns>
        public static async Task<string> GetSourceTextAsync(string srcBucket, string
srcTextFile)
        {
            string srcText = string.Empty;

            var s3Client = new AmazonS3Client();
            TransferUtility utility = new TransferUtility(s3Client);

            using var stream = await utility.OpenStreamAsync(srcBucket,
srcTextFile);

            StreamReader file = new System.IO.StreamReader(stream);

            srcText = file.ReadToEnd();
            return srcText;
        }

        /// <summary>
        /// Use the Amazon Translate Service to translate the document from the
        /// source language to the specified destination language.
        /// </summary>
        /// <param name="client">The Amazon Translate Service client used to
        /// perform the translation.</param>
        /// <param name="srcLang">The language of the source text.</param>
        /// <param name="destLang">The destination language for the translated
        /// text.</param>
```

```
        /// <param name="text">A string representing the text to ranslate.</param>
        /// <returns>The text that has been translated to the destination
        /// language.</returns>
        public static async Task<string> TranslatingTextAsync(AmazonTranslateClient
 client, string srcLang, string destLang, string text)
        {
            var request = new TranslateTextRequest
            {
                SourceLanguageCode = srcLang,
                TargetLanguageCode = destLang,
                Text = text,
            };

            var response = await client.TranslateTextAsync(request);

            return response.TranslatedText;
        }

        /// <summary>
        /// Show the original text followed by the translated text.
        /// </summary>
        /// <param name="srcText">The original text to be translated.</param>
        /// <param name="destText">The translated text.</param>
        public static void ShowText(string srcText, string destText)
        {
            Console.WriteLine("Source text:");
            Console.WriteLine(srcText);
            Console.WriteLine();
            Console.WriteLine("Translated text:");
            Console.WriteLine(destText);
        }
    }
```

- API-Details finden Sie TranslateText in der AWS SDK für .NET API-Referenz.

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für .NET

Zeigt, wie man die .NET-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionalität verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für .NET

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

Codebeispiele für SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für .NET (v4) mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für .NET (v4) Entwicklerhandbuch — Weitere Informationen zur Verwendung von.NET mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- Aurora-Beispiele unter Verwendung SDK für .NET (v4)
- Beispiele für Auto Scaling mit SDK für .NET (v4)
- Beispiele für Amazon Bedrock mit SDK für .NET (v4)
- Beispiele für Amazon Bedrock Runtime mit SDK für .NET (v4)
- CloudFormation Beispiele unter Verwendung SDK für .NET (v4)
- CloudWatch Beispiele unter Verwendung SDK für .NET (v4)

- CloudWatch Protokolliert Beispiele mit SDK für .NET (v4)
- Beispiele für Amazon Cognito Identity Provider unter SDK für .NET (v4)
- AWS Control Tower Beispiele unter Verwendung SDK für .NET (v4)
- DynamoDB-Beispiele mit SDK für .NET (v4)
- Amazon EC2-Beispiele mit SDK für .NET (v4)
- Amazon ECS-Beispiele mit SDK für .NET (v4)
- AWS IoT Beispiele unter Verwendung SDK für .NET (v4)
- AWS IoT data Beispiele unter Verwendung SDK für .NET (v4)
- Amazon Redshift-Beispiele mit SDK für .NET (v4)
- Amazon S3-Beispiele mit SDK für .NET (v4)

Aurora-Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Aurora Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here for
 example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
```

```
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für .NET - API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using the
 DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group using
 the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
    4.  Get some parameters in the group using the DescribeDBClusterParametersAsync
 method.
    5.  Parse and display some parameters in the group.
    6.  Modify the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBClusterParameterGroupAsync method.
    7.  Get and display the updated parameters using the
 DescribeDBClusterParametersAsync method with a source of "user".
    8.  Get a list of allowed engine versions using the
 DescribeDBEngineVersionsAsync method.
    9.  Create an Aurora DB cluster that contains a MySql database and uses the
 parameter group.
        using the CreateDBClusterAsync method.
    10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
 method.
    11. Display and select from a list of instance classes available for the
 selected engine and version
        using the paginated DescribeOrderableDBInstanceOptions method.
```

```
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the CreateDBClusterSnapshotAsync
method.
   16. Wait for DB snapshot to be ready using the DescribeDBClusterSnapshotsAsync
method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonRDS>()
                   .AddTransient<AuroraWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder =>
       {
           builder.AddConsole();
       }).CreateLogger<AuroraScenario>();

       auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

       Console.WriteLine(sepBar);
       Console.WriteLine(
```

```
            "Welcome to the Amazon Aurora: get started with DB clusters example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName,
parameters);

            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
```

```
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
    /// Choose the Aurora DB parameter group family from a list of available
options.
    /// </summary>
    /// <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamilyAsync()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

        Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

        var parameterGroupFamilies =
            engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
        for (var i = 1; i <= parameterGroupFamilies.Count; i++)
        {
            var parameterGroupFamily = parameterGroupFamilies[i - 1];
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
```

```
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("2. Select an available DB parameter group family by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    /// <summary>
    /// Create and get information on a DB parameter group.
    /// </summary>
    /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new
 DB parameter group.</param>
    /// <returns>The new DBParameterGroup.</returns>
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParameterG

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }
```

```
    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</param>
    /// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>> DescribeParametersInGroupAsync(string
parameterGroupName, List<string>? parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);

        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
List<Parameter> parameters)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
group.");

        var parameters =
            await
auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName, "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
```

```
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
 dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family
 {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by entering
 a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
{clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
clusterIdentifier);
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }
```

```
        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption> ChooseDBInstanceClass(string
 engine, string engineVersion)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by entering
 a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
```

```
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateNewInstance(
        string clusterIdentifier,
        string engineName,
        string engineVersion,
        string instanceClass,
        string instanceIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"12. Create a new DB instance with identifier
 {instanceIdentifier}.");
        bool isInstanceReady = false;
        DBInstance newInstance;
        var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
        isInstanceReady = instances.FirstOrDefault(i =>
            i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
 "available";

        if (isInstanceReady)
        {
            Console.WriteLine("Instance already created.");
            newInstance = instances.First(i => i.DBInstanceIdentifier ==
 instanceIdentifier);
        }
        else
        {
            newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
                clusterIdentifier,
                instanceIdentifier,
                engineName,
                engineVersion,
```

```
                instanceClass
            );

            Console.WriteLine("13. Waiting for DB instance to be ready...");
            while (!isInstanceReady)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                instances = await
 auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
                isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
 "available";
                newInstance = instances.First();
            }
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
```

```
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
{cluster.DBClusterIdentifier}.");
        var snapshot = await auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");
```

```
        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
{newCluster.DBClusterIdentifier}.");
            await
auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
newCluster.DBClusterIdentifier);
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
{parameterGroup.DBClusterParameterGroupName}.");
            await
auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGroup
            Console.WriteLine("Parameter group deleted.");
```

```
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
```

```
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
```

```
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string groupName,
List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await _amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }


    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
```

```
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await _amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
```

```
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
```

```
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB cluster.</
param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
 dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
```

```
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name or
size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
```

```
            });

        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
```

```
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
 dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots

- DescribeDBClusters
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
```

```
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK für .NET -API-Referenz.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
```

```
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK für .NET API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
```

```
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK für .NET API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
```

```
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name or
size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für .NET -API-Referenz.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
```

```
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK für .NET -API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await _amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK für .NET API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für .NET -API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK für .NET API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK für .NET API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS SDK für .NET API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB cluster.</
param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
 dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für .NET - API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family name.</
param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für .NET API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für .NET -API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für .NET API-Referenz.

ModifyDBClusterParameterGroup

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string groupName,
List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
```

```
            DBClusterParameterGroupName = groupName,
        };

        var result = await _amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK für .NET API-Referenz.

Beispiele für Auto Scaling mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Auto Scaling Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        if (response.AutoScalingGroups == null || response.AutoScalingGroups.Count
 == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
            return;
        }
        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availability
        });
```

```
    }
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
```

```
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
 host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
 host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
```

```
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when deleting
the
        // launch template at the end of the application.
        var launchTemplateId = await ec2Wrapper.CreateLaunchTemplateAsync(imageId!,
instanceType!, launchTemplateName);

        // Confirm that the template was created by asking for a description of it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones[0].ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
```

```
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for the
group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);

        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size to
3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the current
state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");
```

```
        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups.Any())
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is {group.AutoScalingGroupARN}");
                var instances = group.Instances;
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is {instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred for
the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities.Any())
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are {activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
```

```
        if (metrics.Any())
        {
            Console.WriteLine($"Metrics collected for {groupName}:");
            metrics.ForEach(metric =>
            {
                Console.Write($"Metric name: {metric.MetricName}\t");
                Console.WriteLine($"Namespace: {metric.Namespace}");
            });
        }

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        if (dataPoints.Any())
        {
            Console.WriteLine("Details for the metrics collected:");
            dataPoints.ForEach(detail => { Console.WriteLine(detail); });
        }

        // Disable metrics collection.
        Console.WriteLine("Disabling the collection of metrics for {groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
```

```
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
 autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
```

```
    {
        Console.WriteLine("This code example performs the following operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the group.");
        Console.WriteLine(" 9. Lists the scaling activities that have occurred for
 the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
```

```
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);
```

```
        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Auto Scaling, Amazon EC2 und CloudWatch Aktionen.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
```

```
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };
        try
        {
            var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }
```

```
    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
 response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
 response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }
```

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
 null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }
```

```
        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groups = new List<AutoScalingGroup>();
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        if (response.AutoScalingGroups != null)
        {
            groups = response.AutoScalingGroups;
        }

        return groups;
    }


    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
```

```
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
```

```
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;

public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
```

```
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
```

```
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
```

```
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</returns>
    public async Task<List<Metric>> GetCloudWatchMetricsAsync(string groupName)
    {
        var metrics = new List<Metric>();
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };
```

```
        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);
        if (response.Metrics != null)
        {
            metrics = response.Metrics;
        }
        return metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var dataPoints = new List<Datapoint>();
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
```

```
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);
        if (response.Datapoints != null)
        {
            dataPoints = response.Datapoints;
        }

        return dataPoints;
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
        {
            availabilityZone,
        };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
    try
    {
```

```
            var response = await
 _amazonAutoScaling.CreateAutoScalingGroupAsync(request);
            Console.WriteLine($"{groupName} Auto Scaling Group created");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"{groupName} Auto Scaling Group already exists.");
            return true;
        }
    }
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK für .NET API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
```

```
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };

            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für .NET API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        var instanceDetails = new List<AutoScalingInstanceDetails>();
        if (groups != null)
        {
            groups.ForEach(group =>
            {
                if (group.AutoScalingGroupName == groupName && group.Instances !=
 null)
                {
                    group.Instances.ForEach(instance =>
                    {
                        instanceIds.Add(instance.InstanceId);
                    });
                }
            });

            var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
            {
                MaxRecords = 10,
                InstanceIds = instanceIds,
            };
```

```
            var response =
                await _amazonAutoScaling.DescribeAutoScalingInstancesAsync(
                    scalingGroupsRequest);
            if (response.AutoScalingInstances != null)
            {
                instanceDetails = response.AutoScalingInstances;
            }
        }

        return instanceDetails;
    }
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK für .NET API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Activity>> DescribeScalingActivitiesAsync(
        string groupName)
    {
        var activities = new List<Activity>();
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
```

```
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
_amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        if (response.Activities != null)
        {
            activities = response.Activities;
        }
        return activities;
    }
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK für .NET API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
```

```
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK für .NET API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };
```

```
        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK für .NET API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK für .NET API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK für .NET API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
```

```
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Bedrock mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Aktionen

Erste Schritte

Hallo Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.Bedrock;
using Amazon.Bedrock.Model;

namespace BedrockActions;

/// <summary>
/// This example shows how to list foundation models.
/// </summary>
internal class HelloBedrock
{
    /// <summary>
    /// Main method to call the ListFoundationModelsAsync method.
    /// </summary>
    /// <param name="args"> The command line arguments. </param>
    static async Task Main(string[] args)
    {
        // Specify a region endpoint where Amazon Bedrock is available. For a list
 of supported region see https://docs.aws.amazon.com/bedrock/latest/userguide/what-
is-bedrock.html#bedrock-regions
        AmazonBedrockClient bedrockClient = new(RegionEndpoint.USWest2);

        await ListFoundationModelsAsync(bedrockClient);

    }
```

```
    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
bedrockClient)
    {
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
ListFoundationModelsRequest()
            {
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }


    /// <summary>
    /// Write the foundation model summary to console.
    /// </summary>
    /// <param name="foundationModel"> The foundation model summary to write to
console. </param>
    private static void WriteToConsole(FoundationModelSummary foundationModel)
    {
```

```
        Console.WriteLine($"{foundationModel.ModelId}, Customization:
 {string.Join(", ", foundationModel.CustomizationsSupported)}, Stream:
 {foundationModel.ResponseStreamingSupported}, Input: {string.Join(",
 ", foundationModel.InputModalities)}, Output: {string.Join(", ",
 foundationModel.OutputModalities)}");
    }
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für .NET API-Referenz.

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Bedrock-Basismodelle.

```
    /// <summary>
    /// List foundation models.
    /// </summary>
    /// <param name="bedrockClient"> The Amazon Bedrock client. </param>
    private static async Task ListFoundationModelsAsync(AmazonBedrockClient
 bedrockClient)
    {
        Console.WriteLine("List foundation models with no filter.");

        try
        {
            var response = await bedrockClient.ListFoundationModelsAsync(new
 ListFoundationModelsRequest()
            {
```

```
            });

            if (response?.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                foreach (var fm in response.ModelSummaries)
                {
                    WriteToConsole(fm);
                }
            }
            else
            {
                Console.WriteLine("Something wrong happened");
            }
        }
        catch (AmazonBedrockException e)
        {
            Console.WriteLine(e.Message);
        }
    }
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für .NET API-Referenz.

Beispiele für Amazon Bedrock Runtime mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Anthropic Claude
- Cohere Command
- Meta Llama
- Mistral AI
- Stable Image Core

Erste Schritte

Hallo Amazon Bedrock Runtime

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock Runtime beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

namespace BedrockRuntimeActions;

/// <summary>
/// This example shows how to use the Converse API to send a text message
/// to Amazon Bedrock.
/// </summary>
internal class HelloBedrockRuntime
{
    static async Task Main(string[] args)
    {
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // You can change the region to match your setup.
        var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USWest2);

        // Set the model ID, e.g., Claude Haiku.
        // The "global." prefix enables cross-region inference, allowing the request
        // to be routed to the nearest available region for the specified model.
        var modelId = "global.anthropic.claude-haiku-4-5-20251001-v1:0";

        // Define the user message.
        var userMessage = "Hi. In a short paragraph, explain what you can do.";

        // Create a request with the model ID, the user message, and an inference
 configuration.
```

```
        var request = new ConverseRequest
        {
            ModelId = modelId,
            Messages = new List<Message>
            {
                new Message
                {
                    Role = ConversationRole.User,
                    Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
                }
            }
        };

        try
        {
            // Send the request to the Bedrock Runtime and wait for the response.
            var response = await client.ConverseAsync(request);

            // Extract and print the response text.
            string responseText = response?.Output?.Message?.Content?[0]?.Text ??
 "";
            Console.WriteLine(responseText);
        }
        catch (AmazonBedrockRuntimeException e)
        {
            Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason:
 {e.Message}");
            throw;
        }
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
```

```
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Converse API to send a text message to Anthropic Claude
```

```
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);
```

```
    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für .NET API-Referenz.

Cohere Command

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command.

using System;
using System.Collections.Generic;
using Amazon;
```

```
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
```

```
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Command R.
var modelId = "cohere.command-r-v1:0";

// Define the user message.
```

```
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für .NET API-Referenz.

Meta Llama

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Converse API to send a text message to Meta Llama.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
```

```
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Llama 3 8b Instruct.
var modelId = "meta.llama3-8b-instruct-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
```

```
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);

    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für .NET API-Referenz.

Mistral AI

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral.

using System;
using System.Collections.Generic;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
```

```
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseAsync(request);

    // Extract and print the response text.
    string responseText = response?.Output?.Message?.Content?[0]?.Text ?? "";
    Console.WriteLine(responseText);
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für .NET -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Mistral
```

```
// and print the response stream.

using System;
using System.Collections.Generic;
using System.Linq;
using Amazon;
using Amazon.BedrockRuntime;
using Amazon.BedrockRuntime.Model;

// Create a Bedrock Runtime client in the AWS Region you want to use.
var client = new AmazonBedrockRuntimeClient(RegionEndpoint.USEast1);

// Set the model ID, e.g., Mistral Large.
var modelId = "mistral.mistral-large-2402-v1:0";

// Define the user message.
var userMessage = "Describe the purpose of a 'hello world' program in one line.";

// Create a request with the model ID, the user message, and an inference
 configuration.
var request = new ConverseStreamRequest
{
    ModelId = modelId,
    Messages = new List<Message>
    {
        new Message
        {
            Role = ConversationRole.User,
            Content = new List<ContentBlock> { new ContentBlock { Text =
 userMessage } }
        }
    },
    InferenceConfig = new InferenceConfiguration()
    {
        MaxTokens = 512,
        Temperature = 0.5F,
        TopP = 0.9F
    }
};

try
{
    // Send the request to the Bedrock Runtime and wait for the result.
    var response = await client.ConverseStreamAsync(request);
```

```
    // Extract and print the streamed response text in real-time.
    foreach (var chunk in response.Stream.AsEnumerable())
    {
        if (chunk is ContentBlockDeltaEvent)
        {
            Console.Write((chunk as ContentBlockDeltaEvent).Delta.Text);
        }
    }
}
catch (AmazonBedrockRuntimeException e)
{
    Console.WriteLine($"ERROR: Can't invoke '{modelId}'. Reason: {e.Message}");
    throw;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für .NET API-Referenz.

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Image mit Stable Image Core.

```
        /// <summary>
        /// Asynchronously invokes the Stability.ai Stable Image Core model to run
 an inference based on the provided input.
        /// </summary>
```

```
        /// <param name="prompt">The prompt that describes the image Stability.ai
 Stable Image Core has to generate.</param>
        /// <returns>A base-64 encoded image generated by model</returns>
        /// <remarks>
        /// The different model providers have individual request and response
 formats.
        /// For the format, ranges, and default values for Stability.ai Stable Image
 Core, refer to:
        ///     https://docs.aws.amazon.com/bedrock/latest/userguide/model-
parameters-diffusion-stable-image-core-text-image-request-response.html
        /// </remarks>
        public static async Task<string?> InvokeStableImageCoreAsync(string prompt,
 int seed)
        {
            string stableImageCoreModelId = "stability.stable-image-core-v1:1";

            AmazonBedrockRuntimeClient client = new(RegionEndpoint.USWest2);

            string payload = new JsonObject()
            {
                { "prompt", prompt },
                { "aspect_ratio", "1:1" },
                { "seed", seed },
                { "output_format", "png" }
            }.ToJsonString();

            try
            {
                InvokeModelResponse response = await client.InvokeModelAsync(new
 InvokeModelRequest()
                {
                    ModelId = stableImageCoreModelId,
                    Body = AWSSDKUtils.GenerateMemoryStreamFromString(payload),
                    ContentType = "application/json",
                    Accept = "application/json"
                });

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    var results = JsonNode.ParseAsync(response.Body).Result?
["images"]?.AsArray();

                    return results?[0]?.GetValue<string>();
                }
```

```
                else
                {
                    Console.WriteLine("InvokeModelAsync failed with status code " +
response.HttpStatusCode);
                }
            }
            catch (AmazonBedrockRuntimeException e)
            {
                Console.WriteLine(e.Message);
            }
            return null;
        }
```

- API-Details finden Sie InvokeModel in der AWS SDK für .NET API-Referenz.

CloudFormation Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) with Aktionen ausführen und gängige Szenarien implementieren CloudFormation.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hallo CloudFormation

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudFormation beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.Runtime;

namespace CloudFormationActions;

public static class HelloCloudFormation
{
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    static async Task Main(string[] args)
    {
        // Create the CloudFormation client
        _amazonCloudFormation = new AmazonCloudFormationClient();
        Console.WriteLine($"\nIn Region:
 {_amazonCloudFormation.Config.RegionEndpoint}");

        // List the resources for each stack
        await ListResources();
    }

    /// <summary>
    /// Method to list stack resources and other information.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> ListResources()
    {
        try
        {
            Console.WriteLine("Getting CloudFormation stack information...");

            // Get all stacks using the stack paginator.
            var paginatorForDescribeStacks =
                _amazonCloudFormation.Paginators.DescribeStacks(
```

```
                    new DescribeStacksRequest());
            if (paginatorForDescribeStacks.Stacks != null)
            {
                await foreach (Stack stack in paginatorForDescribeStacks.Stacks)
                {
                    // Basic information for each stack
                    Console.WriteLine(
                        "\n-----------------------------------------------");
                    Console.WriteLine($"\nStack: {stack.StackName}");
                    Console.WriteLine($"  Status: {stack.StackStatus.Value}");
                    Console.WriteLine($"  Created: {stack.CreationTime}");

                    // The tags of each stack (etc.)
                    if (stack.Tags != null && stack.Tags.Count > 0)
                    {
                        Console.WriteLine("  Tags:");
                        foreach (Tag tag in stack.Tags)
                            Console.WriteLine($"    {tag.Key}, {tag.Value}");
                    }

                    // The resources of each stack
                    DescribeStackResourcesResponse responseDescribeResources =
                        await _amazonCloudFormation.DescribeStackResourcesAsync(
                            new DescribeStackResourcesRequest
                            {
                                StackName = stack.StackName
                            });
                    if (responseDescribeResources.StackResources != null &&
responseDescribeResources.StackResources.Count > 0)
                    {
                        Console.WriteLine("  Resources:");
                        foreach (StackResource resource in responseDescribeResources
                                     .StackResources)
                            Console.WriteLine(
                                $"    {resource.LogicalResourceId}:
{resource.ResourceStatus}");
                    }
                }
            }

            Console.WriteLine("\n-----------------------------------------------");
            return true;
        }
        catch (AmazonCloudFormationException ex)
```

```
        {
            Console.WriteLine("Unable to get stack information:\n" + ex.Message);
            return false;
        }
        catch (AmazonServiceException ex)
        {
            if (ex.Message.Contains("Unable to get IAM security credentials"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are usnig SSO, be sure to install" +
                                  " the AWSSDK.SSO and AWSSDK.SSOOIDC packages.");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
        catch (ArgumentNullException ex)
        {
            if (ex.Message.Contains("Options property cannot be empty: ClientName"))
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine("If you are using SSO, have you logged in?");
            }
            else
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
            }

            return false;
        }
    }
```

- API-Details finden Sie DescribeStackResources in der AWS SDK für .NET API-Referenz.

CloudWatch Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo CloudWatch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudWatch beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;
```

```
public static class HelloCloudWatch
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        if (response.Metrics != null)
        {
            foreach (var metric in response.Metrics.Take(5))
            {
                Console.WriteLine($"\tMetric: {metric.MetricName}");
                Console.WriteLine($"\tNamespace: {metric.Namespace}");
                Console.WriteLine(
                    $"\tDimensions: {string.Join(", ", metric.Dimensions.Select(m =>
 $"{m.Name}:{m.Value}"))}");
                Console.WriteLine();
            }
        }
    }
}
```

- API-Details finden Sie ListMetrics in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listet CloudWatch Namespaces und Metriken auf.
- Rufen Sie Statistiken für eine Metrik und die geschätzte Fakturierung ab.
- Erstellen und aktualisieren Sie ein Dashboard.
- Erstellen Sie eine Metrik und fügen Sie ihr Daten hinzu.
- Erstellen und lösen Sie einen Alarm aus und zeigen Sie dann den Alarmverlauf an.
- Fügen Sie einen Anomaliedetektor hinzu.
- Ermitteln Sie ein Metrik-Image, dann bereinigen Sie die Ressourcen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
```

```
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
   */

   private static ILogger logger = null!;
   private static CloudWatchWrapper _cloudWatchWrapper = null!;
   private static IConfiguration _configuration = null!;
   private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
   private static SingleMetricAnomalyDetector? anomalyDetector = null!;

   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonCloudWatch>()
           .AddTransient<CloudWatchWrapper>()
       )
       .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
```

```
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CloudWatchScenario>();

        _cloudWatchWrapper = host.Services.GetRequiredService<CloudWatchWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var selectedNamespace = await SelectNamespace();
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace, selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
```

```
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }

        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");

        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
```

```
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d =>
d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string metricNamespace,
Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
```

```
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };

        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for {metric}
in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the past
seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric {customMetricName}:
\n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes in
the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
```

```
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
```

```
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
```

```
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm {alarmName}?
(y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
```

```
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");

        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for {alarm.MetricName}
{alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];

        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
```

```
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            if (metricData[i].Values != null)
            {
                for (int j = 0; j < metricData[i].Values.Count; j++)
                {
                    Console.WriteLine(
                        $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
                }
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricDataForAlarm()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger an
alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric {customMetricName}:
\n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");
```

```
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10 retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
```

```
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
```

```
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state {detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream, "MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;
```

```
        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");

            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }
```

```
    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

Wrapper-Methoden, die vom Szenario für CloudWatch Aktionen verwendet werden.

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)

    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
```

```
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter> { filter } :
null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions, int
days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
```

```
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard was
 created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
 List<DashboardValidationMessage>();
    }


    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
```

```
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });
```

```
        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string metricNamespace,
string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString = JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
```

```
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData, bool
 useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
```

```
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await _amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
```

```
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota has
already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to append
to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
```

```
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms = _amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }

    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string alarmName,
int historyDays)
```

```
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
```

```
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
```

```
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Beispiel für settings.json-Werte für das Szenario.

```
{
  "dashboardName": "example-new-dashboard",
  "exampleAlarmName": "example-metric-alarm",
  "accountId": "1234567890",
  "region": "us-east-1",
  "emailTopic": "Default_CloudWatch_Alarms_Topic",
  "customMetricNamespace": "example-namespace",
  "customMetricName": "example-custom-metric",
  "dashboardExampleBody": {
    "widgets": [
      {
        "height": 6,
        "width": 6,
        "y": 0,
        "x": 0,
        "type": "text",
        "properties": {
          "markdown": "# Code Example Dashboard \nThis dashboard was created by
 example code.\n"
        }
      },
      {
```

```
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 6,
        "type": "metric",
        "properties": {
          "metrics": [
            [
              "AWS/Billing",
              "EstimatedCharges",
              "Currency",
              "USD",
              { "region": "us-east-1" }
            ]
          ],
          "view": "timeSeries",
          "region": "us-east-1",
          "stat": "Maximum",
          "period": 86400,
          "yAxis": {
            "left": {
              "min": 0,
              "max": 100
            }
          },
          "stacked": false,
          "title": "Estimated Billing",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      },
      {
        "height": 8,
        "width": 8,
        "y": 0,
        "x": 14,
        "type": "metric",
        "properties": {
          "metrics": [
            [ "AWS/Usage", "CallCount", "Type", "API", "Resource", "ListMetrics",
"Service", "CloudWatch", "Class", "None" ],
            [ "...", "GetMetricStatistics", ".", ".", ".", "." ],
```

```
            [ "...", "GetMetricData", ".", ".", ".", "." ],
            [ "...", "PutDashboard", ".", ".", ".", "." ],
            [ "...", "PutMetricData", ".", ".", ".", "." ]
          ],
          "view": "timeSeries",
          "yAxis": {
            "left": {
              "min": 0,
              "max": 200
            }
          },
          "stacked": false,
          "region": "us-east-1",
          "stat": "Sum",
          "period": 300,
          "title": "CloudWatch Usage",
          "setPeriodToTimeRange": false,
          "liveData": true,
          "sparkline": true,
          "trend": true
        }
      }
    ]
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics

- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAlarms`

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- API-Details finden Sie DeleteAlarms in der AWS SDK für .NET API-Referenz.

`DeleteAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAnomalyDetector`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteAnomalyDetector in der AWS SDK für .NET API-Referenz.

`DeleteDashboards`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDashboards`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DeleteDashboards in der AWS SDK für .NET API-Referenz.

`DescribeAlarmHistory`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmHistory`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string alarmName,
int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- API-Details finden Sie DescribeAlarmHistory in der AWS SDK für .NET API-Referenz.

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms = _amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- API-Details finden Sie DescribeAlarms in der AWS SDK für .NET API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
```

```
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms ?? new List<MetricAlarm>();
    }
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK für .NET API-Referenz.

`DescribeAnomalyDetectors`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAnomalyDetectors`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
```

```
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }
```

- API-Details finden Sie DescribeAnomalyDetectors in der AWS SDK für .NET API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
```

```
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK für .NET API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie EnableAlarmActions in der AWS SDK für .NET API-Referenz.

`GetDashboard`

Das folgende Codebeispiel zeigt die Verwendung`GetDashboard`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- API-Details finden Sie GetDashboard in der AWS SDK für .NET API-Referenz.

`GetMetricData`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricData`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData, bool
 useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        // Add the plus sign for positive offsets.
        timeZoneString = timeZoneString.StartsWith('-') ? timeZoneString : "+" +
 timeZoneString;
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
```

```
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        if (paginatedMetricData.MetricDataResults != null)
        {
            await foreach (var data in paginatedMetricData.MetricDataResults)
            {
                metricData.Add(data);
            }
        }

        return metricData;
    }
```

- API-Details finden Sie GetMetricData in der AWS SDK für .NET API-Referenz.

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricStatistics`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the past
seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
```

```
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions, int
days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints ?? new List<Datapoint>();
    }
```

- API-Details finden Sie GetMetricStatistics in der AWS SDK für .NET API-Referenz.

GetMetricWidgetImage

Das folgende Codebeispiel zeigt die VerwendungGetMetricWidgetImage.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string metricNamespace,
string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString = JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
```

```
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- API-Details finden Sie GetMetricWidgetImage in der AWS SDK für .NET API-Referenz.

`ListDashboards`

Das folgende Codebeispiel zeigt die Verwendung`ListDashboards`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
```

```
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }
```

- API-Details finden Sie ListDashboards in der AWS SDK für .NET API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
```

```
                Dimensions = filter != null ? new List<DimensionFilter> { filter } :
null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }
```

- API-Details finden Sie ListMetrics in der AWS SDK für .NET API-Referenz.

`PutAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`PutAnomalyDetector`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });
```

```
        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie PutAnomalyDetector in der AWS SDK für .NET API-Referenz.

`PutDashboard`

Das folgende Codebeispiel zeigt die Verwendung`PutDashboard`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
```

```
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard was
 created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
```

```
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages ?? new
List<DashboardValidationMessage>();
    }
```

- API-Details finden Sie PutDashboard in der AWS SDK für .NET API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
```

```
        try
        {
            var putEmailAlarmResponse = await _amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota has
already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to append
to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
```

```
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK für .NET API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes in
the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
```

```
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace, customData);
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API-Details finden Sie PutMetricData in der AWS SDK für .NET API-Referenz.

CloudWatch Protokolliert Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`GetQueryResults`

Das folgende Codebeispiel zeigt die Verwendung`GetQueryResults`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the results of a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="queryId">The ID of the query.</param>
    /// <returns>The query results response.</returns>
    public async Task<GetQueryResultsResponse?> GetQueryResultsAsync(string queryId)
    {
        try
        {
            var request = new GetQueryResultsRequest
            {
                QueryId = queryId
            };

            var response = await
_amazonCloudWatchLogs.GetQueryResultsAsync(request);
```

```
            return response;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Query not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while getting query results:
 {ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie GetQueryResults in der AWS SDK für .NET API-Referenz.

StartQuery

Das folgende Codebeispiel zeigt die Verwendung`StartQuery`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Starts a CloudWatch Logs Insights query.
    /// </summary>
    /// <param name="logGroupName">The name of the log group to query.</param>
    /// <param name="queryString">The CloudWatch Logs Insights query string.</param>
    /// <param name="startTime">The start time for the query (seconds since
 epoch).</param>
    /// <param name="endTime">The end time for the query (seconds since epoch).</
param>
    /// <param name="limit">The maximum number of results to return.</param>
    /// <returns>The query ID if successful, null otherwise.</returns>
```

```
    public async Task<string?> StartQueryAsync(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit = 10000)
    {
        try
        {
            var request = new StartQueryRequest
            {
                LogGroupName = logGroupName,
                QueryString = queryString,
                StartTime = startTime,
                EndTime = endTime,
                Limit = limit
            };

            var response = await _amazonCloudWatchLogs.StartQueryAsync(request);
            return response.QueryId;
        }
        catch (InvalidParameterException ex)
        {
            _logger.LogError($"Invalid parameter for query: {ex.Message}");
            return null;
        }
        catch (ResourceNotFoundException ex)
        {
            _logger.LogError($"Log group not found: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while starting query:
{ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie StartQuery in der AWS SDK für .NET API-Referenz.

Szenarien

Ausführen einer umfangreichen Abfrage

Das folgende Codebeispiel zeigt, wie CloudWatch Logs verwendet werden, um mehr als 10.000 Datensätze abzufragen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Hauptworkflow, der das Szenario mit großen Abfragen veranschaulicht.

```
using System.Diagnostics;
using System.Text.RegularExpressions;
using Amazon.CloudFormation;
using Amazon.CloudFormation.Model;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
using CloudWatchLogsActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace CloudWatchLogsScenario;

public class LargeQueryWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
    This .NET code example performs the following tasks for the CloudWatch Logs
 Large Query workflow:

    1. Prepare the Application:
       - Prompt the user to deploy CloudFormation stack and generate sample logs.
       - Deploy the CloudFormation template for resource creation.
       - Generate 50,000 sample log entries using CloudWatch Logs API.
       - Wait 5 minutes for logs to be fully ingested.
```

```
    2. Execute Large Query:
       - Perform recursive queries to retrieve all logs using binary search.
       - Display progress for each query executed.
       - Show total execution time and logs found.

    3. Clean up:
       - Prompt the user to delete the CloudFormation stack and all resources.
       - Destroy the CloudFormation stack and wait until removed.
    */

    public static ILogger<LargeQueryWorkflow> _logger = null!;
    public static CloudWatchLogsWrapper _wrapper = null!;
    public static IAmazonCloudFormation _amazonCloudFormation = null!;

    private static string _logGroupName = "/workflows/cloudwatch-logs/large-query";
    private static string _logStreamName = "stream1";
    private static long _queryStartDate;
    private static long _queryEndDate;

    public static bool _interactive = true;
    public static string _stackName = "CloudWatchLargeQueryStack";
    private static string _stackResourcePath = "../../../../../../../scenarios/
features/cloudwatch_logs_large_query/resources/stack.yaml";

    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter("Microsoft", LogLevel.Information))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatchLogs>()
                    .AddAWSService<IAmazonCloudFormation>()
                    .AddTransient<CloudWatchLogsWrapper>()
            )
            .Build();

        if (_interactive)
        {
            _logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
                .CreateLogger<LargeQueryWorkflow>();

            _wrapper = host.Services.GetRequiredService<CloudWatchLogsWrapper>();
```

```
            _amazonCloudFormation =
host.Services.GetRequiredService<IAmazonCloudFormation>();
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to retrieve
more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Runs the scenario workflow. Used for testing.
```

```
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the CloudWatch Logs Large Query Scenario.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("This scenario demonstrates how to perform large-scale
queries on");
        Console.WriteLine("CloudWatch Logs using recursive binary search to retrieve
more than");
        Console.WriteLine("the 10,000 result limit.");
        Console.WriteLine();

        try
        {
            Console.WriteLine(new string('-', 80));
            var prepareSuccess = await PrepareApplication();
            Console.WriteLine(new string('-', 80));

            if (prepareSuccess)
            {
                Console.WriteLine(new string('-', 80));
                await ExecuteLargeQuery();
                Console.WriteLine(new string('-', 80));
            }

            Console.WriteLine(new string('-', 80));
            await Cleanup();
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "There was a problem with the scenario, initiating
cleanup...");
            _interactive = false;
            await Cleanup();
        }

        Console.WriteLine("CloudWatch Logs Large Query scenario completed.");
    }

    /// <summary>
    /// Prepares the application by creating the necessary resources.
    /// </summary>
```

```
    /// <returns>True if the application was prepared successfully.</returns>
    public static async Task<bool> PrepareApplication()
    {
        Console.WriteLine("Preparing the application...");
        Console.WriteLine();

        try
        {
            var deployStack = !_interactive || GetYesNoResponse(
                "Would you like to deploy the CloudFormation stack and generate
sample logs? (y/n) ");

            if (deployStack)
            {
                if (_interactive)
                {
                    Console.Write(
                        $"Enter a path for the CloudFormation stack resource .yaml
file (or press Enter for default '{_stackResourcePath}'): ");
                    string? inputPath = Console.ReadLine();
                    if (!string.IsNullOrWhiteSpace(inputPath))
                    {
                        _stackResourcePath = inputPath;
                    }
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName);

                if (deploySuccess)
                {
                    Console.WriteLine();
                    Console.WriteLine("Generating 50,000 sample log entries...");
                    var generateSuccess = await GenerateSampleLogs();

                    if (generateSuccess)
                    {
                        Console.WriteLine();
                        Console.WriteLine("Sample logs created. Waiting 5 minutes
for logs to be fully ingested...");
                        await WaitWithCountdown(300);

                        Console.WriteLine("Application preparation complete.");
```

```
                        return true;
                    }
                }
            }
            else
            {
                _logGroupName = PromptUserForInput("Enter the log group name ",
_logGroupName);
                _logStreamName = PromptUserForInput("Enter the log stream name ",
_logStreamName);

                var startDateMs = PromptUserForLong("Enter the query start date
(milliseconds since epoch): ");
                var endDateMs = PromptUserForLong("Enter the query end date
(milliseconds since epoch): ");

                _queryStartDate = startDateMs / 1000;
                _queryEndDate = endDateMs / 1000;

                Console.WriteLine("Application preparation complete.");
                return true;
            }
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while preparing the
application.");
        }

        Console.WriteLine("Application preparation failed.");
        return false;
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
        {
```

```
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath)
            };

            var response = await _amazonCloudFormation.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await WaitForStackCompletion(response.StackId);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created successfully.");
                    return true;
                }
                else
                {
                    _logger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                _logger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            _logger.LogWarning($"CloudFormation stack '{stackName}' already exists.
Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName);
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
                {
                    return false;
                }
            }
```

```
            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for CloudFormation stack creation to
complete.");
        return false;
    }

    /// <summary>
    /// Generates sample logs directly using CloudWatch Logs API.
    /// Creates 50,000 log entries spanning 5 minutes.
    /// </summary>
    /// <returns>True if logs were generated successfully.</returns>
    private static async Task<bool> GenerateSampleLogs()
    {
        const int totalEntries = 50000;
        const int entriesPerBatch = 10000;
        const int fiveMinutesMs = 5 * 60 * 1000;

        try
        {
            // Calculate timestamps
            var startTimeMs = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds();
            var timestampIncrement = fiveMinutesMs / totalEntries;

            Console.WriteLine($"Generating {totalEntries} log entries...");

            var entryCount = 0;
            var currentTimestamp = startTimeMs;
            var numBatches = totalEntries / entriesPerBatch;

            // Generate and upload logs in batches
            for (int batchNum = 0; batchNum < numBatches; batchNum++)
            {
                var logEvents = new List<InputLogEvent>();

                for (int i = 0; i < entriesPerBatch; i++)
                {
                    logEvents.Add(new InputLogEvent
                    {
```

```
                        Timestamp =
 DateTimeOffset.FromUnixTimeMilliseconds(currentTimestamp).UtcDateTime,
                        Message = $"Entry {entryCount}"
                    });

                    entryCount++;
                    currentTimestamp += timestampIncrement;
                }

                // Upload batch
                var success = await _wrapper.PutLogEventsAsync(_logGroupName,
 _logStreamName, logEvents);
                if (!success)
                {
                    _logger.LogError($"Failed to upload batch {batchNum + 1}/
{numBatches}");
                    return false;
                }

                Console.WriteLine($"Uploaded batch {batchNum + 1}/{numBatches}");
            }

            // Set query date range (convert milliseconds to seconds for query API)
            _queryStartDate = startTimeMs / 1000;
            _queryEndDate = (currentTimestamp - timestampIncrement) / 1000;

            Console.WriteLine($"Query start date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryStartDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Query end date:
 {DateTimeOffset.FromUnixTimeSeconds(_queryEndDate):yyyy-MM-ddTHH:mm:ss.fffZ}");
            Console.WriteLine($"Successfully uploaded {totalEntries} log entries");

            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while generating sample logs.");
            return false;
        }
    }

    /// <summary>
    /// Executes the large query workflow.
    /// </summary>
```

```
    public static async Task ExecuteLargeQuery()
    {
        Console.WriteLine("Starting recursive query to retrieve all logs...");
        Console.WriteLine();

        var queryLimit = PromptUserForInteger("Enter the query limit (max 10000) ",
10000);
        if (queryLimit > 10000) queryLimit = 10000;

        var queryString = "fields @timestamp, @message | sort @timestamp asc";

        var stopwatch = Stopwatch.StartNew();
        var allResults = await PerformLargeQuery(_logGroupName, queryString,
_queryStartDate, _queryEndDate, queryLimit);
        stopwatch.Stop();

        Console.WriteLine();
        Console.WriteLine($"Queries finished in {stopwatch.Elapsed.TotalSeconds:F3}
seconds.");
        Console.WriteLine($"Total logs found: {allResults.Count}");

        // Check for duplicates
        Console.WriteLine();
        Console.WriteLine("Checking for duplicate logs...");
        var duplicates = FindDuplicateLogs(allResults);
        if (duplicates.Count > 0)
        {
            Console.WriteLine($"WARNING: Found {duplicates.Count} duplicate log
entries!");
            Console.WriteLine("Duplicate entries (showing first 10):");
            foreach (var dup in duplicates.Take(10))
            {
                Console.WriteLine($"  [{dup.Timestamp}] {dup.Message} (appears
{dup.Count} times)");
            }

            var uniqueCount = allResults.Count - duplicates.Sum(d => d.Count - 1);
            Console.WriteLine($"Unique logs: {uniqueCount}");
        }
        else
        {
            Console.WriteLine("No duplicates found. All logs are unique.");
        }
        Console.WriteLine();
```

```
        var viewSample = !_interactive || GetYesNoResponse("Would you like to see a
sample of the logs? (y/n) ");
        if (viewSample)
        {
            Console.WriteLine();
            Console.WriteLine($"Sample logs (first 10 of {allResults.Count}):");
            for (int i = 0; i < Math.Min(10, allResults.Count); i++)
            {
                var timestamp = allResults[i].Find(f => f.Field ==
"@timestamp")?.Value ?? "N/A";
                var message = allResults[i].Find(f => f.Field ==
"@message")?.Value ?? "N/A";
                Console.WriteLine($"[{timestamp}] {message}");
            }
        }
    }

    /// <summary>
    /// Performs a large query using recursive binary search.
    /// </summary>
    private static async Task<List<List<ResultField>>> PerformLargeQuery(
        string logGroupName,
        string queryString,
        long startTime,
        long endTime,
        int limit)
    {
        var queryId = await _wrapper.StartQueryAsync(logGroupName, queryString,
startTime, endTime, limit);
        if (queryId == null)
        {
            return new List<List<ResultField>>();
        }

        var results = await PollQueryResults(queryId);
        if (results == null || results.Count == 0)
        {
            return new List<List<ResultField>>();
        }

        var startDate =
DateTimeOffset.FromUnixTimeSeconds(startTime).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");
```

```
        var endDate = DateTimeOffset.FromUnixTimeSeconds(endTime).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        Console.WriteLine($"Query date range: {startDate} ({startTime}s) to
 {endDate} ({endTime}s). Found {results.Count} logs.");

        if (results.Count < limit)
        {
            Console.WriteLine($"  -> Returning {results.Count} logs (less than limit
 of {limit})");
            return results;
        }

        Console.WriteLine($"  -> Hit limit of {limit}. Need to split and recurse.");

        // Get the timestamp of the last log (sorted to find the actual last one)
        var lastLogTimestamp = GetLastLogTimestamp(results);
        if (lastLogTimestamp == null)
        {
            Console.WriteLine($"  -> No timestamp found in results. Returning
 {results.Count} logs.");
            return results;
        }

        Console.WriteLine($"  -> Last log timestamp: {lastLogTimestamp}");

        // Parse the timestamp and add 1 millisecond to avoid querying the same log
 again
        var lastLogDate = DateTimeOffset.Parse(lastLogTimestamp + " +0000");
        Console.WriteLine($"  -> Last log as DateTimeOffset: {lastLogDate:yyyy-MM-
ddTHH:mm:ss.fffZ} ({lastLogDate.ToUnixTimeSeconds()}s)");

        var offsetLastLogDate = lastLogDate.AddMilliseconds(1);
        Console.WriteLine($"  -> Offset timestamp (last
 + 1ms): {offsetLastLogDate:yyyy-MM-ddTHH:mm:ss.fffZ}
 ({offsetLastLogDate.ToUnixTimeSeconds()}s)");

        // Convert to seconds, but round UP to the next second to avoid overlapping
 with logs in the same second
        // This ensures we don't re-query logs that share the same second as the
 last log
        var offsetLastLogTime = offsetLastLogDate.ToUnixTimeSeconds();
        if (offsetLastLogDate.Millisecond > 0)
        {
            offsetLastLogTime++; // Move to the next full second
```

```
            Console.WriteLine($"  -> Adjusted to next full second:
 {offsetLastLogTime}s ({DateTimeOffset.FromUnixTimeSeconds(offsetLastLogTime):yyyy-
MM-ddTHH:mm:ss.fffZ})");
        }

        Console.WriteLine($"  -> Comparing: offsetLastLogTime={offsetLastLogTime}s
 vs endTime={endTime}s");
        Console.WriteLine($"  -> End time as date:
 {DateTimeOffset.FromUnixTimeSeconds(endTime):yyyy-MM-ddTHH:mm:ss.fffZ}");

        // Check if there's any time range left to query
        if (offsetLastLogTime >= endTime)
        {
            Console.WriteLine($"  -> No time range left to query. Offset time
 ({offsetLastLogTime}s) >= end time ({endTime}s)");
            return results;
        }

        // Split the remaining date range in half
        var (range1Start, range1End, range2Start, range2End) =
 SplitDateRange(offsetLastLogTime, endTime);

        var range1StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range1Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range1EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range1End).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");
        var range2StartDate =
 DateTimeOffset.FromUnixTimeSeconds(range2Start).ToString("yyyy-MM-
ddTHH:mm:ss.fffZ");
        var range2EndDate =
 DateTimeOffset.FromUnixTimeSeconds(range2End).ToString("yyyy-MM-ddTHH:mm:ss.fffZ");

        Console.WriteLine($"  -> Splitting remaining range:");
        Console.WriteLine($"     Range 1: {range1StartDate} ({range1Start}s) to
 {range1EndDate} ({range1End}s)");
        Console.WriteLine($"     Range 2: {range2StartDate} ({range2Start}s) to
 {range2EndDate} ({range2End}s)");

        // Query both halves recursively
        Console.WriteLine($"  -> Querying range 1...");
        var results1 = await PerformLargeQuery(logGroupName, queryString,
 range1Start, range1End, limit);
        Console.WriteLine($"  -> Range 1 returned {results1.Count} logs");
```

```
        Console.WriteLine($"  -> Querying range 2...");
        var results2 = await PerformLargeQuery(logGroupName, queryString,
range2Start, range2End, limit);
        Console.WriteLine($"  -> Range 2 returned {results2.Count} logs");

        // Combine all results
        var allResults = new List<List<ResultField>>(results);
        allResults.AddRange(results1);
        allResults.AddRange(results2);

        Console.WriteLine($"  -> Combined total: {allResults.Count} logs
({results.Count} + {results1.Count} + {results2.Count})");

        return allResults;
    }

    /// <summary>
    /// Gets the timestamp string of the most recent log from a list of logs.
    /// Sorts timestamps to find the actual last one.
    /// </summary>
    private static string? GetLastLogTimestamp(List<List<ResultField>> logs)
    {
        var timestamps = logs
            .Select(log => log.Find(f => f.Field == "@timestamp")?.Value)
            .Where(t => !string.IsNullOrEmpty(t))
            .OrderBy(t => t)
            .ToList();

        if (timestamps.Count == 0)
        {
            return null;
        }

        return timestamps[timestamps.Count - 1];
    }

    /// <summary>
    /// Splits a date range in half.
    /// Range 2 starts at midpoint + 1 second to avoid overlap.
    /// </summary>
    private static (long range1Start, long range1End, long range2Start, long
range2End) SplitDateRange(long startTime, long endTime)
    {
```

```
        var midpoint = startTime + (endTime - startTime) / 2;
        // Range 2 starts at midpoint + 1 to avoid querying the same second twice
        return (startTime, midpoint, midpoint + 1, endTime);
    }

    /// <summary>
    /// Polls for query results until complete.
    /// </summary>
    private static async Task<List<List<ResultField>>?> PollQueryResults(string
queryId)
    {
        int retryCount = 0;
        const int maxRetries = 60;
        const int retryDelay = 1000;

        while (retryCount < maxRetries)
        {
            var response = await _wrapper.GetQueryResultsAsync(queryId);
            if (response == null)
            {
                return null;
            }

            if (response.Status == QueryStatus.Complete)
            {
                return response.Results;
            }

            if (response.Status == QueryStatus.Failed ||
                response.Status == QueryStatus.Cancelled ||
                response.Status == QueryStatus.Timeout ||
                response.Status == QueryStatus.Unknown)
            {
                _logger.LogError($"Query failed with status: {response.Status}");
                return null;
            }

            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError("Timed out waiting for query results.");
        return null;
    }
```

```
    /// <summary>
    /// Cleans up the resources created during the scenario.
    /// </summary>
    public static async Task<bool> Cleanup()
    {
        var cleanup = !_interactive || GetYesNoResponse(
            "Do you want to delete the CloudFormation stack and all resources? (y/n)
");

        if (cleanup)
        {
            try
            {
                var stackDeleteSuccess = await DeleteCloudFormationStack(_stackName,
false);
                return stackDeleteSuccess;
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "An error occurred while cleaning up the
resources.");
                return false;
            }
        }

        Console.WriteLine($"Resources will remain. Stack name: {_stackName}, Log
group: {_logGroupName}");
        _logger.LogInformation("CloudWatch Logs Large Query scenario is complete.");
        return true;
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    private static async Task<bool> DeleteCloudFormationStack(string stackName, bool
forceDelete)
    {
        var request = new DeleteStackRequest
        {
            StackName = stackName,
        };

        if (forceDelete)
```

```
        {
            request.DeletionMode = DeletionMode.FORCE_DELETE_STACK;
        }

        await _amazonCloudFormation.DeleteStackAsync(request);
        Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

        bool stackDeleted = await WaitForStackDeletion(stackName, forceDelete);

        if (stackDeleted)
        {
            Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
            return true;
        }
        else
        {
            _logger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    private static async Task<bool> WaitForStackDeletion(string stackName, bool
forceDelete)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
```

```
                var describeStacksResponse = await
_amazonCloudFormation.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
                    return true;
                }

                if (!forceDelete && describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_FAILED)
                {
                    return await DeleteCloudFormationStack(stackName, true);
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        _logger.LogError($"Timed out waiting for CloudFormation stack '{stackName}'
to be deleted.");
        return false;
    }

    /// <summary>
    /// Waits with a countdown display.
    /// </summary>
    private static async Task WaitWithCountdown(int seconds)
    {
        for (int i = seconds; i > 0; i--)
        {
            Console.Write($"\rWaiting: {i} seconds remaining...  ");
            await Task.Delay(1000);
        }
        Console.WriteLine("\rWait complete.                      ");
```

```
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (_interactive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press Enter
for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
                var regex = "[a-zA-Z][-a-zA-Z0-9]*";
                if (!Regex.IsMatch(input, regex))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
{_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }

    /// <summary>
    /// Prompts the user for input with a default value.
    /// </summary>
    private static string PromptUserForInput(string prompt, string defaultValue)
    {
```

```
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'): ");
            string? input = Console.ReadLine();
            return string.IsNullOrWhiteSpace(input) ? defaultValue : input;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for an integer value.
    /// </summary>
    private static int PromptUserForInteger(string prompt, int defaultValue)
    {
        if (_interactive)
        {
            Console.Write($"{prompt}(press Enter for default '{defaultValue}'): ");
            string? input = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(input) || !int.TryParse(input, out var
result))
            {
                return defaultValue;
            }
            return result;
        }
        return defaultValue;
    }

    /// <summary>
    /// Prompts the user for a long value.
    /// </summary>
    private static long PromptUserForLong(string prompt)
    {
        if (_interactive)
        {
            Console.Write(prompt);
            string? input = Console.ReadLine();
            if (long.TryParse(input, out var result))
            {
                return result;
            }
        }
        return 0;
    }
```

```
    /// <summary>
    /// Finds duplicate log entries based on timestamp and message.
    /// </summary>
    private static List<(string Timestamp, string Message, int Count)>
 FindDuplicateLogs(List<List<ResultField>> logs)
    {
        var logSignatures = new Dictionary<string, int>();

        foreach (var log in logs)
        {
            var timestamp = log.Find(f => f.Field == "@timestamp")?.Value ?? "";
            var message = log.Find(f => f.Field == "@message")?.Value ?? "";
            var signature = $"{timestamp}|{message}";

            if (logSignatures.ContainsKey(signature))
            {
                logSignatures[signature]++;
            }
            else
            {
                logSignatures[signature] = 1;
            }
        }

        return logSignatures
            .Where(kvp => kvp.Value > 1)
            .Select(kvp =>
            {
                var parts = kvp.Key.Split('|');
                return (Timestamp: parts[0], Message: parts[1], Count: kvp.Value);
            })
            .OrderByDescending(x => x.Count)
            .ToList();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - GetQueryResults
  - StartQuery

Beispiele für Amazon Cognito Identity Provider unter SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon Cognito Identity Provider Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListUserPools`

Das folgende Codebeispiel zeigt die Verwendung`ListUserPools`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());
```

```
        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- API-Details finden Sie ListUserPools in der AWS SDK für .NET API-Referenz.

AWS Control Tower Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) with Aktionen ausführen und gängige Szenarien implementieren AWS Control Tower.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Control Tower

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Control Tower beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;
using LogLevel = Microsoft.Extensions.Logging.LogLevel;

namespace ControlTowerActions;

/// <summary>
/// A class that introduces the AWS Control Tower by listing the
/// available baselines for the account.
/// </summary>
public class HelloControlTower
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Control Tower.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
            )
            .Build();
```

```
        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloControlTower>();

        var amazonClient = host.Services.GetRequiredService<IAmazonControlTower>();

        Console.Clear();
        Console.WriteLine("Hello, AWS Control Tower! Let's list available
 baselines:");
        Console.WriteLine();

        var baselines = new List<BaselineSummary>();

        try
        {
            var baselinesPaginator = amazonClient.Paginators.ListBaselines(new
 ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            Console.WriteLine($"{baselines.Count} baseline(s) retrieved.");
            foreach (var baseline in baselines)
            {
                Console.WriteLine($"\t{baseline.Name}");
            }
        }
        catch (Amazon.ControlTower.Model.AccessDeniedException)
        {
            Console.WriteLine("Access denied. Please ensure you have the necessary
 permissions.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
}
```

- API-Details finden Sie ListBaselines in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listen Sie Landing Zones auf.
- Listen Sie Baselines auf, aktivieren Sie sie, rufen Sie sie ab, setzen Sie sie zurück und deaktivieren Sie sie.
- Listen Sie Steuerelemente auf, aktivieren Sie sie, rufen Sie sie ab und deaktivieren Sie sie.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Control Tower Funktionen demonstriert.

```
using Amazon.ControlCatalog;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using Amazon.Organizations;
using Amazon.Organizations.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using ControlTowerActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace ControlTowerBasics;

/// <summary>
/// Scenario class for AWS Control Tower basics.
/// </summary>
public class ControlTowerBasics
{
```

```
    public static bool isInteractive = true;
    public static ILogger logger = null!;
    public static IAmazonOrganizations? orgClient = null;
    public static IAmazonSecurityTokenService? stsClient = null;
    public static ControlTowerWrapper? wrapper = null;
    private static string? ouArn;
    private static bool useLandingZone = false;

    /// <summary>
    /// Main entry point for the AWS Control Tower basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonControlTower>()
                .AddAWSService<IAmazonControlCatalog>()
                .AddAWSService<IAmazonOrganizations>()
                .AddAWSService<IAmazonSecurityTokenService>()
                .AddTransient<ControlTowerWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<ControlTowerBasics>();

        wrapper = host.Services.GetRequiredService<ControlTowerWrapper>();
        orgClient = host.Services.GetRequiredService<IAmazonOrganizations>();
        stsClient = host.Services.GetRequiredService<IAmazonSecurityTokenService>();

        await RunScenario();
    }

    /// <summary>
    /// Runs the example scenario.
    /// </summary>
    public static async Task RunScenario()
    {
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("\tWelcome to the AWS Control Tower with ControlCatalog
example scenario.");
        Console.WriteLine(new string('-', 88));
```

```
        Console.WriteLine("This demo will walk you through working with AWS Control
 Tower for landing zones,");
        Console.WriteLine("managing baselines, and working with controls.");

        try
        {
            var accountId = (await stsClient!.GetCallerIdentityAsync(new
 GetCallerIdentityRequest())).Account;
            Console.WriteLine($"\nAccount ID: {accountId}");

            Console.WriteLine("\nSome demo operations require the use of a landing
 zone.");
            Console.WriteLine("You can use an existing landing zone or opt out of
 these operations in the demo.");
            Console.WriteLine("For instructions on how to set up a landing zone,");
            Console.WriteLine("see https://docs.aws.amazon.com/controltower/latest/
userguide/getting-started-from-console.html");

            // List available landing zones
            var landingZones = await wrapper!.ListLandingZonesAsync();
            if (landingZones.Count > 0)
            {
                Console.WriteLine("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.Count; i++)
                {
                    Console.WriteLine($"{i + 1}. {landingZones[i].Arn}");
                }

                Console.Write($"\nDo you want to use the first landing zone in the
 list ({landingZones[0].Arn})? (y/n): ");
                if (GetUserConfirmation())
                {
                    useLandingZone = true;
                    Console.WriteLine($"Using landing zone: {landingZones[0].Arn}");
                    ouArn = await SetupOrganizationAsync();
                }
            }

            // Managing Baselines
            Console.WriteLine("\nManaging Baselines:");
            var baselines = await wrapper.ListBaselinesAsync();
            Console.WriteLine("\nListing available Baselines:");
            BaselineSummary? controlTowerBaseline = null;
            foreach (var baseline in baselines)
```

```
            {
                if (baseline.Name == "AWSControlTowerBaseline")
                    controlTowerBaseline = baseline;
                Console.WriteLine($"  - {baseline.Name}");
            }

            EnabledBaselineSummary? identityCenterBaseline = null;
            string? baselineArn = null;

            if (useLandingZone && ouArn != null)
            {
                Console.WriteLine("\nListing enabled baselines:");
                var enabledBaselines = await wrapper.ListEnabledBaselinesAsync();
                foreach (var baseline in enabledBaselines)
                {
                    if (baseline.BaselineIdentifier.Contains("baseline/
LN25R72TTG6IGPTQ"))
                        identityCenterBaseline = baseline;
                    Console.WriteLine($"  - {baseline.BaselineIdentifier}");
                }

                if (controlTowerBaseline != null)
                {
                    Console.Write("\nDo you want to enable the Control Tower
 Baseline? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine("\nEnabling Control Tower Baseline.");
                        var icBaselineArn = identityCenterBaseline?.Arn;
                        baselineArn = await wrapper.EnableBaselineAsync(ouArn,
                            controlTowerBaseline.Arn, "5.0", icBaselineArn ?? "");
                        var alreadyEnabled = false;
                        if (baselineArn != null)
                        {
                            Console.WriteLine($"Enabled baseline ARN:
 {baselineArn}");
                        }
                        else
                        {
                            // Find the enabled baseline
                            foreach (var enabled in enabledBaselines)
                            {
                                if (enabled.BaselineIdentifier ==
 controlTowerBaseline.Arn)
```

```
                                {
                                    baselineArn = enabled.Arn;
                                    alreadyEnabled = true;
                                    Console.WriteLine("No change, the selected
baseline was already enabled.");
                                    break;
                                }
                            }
                        }

                        if (baselineArn != null)
                        {
                            Console.Write("\nDo you want to reset the Control Tower
Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"\nResetting Control Tower
Baseline: {baselineArn}");
                                var operationId = await
wrapper.ResetEnabledBaselineAsync(baselineArn);
                                Console.WriteLine($"Reset baseline operation id:
{operationId}");
                            }

                            Console.Write("\nDo you want to disable the Control
Tower Baseline? (y/n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine($"Disabling baseline ARN:
{baselineArn}");
                                var operationId = await
wrapper.DisableBaselineAsync(baselineArn);
                                Console.WriteLine($"Disabled baseline operation id:
{operationId}");
                                if (alreadyEnabled)
                                {
                                    Console.WriteLine($"\nRe-enabling Control Tower
Baseline: {baselineArn}");
                                    // Re-enable the Control Tower baseline if it
was originally enabled.
                                    await wrapper.EnableBaselineAsync(ouArn,
                                        controlTowerBaseline.Arn, "5.0",
icBaselineArn ?? "");
                                }
```

```
                            }
                        }
                    }
                }
            }

            // Managing Controls
            Console.WriteLine("\nManaging Controls:");
            var controls = await wrapper.ListControlsAsync();
            Console.WriteLine("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.Min(5, controls.Count); i++)
            {
                Console.WriteLine($"{i + 1}. {controls[i].Name} -
 {controls[i].Arn}");
            }

            if (useLandingZone && ouArn != null)
            {
                var enabledControls = await wrapper.ListEnabledControlsAsync(ouArn);
                Console.WriteLine("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.Count; i++)
                {
                    Console.WriteLine($"{i + 1}.
 {enabledControls[i].ControlIdentifier}");
                }

                // Find first non-enabled control
                var enabledControlArns = enabledControls.Select(c =>
 c.Arn).ToHashSet();
                var controlArn = controls.FirstOrDefault(c => !
enabledControlArns.Contains(c.Arn))?.Arn;

                if (controlArn != null)
                {
                    Console.Write($"\nDo you want to enable the control
 {controlArn}? (y/n): ");
                    if (GetUserConfirmation())
                    {
                        Console.WriteLine($"\nEnabling control: {controlArn}");
                        var operationId = await
 wrapper.EnableControlAsync(controlArn, ouArn);
                        if (operationId != null)
                        {
```

```
                            Console.WriteLine($"Enabled control with operation id:
 {operationId}");

                            Console.Write("\nDo you want to disable the control? (y/
n): ");
                            if (GetUserConfirmation())
                            {
                                Console.WriteLine("\nDisabling the control...");
                                var disableOpId = await
 wrapper.DisableControlAsync(controlArn, ouArn);
                                Console.WriteLine($"Disable operation ID:
 {disableOpId}");
                            }
                        }
                    }
                }
            }

            Console.WriteLine("\nThis concludes the example scenario.");
            Console.WriteLine("Thanks for watching!");
            Console.WriteLine(new string('-', 88));
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "An error occurred during the Control Tower
 scenario.");
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Sets up AWS Organizations and creates or finds a Sandbox OU.
    /// </summary>
    /// <returns>The ARN of the Sandbox organizational unit.</returns>
    private static async Task<string> SetupOrganizationAsync()
    {
        Console.WriteLine("\nChecking organization status...");

        try
        {
            var orgResponse = await orgClient!.DescribeOrganizationAsync(new
 DescribeOrganizationRequest());
            var orgId = orgResponse.Organization.Id;
            Console.WriteLine($"Account is part of organization: {orgId}");
```

```
        }
        catch (AWSOrganizationsNotInUseException)
        {
            Console.WriteLine("No organization found. Creating a new
organization...");
            var createResponse = await orgClient!.CreateOrganizationAsync(new
CreateOrganizationRequest { FeatureSet = OrganizationFeatureSet.ALL });
            var orgId = createResponse.Organization.Id;
            Console.WriteLine($"Created new organization: {orgId}");
        }

        // Look for Sandbox OU
        var roots = await orgClient.ListRootsAsync(new ListRootsRequest());
        var rootId = roots.Roots[0].Id;

        Console.WriteLine("Checking for Sandbox OU...");
        var ous = await orgClient.ListOrganizationalUnitsForParentAsync(new
ListOrganizationalUnitsForParentRequest { ParentId = rootId });
        var sandboxOu = ous.OrganizationalUnits.FirstOrDefault(ou => ou.Name ==
"Sandbox");

        if (sandboxOu == null)
        {
            Console.WriteLine("Creating Sandbox OU...");
            var createOuResponse = await orgClient.CreateOrganizationalUnitAsync(new
CreateOrganizationalUnitRequest { ParentId = rootId, Name = "Sandbox" });
            sandboxOu = createOuResponse.OrganizationalUnit;
            Console.WriteLine($"Created new Sandbox OU: {sandboxOu.Id}");
        }
        else
        {
            Console.WriteLine($"Found existing Sandbox OU: {sandboxOu.Id}");
        }

        return sandboxOu.Arn;
    }

    /// <summary>
    /// Gets user confirmation by waiting for input or returning true if not
interactive.
    /// </summary>
    /// <returns>True if user enters 'y' or if isInteractive is false, otherwise
false.</returns>
    private static bool GetUserConfirmation()
```

```
    {
        return Console.ReadLine()?.ToLower() == "y" || !isInteractive;
    }
}
```

Wrapper-Methoden, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
using Amazon.ControlCatalog;
using Amazon.ControlCatalog.Model;
using Amazon.ControlTower;
using Amazon.ControlTower.Model;
using ValidationException = Amazon.ControlTower.Model.ValidationException;

namespace ControlTowerActions;

/// <summary>
/// Methods to perform AWS Control Tower actions.
/// </summary>
public class ControlTowerWrapper
{
    private readonly IAmazonControlTower _controlTowerService;
    private readonly IAmazonControlCatalog _controlCatalogService;

    /// <summary>
    /// Constructor for the wrapper class containing AWS Control Tower actions.
    /// </summary>
    /// <param name="controlTowerService">The AWS Control Tower client object.</
param>
    /// <param name="controlCatalogService">The AWS Control Catalog client object.</
param>
    public ControlTowerWrapper(IAmazonControlTower controlTowerService,
 IAmazonControlCatalog controlCatalogService)
    {
        _controlTowerService = controlTowerService;
        _controlCatalogService = controlCatalogService;
    }

    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
```

```
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }
```

```
            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list baselines. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
```

```
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier, string
 baselineIdentifier, string baselineVersion, string identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
            };

            var response = await _controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
 BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
```

```
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a baseline for a specific target and wait for the operation to
complete.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
    /// <returns>The operation ID or null if there was a conflict.</returns>
    public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new DisableBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await _controlTowerService.DisableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
```

```
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}. Skipping
disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };
```

```
            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
```

```
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
 _controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit identifier.</
param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>> ListEnabledControlsAsync(string
 targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };

            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
 _controlTowerService.Paginators.ListEnabledControls(request);
```

```
            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Enable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to enable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID or null if already enabled.</returns>
    public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            Console.WriteLine(controlArn);
            Console.WriteLine(targetIdentifier);

            var request = new EnableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };
```

```
            var response = await _controlTowerService.EnableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// Disable a control for a specified target.
    /// </summary>
    /// <param name="controlArn">The ARN of the control to disable.</param>
```

```
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID.</returns>
   public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           var request = new DisableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await _controlTowerService.DisableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetControlOperationAsync(operationId);
               Console.WriteLine($"Control operation status: {status}");
               if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
               {
                   break;
               }
               await Task.Delay(30000); // Wait 30 seconds
           }

           return operationId;
       }
       catch (Amazon.ControlTower.Model.ResourceNotFoundException)
       {
           Console.WriteLine("Control not found.");
           throw;
       }
       catch (AmazonControlTowerException ex)
       {
           Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
           throw;
       }
   }
```

```
    /// <summary>
    /// Get the status of a control operation.
    /// </summary>
    /// <param name="operationId">The ID of the control operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetControlOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetControlOperationAsync(request);
            return response.ControlOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get control operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }


    /// <summary>
    /// List all controls in the Control Tower control catalog.
    /// </summary>
    /// <returns>A list of control summaries.</returns>
    public async Task<List<ControlSummary>> ListControlsAsync()
    {
        try
        {
            var controls = new List<ControlSummary>();
```

```
            var controlsPaginator =
 _controlCatalogService.Paginators.ListControls(new
 Amazon.ControlCatalog.Model.ListControlsRequest());

            await foreach (var response in controlsPaginator.Responses)
            {
                controls.AddRange(response.Controls);
            }

            return controls;
        }
        catch (AmazonControlCatalogException ex)
        {
            Console.WriteLine($"Couldn't list controls. Here's why: {ex.ErrorCode}:
 {ex.Message}");
            throw;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones

- ResetEnabledBaseline

Aktionen

`DisableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`DisableBaseline`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Disable a baseline for a specific target and wait for the operation to
complete.
   /// </summary>
   /// <param name="enabledBaselineIdentifier">The identifier of the baseline to
disable.</param>
   /// <returns>The operation ID or null if there was a conflict.</returns>
   public async Task<string?> DisableBaselineAsync(string
enabledBaselineIdentifier)
   {
       try
       {
           var request = new DisableBaselineRequest
           {
               EnabledBaselineIdentifier = enabledBaselineIdentifier
           };

           var response = await _controlTowerService.DisableBaselineAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
           while (true)
           {
               var status = await GetBaselineOperationAsync(operationId);
               Console.WriteLine($"Baseline operation status: {status}");
```

```
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (ConflictException ex)
        {
            Console.WriteLine($"Conflict disabling baseline: {ex.Message}. Skipping
disable step.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DisableBaseline in der AWS SDK für .NET API-Referenz.

`DisableControl`

Das folgende Codebeispiel zeigt die Verwendung`DisableControl`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Disable a control for a specified target.
```

```
    /// </summary>
    /// <param name="controlArn">The ARN of the control to disable.</param>
    /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> DisableControlAsync(string controlArn, string
targetIdentifier)
    {
        try
        {
            var request = new DisableControlRequest
            {
                ControlIdentifier = controlArn,
                TargetIdentifier = targetIdentifier
            };

            var response = await _controlTowerService.DisableControlAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Control not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't disable control. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
```

```
        }
    }
```

- API-Details finden Sie DisableControl in der AWS SDK für .NET API-Referenz.

`EnableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`EnableBaseline`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Enable a baseline for the specified target.
    /// </summary>
    /// <param name="targetIdentifier">The ARN of the target.</param>
    /// <param name="baselineIdentifier">The identifier of baseline to enable.</
param>
    /// <param name="baselineVersion">The version of baseline to enable.</param>
    /// <param name="identityCenterBaseline">The identifier of identity center
 baseline if it is enabled.</param>
    /// <returns>The enabled baseline ARN or null.</returns>
    public async Task<string?> EnableBaselineAsync(string targetIdentifier, string
 baselineIdentifier, string baselineVersion, string identityCenterBaseline)
    {
        try
        {
            var parameters = new List<EnabledBaselineParameter>();
            if (!string.IsNullOrEmpty(identityCenterBaseline))
            {
                parameters.Add(
                    new EnabledBaselineParameter
                    {
                        Key = "IdentityCenterEnabledBaselineArn",
                        Value = identityCenterBaseline
```

```
                    });
            }
            var request = new EnableBaselineRequest
            {
                BaselineIdentifier = baselineIdentifier,
                BaselineVersion = baselineVersion,
                TargetIdentifier = targetIdentifier,
                Parameters = parameters
            };

            var response = await _controlTowerService.EnableBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return response.Arn;
        }
        catch (ValidationException ex)
        {
            if (ex.Message.Contains("already enabled"))
                Console.WriteLine("Baseline is already enabled for this target");
            else { Console.WriteLine(ex.Message); }
            // Write the message and return null if baseline cannot be enabled.
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie EnableBaseline in der AWS SDK für .NET API-Referenz.

`EnableControl`

Das folgende Codebeispiel zeigt die Verwendung`EnableControl`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Enable a control for a specified target.
   /// </summary>
   /// <param name="controlArn">The ARN of the control to enable.</param>
   /// <param name="targetIdentifier">The identifier of the target (e.g., OU
ARN).</param>
   /// <returns>The operation ID or null if already enabled.</returns>
   public async Task<string?> EnableControlAsync(string controlArn, string
targetIdentifier)
   {
       try
       {
           Console.WriteLine(controlArn);
           Console.WriteLine(targetIdentifier);

           var request = new EnableControlRequest
           {
               ControlIdentifier = controlArn,
               TargetIdentifier = targetIdentifier
           };

           var response = await _controlTowerService.EnableControlAsync(request);
           var operationId = response.OperationIdentifier;

           // Wait for operation to complete
```

```
            while (true)
            {
                var status = await GetControlOperationAsync(operationId);
                Console.WriteLine($"Control operation status: {status}");
                if (status == ControlOperationStatus.SUCCEEDED || status ==
ControlOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
        }
        catch (Amazon.ControlTower.Model.ValidationException ex) when
(ex.Message.Contains("already enabled"))
        {
            Console.WriteLine("Control is already enabled for this target");
            return null;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return null;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't enable control. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie EnableControl in der AWS SDK für .NET API-Referenz.

`GetBaselineOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetBaselineOperation`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the status of a baseline operation.
    /// </summary>
    /// <param name="operationId">The ID of the baseline operation.</param>
    /// <returns>The operation status.</returns>
    public async Task<BaselineOperationStatus> GetBaselineOperationAsync(string
operationId)
    {
        try
        {
            var request = new GetBaselineOperationRequest
            {
                OperationIdentifier = operationId
            };

            var response = await
_controlTowerService.GetBaselineOperationAsync(request);
            return response.BaselineOperation.Status;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Operation not found.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't get baseline operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie GetBaselineOperation in der AWS SDK für .NET API-Referenz.

`GetControlOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetControlOperation`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Get the status of a control operation.
   /// </summary>
   /// <param name="operationId">The ID of the control operation.</param>
   /// <returns>The operation status.</returns>
   public async Task<ControlOperationStatus> GetControlOperationAsync(string
operationId)
   {
       try
       {
           var request = new GetControlOperationRequest
           {
               OperationIdentifier = operationId
           };

           var response = await
_controlTowerService.GetControlOperationAsync(request);
           return response.ControlOperation.Status;
       }
       catch (Amazon.ControlTower.Model.ResourceNotFoundException)
       {
           Console.WriteLine("Operation not found.");
           throw;
       }
       catch (AmazonControlTowerException ex)
       {
           Console.WriteLine($"Couldn't get control operation status. Here's why:
{ex.ErrorCode}: {ex.Message}");
```

```
            throw;
        }
    }
```

- API-Details finden Sie GetControlOperation in der AWS SDK für .NET API-Referenz.

`ListBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListBaselines`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List all baselines.
    /// </summary>
    /// <returns>A list of baseline summaries.</returns>
    public async Task<List<BaselineSummary>> ListBaselinesAsync()
    {
        try
        {
            var baselines = new List<BaselineSummary>();

            var baselinesPaginator =
_controlTowerService.Paginators.ListBaselines(new ListBaselinesRequest());

            await foreach (var response in baselinesPaginator.Responses)
            {
                baselines.AddRange(response.Baselines);
            }

            return baselines;
        }
        catch (AmazonControlTowerException ex)
        {
```

```
            Console.WriteLine($"Couldn't list baselines. Here's why: {ex.ErrorCode}:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ListBaselines in der AWS SDK für .NET API-Referenz.

`ListEnabledBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledBaselines`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List all enabled baselines.
    /// </summary>
    /// <returns>A list of enabled baseline summaries.</returns>
    public async Task<List<EnabledBaselineSummary>> ListEnabledBaselinesAsync()
    {
        try
        {
            var enabledBaselines = new List<EnabledBaselineSummary>();

            var enabledBaselinesPaginator =
_controlTowerService.Paginators.ListEnabledBaselines(new
ListEnabledBaselinesRequest());

            await foreach (var response in enabledBaselinesPaginator.Responses)
            {
                enabledBaselines.AddRange(response.EnabledBaselines);
            }

            return enabledBaselines;
```

```
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled baselines. Here's why:
 {ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ListEnabledBaselines in der AWS SDK für .NET API-Referenz.

`ListEnabledControls`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledControls`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List enabled controls for a target organizational unit.
    /// </summary>
    /// <param name="targetIdentifier">The target organizational unit identifier.</
param>
    /// <returns>A list of enabled control summaries.</returns>
    public async Task<List<EnabledControlSummary>> ListEnabledControlsAsync(string
 targetIdentifier)
    {
        try
        {
            var request = new ListEnabledControlsRequest
            {
                TargetIdentifier = targetIdentifier
            };
```

```
            var enabledControls = new List<EnabledControlSummary>();

            var enabledControlsPaginator =
_controlTowerService.Paginators.ListEnabledControls(request);

            await foreach (var response in enabledControlsPaginator.Responses)
            {
                enabledControls.AddRange(response.EnabledControls);
            }

            return enabledControls;
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("not registered with AWS Control Tower"))
        {
            Console.WriteLine("AWS Control Tower must be enabled to work with
enabling controls.");
            return new List<EnabledControlSummary>();
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list enabled controls. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ListEnabledControls in der AWS SDK für .NET API-Referenz.

`ListLandingZones`

Das folgende Codebeispiel zeigt die Verwendung`ListLandingZones`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List the AWS Control Tower landing zones for an account.
    /// </summary>
    /// <returns>A list of LandingZoneSummary objects.</returns>
    public async Task<List<LandingZoneSummary>> ListLandingZonesAsync()
    {
        try
        {
            var landingZones = new List<LandingZoneSummary>();

            var landingZonesPaginator =
_controlTowerService.Paginators.ListLandingZones(new ListLandingZonesRequest());

            await foreach (var response in landingZonesPaginator.Responses)
            {
                landingZones.AddRange(response.LandingZones);
            }

            return landingZones;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't list landing zones. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ListLandingZones in der AWS SDK für .NET API-Referenz.

`ResetEnabledBaseline`

Das folgende Codebeispiel zeigt die Verwendung`ResetEnabledBaseline`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Reset an enabled baseline for a specific target.
    /// </summary>
    /// <param name="enabledBaselineIdentifier">The identifier of the enabled
baseline to reset.</param>
    /// <returns>The operation ID.</returns>
    public async Task<string> ResetEnabledBaselineAsync(string
enabledBaselineIdentifier)
    {
        try
        {
            var request = new ResetEnabledBaselineRequest
            {
                EnabledBaselineIdentifier = enabledBaselineIdentifier
            };

            var response = await
_controlTowerService.ResetEnabledBaselineAsync(request);
            var operationId = response.OperationIdentifier;

            // Wait for operation to complete
            while (true)
            {
                var status = await GetBaselineOperationAsync(operationId);
                Console.WriteLine($"Baseline operation status: {status}");
                if (status == BaselineOperationStatus.SUCCEEDED || status ==
BaselineOperationStatus.FAILED)
                {
                    break;
                }
                await Task.Delay(30000); // Wait 30 seconds
            }

            return operationId;
```

```
        }
        catch (Amazon.ControlTower.Model.ResourceNotFoundException)
        {
            Console.WriteLine("Target not found, unable to reset enabled
baseline.");
            throw;
        }
        catch (AmazonControlTowerException ex)
        {
            Console.WriteLine($"Couldn't reset enabled baseline. Here's why:
{ex.ErrorCode}: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ResetEnabledBaseline in der AWS SDK für .NET API-Referenz.

DynamoDB-Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.Model;
using Microsoft.Extensions.DependencyInjection;

namespace DynamoDBActions;

/// <summary>
/// A simple example that demonstrates basic DynamoDB operations.
/// </summary>
public class HelloDynamoDB
{
    /// <summary>
    /// HelloDynamoDB lists the existing DynamoDB tables for the default user.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
            )
            .Build();

        // Now the client is available for injection.
```

```
        var dynamoDbClient = host.Services.GetRequiredService<IAmazonDynamoDB>();

        try
        {
            var request = new ListTablesRequest();
            var tableNames = new List<string>();

            var paginatorForTables = dynamoDbClient.Paginators.ListTables(request);

            await foreach (var tableName in paginatorForTables.TableNames)
            {
                tableNames.Add(tableName);
            }

            Console.WriteLine("Welcome to the DynamoDB Hello Service example. " +
                              "\nLet's list your DynamoDB tables:");
            tableNames.ForEach(table =>
            {
                Console.WriteLine($"Table: {table}");
            });
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB service error occurred while
 listing tables. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing tables.
 {ex.Message}");
        }
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// This example application performs the following basic Amazon DynamoDB
/// functions:
///     CreateTableAsync
///     PutItemAsync
///     UpdateItemAsync
///     BatchWriteItemAsync
///     GetItemAsync
///     DeleteItemAsync
///     Query
///     Scan
///     DeleteItemAsync.
/// </summary>
public class DynamoDbBasics
{
    public static bool IsInteractive = true;

    // Separator for the console display.
    private static readonly string SepBar = new string('-', 80);

    /// <summary>
    /// The main entry point for the DynamoDB Basics example application.
    /// </summary>
```

```
    /// <param name="args">Command line arguments.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon DynamoDB.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonDynamoDB>()
                    .AddTransient<DynamoDbWrapper>())
            .Build();

        // Now the wrapper is available for injection.
        var dynamoDbWrapper = host.Services.GetRequiredService<DynamoDbWrapper>();

        var tableName = "movie_table";

        var movieFileName = @"movies.json";

        DisplayInstructions();

        // Create a new table and wait for it to be active.
        Console.WriteLine($"Creating the new table: {tableName}");

        var success = await dynamoDbWrapper.CreateMovieTableAsync(tableName);

        Console.WriteLine(success
            ? $"\nTable: {tableName} successfully created."
            : $"\nCould not create {tableName}.");

        WaitForEnter();

        // Add a single new movie to the table.
        var newMovie = new Movie
        {
            Year = 2021,
            Title = "Spider-Man: No Way Home",
        };

        success = await dynamoDbWrapper.PutItemAsync(newMovie, tableName);
        if (success)
        {
            Console.WriteLine($"Added {newMovie.Title} to the table.");
        }
```

```
        else
        {
            Console.WriteLine("Could not add movie to table.");
        }

        WaitForEnter();

        // Update the new movie by adding a plot and rank.
        var newInfo = new MovieInfo
        {
            Plot = "With Spider-Man's identity now revealed, Peter asks" +
                   "Doctor Strange for help. When a spell goes wrong, dangerous" +
                   "foes from other worlds start to appear, forcing Peter to" +
                   "discover what it truly means to be Spider-Man.",
            Rank = 9,
        };

        success = await dynamoDbWrapper.UpdateItemAsync(newMovie, newInfo,
tableName);
        if (success)
        {
            Console.WriteLine($"Successfully updated the movie: {newMovie.Title}");
        }
        else
        {
            Console.WriteLine("Could not update the movie.");
        }

        WaitForEnter();

        // Add a batch of movies to the DynamoDB table from a list of
        // movies in a JSON file.
        var itemCount = await dynamoDbWrapper.BatchWriteItemsAsync(movieFileName,
tableName);
        Console.WriteLine($"Added {itemCount} movies to the table.");

        WaitForEnter();

        // Get a movie by key. (partition + sort)
        var lookupMovie = new Movie
        {
            Title = "Jurassic Park",
            Year = 1993,
        };
```

```
        Console.WriteLine("Looking for the movie \"Jurassic Park\".");
        var item = await dynamoDbWrapper.GetItemAsync(lookupMovie, tableName);
        if (item?.Count > 0)
        {
            dynamoDbWrapper.DisplayItem(item);
        }
        else
        {
            Console.WriteLine($"Couldn't find {lookupMovie.Title}");
        }

        WaitForEnter();

        // Delete a movie.
        var movieToDelete = new Movie
        {
            Title = "The Town",
            Year = 2010,
        };

        success = await dynamoDbWrapper.DeleteItemAsync(tableName, movieToDelete);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {movieToDelete.Title}.");
        }

        WaitForEnter();

        // Use Query to find all the movies released in 2010.
        int findYear = 2010;
        Console.WriteLine($"Movies released in {findYear}");
        var queryCount = await dynamoDbWrapper.QueryMoviesAsync(tableName,
findYear);
        Console.WriteLine($"Found {queryCount} movies released in {findYear}");

        WaitForEnter();

        // Use Scan to get a list of movies from 2001 to 2011.
```

```
        int startYear = 2001;
        int endYear = 2011;
        var scanCount = await dynamoDbWrapper.ScanTableAsync(tableName, startYear,
endYear);
        Console.WriteLine($"Found {scanCount} movies released between {startYear}
and {endYear}");

        WaitForEnter();

        // Delete the table.
        success = await dynamoDbWrapper.DeleteTableAsync(tableName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {tableName}");
        }
        else
        {
            Console.WriteLine($"Could not delete {tableName}");
        }

        Console.WriteLine("The DynamoDB Basics example application is complete.");

        WaitForEnter();
    }

    /// <summary>
    /// Displays the description of the application on the console.
    /// </summary>
    private static void DisplayInstructions()
    {
        if (!IsInteractive)
        {
            return;
        }

        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using DynamoDB
with the AWS SDK.");
        Console.WriteLine(SepBar);
```

```
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
 sort:title.");
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that was
 just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition + sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released in a
 given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released within
 a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
 created.");
        WaitForEnter();
    }

    /// <summary>
    /// Simple method to wait for the Enter key to be pressed.
    /// </summary>
    private static void WaitForEnter()
    {
        if (IsInteractive)
        {
            Console.WriteLine("\nPress <Enter> to continue.");
            Console.WriteLine(SepBar);
            _ = Console.ReadLine();
        }
    }
}
```

Verwenden Sie den injizierten Client für Tabellenoperationen.

```
using System.Text.Json;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DataModel;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.DynamoDBv2.Model;

namespace DynamoDBActions;
```

```
/// <summary>
/// Methods of this class perform Amazon DynamoDB operations.
/// </summary>
public class DynamoDbWrapper
{
    private readonly IAmazonDynamoDB _amazonDynamoDB;

    /// <summary>
    /// Constructor for the DynamoDbWrapper class.
    /// </summary>
    /// <param name="amazonDynamoDB">The injected DynamoDB client.</param>
    public DynamoDbWrapper(IAmazonDynamoDB amazonDynamoDB)
    {
        _amazonDynamoDB = amazonDynamoDB;
    }
```

Erstellt eine Tabelle, die Filmdaten enthält.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
 CreateTableRequest
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
```

```
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");
```

```
            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
 table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table {tableName}.
 {ex.Message}");
            throw;
        }
    }
```

Fügt der Tabelle einen einzelnen Film hinzu.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
```

```
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
{ex.Message}");
            throw;
        }
    }
```

Aktualisiert ein einzelnes Element in einer Tabelle.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
```

```
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
```

```
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

Ruft ein einzelnes Element aus der Filmtabelle ab.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
```

```
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while getting item.
{ex.Message}");
            throw;
        }
    }
```

Schreibt einen Stapel von Elementen in die Filmtabelle.

```
    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the DynamoDB table.
    /// </summary>
    /// <param name="movieFileName">The name of the JSON file.</param>
    /// <returns>A generic list of movie objects.</returns>
    public List<Movie> ImportMovies(string movieFileName)
    {
        var moviesList = new List<Movie>();
        if (!File.Exists(movieFileName))
        {
```

```
            return moviesList;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonSerializer.Deserialize<List<Movie>>(
            json,
            new JsonSerializerOptions
            {
                PropertyNameCaseInsensitive = true
            });

        // Now return the first 250 entries.
        if (allMovies != null && allMovies.Any())
        {
            moviesList = allMovies.GetRange(0, 250);
        }
        return moviesList;
    }

    /// <summary>
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();
```

```
            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch write
operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

Löscht ein einzelnes Element aus der Tabelle.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
```

```
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

Fragt die Tabelle nach Filmen ab, die in einem bestimmten Jahr veröffentlicht wurden.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };

        // Value used to track how many movies match the
        // supplied criteria.
        var moviesFound = 0;

        var search = movieTable.Query(config);
        do
        {
```

```
                var movieList = await search.GetNextSetAsync();
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
{ex.Message}");
            throw;
        }
    }
```

Scannt die Tabelle nach Filmen, die in einem bestimmten Zeitraum veröffentlicht wurden.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</returns>
    public async Task<int> ScanTableAsync(
```

```
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string, AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;

            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
```

```
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

Löscht die Filmtabelle.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
```

```
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table {tableName}.
{ex.Message}");
            return false;
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Schreibt einen Stapel von Elementen in die Filmtabelle.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The name of the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public List<Movie> ImportMovies(string movieFileName)
{
    var moviesList = new List<Movie>();
    if (!File.Exists(movieFileName))
    {
        return moviesList;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonSerializer.Deserialize<List<Movie>>(
        json,
        new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        });

    // Now return the first 250 entries.
    if (allMovies != null && allMovies.Any())
    {
        moviesList = allMovies.GetRange(0, 250);
    }
    return moviesList;
}

/// <summary>
```

```
    /// Writes 250 items to the movie table.
    /// </summary>
    /// <param name="movieFileName">A string containing the full path to
    /// the JSON file containing movie data.</param>
    /// <param name="tableName">The name of the table to write items to.</param>
    /// <returns>A long integer value representing the number of movies
    /// imported from the JSON file.</returns>
    public async Task<long> BatchWriteItemsAsync(
        string movieFileName, string tableName)
    {
        try
        {
            var movies = ImportMovies(movieFileName);
            if (!movies.Any())
            {
                Console.WriteLine("Couldn't find the JSON file with movie data.");
                return 0;
            }

            var context = new DynamoDBContextBuilder()
                // Optional call to provide a specific instance of IAmazonDynamoDB
                .WithDynamoDBClient(() => _amazonDynamoDB)
                .Build();

            var movieBatch = context.CreateBatchWrite<Movie>(
                new BatchWriteConfig()
                {
                    OverrideTableName = tableName
                });
            movieBatch.AddPutItems(movies);

            Console.WriteLine("Adding imported movies to the table.");
            await movieBatch.ExecuteAsync();

            return movies.Count;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table was not found during batch write operation.
{ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
```

```
            Console.WriteLine($"An Amazon DynamoDB error occurred during batch write
operation. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during batch write operation.
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für .NET API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    /// </summary>
    /// <param name="tableName">The name of the table to create.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> CreateMovieTableAsync(string tableName)
    {
        try
        {
            var response = await _amazonDynamoDB.CreateTableAsync(new
CreateTableRequest
            {
```

```
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition
                    {
                        AttributeName = "title",
                        AttributeType = ScalarAttributeType.S,
                    },
                    new AttributeDefinition
                    {
                        AttributeName = "year",
                        AttributeType = ScalarAttributeType.N,
                    },
                },
                KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement
                    {
                        AttributeName = "year",
                        KeyType = KeyType.HASH,
                    },
                    new KeySchemaElement
                    {
                        AttributeName = "title",
                        KeyType = KeyType.RANGE,
                    },
                },
                BillingMode = BillingMode.PAY_PER_REQUEST,
            });

            // Wait until the table is ACTIVE and then report success.
            Console.Write("Waiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = response.TableDescription.TableName,
            };

            TableStatus status;

            int sleepDuration = 2000;

            do
            {
```

```
                Thread.Sleep(sleepDuration);

                var describeTableResponse = await
_amazonDynamoDB.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException ex)
        {
            Console.WriteLine($"Table {tableName} already exists. {ex.Message}");
            throw;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while creating
table {tableName}. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating table {tableName}.
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie CreateTable in der AWS SDK für .NET API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a single item from a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table from which the item
    /// will be deleted.</param>
    /// <param name="movieToDelete">A movie object containing the title and
    /// year of the movie to delete.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// delete operation.</returns>
    public async Task<bool> DeleteItemAsync(
        string tableName,
        Movie movieToDelete)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = movieToDelete.Title },
                ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
            };

            var request = new DeleteItemRequest { TableName = tableName, Key =
key, };

            await _amazonDynamoDB.DeleteItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
```

```
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting item.
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DeleteItem in der AWS SDK für .NET API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes a DynamoDB table.
    /// </summary>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the operation.</returns>
    public async Task<bool> DeleteTableAsync(string tableName)
    {
        try
        {
            var request = new DeleteTableRequest
            {
                TableName = tableName,
            };
```

```
            var response = await _amazonDynamoDB.DeleteTableAsync(request);

            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;

        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found and cannot be
deleted. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while deleting
table {tableName}. {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting table {tableName}.
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK für .NET API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets information about an existing movie from the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information about
    /// the movie to retrieve.</param>
    /// <param name="tableName">The name of the table containing the movie.</param>
    /// <returns>A Dictionary object containing information about the item
    /// retrieved.</returns>
    public async Task<Dictionary<string, AttributeValue>> GetItemAsync(Movie
newMovie, string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };

            var request = new GetItemRequest
            {
                Key = key,
                TableName = tableName,
            };

            var response = await _amazonDynamoDB.GetItemAsync(request);
            return response.Item;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return new Dictionary<string, AttributeValue>();
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while getting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"An error occurred while getting item.
 {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie GetItem in der AWS SDK für .NET API-Referenz.

PutItem

Das folgende Codebeispiel zeigt die VerwendungPutItem.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Adds a new item to the table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing informtation for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
 added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
returns>
    public async Task<bool> PutItemAsync(Movie newMovie, string tableName)
    {
        try
        {
            var item = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
```

```
            var request = new PutItemRequest
            {
                TableName = tableName,
                Item = item,
            };

            await _amazonDynamoDB.PutItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while putting
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while putting item.
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie PutItem in der AWS SDK für .NET API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public async Task<int> QueryMoviesAsync(string tableName, int year)
{
    try
    {
        var movieTable = new TableBuilder(_amazonDynamoDB, tableName)
            .AddHashKey("year", DynamoDBEntryType.Numeric)
            .AddRangeKey("title", DynamoDBEntryType.String)
            .Build();

        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
            AttributesToGet = new List<string>
            {
                "title",
                "year",
            },
            ConsistentRead = true,
            Filter = filter,
        };

        // Value used to track how many movies match the
        // supplied criteria.
        var moviesFound = 0;

        var search = movieTable.Query(config);
        do
        {
            var movieList = await search.GetNextSetAsync();
```

```
                moviesFound += movieList.Count;

                foreach (var movie in movieList)
                {
                    DisplayDocument(movie);
                }
            }
            while (!search.IsDone);

            return moviesFound;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while querying
movies. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while querying movies.
{ex.Message}");
            throw;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für .NET -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Scans the table for movies released between the specified years.
    /// </summary>
    /// <param name="tableName">The name of the table to scan.</param>
    /// <param name="startYear">The starting year for the range.</param>
    /// <param name="endYear">The ending year for the range.</param>
    /// <returns>The number of movies found in the specified year range.</returns>
    public async Task<int> ScanTableAsync(
        string tableName,
        int startYear,
        int endYear)
    {
        try
        {
            var request = new ScanRequest
            {
                TableName = tableName,
                ExpressionAttributeNames = new Dictionary<string, string>
                {
                    { "#yr", "year" },
                },
                ExpressionAttributeValues = new Dictionary<string, AttributeValue>
                {
                    { ":y_a", new AttributeValue { N = startYear.ToString() } },
                    { ":y_z", new AttributeValue { N = endYear.ToString() } },
                },
                FilterExpression = "#yr between :y_a and :y_z",
                ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
                Limit = 10 // Set a limit to demonstrate using the LastEvaluatedKey.
            };

            // Keep track of how many movies were found.
            int foundCount = 0;
```

```
            var response = new ScanResponse();
            do
            {
                response = await _amazonDynamoDB.ScanAsync(request);
                foundCount += response.Items.Count;
                response.Items.ForEach(i => DisplayItem(i));
                request.ExclusiveStartKey = response.LastEvaluatedKey;
            }
            while (response?.LastEvaluatedKey?.Count > 0);
            return foundCount;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} was not found. {ex.Message}");
            return 0;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while scanning
table. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while scanning table.
{ex.Message}");
            throw;
        }
    }
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für .NET -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
param>
    /// <returns>A Boolean value that indicates the success of the operation.</
returns>
    public async Task<bool> UpdateItemAsync(
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        try
        {
            var key = new Dictionary<string, AttributeValue>
            {
                ["title"] = new AttributeValue { S = newMovie.Title },
                ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
            };
            var updates = new Dictionary<string, AttributeValueUpdate>
            {
                ["info.plot"] = new AttributeValueUpdate
                {
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { S = newInfo.Plot },
                },

                ["info.rating"] = new AttributeValueUpdate
                {
```

```
                    Action = AttributeAction.PUT,
                    Value = new AttributeValue { N = newInfo.Rank.ToString() },
                },
            };

            var request = new UpdateItemRequest
            {
                AttributeUpdates = updates,
                Key = key,
                TableName = tableName,
            };

            await _amazonDynamoDB.UpdateItemAsync(request);
            return true;
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Table {tableName} or item was not found.
{ex.Message}");
            return false;
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine($"An Amazon DynamoDB error occurred while updating
item. {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while updating item.
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK für .NET API-Referenz.

Amazon EC2-Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon EC2.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace EC2Actions;

public class HelloEc2
{
    /// <summary>
    /// HelloEc2 lists the existing security groups for the default users.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
 EC2).
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                .AddTransient<EC2Wrapper>()
            )
            .Build();

        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();

        try
        {
            // Retrieve information for up to 10 Amazon EC2 security groups.
            var request = new DescribeSecurityGroupsRequest { MaxResults = 10 };
            var securityGroups = new List<SecurityGroup>();

            var paginatorForSecurityGroups =
                ec2Client.Paginators.DescribeSecurityGroups(request);

            await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
            {
                securityGroups.Add(securityGroup);
            }

            // Now print the security groups returned by the call to
            // DescribeSecurityGroupsAsync.
            Console.WriteLine("Welcome to the EC2 Hello Service example. " +
                              "\nLet's list your Security Groups:");
            securityGroups.ForEach(group =>
            {
                Console.WriteLine(
                    $"Security group: {group.GroupName} ID: {group.GroupId}");
            });
        }
        catch (AmazonEC2Exception ex)
        {
            Console.WriteLine($"An Amazon EC2 service error occurred while listing
security groups. {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing security groups.
{ex.Message}");
        }
```

```
    }
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für .NET API-Referenz.

Amazon ECS-Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon ECS Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Amazon ECS

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon ECS beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
using Amazon.ECS;
using Amazon.ECS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace ECSActions;
```

```
/// <summary>
/// A class that introduces the Amazon ECS Client by listing the
/// cluster ARNs for the account.
/// </summary>
public class HelloECS
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon ECS client.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonECS>()
            )
            .Build();

        var amazonECSClient = host.Services.GetRequiredService<IAmazonECS>();

        Console.WriteLine($"Hello Amazon ECS! Following are some cluster ARNS
 available in the your account");
        Console.WriteLine();

        var clusters = new List<string>();

        var clustersPaginator = amazonECSClient.Paginators.ListClusters(new
 ListClustersRequest());

        await foreach (var response in clustersPaginator.Responses)
        {
            clusters.AddRange(response.ClusterArns);
        }

        if (clusters.Count > 0)
        {
            clusters.ForEach(cluster =>
            {
                Console.WriteLine($"\tARN: {cluster}");
```

```
                Console.WriteLine($"Cluster Name: {cluster.Split("/").Last()}");
                Console.WriteLine();
            });
        }
        else
        {
            Console.WriteLine("No clusters were found.");
        }

    }
}
```

- API-Details finden Sie ListClusters in der AWS SDK für .NET API-Referenz.

AWS IoT Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Hello AWS IoT example.
/// </summary>
public class HelloIoT
{
    /// <summary>
    /// Main method to run the Hello IoT example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var iotClient = new AmazonIoTClient();

        try
        {
            Console.WriteLine("Hello AWS IoT! Let's list your IoT Things:");
            Console.WriteLine(new string('-', 80));

            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await iotClient.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
 paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);
```

```
                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await iotClient.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
            }

            if (things is { Count: > 0 })
            {
                Console.WriteLine($"Found {things.Count} IoT Things:");
                foreach (var thing in things)
                {
                    Console.WriteLine($"- Thing Name: {thing.ThingName}");
                    Console.WriteLine($"  Thing ARN: {thing.ThingArn}");
                    Console.WriteLine($"  Thing Type: {thing.ThingTypeName ?? "No
type specified"}");
                    Console.WriteLine($"  Version: {thing.Version}");

                    if (thing.Attributes?.Count > 0)
                    {
                        Console.WriteLine("  Attributes:");
                        foreach (var attr in thing.Attributes)
                        {
                            Console.WriteLine($"    {attr.Key}: {attr.Value}");
                        }
                    }
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No IoT Things found in your account.");
                Console.WriteLine("You can create IoT Things using the IoT Basics
scenario example.");
            }

            Console.WriteLine("Hello IoT completed successfully.");
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
```

```
        {
            Console.WriteLine($"Request throttled, please try again later:
 {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list Things. Here's why: {ex.Message}");
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK für .NET -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Schatten.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche eine AWS IoT Sache.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS IoT Funktionen demonstriert.

```
/// <summary>
/// Scenario class for AWS IoT basics.
/// </summary>
public class IoTBasics
{
    public static bool IsInteractive = true;
    public static IoTWrapper? Wrapper = null;
    public static IAmazonCloudFormation? CloudFormationClient = null;
    public static ILogger<IoTBasics> logger = null!;
    private static IoTWrapper _iotWrapper = null!;
    private static IAmazonCloudFormation _amazonCloudFormation = null!;
    private static ILogger<IoTBasics> _logger = null!;

    private static string _stackName = "IoTBasicsStack";
    private static string _stackResourcePath = "../../../../../../scenarios/basics/
iot/iot_usecase/resources/cfn_template.yaml";

    /// <summary>
    /// Main method for the IoT Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIoT>(new AWSOptions() { Region =
 RegionEndpoint.USEast1 })
                    .AddAWSService<IAmazonCloudFormation>()
                        .AddTransient<IoTWrapper>()
                        .AddLogging(builder => builder.AddConsole())
                        .AddSingleton<IAmazonIotData>(sp =>
```

```
                        {
                            var iotService = sp.GetRequiredService<IAmazonIoT>();
                            var request = new DescribeEndpointRequest
                            {
                                EndpointType = "iot:Data-ATS"
                            };
                            var response =
 iotService.DescribeEndpointAsync(request).Result;
                            return new AmazonIotDataClient($"https://
{response.EndpointAddress}/");
                        })
            )
            .Build();

        logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<IoTBasics>();

        Wrapper = host.Services.GetRequiredService<IoTWrapper>();
        CloudFormationClient =
 host.Services.GetRequiredService<IAmazonCloudFormation>();

        // Set the private fields for backwards compatibility
        _logger = logger;
        _iotWrapper = Wrapper;
        _amazonCloudFormation = CloudFormationClient;

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the AWS IoT example scenario.");
        Console.WriteLine("This example program demonstrates various interactions
 with the AWS Internet of Things (IoT) Core service.");
        Console.WriteLine();
        if (IsInteractive)
        {
            Console.WriteLine("Press Enter to continue...");
            Console.ReadLine();
        }
        Console.WriteLine(new string('-', 80));

        try
        {
            await RunScenarioAsync();
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, "There was a problem running the scenario.");
            Console.WriteLine($"\nAn error occurred: {ex.Message}");
        }

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("The AWS IoT scenario has successfully completed.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Run the IoT Basics scenario.
    /// </summary>
    /// <returns>A Task object.</returns>
    public static async Task RunScenarioAsync()
    {
        // Use static properties if available, otherwise use private fields
        var iotWrapper = Wrapper ?? _iotWrapper;
        var cloudFormationClient = CloudFormationClient ?? _amazonCloudFormation;
        var scenarioLogger = logger ?? _logger;

        await RunScenarioInternalAsync(iotWrapper, cloudFormationClient,
 scenarioLogger);
    }

    /// <summary>
    /// Internal method to run the IoT Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="iotWrapper">The IoT wrapper instance.</param>
    /// <param name="cloudFormationClient">The CloudFormation client instance.</
param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    private static async Task RunScenarioInternalAsync(IoTWrapper iotWrapper,
 IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        string thingName = $"iot-thing-{Guid.NewGuid():N}";
        string certificateArn = "";
        string certificateId = "";
        string ruleName = $"iotruledefault";
        string snsTopicArn = "";

        try
        {
            // Step 1: Create an AWS IoT Thing
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("1. Create an AWS IoT Thing.");
            Console.WriteLine("An AWS IoT Thing represents a virtual entity in the
AWS IoT service that can be associated with a physical device.");
            Console.WriteLine();

            if (IsInteractive)
            {
                Console.Write("Enter Thing name: ");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    thingName = userInput;
            }
            else
            {
                Console.WriteLine($"Using default Thing name: {thingName}");
            }

            var thingArn = await iotWrapper.CreateThingAsync(thingName);
            Console.WriteLine($"{thingName} was successfully created. The ARN value
is {thingArn}");
            Console.WriteLine(new string('-', 80));

            // Step 1.1: List AWS IoT Things
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("2. List AWS IoT Things.");
            Console.WriteLine("Now let's list the IoT Things to see the Thing we
just created.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var things = await iotWrapper.ListThingsAsync();
            Console.WriteLine($"Found {things.Count} IoT Things:");
            foreach (var thing in things.Take(10)) // Show first 10 things
            {
                Console.WriteLine($"Thing Name: {thing.ThingName}");
                Console.WriteLine($"Thing ARN: {thing.ThingArn}");
                if (thing.Attributes != null && thing.Attributes.Any())
                {
                    Console.WriteLine("Attributes:");
```

```
                    foreach (var attr in thing.Attributes)
                    {
                        Console.WriteLine($"  {attr.Key}: {attr.Value}");
                    }
                }
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 2: Generate a Device Certificate
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("3. Generate a device certificate.");
            Console.WriteLine("A device certificate performs a role in securing the
communication between devices (Things) and the AWS IoT platform.");
            Console.WriteLine();

            var createCert = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to create a certificate for {thingName}?
(y/n)");
                createCert = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Creating certificate for {thingName}...");
            }

            if (createCert?.ToLower() == "y")
            {
                var certificateResult = await
iotWrapper.CreateKeysAndCertificateAsync();
                if (certificateResult.HasValue)
                {
                    var (certArn, certPem, certId) = certificateResult.Value;
                    certificateArn = certArn;
                    certificateId = certId;

                    Console.WriteLine($"\nCertificate:");
                    // Show only first few lines of certificate for brevity
                    var lines = certPem.Split('\n');
                    for (int i = 0; i < Math.Min(lines.Length, 5); i++)
                    {
```

```
                        Console.WriteLine(lines[i]);
                    }
                    if (lines.Length > 5)
                    {
                        Console.WriteLine("...");
                    }

                    Console.WriteLine($"\nCertificate ARN:");
                    Console.WriteLine(certificateArn);

                    // Step 3: Attach the Certificate to the AWS IoT Thing
                    Console.WriteLine("Attach the certificate to the AWS IoT
Thing.");
                    var attachResult = await
iotWrapper.AttachThingPrincipalAsync(thingName, certificateArn);
                    if (attachResult)
                    {
                        Console.WriteLine("Certificate attached to Thing
successfully.");
                    }
                    else
                    {
                        Console.WriteLine("Failed to attach certificate to Thing.");
                    }

                    Console.WriteLine("Thing Details:");
                    Console.WriteLine($"Thing Name: {thingName}");
                    Console.WriteLine($"Thing ARN: {thingArn}");
                }
                else
                {
                    Console.WriteLine("Failed to create certificate.");
                }
            }
            Console.WriteLine(new string('-', 80));

            // Step 4: Update an AWS IoT Thing with Attributes
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("4. Update an AWS IoT Thing with Attributes.");
            Console.WriteLine("IoT Thing attributes, represented as key-value pairs,
offer a pivotal advantage in facilitating efficient data");
            Console.WriteLine("management and retrieval within the AWS IoT
ecosystem.");
            Console.WriteLine();
```

```
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var attributes = new Dictionary<string, string>
            {
                { "Location", "Seattle" },
                { "DeviceType", "Sensor" },
                { "Firmware", "1.2.3" }
            };

            await iotWrapper.UpdateThingAsync(thingName, attributes);
            Console.WriteLine("Thing attributes updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 5: Return a unique endpoint specific to the Amazon Web Services
account
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("5. Return a unique endpoint specific to the Amazon
Web Services account.");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var endpoint = await iotWrapper.DescribeEndpointAsync();
            if (endpoint != null)
            {
                var subdomain = endpoint.Split('.')[0];
                Console.WriteLine($"Extracted subdomain: {subdomain}");
                Console.WriteLine($"Full Endpoint URL: https://{endpoint}");
            }
            else
            {
                Console.WriteLine("Failed to retrieve endpoint.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 6: List your AWS IoT certificates
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("6. List your AWS IoT certificates");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var certificates = await iotWrapper.ListCertificatesAsync();
            foreach (var cert in certificates.Take(5)) // Show first 5 certificates
            {
                Console.WriteLine($"Cert id: {cert.CertificateId}");
                Console.WriteLine($"Cert Arn: {cert.CertificateArn}");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 7: Create an IoT shadow
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("7. Update an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
            Console.WriteLine();
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowPayload = JsonSerializer.Serialize(new
            {
                state = new
                {
                    desired = new
                    {
                        temperature = 25,
                        humidity = 50
                    }
                }
            });

            await iotWrapper.UpdateThingShadowAsync(thingName, shadowPayload);
            Console.WriteLine("Thing Shadow updated successfully.");
            Console.WriteLine(new string('-', 80));

            // Step 8: Write out the state information, in JSON format
```

```
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("8. Write out the state information, in JSON
format.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var shadowData = await iotWrapper.GetThingShadowAsync(thingName);
            Console.WriteLine($"Received Shadow Data: {shadowData}");
            Console.WriteLine(new string('-', 80));

            // Step 9: Set up resources (SNS topic and IAM role) and create a rule
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("9. Set up resources and create a rule");
            Console.WriteLine();

            // Deploy CloudFormation stack to create SNS topic and IAM role
            Console.WriteLine("Deploying CloudFormation stack to create SNS topic
and IAM role...");

            var deployStack = !IsInteractive || GetYesNoResponse("Would you like to
deploy the CloudFormation stack? (y/n) ");
            if (deployStack)
            {
                if (IsInteractive)
                {
                    Console.Write(
                        $"Enter stack resource file path (or press Enter for default
'{_stackResourcePath}'): ");
                    var userResourcePath = Console.ReadLine();
                    if (!string.IsNullOrEmpty(userResourcePath))
                        _stackResourcePath = userResourcePath;
                }

                _stackName = PromptUserForStackName();

                var deploySuccess = await DeployCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);

                if (deploySuccess)
                {
                    // Get stack outputs
```

```
                    var stackOutputs = await GetStackOutputs(_stackName,
cloudFormationClient, scenarioLogger);
                    if (stackOutputs != null)
                    {
                        snsTopicArn = stackOutputs["SNSTopicArn"];
                        string roleArn = stackOutputs["RoleArn"];

                        Console.WriteLine($"Successfully deployed stack. SNS topic:
{snsTopicArn}");
                        Console.WriteLine($"Successfully deployed stack. IAM role:
{roleArn}");

                        if (IsInteractive)
                        {
                            Console.Write($"Enter Rule name (press Enter for default
'{ruleName}'): ");
                            var userRuleName = Console.ReadLine();
                            if (!string.IsNullOrEmpty(userRuleName))
                                ruleName = userRuleName;
                        }
                        else
                        {
                            Console.WriteLine($"Using default rule name:
{ruleName}");
                        }

                        // Now create the IoT rule with the CloudFormation outputs
                        var ruleResult = await
iotWrapper.CreateTopicRuleAsync(ruleName, snsTopicArn, roleArn);
                        if (ruleResult)
                        {
                            Console.WriteLine("IoT Rule created successfully.");
                        }
                        else
                        {
                            Console.WriteLine("Failed to create IoT rule.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Failed to get stack outputs. Skipping
rule creation.");
                    }
                }
```

```
                else
                {
                    Console.WriteLine("Failed to deploy CloudFormation stack.
 Skipping rule creation.");
                }
            }
            else
            {
                Console.WriteLine("Skipping CloudFormation stack deployment and rule
 creation.");
            }
            Console.WriteLine(new string('-', 80));

            // Step 10: List your rules
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("10. List your rules.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var rules = await iotWrapper.ListTopicRulesAsync();
            Console.WriteLine("List of IoT Rules:");
            foreach (var rule in rules.Take(5)) // Show first 5 rules
            {
                Console.WriteLine($"Rule Name: {rule.RuleName}");
                Console.WriteLine($"Rule ARN: {rule.RuleArn}");
                Console.WriteLine("--------------");
            }
            Console.WriteLine();
            Console.WriteLine(new string('-', 80));

            // Step 11: Search things using the Thing name
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("11. Search things using the Thing name.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

            var searchResults = await iotWrapper.SearchIndexAsync($"thingName:
{thingName}");
```

```
            if (searchResults.Any())
            {
                Console.WriteLine($"Thing id found using search is
{searchResults.First().ThingId}");
            }
            else
            {
                Console.WriteLine($"No search results found for Thing:
{thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 12: Cleanup - Detach and delete certificate
            if (!string.IsNullOrEmpty(certificateArn))
            {
                Console.WriteLine(new string('-', 80));
                var deleteCert = "y";
                if (IsInteractive)
                {
                    Console.Write($"Do you want to detach and delete the certificate
for {thingName}? (y/n)");
                    deleteCert = Console.ReadLine();
                }
                else
                {
                    Console.WriteLine($"Detaching and deleting certificate for
{thingName}...");
                }

                if (deleteCert?.ToLower() == "y")
                {
                    Console.WriteLine("12. You selected to detach and delete the
certificate.");
                    if (IsInteractive)
                    {
                        Console.WriteLine("Press Enter to continue...");
                        Console.ReadLine();
                    }

                    await iotWrapper.DetachThingPrincipalAsync(thingName,
certificateArn);
                    Console.WriteLine($"{certificateArn} was successfully removed
from {thingName}");
```

```
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                    Console.WriteLine($"{certificateArn} was successfully
deleted.");
                }
                Console.WriteLine(new string('-', 80));
            }

            // Step 13: Delete the AWS IoT Thing
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("13. Delete the AWS IoT Thing.");
            var deleteThing = "y";
            if (IsInteractive)
            {
                Console.Write($"Do you want to delete the IoT Thing? (y/n)");
                deleteThing = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Deleting IoT Thing {thingName}...");
            }

            if (deleteThing?.ToLower() == "y")
            {
                await iotWrapper.DeleteThingAsync(thingName);
                Console.WriteLine($"Deleted Thing {thingName}");
            }
            Console.WriteLine(new string('-', 80));

            // Step 14: Clean up CloudFormation stack
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                Console.WriteLine(new string('-', 80));
                Console.WriteLine("14. Clean up CloudFormation stack.");
                Console.WriteLine("Deleting the CloudFormation stack and all
resources...");

                var cleanup = !IsInteractive || GetYesNoResponse("Do you want to
delete the CloudFormation stack and all resources? (y/n) ");
                if (cleanup)
                {
                    var ruleCleanupSuccess = await
iotWrapper.DeleteTopicRuleAsync(ruleName);
```

```
                    var stackCleanupSuccess = await
 DeleteCloudFormationStack(_stackName, cloudFormationClient, scenarioLogger);
                    if (ruleCleanupSuccess && stackCleanupSuccess)
                    {
                        Console.WriteLine("Successfully cleaned up CloudFormation
 stack and all resources.");
                    }
                    else
                    {
                        Console.WriteLine("Some cleanup operations failed. Check the
 logs for details.");
                    }
                }
                else
                {
                    Console.WriteLine($"Resources will remain. Stack name:
 {_stackName}");
                }
                Console.WriteLine(new string('-', 80));
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "Error occurred during scenario
 execution.");

            // Cleanup on error
            if (!string.IsNullOrEmpty(certificateArn) && !
string.IsNullOrEmpty(thingName))
            {
                try
                {
                    await iotWrapper.DetachThingPrincipalAsync(thingName,
 certificateArn);
                    await iotWrapper.DeleteCertificateAsync(certificateId);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
                }
            }

            if (!string.IsNullOrEmpty(thingName))
            {
```

```
                try
                {
                    await iotWrapper.DeleteThingAsync(thingName);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during Thing
cleanup.");
                }
            }

            // Clean up CloudFormation stack on error
            if (!string.IsNullOrEmpty(snsTopicArn))
            {
                try
                {
                    await _iotWrapper.DeleteTopicRuleAsync(ruleName);
                    await DeleteCloudFormationStack(_stackName,
cloudFormationClient, scenarioLogger);
                }
                catch (Exception cleanupEx)
                {
                    scenarioLogger.LogError(cleanupEx, "Error during CloudFormation
stack cleanup.");
                }
            }

            throw;
        }
    }

    /// <summary>
    /// Deploys the CloudFormation stack with the necessary resources.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deployed successfully.</returns>
    private static async Task<bool> DeployCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        Console.WriteLine($"\nDeploying CloudFormation stack: {stackName}");

        try
```

```
        {
            var request = new CreateStackRequest
            {
                StackName = stackName,
                TemplateBody = await File.ReadAllTextAsync(_stackResourcePath),
                Capabilities = new List<string> { Capability.CAPABILITY_NAMED_IAM }
            };

            var response = await cloudFormationClient.CreateStackAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"CloudFormation stack creation started:
{stackName}");

                bool stackCreated = await WaitForStackCompletion(response.StackId,
cloudFormationClient, scenarioLogger);

                if (stackCreated)
                {
                    Console.WriteLine("CloudFormation stack created successfully.");
                    return true;
                }
                else
                {
                    scenarioLogger.LogError($"CloudFormation stack creation failed:
{stackName}");
                    return false;
                }
            }
            else
            {
                scenarioLogger.LogError($"Failed to create CloudFormation stack:
{stackName}");
                return false;
            }
        }
        catch (AlreadyExistsException)
        {
            scenarioLogger.LogWarning($"CloudFormation stack '{stackName}' already
exists. Please provide a unique name.");
            var newStackName = PromptUserForStackName();
            return await DeployCloudFormationStack(newStackName,
cloudFormationClient, scenarioLogger);
```

```
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deploying the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the CloudFormation stack to be in the CREATE_COMPLETE state.
    /// </summary>
    /// <param name="stackId">The ID of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was created successfully.</returns>
    private static async Task<bool> WaitForStackCompletion(string stackId,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackId
            };

            var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (describeStacksResponse.Stacks.Count > 0)
            {
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_COMPLETE)
                {
                    return true;
                }
                if (describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.CREATE_FAILED ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.ROLLBACK_COMPLETE)
```

```
                {
                    return false;
                }
            }

            Console.WriteLine("Waiting for CloudFormation stack creation to
complete...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError("Timed out waiting for CloudFormation stack creation
to complete.");
        return false;
    }

    /// <summary>
    /// Gets the outputs from the CloudFormation stack.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>A dictionary of stack outputs.</returns>
    private static async Task<Dictionary<string, string>?> GetStackOutputs(string
stackName, IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics>
scenarioLogger)
    {
        try
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            var response = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

            if (response.Stacks.Count > 0)
            {
                var outputs = new Dictionary<string, string>();
                foreach (var output in response.Stacks[0].Outputs)
                {
                    outputs[output.OutputKey] = output.OutputValue;
                }
```

```
                return outputs;
            }

            return null;
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"Failed to get stack outputs for
{stackName}");
            return null;
        }
    }

    /// <summary>
    /// Deletes the CloudFormation stack and waits for confirmation.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> DeleteCloudFormationStack(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        try
        {
            var request = new DeleteStackRequest
            {
                StackName = stackName
            };

            await cloudFormationClient.DeleteStackAsync(request);
            Console.WriteLine($"CloudFormation stack '{stackName}' is being deleted.
This may take a few minutes.");

            bool stackDeleted = await WaitForStackDeletion(stackName,
cloudFormationClient, scenarioLogger);

            if (stackDeleted)
            {
                Console.WriteLine($"CloudFormation stack '{stackName}' has been
deleted.");
                return true;
            }
            else
```

```
            {
                scenarioLogger.LogError($"Failed to delete CloudFormation stack
'{stackName}'.");
                return false;
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, $"An error occurred while deleting the
CloudFormation stack: {stackName}");
            return false;
        }
    }

    /// <summary>
    /// Waits for the stack to be deleted.
    /// </summary>
    /// <param name="stackName">The name of the CloudFormation stack.</param>
    /// <param name="cloudFormationClient">The CloudFormation client.</param>
    /// <param name="scenarioLogger">The logger.</param>
    /// <returns>True if the stack was deleted successfully.</returns>
    private static async Task<bool> WaitForStackDeletion(string stackName,
IAmazonCloudFormation cloudFormationClient, ILogger<IoTBasics> scenarioLogger)
    {
        int retryCount = 0;
        const int maxRetries = 30;
        const int retryDelay = 10000;

        while (retryCount < maxRetries)
        {
            var describeStacksRequest = new DescribeStacksRequest
            {
                StackName = stackName
            };

            try
            {
                var describeStacksResponse = await
cloudFormationClient.DescribeStacksAsync(describeStacksRequest);

                if (describeStacksResponse.Stacks.Count == 0 ||
                    describeStacksResponse.Stacks[0].StackStatus ==
StackStatus.DELETE_COMPLETE)
                {
```

```
                    return true;
                }
            }
            catch (AmazonCloudFormationException ex) when (ex.ErrorCode ==
"ValidationError")
            {
                return true;
            }

            Console.WriteLine($"Waiting for CloudFormation stack '{stackName}' to be
deleted...");
            await Task.Delay(retryDelay);
            retryCount++;
        }

        scenarioLogger.LogError($"Timed out waiting for CloudFormation stack
'{stackName}' to be deleted.");
        return false;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Prompts the user for a stack name.
    /// </summary>
    private static string PromptUserForStackName()
    {
        if (IsInteractive)
        {
            Console.Write($"Enter a name for the CloudFormation stack (press Enter
for default '{_stackName}'): ");
            string? input = Console.ReadLine();
            if (!string.IsNullOrWhiteSpace(input))
            {
```

```
                var regex = new System.Text.RegularExpressions.Regex("[a-zA-Z][-a-
zA-Z0-9]*");
                if (!regex.IsMatch(input))
                {
                    Console.WriteLine($"Invalid stack name. Using default:
 {_stackName}");
                    return _stackName;
                }
                return input;
            }
        }
        return _stackName;
    }
}
```

Eine Wrapper-Klasse für AWS IoT SDK-Methoden.

```
/// <summary>
/// Wrapper methods to use Amazon IoT Core with .NET.
/// </summary>
public class IoTWrapper
{
    private readonly IAmazonIoT _amazonIoT;
    private readonly IAmazonIotData _amazonIotData;
    private readonly ILogger<IoTWrapper> _logger;

    /// <summary>
    /// Constructor for the IoT wrapper.
    /// </summary>
    /// <param name="amazonIoT">The injected IoT client.</param>
    /// <param name="amazonIotData">The injected IoT Data client.</param>
    /// <param name="logger">The injected logger.</param>
    public IoTWrapper(IAmazonIoT amazonIoT, IAmazonIotData amazonIotData,
 ILogger<IoTWrapper> logger)
    {
        _amazonIoT = amazonIoT;
        _amazonIotData = amazonIotData;
        _logger = logger;
    }

    /// <summary>
    /// Creates an AWS IoT Thing.
```

```
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or null
if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };
```

```
            var response = await _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Attaches a certificate to an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to attach.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
        {
            var request = new AttachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.AttachThingPrincipalAsync(request);
            _logger.LogInformation($"Attached certificate {certificateArn} to Thing
{thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
```

```
        {
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
```

```
    /// <returns>List of certificate information, or empty list if listing failed.</
returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
 certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }

    /// <summary>
    /// Updates the Thing's shadow with new state information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="shadowPayload">The shadow payload in JSON format.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingShadowAsync(string thingName, string
 shadowPayload)
    {
        try
        {
            var request = new UpdateThingShadowRequest
            {
                ThingName = thingName,
                Payload = new
 MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
            };
```

```
            await _amazonIotData.UpdateThingShadowAsync(request);
            _logger.LogInformation($"Updated shadow for Thing {thingName}");
            return true;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
{ex.Message}");
```

```
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
{ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Creates an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };
```

```
            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT topic rule.
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteTopicRuleAsync(string ruleName)
    {
        try
        {
            var request = new DeleteTopicRuleRequest
            {
                RuleName = ruleName,
            };

            await _amazonIoT.DeleteTopicRuleAsync(request);
            _logger.LogInformation($"Deleted IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogWarning($"Rule {ruleName} not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete topic rule. Here's why:
{ex.Message}");
```

```
            return false;
        }
    }

    /// <summary>
    /// Lists all IoT topic rules.
    /// </summary>
    /// <returns>List of topic rules, or empty list if listing failed.</returns>
    public async Task<List<TopicRuleListItem>> ListTopicRulesAsync()
    {
        try
        {
            var request = new ListTopicRulesRequest();
            var response = await _amazonIoT.ListTopicRulesAsync(request);

            _logger.LogInformation($"Retrieved {response.Rules.Count} IoT rules");
            return response.Rules;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list topic rules. Here's why:
{ex.Message}");
            return new List<TopicRuleListItem>();
        }
    }

    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
```

```
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count} Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting for
the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
    private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
    {
```

```
        try
        {
            // Update indexing configuration to REGISTRY mode
            _logger.LogInformation("Setting up IoT search indexing
configuration...");
            await _amazonIoT.UpdateIndexingConfigurationAsync(
                new UpdateIndexingConfigurationRequest()
                {
                    ThingIndexingConfiguration = new ThingIndexingConfiguration()
                    {
                        ThingIndexingMode = ThingIndexingMode.REGISTRY
                    }
                });

            _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

            // Wait for the index to be set up - this can take some time
            const int maxRetries = 10;
            const int retryDelaySeconds = 10;

            for (int attempt = 1; attempt <= maxRetries; attempt++)
            {
                try
                {
                    _logger.LogInformation($"Waiting for index to be ready (attempt
{attempt}/{maxRetries})...");
                    await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                    // Try to get the current indexing configuration to see if it's
ready
                    var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                    if (configResponse.ThingIndexingConfiguration?.ThingIndexingMode
== ThingIndexingMode.REGISTRY)
                    {
                        // Try the search again
                        var request = new SearchIndexRequest
                        {
                            QueryString = queryString
                        };

                        var response = await _amazonIoT.SearchIndexAsync(request);
```

```
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready after
configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
```

```
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
```

```
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
```

```
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
                else
                    hasMoreResults = false;
```

```
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Attaches a certificate to an IoT Thing.
   /// </summary>
   /// <param name="thingName">The name of the Thing.</param>
   /// <param name="certificateArn">The ARN of the certificate to attach.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> AttachThingPrincipalAsync(string thingName, string
certificateArn)
   {
       try
       {
           var request = new AttachThingPrincipalRequest
           {
               ThingName = thingName,
               Principal = certificateArn
           };

           await _amazonIoT.AttachThingPrincipalAsync(request);
           _logger.LogInformation($"Attached certificate {certificateArn} to Thing
{thingName}");
           return true;
       }
       catch (Amazon.IoT.Model.ResourceNotFoundException ex)
       {
```

```
            _logger.LogError($"Cannot attach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't attach certificate to Thing. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie AttachThingPrincipal in der AWS SDK für .NET API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates a device certificate for AWS IoT.
    /// </summary>
    /// <returns>The certificate details including ARN and certificate PEM, or null
if creation failed.</returns>
    public async Task<(string CertificateArn, string CertificatePem, string
CertificateId)?> CreateKeysAndCertificateAsync()
    {
        try
        {
            var request = new CreateKeysAndCertificateRequest
            {
                SetAsActive = true
            };
```

```
            var response = await _amazonIoT.CreateKeysAndCertificateAsync(request);
            _logger.LogInformation($"Created certificate with ARN
{response.CertificateArn}");
            return (response.CertificateArn, response.CertificatePem,
response.CertificateId);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create certificate. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS SDK für .NET API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an AWS IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to create.</param>
    /// <returns>The ARN of the Thing created, or null if creation failed.</returns>
    public async Task<string?> CreateThingAsync(string thingName)
    {
```

```
        try
        {
            var request = new CreateThingRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIoT.CreateThingAsync(request);
            _logger.LogInformation($"Created Thing {thingName} with ARN
{response.ThingArn}");
            return response.ThingArn;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Thing {thingName} already exists: {ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't create Thing {thingName}. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie CreateThing in der AWS SDK für .NET API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Creates an IoT topic rule.
```

```
    /// </summary>
    /// <param name="ruleName">The name of the rule.</param>
    /// <param name="snsTopicArn">The ARN of the SNS topic for the action.</param>
    /// <param name="roleArn">The ARN of the IAM role.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> CreateTopicRuleAsync(string ruleName, string
snsTopicArn, string roleArn)
    {
        try
        {
            var request = new CreateTopicRuleRequest
            {
                RuleName = ruleName,
                TopicRulePayload = new TopicRulePayload
                {
                    Sql = "SELECT * FROM 'topic/subtopic'",
                    Description = $"Rule created by .NET example: {ruleName}",
                    Actions = new List<Amazon.IoT.Model.Action>
                    {
                        new Amazon.IoT.Model.Action
                        {
                            Sns = new SnsAction
                            {
                                TargetArn = snsTopicArn,
                                RoleArn = roleArn
                            }
                        }
                    },
                    RuleDisabled = false
                }
            };

            await _amazonIoT.CreateTopicRuleAsync(request);
            _logger.LogInformation($"Created IoT rule {ruleName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceAlreadyExistsException ex)
        {
            _logger.LogWarning($"Rule {ruleName} already exists: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError($"Couldn't create topic rule. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie CreateTopicRule in der AWS SDK für .NET API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an IoT certificate.
    /// </summary>
    /// <param name="certificateId">The ID of the certificate to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteCertificateAsync(string certificateId)
    {
        try
        {
            // First, update the certificate to inactive state
            var updateRequest = new UpdateCertificateRequest
            {
                CertificateId = certificateId,
                NewStatus = CertificateStatus.INACTIVE
            };
            await _amazonIoT.UpdateCertificateAsync(updateRequest);

            // Then delete the certificate
            var deleteRequest = new DeleteCertificateRequest
            {
                CertificateId = certificateId
```

```
            };

            await _amazonIoT.DeleteCertificateAsync(deleteRequest);
            _logger.LogInformation($"Deleted certificate {certificateId}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete certificate. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteCertificate in der AWS SDK für .NET API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Deletes an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing to delete.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DeleteThingAsync(string thingName)
    {
```

```
        try
        {
            var request = new DeleteThingRequest
            {
                ThingName = thingName
            };

            await _amazonIoT.DeleteThingAsync(request);
            _logger.LogInformation($"Deleted Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot delete Thing - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't delete Thing. Here's why: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteThing in der AWS SDK für .NET API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the AWS IoT endpoint URL.
    /// </summary>
```

```
    /// <returns>The endpoint URL, or null if retrieval failed.</returns>
    public async Task<string?> DescribeEndpointAsync()
    {
        try
        {
            var request = new DescribeEndpointRequest
            {
                EndpointType = "iot:Data-ATS"
            };

            var response = await _amazonIoT.DescribeEndpointAsync(request);
            _logger.LogInformation($"Retrieved endpoint:
{response.EndpointAddress}");
            return response.EndpointAddress;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't describe endpoint. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK für .NET API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Detaches a certificate from an IoT Thing.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <param name="certificateArn">The ARN of the certificate to detach.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> DetachThingPrincipalAsync(string thingName, string
certificateArn)
    {
        try
        {
            var request = new DetachThingPrincipalRequest
            {
                ThingName = thingName,
                Principal = certificateArn
            };

            await _amazonIoT.DetachThingPrincipalAsync(request);
            _logger.LogInformation($"Detached certificate {certificateArn} from
Thing {thingName}");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot detach certificate - resource not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't detach certificate from Thing. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DetachThingPrincipal in der AWS SDK für .NET API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists all certificates associated with the account.
    /// </summary>
    /// <returns>List of certificate information, or empty list if listing failed.</
returns>
    public async Task<List<Certificate>> ListCertificatesAsync()
    {
        try
        {
            var request = new ListCertificatesRequest();
            var response = await _amazonIoT.ListCertificatesAsync(request);

            _logger.LogInformation($"Retrieved {response.Certificates.Count}
 certificates");
            return response.Certificates;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
 {ex.Message}");
            return new List<Certificate>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list certificates. Here's why:
 {ex.Message}");
            return new List<Certificate>();
        }
    }
```

- API-Details finden Sie ListCertificates in der AWS SDK für .NET API-Referenz.

`ListThings`

Das folgende Codebeispiel zeigt die Verwendung`ListThings`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Lists IoT Things with pagination support.
    /// </summary>
    /// <returns>List of Things, or empty list if listing failed.</returns>
    public async Task<List<ThingAttribute>> ListThingsAsync()
    {
        try
        {
            // Use pages of 10.
            var request = new ListThingsRequest()
            {
                MaxResults = 10
            };
            var response = await _amazonIoT.ListThingsAsync(request);

            // Since there is not a built-in paginator, use the NextMarker to
paginate.
            bool hasMoreResults = true;

            var things = new List<ThingAttribute>();
            while (hasMoreResults)
            {
                things.AddRange(response.Things);

                // If NextMarker is not null, there are more results. Get the next
page of results.
                if (!String.IsNullOrEmpty(response.NextMarker))
                {
                    request.Marker = response.NextMarker;
                    response = await _amazonIoT.ListThingsAsync(request);
                }
```

```
                else
                    hasMoreResults = false;
            }

            _logger.LogInformation($"Retrieved {things.Count} Things");
            return things;
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingAttribute>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't list Things. Here's why: {ex.Message}");
            return new List<ThingAttribute>();
        }
    }
```

- API-Details finden Sie ListThings in der AWS SDK für .NET API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Searches for IoT Things using the search index.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
    /// <returns>List of Things that match the search criteria, or empty list if
search failed.</returns>
    public async Task<List<ThingDocument>> SearchIndexAsync(string queryString)
```

```
    {
        try
        {
            // First, try to perform the search
            var request = new SearchIndexRequest
            {
                QueryString = queryString
            };

            var response = await _amazonIoT.SearchIndexAsync(request);
            _logger.LogInformation($"Search found {response.Things.Count} Things");
            return response.Things;
        }
        catch (Amazon.IoT.Model.IndexNotReadyException ex)
        {
            _logger.LogWarning($"Search index not ready, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
        {
            _logger.LogWarning($"Search index not configured, setting up indexing
configuration: {ex.Message}");
            return await SetupIndexAndRetrySearchAsync(queryString);
        }
        catch (Amazon.IoT.Model.ThrottlingException ex)
        {
            _logger.LogWarning($"Request throttled, please try again later:
{ex.Message}");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't search index. Here's why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }

    /// <summary>
    /// Sets up the indexing configuration and retries the search after waiting for
the index to be ready.
    /// </summary>
    /// <param name="queryString">The search query string.</param>
```

```
   /// <returns>List of Things that match the search criteria, or empty list if
setup/search failed.</returns>
   private async Task<List<ThingDocument>> SetupIndexAndRetrySearchAsync(string
queryString)
   {
       try
       {
           // Update indexing configuration to REGISTRY mode
           _logger.LogInformation("Setting up IoT search indexing
configuration...");
           await _amazonIoT.UpdateIndexingConfigurationAsync(
               new UpdateIndexingConfigurationRequest()
               {
                   ThingIndexingConfiguration = new ThingIndexingConfiguration()
                   {
                       ThingIndexingMode = ThingIndexingMode.REGISTRY
                   }
               });

           _logger.LogInformation("Indexing configuration updated. Waiting for
index to be ready...");

           // Wait for the index to be set up - this can take some time
           const int maxRetries = 10;
           const int retryDelaySeconds = 10;

           for (int attempt = 1; attempt <= maxRetries; attempt++)
           {
               try
               {
                   _logger.LogInformation($"Waiting for index to be ready (attempt
{attempt}/{maxRetries})...");
                   await Task.Delay(TimeSpan.FromSeconds(retryDelaySeconds));

                   // Try to get the current indexing configuration to see if it's
ready
                   var configResponse = await
_amazonIoT.GetIndexingConfigurationAsync(new GetIndexingConfigurationRequest());
                   if (configResponse.ThingIndexingConfiguration?.ThingIndexingMode
== ThingIndexingMode.REGISTRY)
                   {
                       // Try the search again
                       var request = new SearchIndexRequest
                       {
```

```
                            QueryString = queryString
                        };

                        var response = await _amazonIoT.SearchIndexAsync(request);
                        _logger.LogInformation($"Search found
{response.Things.Count} Things after index setup");
                        return response.Things;
                    }
                }
                catch (Amazon.IoT.Model.IndexNotReadyException)
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
                catch (Amazon.IoT.Model.InvalidRequestException ex) when
(ex.Message.Contains("index") || ex.Message.Contains("Index"))
                {
                    // Index still not ready, continue waiting
                    _logger.LogInformation("Index still not ready, continuing to
wait...");
                    continue;
                }
            }

            _logger.LogWarning("Timeout waiting for search index to be ready after
configuration update");
            return new List<ThingDocument>();
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't set up search index configuration. Here's
why: {ex.Message}");
            return new List<ThingDocument>();
        }
    }
```

- API-Details finden Sie SearchIndex in der AWS SDK für .NET API-Referenz.

`UpdateThing`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThing`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Updates an IoT Thing with attributes.
    /// </summary>
    /// <param name="thingName">The name of the Thing to update.</param>
    /// <param name="attributes">Dictionary of attributes to add.</param>
    /// <returns>True if successful, false otherwise.</returns>
    public async Task<bool> UpdateThingAsync(string thingName, Dictionary<string,
string> attributes)
    {
        try
        {
            var request = new UpdateThingRequest
            {
                ThingName = thingName,
                AttributePayload = new AttributePayload
                {
                    Attributes = attributes,
                    Merge = true
                }
            };

            await _amazonIoT.UpdateThingAsync(request);
            _logger.LogInformation($"Updated Thing {thingName} with attributes");
            return true;
        }
        catch (Amazon.IoT.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot update Thing - resource not found:
{ex.Message}");
            return false;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't update Thing attributes. Here's why:
{ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie UpdateThing in der AWS SDK für .NET API-Referenz.

AWS IoT data Beispiele unter Verwendung SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Gets the Thing's shadow information.
    /// </summary>
    /// <param name="thingName">The name of the Thing.</param>
    /// <returns>The shadow data as a string, or null if retrieval failed.</returns>
    public async Task<string?> GetThingShadowAsync(string thingName)
    {
        try
        {
            var request = new GetThingShadowRequest
            {
                ThingName = thingName
            };

            var response = await _amazonIotData.GetThingShadowAsync(request);
            using var reader = new StreamReader(response.Payload);
            var shadowData = await reader.ReadToEndAsync();

            _logger.LogInformation($"Retrieved shadow for Thing {thingName}");
            return shadowData;
        }
        catch (Amazon.IotData.Model.ResourceNotFoundException ex)
        {
            _logger.LogError($"Cannot get Thing shadow - resource not found:
{ex.Message}");
            return null;
        }
        catch (Exception ex)
        {
            _logger.LogError($"Couldn't get Thing shadow. Here's why:
{ex.Message}");
            return null;
        }
    }
```

- API-Details finden Sie GetThingShadow in der AWS SDK für .NET API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// <summary>
   /// Updates the Thing's shadow with new state information.
   /// </summary>
   /// <param name="thingName">The name of the Thing.</param>
   /// <param name="shadowPayload">The shadow payload in JSON format.</param>
   /// <returns>True if successful, false otherwise.</returns>
   public async Task<bool> UpdateThingShadowAsync(string thingName, string
shadowPayload)
   {
       try
       {
           var request = new UpdateThingShadowRequest
           {
               ThingName = thingName,
               Payload = new
MemoryStream(System.Text.Encoding.UTF8.GetBytes(shadowPayload))
           };

           await _amazonIotData.UpdateThingShadowAsync(request);
           _logger.LogInformation($"Updated shadow for Thing {thingName}");
           return true;
       }
       catch (Amazon.IotData.Model.ResourceNotFoundException ex)
       {
           _logger.LogError($"Cannot update Thing shadow - resource not found:
{ex.Message}");
           return false;
       }
       catch (Exception ex)
       {
           _logger.LogError($"Couldn't update Thing shadow. Here's why:
{ex.Message}");
           return false;
       }
```

```
    }
```

- API-Details finden Sie UpdateThingShadow in der AWS SDK für .NET API-Referenz.

Amazon Redshift-Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Redshift

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Redshift beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Main method to run the Hello Amazon Redshift example.
    /// </summary>
    /// <param name="args">Command line arguments (not used).</param>
    public static async Task Main(string[] args)
    {
        var redshiftClient = new AmazonRedshiftClient();

        Console.WriteLine("Hello, Amazon Redshift! Let's list available clusters:");

        var clusters = new List<Cluster>();

        try
        {
            // Use pagination to retrieve all clusters.
            var clustersPaginator = redshiftClient.Paginators.DescribeClusters(new
DescribeClustersRequest());

            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");

            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }
        }
        catch (AmazonRedshiftException ex)
        {
            Console.WriteLine($"Couldn't list clusters. Here's why: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

- API-Details finden Sie DescribeClusters in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Redshift-Clusters.
- Listen Sie die Datenbanken im Cluster auf.
- Erstellen einer Tabelle mit dem Namen „Movies“.
- Befüllen der Tabelle „Movies“.
- Abfragen der Tabelle „Movies“ nach Jahr.
- Ändern des Redshift-Clusters.
- Löschen Sie den Amazon-Redshift-Cluster.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Redshift-Wrapper-Klasse zur Verwaltung von Vorgängen.

```
/// <summary>
/// Wrapper class for Amazon Redshift operations.
/// </summary>
public class RedshiftWrapper
{
    private readonly IAmazonRedshift _redshiftClient;
    private readonly IAmazonRedshiftDataAPIService _redshiftDataClient;

    /// <summary>
    /// Constructor for RedshiftWrapper.
    /// </summary>
    /// <param name="redshiftClient">Amazon Redshift client.</param>
    /// <param name="redshiftDataClient">Amazon Redshift Data API client.</param>
```

```
    public RedshiftWrapper(IAmazonRedshift redshiftClient,
IAmazonRedshiftDataAPIService redshiftDataClient)
    {
        _redshiftClient = redshiftClient;
        _redshiftDataClient = redshiftDataClient;
    }

    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier, string
databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
            throw;
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"Couldn't create cluster. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string? clusterIdentifier
= null)
    {
        try
        {
            var clusters = new List<Cluster>();
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
```

```
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance window.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
 preferredMaintenanceWindow)
    {
        try
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified and
 has {response.Cluster.PreferredMaintenanceWindow} as the maintenance window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
 {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why: {ex.Message}");
```

```
            return false;
        }
    }

    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
```

```
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create a table in the Redshift database.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <returns>The statement ID.</returns>
    public async Task<string> CreateTableAsync(string clusterIdentifier, string
database, string dbUser)
```

```
    {
        try
        {
            var sqlStatement = @"
                CREATE TABLE Movies (
                    id INTEGER PRIMARY KEY,
                    title VARCHAR(250) NOT NULL,
                    year INTEGER NOT NULL
                )";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine("Table created: Movies");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create table. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Insert a record into the Movies table using parameterized query.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="id">The movie ID.</param>
    /// <param name="title">The movie title.</param>
    /// <param name="year">The movie year.</param>
```

```
    /// <returns>The statement ID.</returns>
    public async Task<string> InsertMovieAsync(string clusterIdentifier, string
database, string dbUser,
        int id, string title, int year)
    {
        try
        {
            var sqlStatement = "INSERT INTO Movies (id, title, year) VALUES
(:id, :title, :year)";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "id", Value = id.ToString() },
                    new SqlParameter { Name = "title", Value = title },
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            await WaitForStatementToCompleteAsync(response.Id);
            Console.WriteLine($"Inserted: {title} ({year})");
            return response.Id;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't insert movie. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Query movies by year using parameterized query.
    /// </summary>
```

```
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="year">The year to query.</param>
    /// <returns>A list of movie titles.</returns>
    public async Task<List<string>> QueryMoviesByYearAsync(string clusterIdentifier,
string database,
        string dbUser, int year)
    {
        try
        {
            var sqlStatement = "SELECT title FROM Movies WHERE year = :year";

            var request = new ExecuteStatementRequest
            {
                ClusterIdentifier = clusterIdentifier,
                Database = database,
                DbUser = dbUser,
                Sql = sqlStatement,
                Parameters = new List<SqlParameter>
                {
                    new SqlParameter { Name = "year", Value = year.ToString() }
                }
            };

            var response = await _redshiftDataClient.ExecuteStatementAsync(request);
            Console.WriteLine($"The identifier of the statement is {response.Id}");

            await WaitForStatementToCompleteAsync(response.Id);

            var results = await GetStatementResultAsync(response.Id);
            var movieTitles = new List<string>();

            foreach (var row in results)
            {
                if (row.Count > 0)
                {
                    var title = row[0].StringValue;
                    Console.WriteLine($"The Movie title field is {title}");
                    movieTitles.Add(title);
                }
            }

            return movieTitles;
```

```
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't query movies. Here's why: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

```
    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
    public async Task<List<List<Field>>> GetStatementResultAsync(string statementId)
    {
        try
        {
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Wait for a statement to complete execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    private async Task WaitForStatementToCompleteAsync(string statementId)
    {
        var status = StatusString.SUBMITTED;
        DescribeStatementResponse? response = null;

        while (status == StatusString.SUBMITTED || status == StatusString.PICKED ||
status == StatusString.STARTED)
```

```
        {
            await Task.Delay(1000); // Wait 1 second
            response = await DescribeStatementAsync(statementId);
            status = response.Status;
            Console.WriteLine($"...{status}");
        }

        if (status == StatusString.FINISHED)
        {
            Console.WriteLine("The statement is finished!");
        }
        else
        {
            var errorMessage = response?.Error ?? "Unknown error";
            Console.WriteLine($"The statement failed with status: {status}");
            Console.WriteLine($"Error message: {errorMessage}");
        }
    }

    /// <summary>
    /// Wait for a cluster to become available.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <returns>A task representing the asynchronous operation.</returns>
    public async Task WaitForClusterAvailableAsync(string clusterIdentifier)
    {
        Console.WriteLine($"Wait until {clusterIdentifier} is available. This may
take a few minutes.");

        var startTime = DateTime.Now;
        var clusters = await DescribeClustersAsync(clusterIdentifier);

        while (clusters[0].ClusterStatus != "available")
        {
            var elapsed = DateTime.Now - startTime;
            Console.WriteLine($"Elapsed Time: {elapsed:mm\\:ss} - Waiting for
cluster...");

            await Task.Delay(5000); // Wait 5 seconds
            clusters = await DescribeClustersAsync(clusterIdentifier);
        }

        var totalElapsed = DateTime.Now - startTime;
```

```
        Console.WriteLine($"Cluster is available! Total Elapsed Time:
 {totalElapsed:mm\\:ss}");
    }
}
```

Führen Sie ein interaktives Szenario durch, das die Grundlagen von Redshift demonstriert.

```
/// <summary>
/// Amazon Redshift Getting Started Scenario.
/// </summary>
public class RedshiftBasics
{
    public static bool IsInteractive = true;
    public static RedshiftWrapper? Wrapper = null;
    public static ILogger logger = null!;
    private static readonly string _moviesFilePath = "../../../../../../resources/
sample_files/movies.json";

    /// <summary>
    /// Main method for the Amazon Redshift Getting Started scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    public static async Task Main(string[] args)
    {
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRedshift>()
                    .AddAWSService<IAmazonRedshiftDataAPIService>()
                    .AddTransient<RedshiftWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<RedshiftBasics>();

        Wrapper = host.Services.GetRequiredService<RedshiftWrapper>();

        await RunScenarioAsync();
    }

    /// <summary>
    /// Run the complete Amazon Redshift scenario.
```

```
    /// </summary>
    public static async Task RunScenarioAsync()
    {
        // Set all variables to default values
        string userName = "awsuser";
        string userPassword = "AwsUser1000";
        string clusterIdentifier = "redshift-cluster-movies";
        var databaseName = "dev";
        int recordCount = 50;
        int year = 2013;
        try
        {
            Console.WriteLine(

"================================================================================");
            Console.WriteLine("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
            Console.WriteLine(
                "This .NET program demonstrates how to interact with Amazon Redshift
by using the AWS SDK for .NET.");
            Console.WriteLine("Let's get started...");
            Console.WriteLine(

"================================================================================");

            // Step 1: Get user credentials (if interactive)
            if (IsInteractive)
            {
                Console.WriteLine("Please enter a user name for the cluster (default
is awsuser):");
                var userInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(userInput))
                    userName = userInput;


Console.WriteLine("================================================================================
                Console.WriteLine("Please enter a user password for the cluster
(default is AwsUser1000):");
                var passwordInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(passwordInput))
                    userPassword = passwordInput;


Console.WriteLine("================================================================================
```

```
                // Step 2: Get cluster identifier
                Console.WriteLine("Enter a cluster id value (default is redshift-
cluster-movies):");
                var clusterInput = Console.ReadLine();
                if (!string.IsNullOrEmpty(clusterInput))
                    clusterIdentifier = clusterInput;
            }
            else
            {
                Console.WriteLine($"Using default values: userName={userName},
 clusterIdentifier={clusterIdentifier}");
            }

            // Step 3: Create Redshift cluster
            await Wrapper!.CreateClusterAsync(clusterIdentifier, databaseName,
 userName, userPassword);

 Console.WriteLine("==========================================================================

            // Step 4: Wait for cluster to become available

 Console.WriteLine("==========================================================================
            await Wrapper.WaitForClusterAvailableAsync(clusterIdentifier);

 Console.WriteLine("==========================================================================

            // Step 5: List databases

 Console.WriteLine("==========================================================================
            Console.WriteLine($" When you created {clusterIdentifier}, the dev
 database is created by default and used in this scenario.");
            Console.WriteLine(" To create a custom database, you need to have a
 CREATEDB privilege.");
            Console.WriteLine(" For more information, see the documentation here:
 https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }

 Console.WriteLine("==========================================================================
```

```
Console.WriteLine("==========================================================================
            Console.WriteLine($"List databases in {clusterIdentifier}");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ListDatabasesAsync(clusterIdentifier, userName,
databaseName);

Console.WriteLine("==========================================================================

            // Step 6: Create Movies table

Console.WriteLine("==========================================================================
            Console.WriteLine("Now you will create a table named Movies.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.CreateTableAsync(clusterIdentifier, databaseName,
userName);

Console.WriteLine("==========================================================================

            // Step 7: Populate the Movies table

Console.WriteLine("==========================================================================
            Console.WriteLine("Populate the Movies table using the Movies.json
file.");

            if (IsInteractive)
            {
                Console.WriteLine("Specify the number of records you would like to
add to the Movies Table.");
                Console.WriteLine("Please enter a value between 50 and 200.");
                Console.Write("Enter a value: ");

                var recordCountInput = Console.ReadLine();
                if (int.TryParse(recordCountInput, out var inputCount) && inputCount
is >= 50 and <= 200)
                {
```

```
                    recordCount = inputCount;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{recordCount}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default record count: {recordCount}");
            }

            await PopulateMoviesTableAsync(clusterIdentifier, databaseName,
userName, recordCount);
            Console.WriteLine($"{recordCount} records were added to the Movies
table.");

Console.WriteLine("==========================================================================

            // Step 8 & 9: Query movies by year

Console.WriteLine("==========================================================================
            Console.WriteLine("Query the Movies table by year. Enter a value between
2012-2014.");

            if (IsInteractive)
            {
                Console.Write("Enter a year: ");
                var yearInput = Console.ReadLine();
                if (int.TryParse(yearInput, out var inputYear) && inputYear is >=
2012 and <= 2014)
                {
                    year = inputYear;
                }
                else
                {
                    Console.WriteLine($"Invalid input. Using default value of
{year}.");
                }
            }
            else
            {
                Console.WriteLine($"Using default year: {year}");
```

```
            }

            await Wrapper.QueryMoviesByYearAsync(clusterIdentifier, databaseName,
 userName, year);

 Console.WriteLine("==========================================================================

            // Step 10: Modify the cluster

 Console.WriteLine("==========================================================================
            Console.WriteLine("Now you will modify the Redshift cluster.");
            if (IsInteractive)
            {
                Console.WriteLine("Press Enter to continue...");
                Console.ReadLine();
            }
            await Wrapper.ModifyClusterAsync(clusterIdentifier, "wed:07:30-
wed:08:00");

 Console.WriteLine("==========================================================================

            // Step 11 & 12: Delete cluster confirmation

 Console.WriteLine("==========================================================================
            if (IsInteractive)
            {
                Console.WriteLine("Would you like to delete the Amazon Redshift
 cluster? (y/n)");
                var deleteResponse = Console.ReadLine();
                if (deleteResponse?.ToLower() == "y")
                {
                    await
 Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
                }
            }
            else
            {
                Console.WriteLine("Deleting the Amazon Redshift cluster...");
                await Wrapper.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            }

 Console.WriteLine("==========================================================================
```

```
Console.WriteLine("==========================================================================
            Console.WriteLine("This concludes the Amazon Redshift SDK Getting
Started scenario.");

Console.WriteLine("==========================================================================
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred during the scenario:
{ex.Message}");
            Console.WriteLine("Deleting the Amazon Redshift cluster...");
            await Wrapper!.DeleteClusterWithoutSnapshotAsync(clusterIdentifier);
            throw;
        }
    }

    /// <summary>
    /// Populate the Movies table with data from the JSON file.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="database">The database name.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="recordCount">Number of records to insert.</param>
    private static async Task PopulateMoviesTableAsync(string clusterIdentifier,
string database, string dbUser, int recordCount)
    {
        if (!File.Exists(_moviesFilePath))
        {
            throw new FileNotFoundException($"Required movies data file not found
at: {_moviesFilePath}");
        }

        var jsonContent = await File.ReadAllTextAsync(_moviesFilePath);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };
        var movies = JsonSerializer.Deserialize<List<Movie>>(jsonContent, options);

        if (movies == null || movies.Count == 0)
        {
            throw new InvalidOperationException("Failed to parse movies JSON file or
file is empty.");
```

```
        }

        var insertCount = Math.Min(recordCount, movies.Count);

        for (int i = 0; i < insertCount; i++)
        {
            var movie = movies[i];
            await Wrapper!.InsertMovieAsync(clusterIdentifier, database, dbUser, i,
 movie.Title, movie.Year);
        }
    }

    /// <summary>
    /// Movie data model.
    /// </summary>
    private class Movie
    {
        public string Title { get; set; } = string.Empty;
        public int Year { get; set; }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Create a new Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="databaseName">The name of the database.</param>
    /// <param name="masterUsername">The master username.</param>
    /// <param name="masterUserPassword">The master user password.</param>
    /// <param name="nodeType">The node type for the cluster.</param>
    /// <returns>The cluster that was created.</returns>
    public async Task<Cluster> CreateClusterAsync(string clusterIdentifier, string
databaseName,
        string masterUsername, string masterUserPassword, string nodeType =
"ra3.large")
    {
        try
        {
            var request = new CreateClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DBName = databaseName,
                MasterUsername = masterUsername,
                MasterUserPassword = masterUserPassword,
                NodeType = nodeType,
                NumberOfNodes = 1,
                ClusterType = "single-node"
            };

            var response = await _redshiftClient.CreateClusterAsync(request);
            Console.WriteLine($"Created cluster {clusterIdentifier}");
            return response.Cluster;
        }
        catch (ClusterAlreadyExistsException ex)
        {
            Console.WriteLine($"Cluster already exists: {ex.Message}");
```

```
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't create cluster. Here's why: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie CreateCluster in der AWS SDK für .NET API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete an Amazon Redshift cluster without a final snapshot.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterWithoutSnapshotAsync(string
clusterIdentifier)
    {
        try
        {
            var request = new DeleteClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                SkipFinalClusterSnapshot = true
            };

            var response = await _redshiftClient.DeleteClusterAsync(request);
```

```
            Console.WriteLine($"The {clusterIdentifier} was deleted");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster not found: {ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't delete cluster. Here's why: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie DeleteCluster in der AWS SDK für .NET API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe Amazon Redshift clusters.
    /// </summary>
    /// <param name="clusterIdentifier">Optional cluster identifier to describe a
specific cluster.</param>
    /// <returns>A list of clusters.</returns>
    public async Task<List<Cluster>> DescribeClustersAsync(string? clusterIdentifier
= null)
    {
        try
        {
            var clusters = new List<Cluster>();
```

```
            var request = new DescribeClustersRequest();
            if (!string.IsNullOrEmpty(clusterIdentifier))
            {
                request.ClusterIdentifier = clusterIdentifier;
            }

            var clustersPaginator =
_redshiftClient.Paginators.DescribeClusters(request);
            await foreach (var response in clustersPaginator.Responses)
            {
                if (response.Clusters != null)
                    clusters.AddRange(response.Clusters);
            }

            Console.WriteLine($"{clusters.Count} cluster(s) retrieved.");
            foreach (var cluster in clusters)
            {
                Console.WriteLine($"\t{cluster.ClusterIdentifier} (Status:
{cluster.ClusterStatus})");
            }

            return clusters;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe clusters. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeClusters in der AWS SDK für .NET API-Referenz.

`DescribeStatement`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStatement`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Describe a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>The statement description.</returns>
    public async Task<DescribeStatementResponse> DescribeStatementAsync(string
statementId)
    {
        try
        {
            var request = new DescribeStatementRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.DescribeStatementAsync(request);
            return response;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't describe statement. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie DescribeStatement in der AWS SDK für .NET API-Referenz.

`GetStatementResult`

Das folgende Codebeispiel zeigt die Verwendung`GetStatementResult`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Get the results of a statement execution.
    /// </summary>
    /// <param name="statementId">The statement ID.</param>
    /// <returns>A list of result rows.</returns>
    public async Task<List<List<Field>>> GetStatementResultAsync(string statementId)
    {
        try
        {
            var request = new GetStatementResultRequest
            {
                Id = statementId
            };

            var response = await
_redshiftDataClient.GetStatementResultAsync(request);
            return response.Records;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ResourceNotFoundException ex)
        {
            Console.WriteLine($"Statement not found: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't get statement result. Here's why:
{ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie GetStatementResult in der AWS SDK für .NET API-Referenz.

`ListDatabases`

Das folgende Codebeispiel zeigt die Verwendung`ListDatabases`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// List databases in a Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The cluster identifier.</param>
    /// <param name="dbUser">The database user.</param>
    /// <param name="dbUser">The database name for authentication.</param>
    /// <returns>A list of database names.</returns>
    public async Task<List<string>> ListDatabasesAsync(string clusterIdentifier,
string dbUser, string databaseName)
    {
        try
        {
            var request = new ListDatabasesRequest
            {
                ClusterIdentifier = clusterIdentifier,
                DbUser = dbUser,
                Database = databaseName
            };

            var response = await _redshiftDataClient.ListDatabasesAsync(request);
            var databases = new List<string>();

            foreach (var database in response.Databases)
            {
                Console.WriteLine($"The database name is : {database}");
                databases.Add(database);
```

```
            }

            return databases;
        }
        catch (Amazon.RedshiftDataAPIService.Model.ValidationException ex)
        {
            Console.WriteLine($"Validation error: {ex.Message}");
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list databases. Here's why: {ex.Message}");
            throw;
        }
    }
```

- API-Details finden Sie ListDatabases in der AWS SDK für .NET API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Modify an Amazon Redshift cluster.
    /// </summary>
    /// <param name="clusterIdentifier">The identifier for the cluster.</param>
    /// <param name="preferredMaintenanceWindow">The preferred maintenance window.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyClusterAsync(string clusterIdentifier, string
 preferredMaintenanceWindow)
    {
        try
```

```
        {
            var request = new ModifyClusterRequest
            {
                ClusterIdentifier = clusterIdentifier,
                PreferredMaintenanceWindow = preferredMaintenanceWindow
            };

            var response = await _redshiftClient.ModifyClusterAsync(request);
            Console.WriteLine($"The modified cluster was successfully modified and
has {response.Cluster.PreferredMaintenanceWindow} as the maintenance window");
            return true;
        }
        catch (ClusterNotFoundException ex)
        {
            Console.WriteLine($"Cluster {clusterIdentifier} not found:
{ex.Message}");
            return false;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't modify cluster. Here's why: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie ModifyCluster in der AWS SDK für .NET API-Referenz.

Amazon S3-Beispiele mit SDK für .NET (v4)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für .NET (v4) mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Hello Amazon Simple Storage Service
// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
```

```
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
            {
                Console.WriteLine($"Found {buckets.Count} S3 buckets:");
                Console.WriteLine();

                foreach (var bucket in buckets)
                {
                    Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                    Console.WriteLine($"  Creation Date: {bucket.CreationDate:yyyy-
MM-dd HH:mm:ss UTC}");
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("No S3 buckets found in your account.");
            }

            Console.WriteLine("Hello S3 completed successfully.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"S3 service error occurred: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Couldn't list S3 buckets. Here's why:
 {ex.Message}");
        }
    }
```

```
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für .NET API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole()))
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;
```

```
        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }

            var success = await s3Wrapper.CreateBucketAsync(bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                // Get the local path and filename for the file to upload.
                while (string.IsNullOrEmpty(filePath))
                {
                    Console.Write("Please enter the path and filename of the file to
upload: ");
                    filePath = Console.ReadLine();

                    // Confirm that the file exists on the local computer.
```

```
                    if (!File.Exists(filePath))
                    {
                        Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                        filePath = string.Empty;
                    }
                }
            }
            else
            {
                // Use the public variable if set, otherwise create a temp file
                if (!string.IsNullOrEmpty(TempFilePath))
                {
                    filePath = TempFilePath;
                    Console.WriteLine($"Using provided test file: {filePath}");
                }
                else
                {
                    // Create a temporary test file for non-interactive mode
                    filePath = Path.GetTempFileName();
                    var testContent = "This is a test file for S3 basics scenario.
\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss UTC");
                    await File.WriteAllTextAsync(filePath, testContent);
                    Console.WriteLine($"Created temporary test file: {filePath}");
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
 filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from {filePath}
 to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set up download path
```

```
            string downloadPath = string.Empty;

            if (IsInteractive)
            {
                // Now get a new location where we can save the file.
                while (string.IsNullOrEmpty(downloadPath))
                {
                    // First get the path to which the file will be downloaded.
                    Console.Write("Please enter the path where the file will be
downloaded: ");
                    downloadPath = Console.ReadLine();

                    // Confirm that the file doesn't already exist on the local
computer.
                    if (File.Exists($"{downloadPath}\\{keyName}"))
                    {
                        Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                        downloadPath = string.Empty;
                    }
                }
            }
            else
            {
                downloadPath = Path.GetTempPath();
                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                }

                Console.WriteLine($"Using download path: {downloadPath}");
            }

            // Download an object from a bucket.
            success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
```

```
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            if (IsInteractive)
            {
                while (string.IsNullOrEmpty(folderName))
                {
                    Console.Write("Please enter the name of the folder to copy your
 object to: ");
                    folderName = Console.ReadLine();
                }
            }
            else
            {
                folderName = "test-folder";
                Console.WriteLine($"Using folder name: {folderName}");
            }

            await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
 folderName);

            // List the objects in the bucket.
            await s3Wrapper.ListBucketContentsAsync(bucketName);

            // Delete the contents of the bucket.
            if (IsInteractive)
            {
                Console.WriteLine("Press <Enter> when you are ready to delete the
 bucket contents.");
                _ = Console.ReadLine();
            }

            var deleteContentsSuccess = await
 s3Wrapper.DeleteBucketContentsAsync(bucketName);
            if (deleteContentsSuccess)
            {
                Console.WriteLine($"Successfully deleted contents of {bucketName}.
\n");
            }
            else
            {
```

```
                Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
            }

            if (IsInteractive)
            {
                // Deleting the bucket too quickly after separately deleting its
contents can
                // cause an error that the bucket isn't empty. To delete contents
and bucket in one
                // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
                Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
                _ = Console.ReadLine();
            }
            else
            {
                // Add a small delay for non-interactive mode to ensure objects are
fully deleted.
                Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
                await Task.Delay(2000);
            }

            // Delete the bucket.
            var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
            if (deleteSuccess)
            {
                Console.WriteLine($"Successfully deleted {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (File.Exists(filePath))
                    {
                        File.Delete(filePath);
                        Console.WriteLine("Cleaned up temporary test file.");
```

```
                    }

                    var downloadFile = Path.Combine(downloadPath, keyName);
                    if (File.Exists(downloadFile))
                    {
                        File.Delete(downloadFile);
                        Console.WriteLine("Cleaned up downloaded test file.");
                    }
                }
                catch (Exception ex)
                {
                    scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
                }
            }
        }
        catch (Exception ex)
        {
            scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

            // Clean up on error - delete bucket if it exists
            try
            {
                if (!string.IsNullOrEmpty(bucketName))
                {
                    await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                    await s3Wrapper.DeleteBucketAsync(bucketName);
                }
            }
            catch (Exception cleanupEx)
            {
                scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
            }

            // Clean up temporary files in non-interactive mode
            if (!IsInteractive)
            {
                try
                {
                    if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                    {
                        File.Delete(filePath);
                    }
```

```
                }
                catch (Exception fileCleanupEx)
                {
                    scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
 temporary files during error handling.");
                }
            }

            throw;
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }


    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
```

```
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
```

```
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

        using GetObjectResponse response = await _amazonS3.GetObjectAsync(request);

        try
        {
```

```
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
```

```
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }



    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
    {
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }
```

```
            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server. Message:'{ex.Message}'
getting list of objects.");
            return null;
        }
    }



    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
```

```
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }



    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET -API-Referenz.
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
```

```
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie CopyObject in der AWS SDK für .NET API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
```

```
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- API-Details finden Sie CreateBucket in der AWS SDK für .NET API-Referenz.

`CreatePresignedPost`

Das folgende Codebeispiel zeigt die Verwendung`CreatePresignedPost`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte POST URL.

```
    /// <summary>
    /// Create a presigned POST URL with conditions.
    /// </summary>
    /// <param name="s3Client">The Amazon S3 client.</param>
    /// <param name="bucketName">The name of the bucket.</param>
```

```
    /// <param name="objectKey">The object key (path) where the uploaded file will
 be stored.</param>
    /// <param name="expires">When the presigned URL expires.</param>
    /// <param name="fields">Dictionary of fields to add to the form.</param>
    /// <param name="conditions">List of conditions to apply.</param>
    /// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
    public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
        IAmazonS3 s3Client,
        string bucketName,
        string objectKey,
        DateTime expires,
        Dictionary<string, string>? fields = null,
        List<S3PostCondition>? conditions = null)
    {
        var request = new CreatePresignedPostRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = expires
        };

        // Add custom fields if provided
        if (fields != null)
        {
            foreach (var field in fields)
            {
                request.Fields.Add(field.Key, field.Value);
            }
        }

        // Add conditions if provided
        if (conditions != null)
        {
            foreach (var condition in conditions)
            {
                request.Conditions.Add(condition);
            }
        }

        return await s3Client.CreatePresignedPostAsync(request);
    }
```

- API-Details finden Sie CreatePresignedPost in der AWS SDK für .NET API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- API-Details finden Sie DeleteBucket in der AWS SDK für .NET API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Delete all of the objects stored in an existing Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket from which the
    /// contents will be deleted.</param>
    /// <returns>A boolean value that represents the success or failure of
    /// deleting all of the objects in the bucket.</returns>
    public async Task<bool> DeleteBucketContentsAsync(string bucketName)
    {
        // Iterate over the contents of the bucket and delete all objects.
        try
        {
            // Delete all objects in the bucket.
            var deleteList = await ListBucketContentsAsync(bucketName, false);
            if (deleteList != null && deleteList.Any())
            {
                await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
                {
                    BucketName = bucketName,
                    Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
                });
            }

            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
```

```
        }
    }
```

- API-Details finden Sie DeleteObjects in der AWS SDK für .NET API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to download an object from an Amazon S3 bucket to the
    /// local computer.
    /// </summary>
    /// <param name="bucketName">The name of the bucket where the object is
    /// currently stored.</param>
    /// <param name="objectName">The name of the object to download.</param>
    /// <param name="filePath">The path, including filename, where the
    /// downloaded object will be stored.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// download process.</returns>
    public async Task<bool> DownloadObjectFromBucketAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };
```

```
        using GetObjectResponse response = await _amazonS3.GetObjectAsync(request);

        try
        {
            // Save object to local file
            await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie GetObject in der AWS SDK für .NET API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
 bool printList = true)
    {
```

```
        try
        {
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
                MaxKeys = 5,
            };

            if (printList)
            {
                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");
            }

            var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });
            var s3Objects = new List<S3Object>();
            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                if (response.S3Objects != null)
                {
                    s3Objects.AddRange(response.S3Objects);
                }
            }

            if (printList)
            {
                Console.WriteLine($"Number of Objects: {s3Objects.Count}");
                foreach (var entry in s3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }

            return s3Objects;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server. Message:'{ex.Message}'
getting list of objects.");
```

```
            return null;
        }
    }
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für .NET API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
```

```
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
```

- API-Details finden Sie PutObject in der AWS SDK für .NET API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für .NET (v4)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und verwenden Sie vorsignierte POST URLs für direkte Browser-Uploads.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
```

```
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetUpServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory = host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }

    /// <summary>
    /// Perform the actions defined for the Amazon S3 Presigned POST scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        _isInteractive = !args.Contains("--non-interactive");

        // Set up dependency injection for Amazon S3
        using var host =
 Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
```

```
            .Build();

        SetUpServices(host);

        try
        {
            // Display overview
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(_isInteractive);

            // Step 1: Create bucket
            await CreateBucketAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 2: Create presigned URL
            _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
            var response = await CreatePresignedPostAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 3: Display URL and fields
            _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
            DisplayPresignedPostFields(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 4: Upload file
            _uiMethods.DisplayTitle("Step 4: Upload test file using presigned POST
URL");
            await UploadFileAsync(response);
            _uiMethods.PressEnter(_isInteractive);

            // Step 5: Verify file exists
            await VerifyFileExistsAsync();
            _uiMethods.PressEnter(_isInteractive);

            // Step 6: Cleanup
            _uiMethods.DisplayTitle("Step 6: Clean up resources");
            await CleanupAsync();

            _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
            _uiMethods.PressEnter(_isInteractive);
        }
        catch (Exception ex)
        {
```

```
            _logger.LogError(ex, "Error in scenario");
            Console.WriteLine($"Error: {ex.Message}");

            // Attempt cleanup if there was an error
            if (!string.IsNullOrEmpty(_bucketName))
            {
                _uiMethods.DisplayTitle("Cleaning up resources after error");
                await _s3Wrapper.DeleteBucketAsync(_bucketName);
                Console.WriteLine($"Cleaned up bucket: {_bucketName}");
            }
        }
    }

    /// <summary>
    /// Create an S3 bucket for the scenario.
    /// </summary>
    private static async Task CreateBucketAsync()
    {
        _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

        // Generate a default bucket name for the scenario
        var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

        // Prompt user for bucket name or use default in non-interactive mode
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'): ",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrWhiteSpace(_bucketName))
        {
            _bucketName = defaultBucketName;
        }

        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }
```

```
    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
 CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for the
 demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
 scenario.";

        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
 testFilePath);

        // Display the upload result
        if (uploadResult.Success)
```

```
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
    {
        try
        {
            _logger.LogInformation("Uploading file {filePath} using presigned POST
URL", filePath);

            using var httpClient = _httpClientFactory.CreateClient();
            using var formContent = new MultipartFormDataContent();

            // Add all the fields from the presigned POST response
            foreach (var field in response.Fields)
            {
                formContent.Add(new StringContent(field.Value), field.Key);
            }

            // Add the file content
            var fileStream = File.OpenRead(filePath);
            var fileName = Path.GetFileName(filePath);
            var fileContent = new StreamContent(fileStream);
```

```
            fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
            formContent.Add(fileContent, "file", fileName);

            // Send the POST request
            var httpResponse = await httpClient.PostAsync(response.Url,
 formContent);
            var responseContent = await httpResponse.Content.ReadAsStringAsync();

            // Log and return the result
            _logger.LogInformation("Upload completed with status code {statusCode}",
 httpResponse.StatusCode);

            return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
 responseContent);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error uploading file");
            return (false, HttpStatusCode.InternalServerError, ex.Message);
        }
    }

    /// <summary>
    /// Verify that the uploaded file exists in the S3 bucket.
    /// </summary>
    private static async Task VerifyFileExistsAsync()
    {
        _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

        Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

        try
        {
            var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
 _objectKey!);

            Console.WriteLine($"File verification successful! File exists in the
 bucket.");
            Console.WriteLine($"File size: {metadata.ContentLength} bytes");
            Console.WriteLine($"File type: {metadata.Headers.ContentType}");
            Console.WriteLine($"Last modified: {metadata.LastModified}");
        }
```

```
        catch (AmazonS3Exception ex) when (ex.StatusCode ==
 System.Net.HttpStatusCode.NotFound)
        {
            Console.WriteLine($"Error: File was not found in the bucket.");
            throw;
        }
    }

    private static void DisplayPresignedPostFields(CreatePresignedPostResponse
 response)
    {
        Console.WriteLine($"Presigned POST URL: {response.Url}");
        Console.WriteLine("Form fields to include:");

        foreach (var field in response.Fields)
        {
            Console.WriteLine($"  {field.Key}: {field.Value}");
        }
    }

    /// <summary>
    /// Clean up resources created by the scenario.
    /// </summary>
    private static async Task CleanupAsync()
    {
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
 already deleted");
            }
        }
    }
}
```

Codebeispiele für AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie das with Bash-Skript AWS Command Line Interface mit verwenden. AWS

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- AWS CLI mit dem Bash-Skript-Entwicklerhandbuch – weitere Informationen zur Verwendung von Bash mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- Beispiele für API Gateway unter Verwendung AWS CLI mit Bash-Skript
- Athena-Beispiele mit AWS CLI mit Bash-Skript
- AWS Batch Beispiele mit AWS CLI mit Bash-Skript
- AWS Cloud Map Beispiele mit AWS CLI mit Bash-Skript
- CloudFormation Beispiele mit AWS CLI mit Bash-Skript
- CloudFront Beispiele mit AWS CLI mit Bash-Skript
- CloudWatch Beispiele mit AWS CLI mit Bash-Skript
- CloudWatch Loggt Beispiele mit AWS CLI mit Bash-Skript
- Beispiele für Amazon Cognito Identity Provider unter AWS CLI mit Bash-Skript

- AWS Config Beispiele mit AWS CLI mit Bash-Skript
- Verbinden Sie Kundenbeispiele mit AWS CLI mit Bash-Skript
- Direct Connect Beispiele mit AWS CLI mit Bash-Skript
- Amazon DocumentDB-Beispiele mit AWS CLI mit Bash-Skript
- DynamoDB-Beispiele mit AWS CLI mit Bash-Skript
- Amazon EC2-Beispiele mit AWS CLI mit Bash-Skript
- Amazon ECR-Beispiele mit AWS CLI mit Bash-Skript
- Amazon ECS-Beispiele mit AWS CLI mit Bash-Skript
- Amazon EKS-Beispiele mit AWS CLI mit Bash-Skript
- Beispiele für Elastic Load Balancing — Version 2 mit AWS CLI mit Bash-Skript
- ElastiCache Beispiele mit AWS CLI mit Bash-Skript
- Amazon EMR-Beispiele mit AWS CLI mit Bash-Skript
- OpenSearch Service-Beispiele mit AWS CLI mit Bash-Skript
- AWS FIS Beispiele mit AWS CLI mit Bash-Skript
- AWS Glue Beispiele mit AWS CLI mit Bash-Skript
- HealthImaging Beispiele mit AWS CLI mit Bash-Skript
- IAM-Beispiele mit AWS CLI mit Bash-Skript
- AWS IoT Beispiele mit AWS CLI mit Bash-Skript
- AWS KMS Beispiele mit AWS CLI mit Bash-Skript
- Lambda-Beispiele mit AWS CLI mit Bash-Skript
- Lightsail-Beispiele mit AWS CLI mit Bash-Skript
- Amazon Managed Grafana-Beispiele mit AWS CLI mit Bash-Skript
- MediaConnect Beispiele mit AWS CLI mit Bash-Skript
- Amazon MSK-Beispiele mit AWS CLI mit Bash-Skript
- Neptun-Beispiele mit AWS CLI mit Bash-Skript
- AWS Payment Cryptography Beispiele mit AWS CLI mit Bash-Skript
- Amazon Pinpoint-Beispiele mit AWS CLI mit Bash-Skript
- Beispiele für Amazon Polly unter Verwendung AWS CLI mit Bash-Skript
- Amazon RDS-Beispiele mit AWS CLI mit Bash-Skript
- Beispiele für Amazon Redshift mit AWS CLI mit Bash-Skript

- Amazon S3-Beispiele mit AWS CLI mit Bash-Skript
- SageMaker KI-Beispiele unter Verwendung AWS CLI mit Bash-Skript
- Secrets Manager-Beispiele mit AWS CLI mit Bash-Skript
- Amazon SES-Beispiele mit AWS CLI mit Bash-Skript
- Amazon SNS-Beispiele unter Verwendung AWS CLI mit Bash-Skript
- Beispiele für Step Functions mit AWS CLI mit Bash-Skript
- AWS STS Beispiele mit AWS CLI mit Bash-Skript
- Support Beispiele mit AWS CLI mit Bash-Skript
- Beispiele für Systems Manager mit AWS CLI mit Bash-Skript
- Beispiele für Amazon Textract mit AWS CLI mit Bash-Skript
- Beispiele für VPC-Gitter mit AWS CLI mit Bash-Skript
- AWS WAFV2 Beispiele mit AWS CLI mit Bash-Skript
- WorkSpaces Beispiele mit AWS CLI mit Bash-Skript

Beispiele für API Gateway unter Verwendung AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das with Bash-Skript AWS Command Line Interface mit API Gateway verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung

- Erstellen Sie eine Lambda-Funktion und stellen Sie sie bereit
- Erstellen Sie eine REST-API
- Konfigurieren Sie die Lambda-Proxy-Integration
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
```

```
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))
```

```
        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
```

```
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
```

```
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
```

```
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
```

```
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Athena-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Athena Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Abfrageanalysen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen S3-Bucket für Abfrageergebnisse
- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Ausführen einer -Abfrage
- Erstellen und verwenden Sie benannte Abfragen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail
```

```
# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] || [ ${#bucket_name}
 -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
```

```
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
```

```
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-
z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION" 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
athena-gs}]'
```

```
    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi

    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)
```

```
if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
```

```
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^\\\
\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
```

```
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
```

```
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
```

```
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/dev/
null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -d'"' -
f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
```

```
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
```

```
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
```

```
   DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
       --region "$AWS_REGION" 2>&1)
   if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
       echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
   else
       echo "Named query deleted successfully."
   fi

   # Drop table
   echo "Dropping table: $TABLE_NAME"
   DROP_TABLE_RESULT=$(aws athena start-query-execution \
       --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
       --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
       --region "$AWS_REGION" 2>&1)

   if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
       echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
   else
       QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
       if [ -n "$QUERY_ID" ]; then
           echo "Waiting for table deletion to complete..."

           ELAPSED=0
           while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
               QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
                   --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
               if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                   echo "Table dropped successfully."
                   break
               elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                   echo "Warning: Table deletion failed with status: $QUERY_STATUS"
                   break
               fi
               echo "Table deletion in progress, status: $QUERY_STATUS"
               sleep 2
               ((ELAPSED+=2))
           done
       fi
```

```
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
```

```
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults

- ListDatabases
- ListNamedQueries
- ListTableMetadata
- Mb
- Rb
- Rm
- StartQueryExecution

AWS Batch Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS Batch

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Batchverarbeitung und serverlosen Containern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Ausführungsrolle
- Erstellen Sie eine Rechenumgebung
- Erstellen Sie eine Job-Warteschlange
- Erstellen Sie eine Auftragsdefinition
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Batch Fargate Getting Started Script - Security Hardened Version
# This script demonstrates creating AWS Batch resources with Fargate orchestration
#

set -euo pipefail  # Exit on any error, undefined variables, and pipe failures

# Configuration
SCRIPT_NAME="batch-fargate-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
RANDOM_SUFFIX=$(openssl rand -hex 6)
COMPUTE_ENV_NAME="batch-fargate-compute-${RANDOM_SUFFIX}"
JOB_QUEUE_NAME="batch-fargate-queue-${RANDOM_SUFFIX}"
JOB_DEF_NAME="batch-fargate-jobdef-${RANDOM_SUFFIX}"
JOB_NAME="batch-hello-world-${RANDOM_SUFFIX}"
ROLE_NAME="BatchEcsTaskExecutionRole-${RANDOM_SUFFIX}"
TRUST_POLICY_FILE="batch-trust-policy-${RANDOM_SUFFIX}.json"

# Security: Set restrictive umask
umask 0077

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Logging function with sanitization
log() {
    local message="${1//[$'\t\r\n']/}"
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] ${message}" | tee -a "${LOG_FILE}"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
```

```
    log "Attempting to clean up resources created so far..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error ${LINENO}' ERR

# Validate AWS credentials
validate_aws_credentials() {
    if ! aws sts get-caller-identity &>/dev/null; then
        log "ERROR: AWS credentials are not configured or invalid"
        exit 1
    fi
}

# Function to wait for resource to be ready
wait_for_compute_env() {
    local env_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for compute environment ${env_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-compute-environments \
            --compute-environments "${env_name}" \
            --query 'computeEnvironments[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "VALID" ]; then
            log "Compute environment ${env_name} is ready"
            return 0
        elif [ "${status}" = "INVALID" ] || [ "${status}" = "NOT_FOUND" ]; then
            log "ERROR: Compute environment ${env_name} failed to create properly"
            return 1
        fi

        log "Compute environment status: ${status}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done
```

```
    log "ERROR: Timeout waiting for compute environment ${env_name}"
    return 1
}

# Function to wait for job queue to be ready
wait_for_job_queue() {
    local queue_name="${1}"
    local max_attempts=60
    local attempt=0

    log "Waiting for job queue ${queue_name} to be VALID..."

    while [ ${attempt} -lt ${max_attempts} ]; do
        local state
        state=$(aws batch describe-job-queues \
            --job-queues "${queue_name}" \
            --query 'jobQueues[0].state' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${state}" = "ENABLED" ]; then
            log "Job queue ${queue_name} is ready"
            return 0
        elif [ "${state}" = "DISABLED" ] || [ "${state}" = "NOT_FOUND" ]; then
            log "ERROR: Job queue ${queue_name} failed to create properly"
            return 1
        fi

        log "Job queue state: ${state}. Waiting 10 seconds..."
        sleep 10
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job queue ${queue_name}"
    return 1
}

# Function to wait for job completion
wait_for_job() {
    local job_id="${1}"
    local max_attempts=120
    local attempt=0

    log "Waiting for job ${job_id} to complete..."
```

```
    while [ ${attempt} -lt ${max_attempts} ]; do
        local status
        status=$(aws batch describe-jobs \
            --jobs "${job_id}" \
            --query 'jobs[0].status' \
            --output text 2>/dev/null || echo "NOT_FOUND")

        if [ "${status}" = "SUCCEEDED" ]; then
            log "Job ${job_id} completed successfully"
            return 0
        elif [ "${status}" = "FAILED" ]; then
            log "ERROR: Job ${job_id} failed"
            return 1
        fi

        log "Job status: ${status}. Waiting 30 seconds..."
        sleep 30
        ((attempt++))
    done

    log "ERROR: Timeout waiting for job ${job_id}"
    return 1
}

# Function to wait for resource state before deletion
wait_for_resource_state() {
    local resource_type="${1}"
    local resource_name="${2}"
    local expected_state="${3}"
    local max_attempts=30
    local attempt=0

    log "Waiting for ${resource_type} ${resource_name} to reach state:
 ${expected_state}"

    while [ ${attempt} -lt ${max_attempts} ]; do
        local current_state=""

        case "${resource_type}" in
            "JOB_QUEUE")
                current_state=$(aws batch describe-job-queues \
                    --job-queues "${resource_name}" \
                    --query 'jobQueues[0].state' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
```

```
                ;;
            "COMPUTE_ENV")
                current_state=$(aws batch describe-compute-environments \
                    --compute-environments "${resource_name}" \
                    --query 'computeEnvironments[0].status' \
                    --output text 2>/dev/null || echo "NOT_FOUND")
                ;;
            *)
                log "WARNING: Unknown resource type: ${resource_type}"
                return 1
                ;;
        esac

        if [ "${current_state}" = "${expected_state}" ]; then
            log "${resource_type} ${resource_name} is now in state:
 ${expected_state}"
            return 0
        fi

        log "${resource_type} ${resource_name} state: ${current_state} (waiting for
 ${expected_state})"
        sleep 10
        ((attempt++))
    done

    log "WARNING: ${resource_type} ${resource_name} did not reach expected state
 after ${max_attempts} attempts"
    return 1
}

# Cleanup function
cleanup_resources() {
    log "Starting cleanup of created resources..."

    # Clean up in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local resource="${CREATED_RESOURCES[i]}"
        local resource_type
        local resource_name

        resource_type=$(echo "${resource}" | cut -d: -f1)
        resource_name=$(echo "${resource}" | cut -d: -f2-)

        log "Cleaning up ${resource_type}: ${resource_name}"
```

```
        case "${resource_type}" in
            "JOB_QUEUE")
                aws batch update-job-queue --job-queue "${resource_name}" --state
 DISABLED 2>/dev/null || true
                wait_for_resource_state "JOB_QUEUE" "${resource_name}" "DISABLED" ||
 true
                aws batch delete-job-queue --job-queue "${resource_name}" 2>/dev/
null || true
                ;;
            "COMPUTE_ENV")
                aws batch update-compute-environment --compute-environment
 "${resource_name}" --state DISABLED 2>/dev/null || true
                wait_for_resource_state "COMPUTE_ENV" "${resource_name}" "DISABLED"
 || true
                aws batch delete-compute-environment --compute-environment
 "${resource_name}" 2>/dev/null || true
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "${resource_name}" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/
null || true
                aws iam delete-role --role-name "${resource_name}" 2>/dev/null ||
 true
                ;;
            "FILE")
                rm -f "${resource_name}" 2>/dev/null || true
                ;;
            *)
                log "WARNING: Unknown resource type for cleanup: ${resource_type}"
                ;;
        esac
    done

    log "Cleanup completed"
}

# Validate input parameters
validate_inputs() {
    if [ -z "${ACCOUNT_ID:-}" ]; then
        log "ERROR: ACCOUNT_ID is not set"
        return 1
    fi
```

```
    if [ -z "${DEFAULT_VPC:-}" ]; then
        log "ERROR: DEFAULT_VPC is not set"
        return 1
    fi

    if [ -z "${SUBNETS:-}" ]; then
        log "ERROR: SUBNETS is not set"
        return 1
    fi
}

# Validate container image format
validate_container_image() {
    local image="${1}"

    # Check if image contains any shell metacharacters that could be dangerous
    if [[ "${image}" =~ [';$`|&<>()[]{}\\'] ]]; then
        log "ERROR: Container image contains potentially dangerous characters:
 ${image}"
        return 1
    fi

    # Basic ECR/Docker image format validation
    if ! [[ "${image}" =~ ^[a-zA-Z0-9._/:-]+$ ]]; then
        log "ERROR: Container image format is invalid: ${image}"
        return 1
    fi

    return 0
}

# Main script execution
main() {
    log "Starting AWS Batch Fargate tutorial script - Security Hardened Version"
    log "Log file: ${LOG_FILE}"

    # Validate AWS credentials
    validate_aws_credentials

    # Get AWS account ID
    log "Getting AWS account ID..."
    local ACCOUNT_ID
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    if [ -z "${ACCOUNT_ID}" ] || [ "${ACCOUNT_ID}" = "None" ]; then
```

```
        log "ERROR: Could not retrieve AWS account ID"
        exit 1
    fi
    log "Account ID: ${ACCOUNT_ID}"

    # Get default VPC and subnets
    log "Getting default VPC and subnets..."
    local DEFAULT_VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs \
        --filters "Name=is-default,Values=true" \
        --query 'Vpcs[0].VpcId' \
        --output text)

    if [ "${DEFAULT_VPC}" = "None" ] || [ "${DEFAULT_VPC}" = "null" ] || [ -z
 "${DEFAULT_VPC}" ]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi

    log "Default VPC: ${DEFAULT_VPC}"

    # Get subnets in the default VPC
    local SUBNETS
    SUBNETS=$(aws ec2 describe-subnets \
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" \
        --query 'Subnets[*].SubnetId' \
        --output text)

    if [ -z "${SUBNETS}" ]; then
        log "ERROR: No subnets found in default VPC"
        exit 1
    fi

    # Convert tab/space-separated subnets to JSON array format
    local SUBNET_ARRAY
    SUBNET_ARRAY=$(echo "${SUBNETS}" | tr '\t ' '\n' | sed 's/^/"/;s/$/"/' | paste -
sd ',' -)
    log "Subnets: ${SUBNETS}"
    log "Subnet array: [${SUBNET_ARRAY}]"

    # Get default security group for the VPC
    local DEFAULT_SG
    DEFAULT_SG=$(aws ec2 describe-security-groups \
```

```
        --filters "Name=vpc-id,Values=${DEFAULT_VPC}" "Name=group-
name,Values=default" \
        --query 'SecurityGroups[0].GroupId' \
        --output text)

    if [ "${DEFAULT_SG}" = "None" ] || [ "${DEFAULT_SG}" = "null" ] || [ -z
 "${DEFAULT_SG}" ]; then
        log "ERROR: No default security group found in VPC"
        exit 1
    fi

    log "Default security group: ${DEFAULT_SG}"

    # Step 1: Create IAM execution role
    log "Step 1: Creating IAM execution role..."

    # Create trust policy document with proper escaping
    cat > "${TRUST_POLICY_FILE}" << 'EOFPOLICY'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs-tasks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOFPOLICY
    CREATED_RESOURCES+=("FILE:${TRUST_POLICY_FILE}")

    # Validate trust policy file before using it
    if ! jq empty "${TRUST_POLICY_FILE}" 2>/dev/null; then
        log "ERROR: Trust policy file is not valid JSON"
        exit 1
    fi

    # Create the role
    aws iam create-role \
        --role-name "${ROLE_NAME}" \
        --assume-role-policy-document "file://${TRUST_POLICY_FILE}"
```

```
    aws iam tag-role --role-name "${ROLE_NAME}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=getting-started-batch-fargate
    CREATED_RESOURCES+=("IAM_ROLE:${ROLE_NAME}")

    # Attach policy
    aws iam attach-role-policy \
        --role-name "${ROLE_NAME}" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"

    log "IAM role created: ${ROLE_NAME}"

    # Wait for IAM role propagation
    log "Waiting for IAM role propagation (15 seconds)..."
    sleep 15

    # Step 2: Create compute environment
    log "Step 2: Creating Fargate compute environment..."

    aws batch create-compute-environment \
        --compute-environment-name "${COMPUTE_ENV_NAME}" \
        --type MANAGED \
        --state ENABLED \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-resources "{
            \"type\": \"FARGATE\",
            \"maxvCpus\": 256,
            \"subnets\": [${SUBNET_ARRAY}],
            \"securityGroupIds\": [\"${DEFAULT_SG}\"]
        }"
    CREATED_RESOURCES+=("COMPUTE_ENV:${COMPUTE_ENV_NAME}")

    # Wait for compute environment to be ready
    if ! wait_for_compute_env "${COMPUTE_ENV_NAME}"; then
        log "ERROR: Compute environment failed to reach VALID state"
        exit 1
    fi

    # Step 3: Create job queue
    log "Step 3: Creating job queue..."

    aws batch create-job-queue \
        --job-queue-name "${JOB_QUEUE_NAME}" \
        --state ENABLED \
```

```
        --priority 900 \
        --tags project=doc-smith,tutorial=getting-started-batch-fargate \
        --compute-environment-order "order=1,computeEnvironment=${COMPUTE_ENV_NAME}"
    CREATED_RESOURCES+=("JOB_QUEUE:${JOB_QUEUE_NAME}")

    # Wait for job queue to be ready
    if ! wait_for_job_queue "${JOB_QUEUE_NAME}"; then
        log "ERROR: Job queue failed to reach ENABLED state"
        exit 1
    fi

    # Step 4: Create job definition
    log "Step 4: Creating job definition..."

    local CONTAINER_IMAGE="busybox:latest"
    validate_container_image "${CONTAINER_IMAGE}"

    aws batch register-job-definition \
        --job-definition-name "${JOB_DEF_NAME}" \
        --type container \
        --platform-capabilities FARGATE \
        --container-properties "{
            \"image\": \"${CONTAINER_IMAGE}\",
            \"resourceRequirements\": [
                {\"type\": \"VCPU\", \"value\": \"0.25\"},
                {\"type\": \"MEMORY\", \"value\": \"512\"}
            ],
            \"command\": [\"echo\", \"hello world\"],
            \"networkConfiguration\": {
                \"assignPublicIp\": \"DISABLED\"
            },
            \"executionRoleArn\": \"arn:aws:iam::${ACCOUNT_ID}:role/${ROLE_NAME}\"
        }"

    log "Job definition created: ${JOB_DEF_NAME}"

    # Step 5: Submit job
    log "Step 5: Submitting job..."

    local JOB_ID
    JOB_ID=$(aws batch submit-job \
        --job-name "${JOB_NAME}" \
        --job-queue "${JOB_QUEUE_NAME}" \
        --job-definition "${JOB_DEF_NAME}" \
```

```
        --query 'jobId' \
        --output text)

    if [ -z "${JOB_ID}" ] || [ "${JOB_ID}" = "None" ]; then
        log "ERROR: Failed to submit job"
        exit 1
    fi

    log "Job submitted with ID: ${JOB_ID}"

    # Step 6: Wait for job completion and view output
    log "Step 6: Waiting for job completion..."
    if ! wait_for_job "${JOB_ID}"; then
        log "ERROR: Job failed or timed out"
        exit 1
    fi

    # Get log stream name
    log "Getting job logs..."
    local LOG_STREAM
    LOG_STREAM=$(aws batch describe-jobs \
        --jobs "${JOB_ID}" \
        --query 'jobs[0].attempts[0].taskProperties.containers[0].logStreamName' \
        --output text)

    if [ "${LOG_STREAM}" != "None" ] && [ "${LOG_STREAM}" != "null" ] && [ -n
"${LOG_STREAM}" ]; then
        log "Log stream: ${LOG_STREAM}"
        log "Job output:"
        aws logs get-log-events \
            --log-group-name "/aws/batch/job" \
            --log-stream-name "${LOG_STREAM}" \
            --query 'events[*].message' \
            --output text 2>/dev/null | tee -a "${LOG_FILE}" || true
    else
        log "No log stream available for job"
    fi

    log "Tutorial completed successfully!"

    # Show created resources
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
```

```
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${resource}"
    done
    echo ""
    echo "==========================================="
    echo "CLEANUP"
    echo "==========================================="

    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

AWS Cloud Map Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS Cloud Map

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Konfigurieren Sie die Container-Dienstkonnektivität

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
```

```
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"
```

```
    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
```

```
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
```

```
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
```

```
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
```

```
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
```

```
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
```

```
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF
```

```
    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
```

```
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}
```

```
# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
```

```
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources
```

```
    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace

- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Serviceerkennung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen öffentlichen DNS-Namespace mit gehosteter Route-53-Zonenintegration
- Erstellen Sie Services, die über DNS-Abfragen und API-Aufrufe auffindbar sind.
- Registrieren Sie Service-Instances mit unterschiedlichen Erkennungskonfigurationen.
- Entdecken Sie Services, die sowohl DNS-Abfragen als auch Cloud-Map-API verwenden.
- Überprüfen Sie die Serviceerkennung mithilfe von Dig-Befehlen und API-Aufrufen.
- Bereinigen Sie Ressourcen in der richtigen Reihenfolge (Instances, Services, Namespace).

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Cloud Map Private Namespace Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery
# with DNS queries and API calls

# Exit on error
set -e

# Configuration
REGION="us-east-2"
NAMESPACE_NAME="cloudmap-tutorial.com"
LOG_FILE="cloudmap-tutorial.log"
CREATOR_REQUEST_ID=$(date +%s)

# Function to log messages
log() {
    local message="$1"
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $message" | tee -a "$LOG_FILE"
}

# Function to check operation status
check_operation() {
    local operation_id="$1"
    local status=""

    log "Checking operation status for $operation_id..."

    while [[ "$status" != "SUCCESS" ]]; do
        sleep 5
        status=$(aws servicediscovery get-operation \
            --operation-id "$operation_id" \
            --region "$REGION" \
            --query "Operation.Status" \
```

```
            --output text)

        log "Operation status: $status"

        if [[ "$status" == "FAIL" ]]; then
            log "Operation failed. Exiting."
            exit 1
        fi
    done

    log "Operation completed successfully."
}

# Function to clean up resources
cleanup() {
    log "Starting cleanup process..."

    if [[ -n "$FIRST_INSTANCE_ID" ]]; then
        log "Deregistering first service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$PUBLIC_SERVICE_ID" \
            --instance-id "$FIRST_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister first instance"
    fi

    if [[ -n "$SECOND_INSTANCE_ID" ]]; then
        log "Deregistering second service instance..."
        aws servicediscovery deregister-instance \
            --service-id "$BACKEND_SERVICE_ID" \
            --instance-id "$SECOND_INSTANCE_ID" \
            --region "$REGION" || log "Failed to deregister second instance"
    fi

    if [[ -n "$PUBLIC_SERVICE_ID" ]]; then
        log "Deleting public service..."
        aws servicediscovery delete-service \
            --id "$PUBLIC_SERVICE_ID" \
            --region "$REGION" || log "Failed to delete public service"
    fi

    if [[ -n "$BACKEND_SERVICE_ID" ]]; then
        log "Deleting backend service..."
        aws servicediscovery delete-service \
            --id "$BACKEND_SERVICE_ID" \
```

```
            --region "$REGION" || log "Failed to delete backend service"
    fi

    if [[ -n "$NAMESPACE_ID" ]]; then
        log "Deleting namespace..."
        aws servicediscovery delete-namespace \
            --id "$NAMESPACE_ID" \
            --region "$REGION" || log "Failed to delete namespace"
    fi

    log "Cleanup completed."
}

# Set up trap for cleanup on script exit
trap cleanup EXIT INT TERM

# Initialize log file
> "$LOG_FILE"
log "Starting AWS Cloud Map tutorial script"

# Step 1: Create an AWS Cloud Map namespace
log "Creating AWS Cloud Map namespace: $NAMESPACE_NAME"
OPERATION_RESULT=$(aws servicediscovery create-public-dns-namespace \
    --name "$NAMESPACE_NAME" \
    --creator-request-id "cloudmap-tutorial-$CREATOR_REQUEST_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

OPERATION_ID=$(echo "$OPERATION_RESULT" | jq -r '.OperationId')
log "Namespace creation initiated. Operation ID: $OPERATION_ID"

# Check operation status
check_operation "$OPERATION_ID"

# Get the namespace ID
log "Getting namespace ID..."
NAMESPACE_ID=$(aws servicediscovery list-namespaces \
    --region "$REGION" \
    --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" \
    --output text)

log "Namespace ID: $NAMESPACE_ID"
```

```
# Get the hosted zone ID
log "Getting Route 53 hosted zone ID..."
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
    --dns-name "$NAMESPACE_NAME" \
    --query "HostedZones[0].Id" \
    --output text | sed 's|/hostedzone/||')

log "Hosted Zone ID: $HOSTED_ZONE_ID"

# Step 2: Create the AWS Cloud Map services
log "Creating public service..."
PUBLIC_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "public-service" \
    --namespace-id "$NAMESPACE_ID" \
    --dns-config "RoutingPolicy=MULTIVALUE,DnsRecords=[{Type=A,TTL=300}]" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

PUBLIC_SERVICE_ID=$(echo "$PUBLIC_SERVICE_RESULT" | jq -r '.Service.Id')
log "Public service created. Service ID: $PUBLIC_SERVICE_ID"

log "Creating backend service..."
BACKEND_SERVICE_RESULT=$(aws servicediscovery create-service \
    --name "backend-service" \
    --namespace-id "$NAMESPACE_ID" \
    --type "HTTP" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-service-discovery
 \
    --region "$REGION")

BACKEND_SERVICE_ID=$(echo "$BACKEND_SERVICE_RESULT" | jq -r '.Service.Id')
log "Backend service created. Service ID: $BACKEND_SERVICE_ID"

# Step 3: Register the AWS Cloud Map service instances
log "Registering first service instance..."
FIRST_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$PUBLIC_SERVICE_ID" \
    --instance-id "first" \
    --attributes "AWS_INSTANCE_IPV4=192.168.2.1" \
    --region "$REGION")

FIRST_INSTANCE_ID="first"
FIRST_OPERATION_ID=$(echo "$FIRST_INSTANCE_RESULT" | jq -r '.OperationId')
```

```
log "First instance registration initiated. Operation ID: $FIRST_OPERATION_ID"

# Check operation status
check_operation "$FIRST_OPERATION_ID"

log "Registering second service instance..."
SECOND_INSTANCE_RESULT=$(aws servicediscovery register-instance \
    --service-id "$BACKEND_SERVICE_ID" \
    --instance-id "second" \
    --attributes "service-name=backend" \
    --region "$REGION")

SECOND_INSTANCE_ID="second"
SECOND_OPERATION_ID=$(echo "$SECOND_INSTANCE_RESULT" | jq -r '.OperationId')
log "Second instance registration initiated. Operation ID: $SECOND_OPERATION_ID"

# Check operation status
check_operation "$SECOND_OPERATION_ID"

# Step 4: Discover the AWS Cloud Map service instances
log "Getting Route 53 name servers..."
NAME_SERVERS=$(aws route53 get-hosted-zone \
    --id "$HOSTED_ZONE_ID" \
    --query "DelegationSet.NameServers[0]" \
    --output text)

log "Name server: $NAME_SERVERS"

log "Using dig to query DNS records (this will be simulated)..."
log "Command: dig @$NAME_SERVERS public-service.$NAMESPACE_NAME"
log "Expected output would show: public-service.$NAMESPACE_NAME. 300 IN A
 192.168.2.1"

log "Using AWS CLI to discover backend service instances..."
DISCOVER_RESULT=$(aws servicediscovery discover-instances \
    --namespace-name "$NAMESPACE_NAME" \
    --service-name "backend-service" \
    --region "$REGION")

log "Discovery result: $(echo "$DISCOVER_RESULT" | jq -c '.')"

# Display created resources
log "Resources created:"
log "- Namespace: $NAMESPACE_NAME (ID: $NAMESPACE_ID)"
```

```
log "- Public Service: public-service (ID: $PUBLIC_SERVICE_ID)"
log "- Backend Service: backend-service (ID: $BACKEND_SERVICE_ID)"
log "- Service Instance: first (Service: public-service)"
log "- Service Instance: second (Service: backend-service)"

# Ask user if they want to clean up resources
read -p "Do you want to clean up all created resources? (y/n): " CLEANUP_RESPONSE

if [[ "$CLEANUP_RESPONSE" == "y" || "$CLEANUP_RESPONSE" == "Y" ]]; then
    log "User confirmed cleanup. Proceeding with resource deletion."
    # Cleanup function will be called automatically on exit
else
    log "User chose not to clean up resources. Exiting without cleanup."
    trap - EXIT
    exit 0
fi
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreatePublicDnsNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - DiscoverInstances
  - GetOperation
  - ListNamespaces
  - RegisterInstance

Benutzerdefinierte Attribute für die Serviceerkennung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Cloud Map-Namespace
- Erstellen einer DynamoDB-Tabelle
- Erstellen Sie einen Cloud Map-Datendienst und registrieren Sie die DynamoDB-Tabelle
- Erstellen Sie eine IAM-Rolle für Lambda-Funktionen

- Erstellen Sie die Lambda-Funktion zum Schreiben von Daten
- Erstellen Sie einen Cloud Map-App-Service und registrieren Sie die Lambda-Schreibfunktion
- Erstellen Sie die Lambda-Funktion zum Lesen von Daten

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Cloud Map Tutorial Script
# This script demonstrates how to use AWS Cloud Map for service discovery with
 custom attributes

set -euo pipefail

# Set up logging
LOG_FILE="cloudmap-tutorial.log"
echo "AWS Cloud Map Tutorial Script" > "$LOG_FILE"
echo "Started at $(date)" >> "$LOG_FILE"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log commands and their output
log_cmd() {
  echo "$ $1" | tee -a "$LOG_FILE"
  eval "$1" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
  local LINE=$1
  echo "An error occurred at line $LINE" | tee -a "$LOG_FILE"
  echo "Resources created so far:" | tee -a "$LOG_FILE"
  for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
```

```
  done
  echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"
  cleanup
  exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Helper function to wait for Cloud Map operations to complete
wait_for_operation() {
  local OPERATION_ID=$1
  local TIMEOUT=300  # 5 minutes timeout
  local START_TIME
  START_TIME=$(date +%s)

  while true; do
    local STATUS
    STATUS=$(aws servicediscovery get-operation --operation-id "$OPERATION_ID" --
query 'Operation.Status' --output text 2>/dev/null || echo "UNKNOWN")

    if [ "$STATUS" == "SUCCESS" ]; then
      echo "Operation completed successfully" | tee -a "$LOG_FILE"
      break
    elif [ "$STATUS" == "FAIL" ]; then
      echo "Operation failed" | tee -a "$LOG_FILE"
      return 1
    fi

    local CURRENT_TIME
    CURRENT_TIME=$(date +%s)
    if [ $((CURRENT_TIME - START_TIME)) -gt $TIMEOUT ]; then
      echo "Operation timed out" | tee -a "$LOG_FILE"
      return 1
    fi

    sleep 5
  done

  return 0
}

# Function to clean up resources
cleanup() {
```

```
 echo "Cleaning up resources..." | tee -a "$LOG_FILE"

 # Reverse the order of created resources for proper deletion
 for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
   resource="${CREATED_RESOURCES[$i]}"
   echo "Deleting $resource..." | tee -a "$LOG_FILE"

   if [[ $resource == "instance:"* ]]; then
     # Extract service ID and instance ID
     SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
     INSTANCE_ID=$(echo "$resource" | cut -d':' -f3)

     # Check if instance exists before trying to deregister
     INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
"$SERVICE_ID" --query "Instances[?Id=='$INSTANCE_ID'].Id" --output text 2>/dev/null
|| echo "")
     if [[ -n "$INSTANCE_EXISTS" ]]; then
       OPERATION_ID=$(aws servicediscovery deregister-instance --service-id
"$SERVICE_ID" --instance-id "$INSTANCE_ID" --query 'OperationId' --output text)

       # Wait for deregistration to complete
       echo "Waiting for instance deregistration to complete..." | tee -a
"$LOG_FILE"
       wait_for_operation "$OPERATION_ID" || true
     else
       echo "Instance $INSTANCE_ID already deregistered" | tee -a "$LOG_FILE"
     fi
   elif [[ $resource == "lambda:"* ]]; then
     # Extract function name
     FUNCTION_NAME=$(echo "$resource" | cut -d':' -f2)
     aws lambda delete-function --function-name "$FUNCTION_NAME" 2>/dev/null ||
echo "Function already deleted" | tee -a "$LOG_FILE"
   elif [[ $resource == "role:"* ]]; then
     # Extract role name
     ROLE_NAME=$(echo "$resource" | cut -d':' -f2)

     # Detach all policies first
     for POLICY_ARN in $(aws iam list-attached-role-policies --role-name
"$ROLE_NAME" --query 'AttachedPolicies[*].PolicyArn' --output text); do
       aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$POLICY_ARN"
     done

     # Delete the role
```

```
      aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Role already
 deleted" | tee -a "$LOG_FILE"
    elif [[ $resource == "dynamodb:"* ]]; then
      # Extract table name
      TABLE_NAME=$(echo "$resource" | cut -d':' -f2)
      aws dynamodb delete-table --table-name "$TABLE_NAME" 2>/dev/null || echo
 "Table already deleted" | tee -a "$LOG_FILE"

      # Wait for table deletion to complete
      echo "Waiting for DynamoDB table deletion to complete..." | tee -a "$LOG_FILE"
      aws dynamodb wait table-not-exists --table-name "$TABLE_NAME" 2>/dev/null ||
 true
    fi
  done

  # Handle services separately to ensure all instances are deregistered first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "service:"* ]]; then
      # Extract service ID
      SERVICE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"

      # Make sure all instances are deregistered
      INSTANCES=$(aws servicediscovery list-instances --service-id "$SERVICE_ID" --
query 'Instances[*].Id' --output text 2>/dev/null || echo "")
      if [[ -n "$INSTANCES" ]]; then
        echo "Service still has instances. Waiting before deletion..." | tee -a
 "$LOG_FILE"
        sleep 10
      fi

      # Try to delete the service
      aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || echo
 "Service already deleted" | tee -a "$LOG_FILE"
      sleep 5
    fi
  done

  # Handle namespaces last to ensure all services are deleted first
  for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
    resource="${CREATED_RESOURCES[$i]}"
    if [[ $resource == "namespace:"* ]]; then
      # Extract namespace ID
```

```
      NAMESPACE_ID=$(echo "$resource" | cut -d':' -f2)
      echo "Deleting namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"

      # Check if namespace still has services
      SERVICES=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query 'Services[*].Id' --
output text 2>/dev/null || echo "")
      if [[ -n "$SERVICES" ]]; then
        echo "Namespace still has services. Deleting them first..." | tee -a
 "$LOG_FILE"
        for SERVICE_ID in $SERVICES; do
          echo "Deleting service $SERVICE_ID..." | tee -a "$LOG_FILE"
          aws servicediscovery delete-service --id "$SERVICE_ID" 2>/dev/null || true
        done
        sleep 5
      fi

      # Try to delete the namespace
      OPERATION_ID=$(aws servicediscovery delete-namespace --id "$NAMESPACE_ID" --
query 'OperationId' --output text 2>/dev/null || echo "")
      if [[ -n "$OPERATION_ID" ]]; then
        echo "Waiting for namespace deletion to complete..." | tee -a "$LOG_FILE"
        wait_for_operation "$OPERATION_ID" || true
      else
        echo "Failed to delete namespace or namespace already deleted" | tee -a
 "$LOG_FILE"
      fi
    fi
  done

  echo "Cleanup complete" | tee -a "$LOG_FILE"
}

# Step 1: Create an AWS Cloud Map namespace
echo "Step 1: Creating AWS Cloud Map namespace..." | tee -a "$LOG_FILE"

# Check if namespace already exists
NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$NAMESPACE_ID" || "$NAMESPACE_ID" == "None" ]]; then
  log_cmd "aws servicediscovery create-http-namespace --name cloudmap-tutorial --
creator-request-id namespace-request-\$(date +%s)"
```

```
  OPERATION_ID=$(aws servicediscovery create-http-namespace --name cloudmap-tutorial
 --creator-request-id "namespace-request-$(date +%s)" --tags Key=project,Value=doc-
smith Key=tutorial,Value=cloudmap-custom-attributes --query 'OperationId' --output
 text)

  # Wait for namespace creation to complete
  echo "Waiting for namespace creation to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"

  # Get the namespace ID
  NAMESPACE_ID=$(aws servicediscovery list-namespaces --query "Namespaces[?
Name=='cloudmap-tutorial'].Id" --output text)
  echo "Namespace created with ID: $NAMESPACE_ID" | tee -a "$LOG_FILE"
else
  echo "Namespace cloudmap-tutorial already exists with ID: $NAMESPACE_ID" | tee -a
 "$LOG_FILE"
fi

CREATED_RESOURCES+=("namespace:$NAMESPACE_ID")

# Step 2: Create a DynamoDB table
echo "Step 2: Creating DynamoDB table..." | tee -a "$LOG_FILE"

# Check if table already exists
TABLE_EXISTS=$(aws dynamodb describe-table --table-name cloudmap 2>&1 || echo
 "NOT_EXISTS")

if [[ $TABLE_EXISTS == *"ResourceNotFoundException"* || $TABLE_EXISTS ==
 "NOT_EXISTS" ]]; then
  log_cmd "aws dynamodb create-table --table-name cloudmap --attribute-definitions
 AttributeName=id,AttributeType=S --key-schema AttributeName=id,KeyType=HASH
 --billing-mode PAY_PER_REQUEST --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes"

  # Wait for DynamoDB table to become active
  echo "Waiting for DynamoDB table to become active..." | tee -a "$LOG_FILE"
  aws dynamodb wait table-exists --table-name cloudmap
else
  echo "DynamoDB table cloudmap already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("dynamodb:cloudmap")

# Step 3: Create an AWS Cloud Map data service
```

```
echo "Step 3: Creating AWS Cloud Map data service..." | tee -a "$LOG_FILE"

# Get all services in the namespace
echo "Listing all services in namespace $NAMESPACE_ID..." | tee -a "$LOG_FILE"
SERVICES=$(aws servicediscovery list-services --filters "Name=NAMESPACE_ID,Values=
$NAMESPACE_ID,Condition=EQ" --query 'Services[*].[Id,Name]' --output text 2>/dev/
null || echo "")
echo "Services found: $SERVICES" | tee -a "$LOG_FILE"

# Check if data service already exists
DATA_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  echo "Checking service: ID=$id, Name=$name" | tee -a "$LOG_FILE"
  if [[ "$name" == "data-service" ]]; then
    DATA_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$DATA_SERVICE_ID" ]]; then
  echo "Data service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name data-service --namespace-id
 $NAMESPACE_ID --creator-request-id data-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name data-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "data-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  DATA_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='data-service'].Id" --output text)
  echo "Data service created with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "Data service already exists with ID: $DATA_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$DATA_SERVICE_ID")

# Register DynamoDB table as a service instance
echo "Registering DynamoDB table as a service instance..." | tee -a "$LOG_FILE"
```

```
# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id
 "$DATA_SERVICE_ID" --query "Instances[?Id=='data-instance'].Id" --output text 2>/
dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  AWS_REGION=$(aws configure get region || echo "us-east-1")
  log_cmd "aws servicediscovery register-instance --service-id $DATA_SERVICE_ID --
instance-id data-instance --attributes tablename=cloudmap,region=$AWS_REGION"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$DATA_SERVICE_ID" --instance-id data-instance --attributes
 "tablename=cloudmap,region=$AWS_REGION" --query 'OperationId' --output text)

  # Wait for instance registration to complete
  echo "Waiting for instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance data-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$DATA_SERVICE_ID:data-instance")

# Step 4: Create an IAM role for Lambda
echo "Step 4: Creating IAM role for Lambda..." | tee -a "$LOG_FILE"

# Create a trust policy for Lambda
cat > lambda-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Check if role already exists
echo "Checking if IAM role already exists..." | tee -a "$LOG_FILE"
```

```
ROLE_EXISTS=$(aws iam get-role --role-name cloudmap-tutorial-role 2>&1 || echo
 "NOT_EXISTS")

if [[ $ROLE_EXISTS == *"NoSuchEntity"* || $ROLE_EXISTS == "NOT_EXISTS" ]]; then
    log_cmd "aws iam create-role --role-name cloudmap-tutorial-role --assume-role-
policy-document file://lambda-trust-policy.json"
    aws iam tag-role --role-name cloudmap-tutorial-role --tags
 Key=project,Value=doc-smith Key=tutorial,Value=cloudmap-custom-attributes
else
    echo "Role cloudmap-tutorial-role already exists, using existing role" | tee -a
 "$LOG_FILE"
fi

# Create a custom policy with least privilege
cat > cloudmap-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:GetItem",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/cloudmap"
    },
    {
      "Effect": "Allow",
      "Action": [
        "servicediscovery:DiscoverInstances"
      ],
      "Resource": "*"
    }
  ]
```

```
}
EOF

# Check if policy already exists
POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text 2>/dev/null || echo "")

if [[ -z "$POLICY_ARN" ]]; then
  echo "Creating CloudMapTutorialPolicy..." | tee -a "$LOG_FILE"
  echo "$ aws iam create-policy --policy-name CloudMapTutorialPolicy --policy-
document file://cloudmap-policy.json" | tee -a "$LOG_FILE"
  CREATE_OUTPUT=$(aws iam create-policy --policy-name CloudMapTutorialPolicy --
policy-document file://cloudmap-policy.json)
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"
  POLICY_ARN=$(aws iam list-policies --query "Policies[?
PolicyName=='CloudMapTutorialPolicy'].Arn" --output text)
else
  echo "Policy CloudMapTutorialPolicy already exists with ARN: $POLICY_ARN" | tee -a
 "$LOG_FILE"
fi

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 $POLICY_ARN" | tee -a "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 "$POLICY_ARN" | tee -a "$LOG_FILE"

echo "$ aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-
arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" | tee -a
 "$LOG_FILE"
aws iam attach-role-policy --role-name cloudmap-tutorial-role --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole | tee -a
 "$LOG_FILE"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

ROLE_ARN=$(aws iam get-role --role-name cloudmap-tutorial-role --query 'Role.Arn' --
output text)
CREATED_RESOURCES+=("role:cloudmap-tutorial-role")

# Step 5: Create an AWS Cloud Map app service
echo "Step 5: Creating AWS Cloud Map app service..." | tee -a "$LOG_FILE"
```

```
# Get all services in the namespace
SERVICES=$(aws servicediscovery list-services --filters "Name=NAMESPACE_ID,Values=
$NAMESPACE_ID,Condition=EQ" --query 'Services[*].[Id,Name]' --output text 2>/dev/
null || echo "")

# Check if app service already exists
APP_SERVICE_ID=""
while read -r id name || [[ -n "$id" ]]; do
  if [[ "$name" == "app-service" ]]; then
    APP_SERVICE_ID="$id"
    break
  fi
done <<< "$SERVICES"

if [[ -z "$APP_SERVICE_ID" ]]; then
  echo "App service does not exist, creating it..." | tee -a "$LOG_FILE"
  # Create the service and capture the ID directly
  echo "$ aws servicediscovery create-service --name app-service --namespace-id
 $NAMESPACE_ID --creator-request-id app-service-request-\$(date +%s)" | tee -a
 "$LOG_FILE"
  CREATE_OUTPUT=$(aws servicediscovery create-service --name app-service
 --namespace-id "$NAMESPACE_ID" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudmap-custom-attributes --creator-request-id "app-service-
request-$(date +%s)")
  echo "$CREATE_OUTPUT" | tee -a "$LOG_FILE"

  # Extract the service ID using AWS CLI query
  APP_SERVICE_ID=$(aws servicediscovery list-services --filters
 "Name=NAMESPACE_ID,Values=$NAMESPACE_ID,Condition=EQ" --query "Services[?
Name=='app-service'].Id" --output text)
  echo "App service created with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
else
  echo "App service already exists with ID: $APP_SERVICE_ID" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("service:$APP_SERVICE_ID")

# Step 6: Create a Lambda function to write data
echo "Step 6: Creating Lambda function to write data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > writefunction.py << 'EOF'
import boto3
import json
```

```
import random
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)

        # Write data to the table
        table.put_item(
            Item={
                'id': str(random.randint(1, 100)),
                'todo': str(event)
            }
        )

        return {
            'statusCode': 200,
            'body': json.dumps('Data written successfully!')
        }
    except Exception as e:
        return {
            'statusCode': 500,
```

```
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip writefunction.zip writefunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='writefunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name writefunction --runtime
 python3.12 --role $ROLE_ARN --handler writefunction.lambda_handler --zip-file
 fileb://writefunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name writefunction --query
 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function writefunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:writefunction")

# Step 7: Register the Lambda write function as an AWS Cloud Map service instance
echo "Step 7: Registering Lambda write function as a service instance..." | tee -a
 "$LOG_FILE"

# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id "$APP_SERVICE_ID"
 --query "Instances[?Id=='write-instance'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id write-instance --attributes action=write,functionname=writefunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id write-instance --attributes
 action=write,functionname=writefunction --query 'OperationId' --output text)
```

```
  # Wait for instance registration to complete
  echo "Waiting for write instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance write-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:write-instance")

# Step 8: Create a Lambda function to read data
echo "Step 8: Creating Lambda function to read data..." | tee -a "$LOG_FILE"

# Create Lambda function code
cat > readfunction.py << 'EOF'
import boto3
import json
import os

def lambda_handler(event, context):
    # Use AWS Cloud Map to discover the DynamoDB table
    serviceclient = boto3.client('servicediscovery')

    try:
        # Discover the data service instance
        response = serviceclient.discover_instances(
            NamespaceName='cloudmap-tutorial',
            ServiceName='data-service'
        )

        if not response.get('Instances'):
            return {
                'statusCode': 404,
                'body': json.dumps('No service instances found')
            }

        # Extract table name and region from the instance attributes
        attributes = response['Instances'][0].get('Attributes', {})
        tablename = attributes.get('tablename', 'cloudmap')
        region = attributes.get('region', os.environ.get('AWS_REGION', 'us-east-1'))

        # Create DynamoDB client in the specified region
        dynamodb = boto3.resource('dynamodb', region_name=region)
        table = dynamodb.Table(tablename)
```

```
        # Read data from the table
        response = table.scan()

        return {
            'statusCode': 200,
            'body': json.dumps(response.get('Items', []))
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps(f'Error: {str(e)}')
        }
EOF

# Zip the function code
log_cmd "zip readfunction.zip readfunction.py"

# Create the Lambda function
FUNCTION_EXISTS=$(aws lambda list-functions --query "Functions[?
FunctionName=='readfunction'].FunctionName" --output text 2>/dev/null || echo "")
if [[ -z "$FUNCTION_EXISTS" ]]; then
  log_cmd "aws lambda create-function --function-name readfunction --runtime
 python3.12 --role $ROLE_ARN --handler readfunction.lambda_handler --zip-file
 fileb://readfunction.zip --architectures x86_64 --timeout 10 --tags project=doc-
smith,tutorial=cloudmap-custom-attributes"

  # Wait for the Lambda function to be active before updating
  echo "Waiting for Lambda function to become active..." | tee -a "$LOG_FILE"
  function_state="Pending"
  while [ "$function_state" == "Pending" ]; do
      sleep 5
      function_state=$(aws lambda get-function --function-name readfunction --query
 'Configuration.State' --output text)
      echo "Current function state: $function_state" | tee -a "$LOG_FILE"
  done
else
  echo "Lambda function readfunction already exists" | tee -a "$LOG_FILE"
fi
CREATED_RESOURCES+=("lambda:readfunction")

# Step 9: Register the Lambda read function as an AWS Cloud Map service instance
echo "Step 9: Registering Lambda read function as a service instance..." | tee -a
 "$LOG_FILE"
```

```
# Check if instance already exists
INSTANCE_EXISTS=$(aws servicediscovery list-instances --service-id "$APP_SERVICE_ID"
 --query "Instances[?Id=='read-instance'].Id" --output text 2>/dev/null || echo "")

if [[ -z "$INSTANCE_EXISTS" ]]; then
  log_cmd "aws servicediscovery register-instance --service-id $APP_SERVICE_ID --
instance-id read-instance --attributes action=read,functionname=readfunction"
  OPERATION_ID=$(aws servicediscovery register-instance --service-
id "$APP_SERVICE_ID" --instance-id read-instance --attributes
 action=read,functionname=readfunction --query 'OperationId' --output text)

  # Wait for read instance registration to complete
  echo "Waiting for read instance registration to complete..." | tee -a "$LOG_FILE"
  wait_for_operation "$OPERATION_ID"
else
  echo "Instance read-instance already exists" | tee -a "$LOG_FILE"
fi

CREATED_RESOURCES+=("instance:$APP_SERVICE_ID:read-instance")

# Step 10: Create Python clients to interact with the services
echo "Step 10: Creating Python clients..." | tee -a "$LOG_FILE"

cat > writeclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'write'}
    )

    if not response.get('Instances'):
        print("No write service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]
```

```
    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        Payload=json.dumps('This is a test data'),
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

cat > readclient.py << 'EOF'
#!/usr/bin/env python3
import boto3
import json
import sys

try:
    serviceclient = boto3.client('servicediscovery')

    response = serviceclient.discover_instances(
        NamespaceName='cloudmap-tutorial',
        ServiceName='app-service',
        QueryParameters={'action': 'read'}
    )

    if not response.get('Instances'):
        print("No read service instances found", file=sys.stderr)
        sys.exit(1)

    functionname = response["Instances"][0]["Attributes"]["functionname"]

    lambdaclient = boto3.client('lambda')

    resp = lambdaclient.invoke(
        FunctionName=functionname,
        InvocationType='RequestResponse'
    )

    print(resp["Payload"].read().decode())
```

```
except Exception as e:
    print(f"Error: {str(e)}", file=sys.stderr)
    sys.exit(1)
EOF

chmod +x writeclient.py readclient.py

echo "Running write client..." | tee -a "$LOG_FILE"
log_cmd "python3 writeclient.py" || echo "Write client execution completed with
 warnings" | tee -a "$LOG_FILE"

echo "Running read client..." | tee -a "$LOG_FILE"
log_cmd "python3 readclient.py" || echo "Read client execution completed with
 warnings" | tee -a "$LOG_FILE"

# Step 11: Clean up resources
echo "Resources created:" | tee -a "$LOG_FILE"
for resource in "${CREATED_RESOURCES[@]}"; do
  echo "- $resource" | tee -a "$LOG_FILE"
done

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "Script completed at $(date)" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateHttpNamespace
  - CreateService
  - DeleteNamespace
  - DeleteService
  - DeregisterInstance
  - GetOperation
  - ListNamespaces
  - ListServices
  - RegisterInstance

CloudFormation Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. CloudFormation

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie Ihren ersten Infrastruktur-Stack

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine CloudFormation Vorlage
- Testen Sie den Webserver
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudFormation Getting Started Script
# This script creates a CloudFormation stack with a web server and security group,
# monitors the stack creation, and provides cleanup options.

set -euo pipefail
```

```
# Set up logging with secure permissions
LOG_FILE="cloudformation-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS CloudFormation Getting Started Tutorial"
echo "====================================================="
echo "This script will create a CloudFormation stack with:"
echo "- An EC2 instance running a simple web server"
echo "- A security group allowing HTTP access from your IP"
echo ""
echo "Starting at: $(date)"
echo ""

# Function to validate IP address format
validate_ip() {
    local ip=$1
    if ! [[ "$ip" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        return 1
    fi

    local IFS=.
    local -a octets=($ip)
    for octet in "${octets[@]}"; do
        if ((octet > 255)); then
            return 1
        fi
    done
    return 0
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack: $STACK_NAME"
```

```
        aws cloudformation delete-stack --stack-name "$STACK_NAME" --region
 "${AWS_REGION:-us-east-1}" || true

        echo "Waiting for stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME" --
region "${AWS_REGION:-us-east-1}" 2>/dev/null || true

        echo "Stack deletion complete."
    fi

    if [ -f "${TEMPLATE_FILE:-}" ]; then
        echo "Removing local template file: $TEMPLATE_FILE"
        shred -vfz -n 3 "$TEMPLATE_FILE" 2>/dev/null || rm -f "$TEMPLATE_FILE"
    fi

    echo "Cleanup completed at: $(date)"
}

# Function to handle errors
handle_error() {
    echo ""
    echo "====================================================="
    echo "ERROR: $1"
    echo "====================================================="
    echo "Resources created before error:"
    if [ -n "${STACK_NAME:-}" ]; then
        echo "- CloudFormation stack: $STACK_NAME"
    fi
    echo ""

    echo "Cleaning up resources automatically..."
    cleanup

    exit 1
}

# Set up trap for script interruption
trap 'handle_error "Script interrupted"' INT TERM
trap 'cleanup' EXIT

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS_REGION format: $AWS_REGION"
```

```
fi

# Generate a unique stack name with timestamp
TIMESTAMP=$(date +%s)
STACK_NAME="MyTestStack-${TIMESTAMP}"
TEMPLATE_FILE="webserver-template-${TIMESTAMP}.yaml"

# Verify AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Verify curl is installed
if ! command -v curl &> /dev/null; then
    handle_error "curl is not installed or not in PATH"
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Step 1: Create the CloudFormation template file
echo "Creating CloudFormation template file: $TEMPLATE_FILE"
cat > "$TEMPLATE_FILE" << 'EOF'
AWSTemplateFormatVersion: 2010-09-09
Description: CloudFormation Template for WebServer with Security Group and EC2
 Instance

Metadata:
  AWS::CloudFormation::Init:
    config:
      packages:
        yum:
          httpd: []
      services:
        sysvinit:
          httpd:
            enabled: true
            ensureRunning: true

Parameters:
  LatestAmiId:
    Description: The latest Amazon Linux 2 AMI from the Parameter Store
```

```
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t3.micro
      - t2.micro
    ConstraintDescription: must be a valid EC2 instance type.

  MyIP:
    Description: Your IP address in CIDR format (e.g 203.0.113.1/32).
    Type: String
    MinLength: '9'
    MaxLength: '18'
    AllowedPattern: '^(\d{1,3}\.){3}\d{1,3}/\d{1,2}$'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.

Resources:
  WebServerRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'

  WebServerInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref WebServerRole

  WebServerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Allow HTTP access via specified IP address
```

```
      SecurityGroupEgress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
          Description: Allow HTTP outbound
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
          Description: Allow HTTPS outbound for package updates
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: !Ref MyIP
          Description: HTTP access from specified IP

  WebServer:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref LatestAmiId
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref WebServerInstanceProfile
      SecurityGroupIds:
        - !Ref WebServerSecurityGroup
      Monitoring: true
      MetadataOptions:
        HttpEndpoint: enabled
        HttpTokens: required
        HttpPutResponseHopLimit: 1
      UserData: !Base64 |
        #!/bin/bash
        set -euo pipefail
        exec > >(tee /var/log/user-data.log)
        exec 2>&1
        yum update -y
        yum install -y httpd
        systemctl start httpd
        systemctl enable httpd
        echo "<html><body><h1>Hello World!</h1></body></html>" > /var/www/html/
index.html
        chmod 644 /var/www/html/index.html
```

```
Outputs:
  WebsiteURL:
    Value: !Join
      - ''
      - - http://
        - !GetAtt WebServer.PublicDnsName
    Description: Website URL
  InstanceId:
    Value: !Ref WebServer
    Description: EC2 Instance ID
  SecurityGroupId:
    Value: !Ref WebServerSecurityGroup
    Description: Security Group ID
EOF

chmod 600 "$TEMPLATE_FILE"

if [ ! -f "$TEMPLATE_FILE" ]; then
    handle_error "Failed to create template file"
fi

# Step 2: Validate the template
echo ""
echo "Validating CloudFormation template..."
if ! VALIDATION_RESULT=$(aws cloudformation validate-template \
    --template-body "file://$TEMPLATE_FILE" \
    --region "$AWS_REGION" 2>&1); then
    handle_error "Template validation failed: $VALIDATION_RESULT"
fi
echo "Template validation successful."

# Step 3: Get the user's public IP address
echo ""
echo "Retrieving your public IP address..."

MY_IP=""
for endpoint in "https://checkip.amazonaws.com" "https://api.ipify.org" "https://
icanhazip.com"; do
    if MY_IP=$(curl -s --max-time 5 "$endpoint" 2>/dev/null); then
        MY_IP="${MY_IP//[[:space:]]/}"
        if validate_ip "$MY_IP"; then
            break
        fi
        MY_IP=""
```

```
    fi
done

if [ -z "$MY_IP" ]; then
    handle_error "Failed to retrieve public IP address from multiple sources"
fi

MY_IP_CIDR="${MY_IP}/32"
echo "Your public IP address: $MY_IP_CIDR"

# Step 4: Create the CloudFormation stack
echo ""
echo "Creating CloudFormation stack: $STACK_NAME"
echo "This will create an EC2 instance and security group."
if ! CREATE_RESULT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-body "file://$TEMPLATE_FILE" \
  --parameters \
    ParameterKey=InstanceType,ParameterValue=t2.micro \
    ParameterKey=MyIP,ParameterValue="$MY_IP_CIDR" \
  --capabilities CAPABILITY_IAM \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudformation-gs \
  --region "$AWS_REGION" \
  --output text 2>&1); then
    handle_error "Stack creation failed: $CREATE_RESULT"
fi

STACK_ID="${CREATE_RESULT//[[:space:]]/}"
echo "Stack creation initiated. Stack ID: $STACK_ID"

# Step 5: Monitor stack creation
echo ""
echo "Monitoring stack creation..."
echo "This may take a few minutes."

if ! aws cloudformation wait stack-create-complete \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" 2>/dev/null; then
    STACK_STATUS=$(aws cloudformation describe-stacks \
        --stack-name "$STACK_NAME" \
        --query "Stacks[0].StackStatus" \
        --region "$AWS_REGION" \
        --output text 2>/dev/null || echo "UNKNOWN")
    if [[ "$STACK_STATUS" =~ ROLLBACK|FAILED ]]; then
```

```
        handle_error "Stack creation failed. Status: $STACK_STATUS"
    fi
fi

echo "Stack creation completed successfully."

# Step 6: List stack resources
echo ""
echo "Resources created by the stack:"
aws cloudformation list-stack-resources \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "StackResourceSummaries[*].{LogicalID:LogicalResourceId,
 Type:ResourceType, Status:ResourceStatus}" \
    --output table

# Step 7: Get stack outputs
echo ""
echo "Stack outputs:"
if ! OUTPUTS=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs" \
    --output json 2>&1); then
    handle_error "Failed to retrieve stack outputs"
fi

echo "$OUTPUTS"

# Extract the WebsiteURL
WEBSITE_URL=$(aws cloudformation describe-stacks \
    --stack-name "$STACK_NAME" \
    --region "$AWS_REGION" \
    --query "Stacks[0].Outputs[?OutputKey=='WebsiteURL'].OutputValue" \
    --output text 2>/dev/null || echo "")

if [ -z "$WEBSITE_URL" ]; then
    handle_error "Failed to extract WebsiteURL from stack outputs"
fi

echo ""
echo "WebsiteURL: $WEBSITE_URL"
echo ""
echo "You can access the web server by opening the above URL in your browser."
```

```
echo "You should see a simple 'Hello World!' message."

# Step 8: Test the connection via CLI with timeout
echo ""
echo "Testing connection to the web server..."
MAX_RETRIES=5
RETRY_COUNT=0

while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
    if HTTP_RESPONSE=$(curl -s -o /dev/null -w "%{http_code}" --max-time 10
 "$WEBSITE_URL" 2>/dev/null); then
        if [ "$HTTP_RESPONSE" == "200" ]; then
            echo "Connection successful! HTTP status code: $HTTP_RESPONSE"
            break
        elif [ "$HTTP_RESPONSE" == "000" ]; then
            RETRY_COUNT=$((RETRY_COUNT + 1))
            if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
                echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying
 in 10 seconds..."
                sleep 10
            else
                echo "Warning: Connection test failed after $MAX_RETRIES attempts"
            fi
        else
            echo "Warning: Connection test returned HTTP status code:
 $HTTP_RESPONSE"
            break
        fi
    else
        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Connection test attempt $RETRY_COUNT/$MAX_RETRIES - retrying in 10
 seconds..."
            sleep 10
        else
            echo "Warning: Connection test failed after $MAX_RETRIES attempts"
        fi
    fi
done

# Step 9: Auto-confirm cleanup
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
```

```
echo "==================================================="
echo "Resources created:"
echo "- CloudFormation stack: $STACK_NAME"
echo "  - EC2 instance"
echo "  - Security group"
echo "  - IAM role and instance profile"
echo ""
echo "Proceeding with cleanup of all created resources..."

cleanup

echo ""
echo "==================================================="
echo "Tutorial completed at: $(date)"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateStack
  - DeleteStack
  - DescribeStacks
  - ListStackResources
  - ValidateTemplate
  - Wait

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
```

```
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
```

```
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
```

```
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
```

```
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
```

```
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
```

```
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"
```

```
CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
```

```
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register
```

```
# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
```

```
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy

- CreateCluster
- CreateNodegroup
- CreateRole
- CreateStack
- DeleteCluster
- DeleteNodegroup
- DeleteRole
- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

CloudFront Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. CloudFront

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Beginnen Sie mit einem einfachen Netzwerk zur Verteilung von Inhalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Laden Sie Inhalte in den Bucket hoch
- Erstellen Sie eine CloudFront Distribution mit OAC
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudFront Getting Started Tutorial Script
# This script creates an S3 bucket, uploads sample content, creates a CloudFront
 distribution with OAC,
# and demonstrates how to access content through CloudFront.

set -euo pipefail

# Set up logging
LOG_FILE="cloudfront-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting CloudFront Getting Started Tutorial at $(date)"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created before error:"
    if [ -n "${BUCKET_NAME:-}" ]; then
        echo "- S3 Bucket: $BUCKET_NAME"
```

```
    fi
    if [ -n "${OAC_ID:-}" ]; then
        echo "- CloudFront Origin Access Control: $OAC_ID"
    fi
    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "- CloudFront Distribution: $DISTRIBUTION_ID"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "Cleaning up resources..."

    if [ -n "${DISTRIBUTION_ID:-}" ]; then
        echo "Disabling CloudFront distribution $DISTRIBUTION_ID..."

        # Get the current configuration and ETag in one call
        DIST_CONFIG=$(aws cloudfront get-distribution-config --id "$DISTRIBUTION_ID"
 2>/dev/null) || {
            echo "Failed to get distribution config. Continuing with cleanup..."
            DIST_CONFIG=""
        }

        if [ -n "$DIST_CONFIG" ]; then
            ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')

            # Modify and update distribution in one pipeline
            if echo "$DIST_CONFIG" | jq '.DistributionConfig.Enabled = false' | \
                aws cloudfront update-distribution \
                    --id "$DISTRIBUTION_ID" \
                    --distribution-config "$(cat)" \
                    --if-match "$ETAG" 2>/dev/null; then

                echo "Waiting for distribution to be disabled (this may take several
 minutes)..."
                aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID"
 2>/dev/null || {
                    echo "Distribution deployment wait timed out. Proceeding with
 deletion..."
                }
```

```
                # Get fresh ETag for deletion
                DIST_CONFIG=$(aws cloudfront get-distribution-config --id
 "$DISTRIBUTION_ID" 2>/dev/null) || {
                    echo "Failed to get updated config. Skipping distribution
 deletion..."
                    DIST_CONFIG=""
                }

                if [ -n "$DIST_CONFIG" ]; then
                    ETAG=$(echo "$DIST_CONFIG" | jq -r '.ETag')
                    aws cloudfront delete-distribution --id "$DISTRIBUTION_ID" --if-
match "$ETAG" 2>/dev/null && \
                        echo "CloudFront distribution deleted." || \
                        echo "Failed to delete distribution. You may need to delete
 it manually."
                fi
            else
                echo "Failed to disable distribution. Continuing with cleanup..."
            fi
        fi
    fi

    if [ -n "${OAC_ID:-}" ]; then
        echo "Deleting Origin Access Control $OAC_ID..."
        OAC_DATA=$(aws cloudfront get-origin-access-control --id "$OAC_ID" 2>/dev/
null) || {
            echo "Failed to get Origin Access Control. You may need to delete it
 manually."
            OAC_DATA=""
        }

        if [ -n "$OAC_DATA" ]; then
            OAC_ETAG=$(echo "$OAC_DATA" | jq -r '.ETag')
            aws cloudfront delete-origin-access-control --id "$OAC_ID" --if-match
 "$OAC_ETAG" 2>/dev/null && \
                echo "Origin Access Control deleted." || \
                echo "Failed to delete Origin Access Control. You may need to delete
 it manually."
        fi
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "$BUCKET_IS_SHARED" != "true" ]; then
        echo "Deleting S3 bucket $BUCKET_NAME and its contents..."
```

```
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || {
            echo "Failed to remove bucket contents. Continuing with bucket
 deletion..."
        }

        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null && \
            echo "S3 bucket deleted." || \
            echo "Failed to delete bucket. You may need to delete it manually."
    fi

    # Clean up temporary files
    rm -f temp_disabled_config.json distribution-config.json bucket-policy.json 2>/
dev/null || true
    rm -rf temp_content 2>/dev/null || true
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Validate jq is available
if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

# Validate AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Initialize variables
BUCKET_NAME=""
OAC_ID=""
DISTRIBUTION_ID=""
BUCKET_IS_SHARED=false

# Generate a random identifier for the bucket name using secure random
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
    handle_error "Failed to generate random identifier"
```

```
fi

# Check for shared prereq bucket and get account ID in parallel calls
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null) || PREREQ_BUCKET=""

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="cloudfront-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Get AWS account ID early
ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
if [ $? -ne 0 ]; then
    handle_error "Failed to get AWS account ID"
fi

# Validate account ID format
if ! [[ "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    handle_error "Invalid AWS account ID format: $ACCOUNT_ID"
fi

# Create a temporary directory for content with restrictive permissions
TEMP_DIR="temp_content"
mkdir -p "$TEMP_DIR/css"
chmod 700 "$TEMP_DIR"
if [ $? -ne 0 ]; then
    handle_error "Failed to create temporary directory"
fi

# Step 1: Create an S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" != "true" ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    aws s3 mb "s3://$BUCKET_NAME" --region us-east-1
    if [ $? -ne 0 ]; then
        handle_error "Failed to create S3 bucket"
    fi
```

```
    aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=cloudfront-
gettingstarted}]'

    # Batch bucket configuration calls for efficiency
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled &
    aws s3api put-public-access-block \
        --bucket "$BUCKET_NAME" \
        --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 &
    aws s3api put-bucket-encryption \
        --bucket "$BUCKET_NAME" \
        --server-side-encryption-configuration '{
            "Rules": [
                {
                    "ApplyServerSideEncryptionByDefault": {
                        "SSEAlgorithm": "AES256"
                    }
                }
            ]
        }' &
    wait
fi

# Step 2: Create sample content
echo "Creating sample content..."
cat > "$TEMP_DIR/index.html" << 'EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
EOF

cat > "$TEMP_DIR/css/styles.css" << 'EOF'
body {
```

```
    font-family: Arial, sans-serif;
    margin: 40px;
    background-color: #f5f5f5;
}
h1 {
    color: #333;
    text-align: center;
}
EOF

# Set restrictive permissions on content files
chmod 600 "$TEMP_DIR/index.html" "$TEMP_DIR/css/styles.css"

# Step 3: Upload content to the S3 bucket with encryption and metadata
echo "Uploading content to S3 bucket..."
aws s3 cp "$TEMP_DIR/" "s3://$BUCKET_NAME/" --recursive \
    --sse AES256 \
    --metadata "Source=CloudFrontTutorial"
if [ $? -ne 0 ]; then
    handle_error "Failed to upload content to S3 bucket"
fi

# Step 4: Create Origin Access Control
echo "Creating Origin Access Control..."
OAC_RESPONSE=$(aws cloudfront create-origin-access-control \
    --origin-access-control-config Name="oac-for-
$BUCKET_NAME",SigningProtocol=sigv4,SigningBehavior=always,OriginAccessControlOriginType=s3)

if [ $? -ne 0 ]; then
    handle_error "Failed to create Origin Access Control"
fi

OAC_ID=$(echo "$OAC_RESPONSE" | jq -r '.OriginAccessControl.Id')
if [ -z "$OAC_ID" ] || [ "$OAC_ID" = "null" ]; then
    handle_error "Failed to extract OAC ID from response"
fi

# Validate OAC ID format (alphanumeric and hyphens)
if ! [[ "$OAC_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid OAC ID format: $OAC_ID"
fi

echo "Created Origin Access Control with ID: $OAC_ID"
```

```
# Step 5: Create CloudFront distribution
echo "Creating CloudFront distribution..."

# Validate bucket name format
if ! [[ "$BUCKET_NAME" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]]; then
    handle_error "Invalid S3 bucket name format: $BUCKET_NAME"
fi

# Create distribution configuration with improved security settings
cat > distribution-config.json << EOF
{
    "CallerReference": "cli-tutorial-$(date +%s)",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "S3-$BUCKET_NAME",
                "DomainName": "$BUCKET_NAME.s3.amazonaws.com",
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                },
                "OriginAccessControlId": "$OAC_ID"
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "S3-$BUCKET_NAME",
        "ViewerProtocolPolicy": "redirect-to-https",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": ["GET", "HEAD"],
            "CachedMethods": {
                "Quantity": 2,
                "Items": ["GET", "HEAD"]
            }
        },
        "DefaultTTL": 86400,
        "MinTTL": 0,
        "MaxTTL": 31536000,
        "Compress": true,
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
```

```
            }
        }
    },
    "Comment": "CloudFront distribution for tutorial",
    "Enabled": true,
    "WebACLId": "",
    "HttpVersion": "http2and3"
}
EOF

# Set restrictive permissions on config file before passing credentials
chmod 600 distribution-config.json

DIST_RESPONSE=$(aws cloudfront create-distribution --distribution-config file://
distribution-config.json)
if [ $? -ne 0 ]; then
    handle_error "Failed to create CloudFront distribution"
fi

DISTRIBUTION_ID=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.Id')
DOMAIN_NAME=$(echo "$DIST_RESPONSE" | jq -r '.Distribution.DomainName')

if [ -z "$DISTRIBUTION_ID" ] || [ "$DISTRIBUTION_ID" = "null" ] || [ -z
 "$DOMAIN_NAME" ] || [ "$DOMAIN_NAME" = "null" ]; then
    handle_error "Failed to extract distribution ID or domain name from response"
fi

# Validate distribution ID format
if ! [[ "$DISTRIBUTION_ID" =~ ^[A-Z0-9]+$ ]]; then
    handle_error "Invalid distribution ID format: $DISTRIBUTION_ID"
fi

echo "Created CloudFront distribution with ID: $DISTRIBUTION_ID"
echo "CloudFront domain name: $DOMAIN_NAME"

# Tag the CloudFront distribution
aws cloudfront tag-resource --resource "arn:aws:cloudfront::
$ACCOUNT_ID:distribution/$DISTRIBUTION_ID" --tags 'Items=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=cloudfront-gettingstarted}]'

# Step 6: Update S3 bucket policy
echo "Updating S3 bucket policy..."

cat > bucket-policy.json << EOF
```

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::$BUCKET_NAME/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn": "arn:aws:cloudfront::$ACCOUNT_ID:distribution/
$DISTRIBUTION_ID"
                }
            }
        }
    ]
}
EOF

# Set restrictive permissions on policy file
chmod 600 bucket-policy.json

aws s3api put-bucket-policy --bucket "$BUCKET_NAME" --policy file://bucket-
policy.json
if [ $? -ne 0 ]; then
    handle_error "Failed to update S3 bucket policy"
fi

# Step 7: Wait for distribution to deploy
echo "Waiting for CloudFront distribution to deploy (this may take 5-10 minutes)..."
aws cloudfront wait distribution-deployed --id "$DISTRIBUTION_ID" 2>/dev/null || {
    echo "Warning: Distribution deployment wait timed out. The distribution may
 still be deploying."
}

echo "CloudFront distribution is now deployed."

# Step 8: Display access information
echo ""
echo "===== CloudFront Distribution Setup Complete ======"
echo "You can access your content at: https://$DOMAIN_NAME/index.html"
```

```
echo ""
echo "Resources created:"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- CloudFront Origin Access Control: $OAC_ID"
echo "- CloudFront Distribution: $DISTRIBUTION_ID"
echo ""
echo "To clean up resources, run: cleanup"
echo ""

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateDistribution
  - CreateOriginAccessControl
  - DeleteDistribution
  - DeleteOriginAccessControl
  - GetDistribution
  - GetDistributionConfig
  - GetOriginAccessControl
  - UpdateDistribution
  - WaitDistributionDeployed

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleichen von Zeichenfolgen hinzu
- Verwaltete Regeln hinzufügen
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
```

```
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
```

```
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
```

```
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "====================================================="
echo "STEP 1: Creating Web ACL"
echo "====================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
```

```
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "======================================================"
echo "STEP 2: Adding String Match Rule"
echo "======================================================"

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
```

```
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true
```

```
    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
```

```
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
```

```
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="
```

```
    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
```

```
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

CloudWatch Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. CloudWatch

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Monitoring-Dashboards mit einem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
```

```
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
```

```
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi
```

```
    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
```

```
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
```

```
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
```

```
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
```

```
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
```

```
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
```

```
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi
```

```
echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Versuchsvorlage
- Führen Sie das Experiment aus
- Überprüfe die Ergebnisse
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
```

```
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi
```

```
    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
```

```
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10
```

```
echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"
```

```
echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"
```

```
# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
```

```
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
```

```
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
```

```
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
```

```
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
"stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
```

```
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole

- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Lambda-Funktionen für die Überwachung
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
```

```
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
```

```
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
```

```
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
```

```
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

CloudWatch Loggt Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit CloudWatch Logs Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Konfigurieren Sie die Container-Dienstkonnektivität

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
```

```
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
```

```
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}
```

```
# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"
```

```
    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}
```

```
# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
```

```
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null
```

```
        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
```

```
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
```

```
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
```

```
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi
```

```
    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)
```

```
        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
```

```
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
```

```
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
```

```
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Lambda
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
```

```
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

###############################################################################
# Setup
###############################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

###############################################################################
# Helper functions
###############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"
```

```
        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
```

```
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
```

```
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "============================================"
echo "AWS Lambda - Create Your First Function"
echo "============================================"
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
```

```
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
```

```
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi
```

```
echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
```

```
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
```

```
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

##############################################################################
# Step 4: View CloudWatch logs
##############################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""
```

```
echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################
```

```
echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

Beispiele für Amazon Cognito Identity Provider unter AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie das with Bash-Skript AWS Command Line Interface mit Amazon Cognito Identity Provider verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Benutzerpools

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Benutzerpools
- Erstellen eines App-Clients
- Richten Sie eine Domain für Ihren Benutzerpool ein
- Erstellen Sie einen Benutzer als Administrator
- Aktivieren Sie die Selbstregistrierung
- Benutzer im Benutzerpool auflisten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Cognito User Pools Getting Started Script
# This script creates and configures an Amazon Cognito user pool with an app client

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cognito-user-pool-setup.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Cognito User Pool setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -qi "error\|failed"; then
    echo "ERROR: Command failed: $cmd" >&2
    echo "Output: $output" >&2
    cleanup_on_error
    exit 1
  fi
}

# Function to check AWS CLI return code
check_aws_error() {
  local exit_code=$?
  local cmd=$1

  if [ $exit_code -ne 0 ]; then
    echo "ERROR: AWS CLI command failed with exit code $exit_code: $cmd" >&2
    cleanup_on_error
    exit "$exit_code"
  fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
  echo "Error encountered. Attempting to clean up resources..." >&2

  if [ -n "${DOMAIN_NAME:-}" ] && [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool domain: $DOMAIN_NAME" >&2
    aws cognito-idp delete-user-pool-domain \
      --user-pool-id "$USER_POOL_ID" \
      --domain "$DOMAIN_NAME" 2>/dev/null || true
  fi

  if [ -n "${USER_POOL_ID:-}" ]; then
    echo "Deleting user pool: $USER_POOL_ID" >&2
    aws cognito-idp delete-user-pool \
      --user-pool-id "$USER_POOL_ID" 2>/dev/null || true
  fi
}

# Set trap for cleanup on exit
trap cleanup_on_error EXIT ERR

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
  echo "ERROR: AWS CLI is not installed" >&2
  exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
  echo "ERROR: jq is not installed" >&2
  exit 1
fi

# Validate openssl is installed
if ! command -v openssl &> /dev/null; then
  echo "ERROR: openssl is not installed" >&2
  exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
  echo "ERROR: AWS CLI is not configured or credentials are invalid" >&2
  exit 1
fi
```

```
# Get the current AWS region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
  AWS_REGION="us-east-1" # Default region if not configured
fi
echo "Using AWS Region: $AWS_REGION"

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
  echo "ERROR: Invalid AWS region format: $AWS_REGION" >&2
  exit 1
fi

# Generate random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ]; then
  echo "ERROR: Failed to generate random identifier" >&2
  exit 1
fi

USER_POOL_NAME="MyUserPool-${RANDOM_ID}"
APP_CLIENT_NAME="MyAppClient-${RANDOM_ID}"
DOMAIN_NAME="my-auth-domain-${RANDOM_ID}"

# Validate resource names don't exceed limits
if [ ${#USER_POOL_NAME} -gt 128 ]; then
  echo "ERROR: User pool name exceeds maximum length of 128 characters" >&2
  exit 1
fi

if [ ${#APP_CLIENT_NAME} -gt 128 ]; then
  echo "ERROR: App client name exceeds maximum length of 128 characters" >&2
  exit 1
fi

echo "Using random identifier: $RANDOM_ID"
echo "User pool name: $USER_POOL_NAME"
echo "App client name: $APP_CLIENT_NAME"
echo "Domain name: $DOMAIN_NAME"

# Step 1: Create a User Pool
echo "Creating user pool..."
USER_POOL_OUTPUT=$(aws cognito-idp create-user-pool \
```

```
  --pool-name "$USER_POOL_NAME" \
  --auto-verified-attributes email \
  --username-attributes email \
  --policies '{"PasswordPolicy":
{"MinimumLength":12,"RequireUppercase":true,"RequireLowercase":true,"RequireNumbers":true,"R
 \
  --schema '[{"Name":"email","Required":true,"Mutable":true}]' \
  --mfa-configuration OFF \
  --user-attribute-update-settings '{"AttributesRequireVerificationBeforeUpdate":
["email"]}' \
  --account-recovery-setting 'RecoveryMechanisms=[{Name=verified_email,Priority=1}]'
 \
  --deletion-protection INACTIVE \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool"

# Extract the User Pool ID using jq for safety
USER_POOL_ID=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Id // empty')
if [ -z "$USER_POOL_ID" ]; then
  echo "ERROR: Failed to extract User Pool ID" >&2
  exit 1
fi

# Validate User Pool ID format
if ! [[ "$USER_POOL_ID" =~ ^[a-z]{2}-[a-z]+-[0-9]+_[a-zA-Z0-9]+$ ]]; then
  echo "ERROR: Invalid User Pool ID format: $USER_POOL_ID" >&2
  exit 1
fi

echo "User Pool created with ID: $USER_POOL_ID"

USER_POOL_ARN=$(echo "$USER_POOL_OUTPUT" | jq -r '.UserPool.Arn // empty')
aws cognito-idp tag-resource \
  --resource-arn "$USER_POOL_ARN" \
  --tags project=doc-smith,tutorial=amazon-cognito-gs

# Wait for user pool to be ready
echo "Waiting for user pool to be ready..."
sleep 5

# Step 2: Create an App Client with enhanced security
echo "Creating app client..."
APP_CLIENT_OUTPUT=$(aws cognito-idp create-user-pool-client \
```

```
  --user-pool-id "$USER_POOL_ID" \
  --client-name "$APP_CLIENT_NAME" \
  --no-generate-secret \
  --explicit-auth-flows ALLOW_REFRESH_TOKEN_AUTH ALLOW_USER_PASSWORD_AUTH \
  --callback-urls '["https://localhost:3000/callback"]' \
  --allowed-o-auth-flows 'code' \
  --allowed-o-auth-scopes 'openid' 'email' 'profile' \
  --allowed-o-auth-flows-user-pool-client \
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --access-token-validity 1 \
  --id-token-validity 1 \
  --refresh-token-validity 30 \
  --token-validity-units 'AccessToken=hours,IdToken=hours,RefreshToken=days' \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-client"

# Extract the Client ID using jq for safety
CLIENT_ID=$(echo "$APP_CLIENT_OUTPUT" | jq -r '.UserPoolClient.ClientId // empty')
if [ -z "$CLIENT_ID" ]; then
  echo "ERROR: Failed to extract Client ID" >&2
  cleanup_on_error
  exit 1
fi

# Validate Client ID format
if ! [[ "$CLIENT_ID" =~ ^[a-z0-9]{26}$ ]]; then
  echo "ERROR: Invalid Client ID format: $CLIENT_ID" >&2
  cleanup_on_error
  exit 1
fi

echo "App Client created with ID: $CLIENT_ID"

# Step 3: Set Up a Domain for Your User Pool
echo "Setting up user pool domain..."
DOMAIN_OUTPUT=$(aws cognito-idp create-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "create-user-pool-domain"
echo "Domain created: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
```

```
# Step 4: View User Pool Details
echo "Retrieving user pool details..."
USER_POOL_DETAILS=$(aws cognito-idp describe-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool"
echo "User Pool details retrieved successfully"

# Step 5: View App Client Details
echo "Retrieving app client details..."
APP_CLIENT_DETAILS=$(aws cognito-idp describe-user-pool-client \
  --user-pool-id "$USER_POOL_ID" \
  --client-id "$CLIENT_ID" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "describe-user-pool-client"
echo "App Client details retrieved successfully"

# Step 6: Create a User (Admin)
echo "Creating admin user..."
ADMIN_USER_EMAIL="admin@example.com"
TEMP_PASSWORD="$(openssl rand -base64 12 | tr -d '\n')!@#"
if [ -z "$TEMP_PASSWORD" ]; then
  echo "ERROR: Failed to generate temporary password" >&2
  cleanup_on_error
  exit 1
fi

ADMIN_USER_OUTPUT=$(aws cognito-idp admin-create-user \
  --user-pool-id "$USER_POOL_ID" \
  --username "$ADMIN_USER_EMAIL" \
  --user-attributes Name=email,Value="$ADMIN_USER_EMAIL"
 Name=email_verified,Value=true \
  --temporary-password "$TEMP_PASSWORD" \
  --message-action SUPPRESS \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-create-user"
echo "Admin user created: $ADMIN_USER_EMAIL"

# Securely clear temporary password from memory
unset TEMP_PASSWORD
```

```
# Step 7: Self-Registration
echo "Demonstrating self-registration..."
USER_EMAIL="user@example.com"
USER_PASSWORD="SecurePassword123!"
SIGNUP_OUTPUT=$(aws cognito-idp sign-up \
  --client-id "$CLIENT_ID" \
  --username "$USER_EMAIL" \
  --password "$USER_PASSWORD" \
  --user-attributes Name=email,Value="$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "sign-up"
echo "User signed up: $USER_EMAIL"
echo "A confirmation code would be sent to the user's email in a real scenario"

# Securely clear password from memory
unset USER_PASSWORD

echo ""
echo "====================================================="
echo "IMPORTANT: In a real scenario, the user would receive"
echo "a confirmation code via email. For this demo, we'll"
echo "use admin-confirm-sign-up instead."
echo "====================================================="
echo ""

# Step 8: Confirm User Registration (using admin privileges for demo)
echo "Confirming user registration (admin method)..."
CONFIRM_OUTPUT=$(aws cognito-idp admin-confirm-sign-up \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-confirm-sign-up"
echo "User confirmed: $USER_EMAIL"

# Step 9: Set permanent password for user
echo "Setting permanent password for user..."
SET_PASSWORD="SecureUserPassword123!"
SET_PASS_OUTPUT=$(aws cognito-idp admin-set-user-password \
  --user-pool-id "$USER_POOL_ID" \
  --username "$USER_EMAIL" \
  --password "$SET_PASSWORD" \
```

```
  --permanent \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "admin-set-user-password"
echo "Permanent password set for user"

unset SET_PASSWORD

# Step 10: Authenticate a User
echo "Authenticating user..."
AUTH_PASSWORD="SecureUserPassword123!"
AUTH_OUTPUT=$(aws cognito-idp initiate-auth \
  --client-id "$CLIENT_ID" \
  --auth-flow USER_PASSWORD_AUTH \
  --auth-parameters "USERNAME=$USER_EMAIL,PASSWORD=$AUTH_PASSWORD" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "initiate-auth"
echo "User authenticated successfully"

unset AUTH_PASSWORD

# Extract auth tokens securely
ID_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.IdToken // empty')
ACCESS_TOKEN=$(echo "$AUTH_OUTPUT" | jq -r '.AuthenticationResult.AccessToken //
 empty')

# Validate tokens exist
if [ -z "$ID_TOKEN" ] || [ -z "$ACCESS_TOKEN" ]; then
  echo "WARNING: Failed to extract authentication tokens" >&2
else
  echo "Authentication tokens obtained successfully"
fi

# Securely clear tokens from memory
unset ID_TOKEN
unset ACCESS_TOKEN

# Step 11: List Users in the User Pool
echo "Listing users in the user pool..."
USERS_OUTPUT=$(aws cognito-idp list-users \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
```

```
check_aws_error "list-users"
echo "Users listed successfully"

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "User Pool ID: $USER_POOL_ID"
echo "User Pool Name: $USER_POOL_NAME"
echo "App Client ID: $CLIENT_ID"
echo "App Client Name: $APP_CLIENT_NAME"
echo "Domain: $DOMAIN_NAME.auth.$AWS_REGION.amazoncognito.com"
echo "Admin User: $ADMIN_USER_EMAIL"
echo "Regular User: $USER_EMAIL"
echo "===================================================="
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting cleanup process..."

# Step 12: Clean Up Resources
echo "Deleting user pool domain..."
DELETE_DOMAIN_OUTPUT=$(aws cognito-idp delete-user-pool-domain \
  --user-pool-id "$USER_POOL_ID" \
  --domain "$DOMAIN_NAME" \
  --region "$AWS_REGION" \
  2>&1)
check_aws_error "delete-user-pool-domain"
echo "Domain deleted successfully"

# Wait for domain deletion to complete
echo "Waiting for domain deletion to complete..."
sleep 5

echo "Deleting user pool (this will also delete the app client)..."
DELETE_POOL_OUTPUT=$(aws cognito-idp delete-user-pool \
  --user-pool-id "$USER_POOL_ID" \
  --region "$AWS_REGION" \
  2>&1)
```

```
check_aws_error "delete-user-pool"
echo "User pool deleted successfully"

echo "All resources have been cleaned up"

echo "Script completed at $(date)"

# Remove trap to prevent cleanup on successful exit
trap - EXIT ERR
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AdminConfirmSignUp
  - AdminCreateUser
  - CreateUserPool
  - CreateUserPoolClient
  - CreateUserPoolDomain
  - DeleteUserPool
  - DeleteUserPoolDomain
  - DescribeUserPool
  - DescribeUserPoolClient
  - InitiateAuth
  - ListUsers
  - SignUp

AWS Config Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS Config

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit der Konfigurationsverwaltung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config-Konfigurationsrekorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
```

```
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=========================================="
    echo "CREATED RESOURCES"
    echo "=========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
```

```
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "==========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
```

```
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
```

```
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
```

```
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
```

```
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
```

```
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources
```

```
# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel

- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Verbinden Sie Kundenbeispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Connect Customer Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Eine Contact-Center-Instanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Amazon-Connect-Instance
- Konfigurieren Sie einen Administratorbenutzer
- Konfigurieren Sie die Telefonieoptionen
- Richten Sie eine Telefonnummer ein
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon Connect instance using AWS CLI
# This script follows the steps in the Amazon Connect instance creation tutorial

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="connect-instance-creation.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting Amazon Connect instance creation at $(date)" > "$LOG_FILE"

# Set default region
AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"

# Validate AWS CLI is installed and credentials are available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid" | tee -a "$LOG_FILE"
    exit 1
fi

# Function to log commands and their output
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
```

```
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"

    if [[ -n "${INSTANCE_ID:-}" ]]; then
        echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
        log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --region
 '$AWS_REGION'" || true
    fi

    echo "Cleanup completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
}

# Set trap to clean up on error
trap cleanup_on_error ERR EXIT

# Function to wait for instance to be fully active
wait_for_instance() {
    local instance_id="$1"
    local max_attempts=30
    local attempt=1

    echo "Waiting for instance $instance_id to become fully active..." | tee -a
 "$LOG_FILE"

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt of $max_attempts: Checking instance status..." | tee
 -a "$LOG_FILE"
```

```
        # Try to describe the instance
        local result
        result=$(log_cmd "aws connect describe-instance --instance-id '$instance_id'
 --region '$AWS_REGION' --output json" 2>&1) || true

        # Check if the command was successful and instance status is ACTIVE
        if [[ $? -eq 0 && "$result" =~ "ACTIVE" ]]; then
            echo "Instance is now fully active and ready to use." | tee -a
 "$LOG_FILE"
            return 0
        fi

        echo "Instance not fully active yet. Waiting 30 seconds before next
 check..." | tee -a "$LOG_FILE"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for instance to become fully active." | tee -a
 "$LOG_FILE"
    return 1
}

# Function to check and handle existing instances
check_existing_instances() {
    echo "Checking for existing Amazon Connect instances..." | tee -a "$LOG_FILE"

    local instances
    instances=$(log_cmd "aws connect list-instances --region '$AWS_REGION' --output
 json" 2>&1) || true
    if [[ $? -ne 0 ]]; then
        echo "ERROR: Failed to list existing instances" | tee -a "$LOG_FILE"
        return 1
    fi

    # Check if there are any instances
    local instance_count
    instance_count=$(echo "$instances" | jq '.InstanceSummaryList | length' 2>/dev/
null || echo 0)

    if [[ $instance_count -gt 0 ]]; then
        echo "Found $instance_count existing Amazon Connect instance(s)" | tee -a
 "$LOG_FILE"
```

```
        echo "$instances" | jq '.InstanceSummaryList[] | {Id, Alias}' 2>/dev/null |
 tee -a "$LOG_FILE" || true

        echo ""
        echo "============================================"
        echo "EXISTING INSTANCES FOUND"
        echo "============================================"
        echo "Found $instance_count existing Amazon Connect instance(s)."
        echo "Auto-deleting existing instances to free up quota..." | tee -a
 "$LOG_FILE"

        echo "Deleting existing instances..." | tee -a "$LOG_FILE"

        # Extract instance IDs and delete each one
        local instance_ids
        instance_ids=$(echo "$instances" | jq -r '.InstanceSummaryList[].Id' 2>/dev/
null || echo "")

        while IFS= read -r id; do
            if [[ -n "$id" ]]; then
                echo "Deleting instance: $id" | tee -a "$LOG_FILE"
                log_cmd "aws connect delete-instance --instance-id '$id' --region
 '$AWS_REGION'" || {
                    echo "WARNING: Failed to delete instance $id" | tee -a
 "$LOG_FILE"
                }

                # Wait a bit between deletions
                sleep 5
            fi
        done <<< "$instance_ids"

        echo "Waiting for deletions to complete..." | tee -a "$LOG_FILE"
        sleep 30
    else
        echo "No existing Amazon Connect instances found" | tee -a "$LOG_FILE"
    fi

    return 0
}

# Check for existing instances before proceeding
check_existing_instances
```

```
# Generate a random instance alias to avoid naming conflicts
INSTANCE_ALIAS="connect-instance-$(openssl rand -hex 6 2>/dev/null || date +%s%N)"
echo "Using instance alias: $INSTANCE_ALIAS" | tee -a "$LOG_FILE"

# Step 1: Create Amazon Connect instance
echo "Step 1: Creating Amazon Connect instance..." | tee -a "$LOG_FILE"
INSTANCE_RESULT=$(log_cmd "aws connect create-instance --identity-management-type
 CONNECT_MANAGED --instance-alias '$INSTANCE_ALIAS' --inbound-calls-enabled --
outbound-calls-enabled --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_RESULT" $? "Failed to create Amazon Connect instance";
 then
    # Check if the error is due to quota limit
    if [[ "$INSTANCE_RESULT" =~ "ServiceQuotaExceededException" ||
 "$INSTANCE_RESULT" =~ "Quota limit reached" ]]; then
        echo "Quota limit reached for Amazon Connect instances. Please delete
 existing instances or request a quota increase." | tee -a "$LOG_FILE"
    fi
    exit 1
fi

# Extract instance ID from the result using jq
INSTANCE_ID=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceId' 2>/dev/null || echo "")
INSTANCE_ARN=$(echo "$INSTANCE_RESULT" | jq -r '.InstanceArn' 2>/dev/null || echo
 "")

if [[ -z "$INSTANCE_ID" ]]; then
    echo "ERROR: Failed to extract instance ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created Amazon Connect instance with ID: $INSTANCE_ID" | tee -a
 "$LOG_FILE"
echo "Instance ARN: $INSTANCE_ARN" | tee -a "$LOG_FILE"

# Tag the instance
aws connect tag-resource --resource-arn "$INSTANCE_ARN" \
    --tags project=doc-smith,tutorial=connect-gs \
    --region "$AWS_REGION"

# Wait for the instance to be fully created and active
if ! wait_for_instance "$INSTANCE_ID"; then
    echo "ERROR: Instance did not become fully active within the timeout period" |
 tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Step 2: Get security profiles to find the Admin profile ID
echo "Step 2: Getting security profiles..." | tee -a "$LOG_FILE"
SECURITY_PROFILES=$(log_cmd "aws connect list-security-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$SECURITY_PROFILES" $? "Failed to list security profiles"; then
    exit 1
fi

# Extract Admin security profile ID using jq
ADMIN_PROFILE_ID=$(echo "$SECURITY_PROFILES" | jq -r '.SecurityProfileSummaryList[]
 | select(.Name=="Admin") | .Id' 2>/dev/null | head -1 || echo "")

if [[ -z "$ADMIN_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find Admin security profile ID" | tee -a "$LOG_FILE"
    echo "Available security profiles:" | tee -a "$LOG_FILE"
    echo "$SECURITY_PROFILES" | jq '.SecurityProfileSummaryList[] | {Id, Name}' 2>/
dev/null | tee -a "$LOG_FILE" || echo "$SECURITY_PROFILES" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Found Admin security profile ID: $ADMIN_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 3: Get routing profiles to find a default routing profile ID
echo "Step 3: Getting routing profiles..." | tee -a "$LOG_FILE"
ROUTING_PROFILES=$(log_cmd "aws connect list-routing-profiles --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$ROUTING_PROFILES" $? "Failed to list routing profiles"; then
    exit 1
fi

# Extract the first routing profile ID using jq
ROUTING_PROFILE_ID=$(echo "$ROUTING_PROFILES" | jq -r
 '.RoutingProfileSummaryList[0].Id' 2>/dev/null || echo "")

if [[ -z "$ROUTING_PROFILE_ID" ]]; then
    echo "ERROR: Failed to find a routing profile ID" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
echo "Found routing profile ID: $ROUTING_PROFILE_ID" | tee -a "$LOG_FILE"

# Step 4: Create an admin user
echo "Step 4: Creating admin user..." | tee -a "$LOG_FILE"

# Generate a secure password
ADMIN_PASSWORD="Connect$(openssl rand -base64 12 2>/dev/null || head -c 12 /dev/
urandom | base64)"

USER_RESULT=$(log_cmd "aws connect create-user --instance-id '$INSTANCE_ID'
 --username admin --password '$ADMIN_PASSWORD' --identity-info
 FirstName=Admin,LastName=User,Email=admin@example.com --phone-config
 PhoneType=DESK_PHONE,AutoAccept=true,AfterContactWorkTimeLimit=30,DeskPhoneNumber=
+12065550100 --security-profile-ids '$ADMIN_PROFILE_ID' --routing-profile-id
 '$ROUTING_PROFILE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$USER_RESULT" $? "Failed to create admin user"; then
    exit 1
fi

# Extract user ID using jq
USER_ID=$(echo "$USER_RESULT" | jq -r '.UserId' 2>/dev/null || echo "")

if [[ -z "$USER_ID" ]]; then
    echo "ERROR: Failed to extract user ID from the result" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Successfully created admin user with ID: $USER_ID" | tee -a "$LOG_FILE"
echo "Admin password: $ADMIN_PASSWORD" | tee -a "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Step 5: Configure telephony options
echo "Step 5: Configuring telephony options..." | tee -a "$LOG_FILE"

# Enable early media
EARLY_MEDIA_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-id
 '$INSTANCE_ID' --attribute-type EARLY_MEDIA --value true --region '$AWS_REGION'"
 2>&1) || true

if ! check_error "$EARLY_MEDIA_RESULT" $? "Failed to enable early media"; then
    exit 1
fi
```

```
# Enable multi-party calls and enhanced monitoring for voice
MULTI_PARTY_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_RESULT" $? "Failed to enable multi-party calls"; then
    exit 1
fi

# Enable multi-party chats and enhanced monitoring for chat
MULTI_PARTY_CHAT_RESULT=$(log_cmd "aws connect update-instance-attribute --instance-
id '$INSTANCE_ID' --attribute-type MULTI_PARTY_CHAT_CONFERENCE --value true --region
 '$AWS_REGION'" 2>&1) || true

if ! check_error "$MULTI_PARTY_CHAT_RESULT" $? "Failed to enable multi-party chats";
 then
    exit 1
fi

echo "Successfully configured telephony options" | tee -a "$LOG_FILE"

# Step 6: View storage configurations
echo "Step 6: Viewing storage configurations..." | tee -a "$LOG_FILE"

# List storage configurations for chat transcripts
STORAGE_CONFIGS=$(log_cmd "aws connect list-instance-storage-configs --instance-id
 '$INSTANCE_ID' --resource-type CHAT_TRANSCRIPTS --region '$AWS_REGION' --output
 json" 2>&1) || true

if ! check_error "$STORAGE_CONFIGS" $? "Failed to list storage configurations"; then
    exit 1
fi

echo "Successfully retrieved storage configurations" | tee -a "$LOG_FILE"

# Step 7: Verify instance details
echo "Step 7: Verifying instance details..." | tee -a "$LOG_FILE"
INSTANCE_DETAILS=$(log_cmd "aws connect describe-instance --instance-id
 '$INSTANCE_ID' --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$INSTANCE_DETAILS" $? "Failed to describe instance"; then
    exit 1
fi
```

```
echo "Successfully verified instance details" | tee -a "$LOG_FILE"

# Step 8: Search for available phone numbers (optional)
echo "Step 8: Searching for available phone numbers..." | tee -a "$LOG_FILE"
PHONE_NUMBERS=$(log_cmd "aws connect search-available-phone-numbers --target-arn
 '$INSTANCE_ARN' --phone-number-type TOLL_FREE --phone-number-country-code US --max-
results 5 --region '$AWS_REGION' --output json" 2>&1) || true

if ! check_error "$PHONE_NUMBERS" $? "Failed to search for available phone numbers";
 then
    exit 1
fi

# Extract the first phone number if available using jq
PHONE_NUMBER=$(echo "$PHONE_NUMBERS" | jq -r '.AvailableNumbersList[0].PhoneNumber'
 2>/dev/null || echo "")

if [[ -n "$PHONE_NUMBER" ]]; then
    echo "Found available phone number: $PHONE_NUMBER" | tee -a "$LOG_FILE"

    echo ""
    echo "==========================================="
    echo "CLAIM PHONE NUMBER"
    echo "==========================================="
    echo "Auto-claiming available phone number $PHONE_NUMBER..." | tee -a
 "$LOG_FILE"

    CLAIM_RESULT=$(log_cmd "aws connect claim-phone-number --target-arn
 '$INSTANCE_ARN' --phone-number '$PHONE_NUMBER' --region '$AWS_REGION' --output
 json" 2>&1) || true

    if ! check_error "$CLAIM_RESULT" $? "Failed to claim phone number"; then
        echo "WARNING: Failed to claim phone number, but continuing with script" |
 tee -a "$LOG_FILE"
    else
        echo "Successfully claimed phone number" | tee -a "$LOG_FILE"
        # Extract the phone number ID from the claim result using jq
        PHONE_NUMBER_ID=$(echo "$CLAIM_RESULT" | jq -r '.PhoneNumberId' 2>/dev/null
 || echo "")
    fi
else
    echo "No available phone numbers found" | tee -a "$LOG_FILE"
fi
```

```
# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Amazon Connect Instance ID: $INSTANCE_ID"
echo "Amazon Connect Instance ARN: $INSTANCE_ARN"
echo "Admin User ID: $USER_ID"
echo "Admin Username: admin"
echo "Admin Password: $ADMIN_PASSWORD"
if [[ -n "${PHONE_NUMBER:-}" ]]; then
    echo "Claimed Phone Number: $PHONE_NUMBER"
    if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
        echo "Claimed Phone Number ID: $PHONE_NUMBER_ID"
    fi
fi
echo "============================================"
echo ""

# Auto-confirm cleanup and clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Auto-starting cleanup..." | tee -a "$LOG_FILE"

# Release claimed phone number if applicable
if [[ -n "${PHONE_NUMBER_ID:-}" ]]; then
    echo "Releasing phone number: $PHONE_NUMBER_ID" | tee -a "$LOG_FILE"
    RELEASE_RESULT=$(log_cmd "aws connect release-phone-number --phone-number-id
 '$PHONE_NUMBER_ID' --region '$AWS_REGION'" 2>&1) || true

    if ! check_error "$RELEASE_RESULT" $? "Failed to release phone number"; then
        echo "WARNING: Failed to release phone number" | tee -a "$LOG_FILE"
    else
        echo "Successfully released phone number" | tee -a "$LOG_FILE"
    fi

    echo "Waiting for phone number release to complete..." | tee -a "$LOG_FILE"
    sleep 10
fi

# Delete the Amazon Connect instance (this will also delete all associated
 resources)
```

```
echo "Deleting Amazon Connect instance: $INSTANCE_ID" | tee -a "$LOG_FILE"
DELETE_RESULT=$(log_cmd "aws connect delete-instance --instance-id '$INSTANCE_ID' --
region '$AWS_REGION'" 2>&1) || true

if ! check_error "$DELETE_RESULT" $? "Failed to delete instance"; then
    echo "WARNING: Failed to delete instance" | tee -a "$LOG_FILE"
else
    echo "Successfully deleted instance" | tee -a "$LOG_FILE"
fi

echo "Cleanup completed. All resources have been deleted." | tee -a "$LOG_FILE"

echo "Script completed successfully. See $LOG_FILE for details." | tee -a
 "$LOG_FILE"

trap - ERR EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - ClaimPhoneNumber
  - CreateInstance
  - CreateUser
  - DeleteInstance
  - DescribeInstance
  - ListInstanceStorageConfigs
  - ListInstances
  - ListRoutingProfiles
  - ListSecurityProfiles
  - ReleasePhoneNumber
  - SearchAvailablePhoneNumbers
  - UpdateInstanceAttribute

Direct Connect Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. Direct Connect

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Beginnen Sie mit dedizierten Netzwerkverbindungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 CreateVpnGateway
- Benutze ec2 DeleteVpnGateway
- Benutze ec2 DescribeVpnGateways
- Verwenden Sie DirectConnect CreateConnection
- Verwenden Sie DirectConnect CreatePrivateVirtualInterface
- Verwenden Sie DirectConnect DeleteConnection

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect connections
 using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
```

```
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}

# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done
```

```
    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)

        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is $CONNECTION_STATE,
 waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
```

```
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
```

```
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"

# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID" 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
```

```
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"' -
f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
```

```
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o '"virtualInterfaceId":
 "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
fi

# Step 8: Check connection state and provide guidance for public virtual interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
```

```
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways

- UpdateConnection

Amazon DocumentDB-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Dokumentdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

###############################################################################
# Configuration
###############################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""

###############################################################################
# Region pre-check
###############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
```

```
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
```

```
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."
```

```
    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
```

```
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
secret."
    fi

    echo ""
```

```
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "============================================"
echo "Step 1: Create master password in Secrets Manager"
echo "============================================"
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi

SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

################################################################################
# Step 2: Find default VPC and subnets
```

```
##############################################################################
echo "============================================="
echo "Step 2: Find default VPC and subnets"
echo "============================================="
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
```

```
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

##############################################################################
# Step 3: Create subnet group
##############################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

##############################################################################
# Step 4: Create DocumentDB cluster
##############################################################################
echo "==========================================="
```

```
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""

##############################################################################
# Step 5: Create DocumentDB instance
##############################################################################
echo "==========================================="
echo "Step 5: Create DocumentDB instance"
echo "==========================================="
echo ""
```

```
INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

################################################################################
# Step 6: Get cluster endpoint and security group
################################################################################
echo "============================================"
echo "Step 6: Get cluster endpoint and security group"
echo "============================================"
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""
```

```
################################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
################################################################################
echo "============================================="
echo "Step 7: Add security group ingress rule"
echo "============================================="
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
```

```
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "============================================"
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================"
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

################################################################################
```

```
# Step 9: Display connection information
################################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

################################################################################
# Step 10: Cleanup
################################################################################
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo ""
echo "Resources created:"
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources
```

```
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

DynamoDB-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das DynamoDB-Szenario „Erste Schritte“.

```
###############################################################################
# function dynamodb_getting_started_movies
#
# Scenario to create an Amazon DynamoDB table and perform a series of operations on
 the table.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function dynamodb_getting_started_movies() {

  source ./dynamodb_operations.sh

  key_schema_json_file="dynamodb_key_schema.json"
  attribute_definitions_json_file="dynamodb_attr_def.json"
  item_json_file="movie_item.json"
  key_json_file="movie_key.json"
  batch_json_file="batch.json"
  attribute_names_json_file="attribute_names.json"
  attributes_values_json_file="attribute_values.json"

  echo_repeat "*" 88
  echo
  echo "Welcome to the Amazon DynamoDB getting started demo."
  echo
  echo_repeat "*" 88
  echo

  local table_name
  echo -n "Enter a name for a new DynamoDB table: "
  get_input
  table_name=$get_input_result

  echo '[
  {"AttributeName": "year", "KeyType": "HASH"},
   {"AttributeName": "title", "KeyType": "RANGE"}
  ]' >"$key_schema_json_file"

  echo '[
  {"AttributeName": "year", "AttributeType": "N"},
   {"AttributeName": "title", "AttributeType": "S"}
  ]' >"$attribute_definitions_json_file"

  if dynamodb_create_table -n "$table_name" -a "$attribute_definitions_json_file" \
    -k "$key_schema_json_file" 1>/dev/null; then
    echo "Created a DynamoDB table named $table_name"
  else
    errecho "The table failed to create. This demo will exit."
    clean_up
```

```
    return 1
  fi

  echo "Waiting for the table to become active...."

  if dynamodb_wait_table_active -n "$table_name"; then
    echo "The table is now active."
  else
    errecho "The table failed to become active. This demo will exit."
    cleanup "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo -n "Enter the title of a movie you want to add to the table: "
  get_input
  local added_title
  added_title=$get_input_result

  local added_year
  get_int_input "What year was it released? "
  added_year=$get_input_result

  local rating
  get_float_input "On a scale of 1 - 10, how do you rate it? " "1" "10"
  rating=$get_input_result

  local plot
  echo -n "Summarize the plot for me: "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"},
    "info": {"M" : {"plot": {"S" : "'"$plot"'"}, "rating": {"N" :"'"$rating"'"} } }
   }' >"$item_json_file"

  if dynamodb_put_item -n "$table_name" -i "$item_json_file"; then
    echo "The movie '$added_title' was successfully added to the table
'$table_name'."
```

```
  else
    errecho "Put item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo

  echo "Let's update your movie '$added_title'."
  get_float_input "You rated it $rating, what new rating would you give it? " "1"
"10"
  rating=$get_input_result

  echo -n "You summarized the plot as '$plot'."
  echo "What would you say now? "
  get_input
  plot=$get_input_result

  echo '{
    "year": {"N" :"'"$added_year"'"},
    "title": {"S" :  "'"$added_title"'"}
    }' >"$key_json_file"

  echo '{
    ":r": {"N" :"'"$rating"'"},
    ":p": {"S" : "'"$plot"'"}
   }' >"$item_json_file"

  local update_expression="SET info.rating = :r, info.plot = :p"

  if dynamodb_update_item -n "$table_name" -k "$key_json_file" -e
"$update_expression" -v "$item_json_file"; then
    echo "Updated '$added_title' with new attributes."
  else
    errecho "Update item failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo
  echo_repeat "*" 88
  echo
```

```
  echo "We will now use batch write to upload 150 movie entries into the table."

  local batch_json
  for batch_json in movie_files/movies_*.json; do
    echo "{ \"$table_name\" : $(<"$batch_json") }" >"$batch_json_file"
    if dynamodb_batch_write_item -i "$batch_json_file" 1>/dev/null; then
      echo "Entries in $batch_json added to table."
    else
      errecho "Batch write failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  done

  local title="The Lord of the Rings: The Fellowship of the Ring"
  local year="2001"

  if get_yes_no_input "Let's move on...do you want to get info about '$title'? (y/n)
"; then
    echo '{
  "year": {"N" :"'"$year"'"},
  "title": {"S" :  "'"$title"'"}
  }' >"$key_json_file"
    local info
    info=$(dynamodb_get_item -n "$table_name" -k "$key_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Get item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$info"
  fi

  local ask_for_year=true
  while [[ "$ask_for_year" == true ]]; do
    echo "Let's get a list of movies released in a given year."
    get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
    year=$get_input_result
    echo '{
```

```
    "#n": "year"
    }' >"$attribute_names_json_file"

    echo '{
    ":v": {"N" :"'"$year"'"}
    }' >"$attributes_values_json_file"

    response=$(dynamodb_query -n "$table_name" -k "#n=:v" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
      errecho "Query table failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi

    echo "Here is what I found:"
    echo "$response"

    if ! get_yes_no_input "Try another year? (y/n) "; then
      ask_for_year=false
    fi
  done

  echo "Now let's scan for movies released in a range of years. Enter a year: "
  get_int_input "Enter a year between 1972 and 2018: " "1972" "2018"
  local start=$get_input_result

  get_int_input "Enter another year: " "1972" "2018"
  local end=$get_input_result

  echo '{
    "#n": "year"
    }' >"$attribute_names_json_file"

  echo '{
    ":v1": {"N" : "'"$start"'"},
    ":v2": {"N" : "'"$end"'"}
    }' >"$attributes_values_json_file"

  response=$(dynamodb_scan -n "$table_name" -f "#n BETWEEN :v1 AND :v2" -a
"$attribute_names_json_file" -v "$attributes_values_json_file")
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "Scan table failed. This demo will exit."
    clean_up "$table_name"
    return 1
  fi

  echo "Here is what I found:"
  echo "$response"

  echo
  echo_repeat "*" 88
  echo

  echo "Let's remove your movie '$added_title' from the table."

  if get_yes_no_input "Do you want to remove '$added_title'? (y/n) "; then
    echo '{
  "year": {"N" :"'"$added_year"'"},
  "title": {"S" :  "'"$added_title"'"}
  }' >"$key_json_file"

    if ! dynamodb_delete_item -n "$table_name" -k "$key_json_file"; then
      errecho "Delete item failed. This demo will exit."
      clean_up "$table_name"
      return 1
    fi
  fi

  if get_yes_no_input "Do you want to delete the table '$table_name'? (y/n) "; then
    if ! clean_up "$table_name"; then
      return 1
    fi
  else
    if ! clean_up; then
      return 1
    fi
  fi

  return 0
}
```

Die in diesem Szenario verwendeten „DynamoDB“-Funktionen.

```
###############################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and their
 types.
#       -k key_schema -- JSON file path of a list of attributes and their key types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_definitions" ]]; then
    errecho "ERROR: You must provide an attribute definitions json file path the -a
parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}

###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
###############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_describe_table"
    echo "Describe the status of a DynamoDB table."
    echo "  -n table_name  -- The name of the table."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
```

```
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  local table_status
    table_status=$(
      aws dynamodb describe-table \
        --table-name "$table_name" \
        --output text \
        --query 'Table.TableStatus'
    )

   local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log "$error_code"
      errecho "ERROR: AWS reports describe-table operation failed.$table_status"
      return 1
    fi

  echo "$table_status"

  return 0
}

###########################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
```

```
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_put_item"
    echo "Put an item into a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -i item  -- Path to json file containing the item values."
    echo ""
  }

  while getopts "n:i:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$item" ]]; then
```

```
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""
  iecho ""

  response=$(aws dynamodb put-item \
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 update.
#       -e update expression  -- An expression that defines one or more attributes
 to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
##############################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_update_item"
    echo "Update an item in a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to update."
    echo " -e update expression  -- An expression that defines one or more
 attributes to be updated."
    echo " -v values  -- Path to json file containing the update values."
    echo ""
  }

  while getopts "n:k:e:v:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      e) update_expression="${OPTARG}" ;;
      v) values="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi
  if [[ -z "$update_expression" ]]; then
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi

  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_batch_write_item
```

```
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}

###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
```

```
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
```

```
      --output text \
      --query "$query")
  else
    response=$(
      aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"$keys" \
        --output text
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}

###########################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
```

```
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}

###########################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the expression
 attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
 expression attribute names."
```

```
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
    errecho "ERROR: You must provide expression attribute names with the -a
parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
```

```
#       -k keys  -- Path to json file containing the keys that identify the item to
 delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_delete_item"
    echo "Delete an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to delete."
    echo ""
  }
  while getopts "n:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$keys" ]]; then
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho ""

  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}

###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function dynamodb_delete_table"
  echo "Deletes an Amazon DynamoDB table."
  echo " -n table_name  -- The name of the table to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho ""

response=$(aws dynamodb delete-table \
  --table-name "$table_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-table operation failed.$response"
  return 1
```

```
  fi

  return 0
}
```

Die in diesem Szenario verwendeten Hilfsfunktionen.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
```

```
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_batch_get_item
#
# This function gets a batch of items from a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the keys of the items to get.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_batch_get_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_batch_get_item"
    echo "Get a batch of items from a DynamoDB table."
    echo " -i item  -- Path to json file containing the keys of the items to get."
    echo ""
  }
```

```
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  response=$(aws dynamodb batch-get-item \
    --request-items file://"$item")
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-get-item operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
```

```
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie BatchGetItem in der AWS CLI Befehlsreferenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_batch_write_item
#
# This function writes a batch of items into a DynamoDB table.
#
# Parameters:
#       -i item  -- Path to json file containing the items to write.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_batch_write_item() {
  local item response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_batch_write_item"
    echo "Write a batch of items into a DynamoDB table."
    echo " -i item  -- Path to json file containing the items to write."
    echo ""
  }
  while getopts "i:h" option; do
    case "${option}" in
      i) item="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$item" ]]; then
    errecho "ERROR: You must provide an item with the -i parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    item:   $item"
  iecho ""

  response=$(aws dynamodb batch-write-item \
    --request-items file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
```

```
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie BatchWriteItem in der AWS CLI Befehlsreferenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_create_table
#
# This function creates an Amazon DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to create.
#       -a attribute_definitions -- JSON file path of a list of attributes and their
 types.
#       -k key_schema -- JSON file path of a list of attributes and their key types.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
```

```
function dynamodb_create_table() {
  local table_name attribute_definitions key_schema response
  local option OPTARG # Required to use getopts command in a function.

  ########################################
  # Function usage explanation
  ########################################
  function usage() {
    echo "function dynamodb_create_table"
    echo "Creates an Amazon DynamoDB table with on-demand billing."
    echo " -n table_name  -- The name of the table to create."
    echo " -a attribute_definitions -- JSON file path of a list of attributes and
 their types."
    echo " -k key_schema -- JSON file path of a list of attributes and their key
 types."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:a:k:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      a) attribute_definitions="${OPTARG}" ;;
      k) key_schema="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_definitions" ]]; then
```

```
    errecho "ERROR: You must provide an attribute definitions json file path the -a
 parameter."
    usage
    return 1
  fi

  if [[ -z "$key_schema" ]]; then
    errecho "ERROR: You must provide a key schema json file path the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    attribute_definitions:   $attribute_definitions"
  iecho "    key_schema:   $key_schema"
  iecho ""

  response=$(aws dynamodb create-table \
    --table-name "$table_name" \
    --attribute-definitions file://"$attribute_definitions" \
    --billing-mode PAY_PER_REQUEST \
    --key-schema file://"$key_schema" )

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-table operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
```

```
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie CreateTable in der AWS CLI Befehlsreferenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_delete_item
#
# This function deletes an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_delete_item() {
  local table_name keys response
  local option OPTARG # Required to use getopts command in a function.
```

```
# ######################################
# Function usage explanation
######################################
function usage() {
  echo "function dynamodb_delete_item"
  echo "Delete an item from a DynamoDB table."
  echo " -n table_name  -- The name of the table."
  echo " -k keys  -- Path to json file containing the keys that identify the item
to delete."
  echo ""
}
while getopts "n:k:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    k) keys="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$keys" ]]; then
  errecho "ERROR: You must provide a keys json file path the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    keys:   $keys"
iecho ""
```

```
  response=$(aws dynamodb delete-item \
    --table-name "$table_name" \
    --key file://"$keys")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-item operation failed.$response"
    return 1
  fi

  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
```

```
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DeleteItem in der AWS CLI Befehlsreferenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_delete_table
#
# This function deletes a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table to delete.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function dynamodb_delete_table() {
  local table_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function dynamodb_delete_table"
    echo "Deletes an Amazon DynamoDB table."
    echo " -n table_name  -- The name of the table to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho ""

  response=$(aws dynamodb delete-table \
    --table-name "$table_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-table operation failed.$response"
    return 1
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
```

```
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
```

```
  fi

  return 0
}
```

- API-Details finden Sie DeleteTable in der AWS CLI Befehlsreferenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_describe_table
#
# This function returns the status of a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#
#  Response:
#       - TableStatus:
#     And:
#       0 - Table is active.
#       1 - If it fails.
###############################################################################
function dynamodb_describe_table {
  local table_name
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
  function usage() {
```

```
  echo "function dynamodb_describe_table"
  echo "Describe the status of a DynamoDB table."
  echo "  -n table_name  -- The name of the table."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

local table_status
  table_status=$(
    aws dynamodb describe-table \
      --table-name "$table_name" \
      --output text \
      --query 'Table.TableStatus'
  )

 local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log "$error_code"
    errecho "ERROR: AWS reports describe-table operation failed.$table_status"
    return 1
  fi
```

```
  echo "$table_status"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
```

```
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeTable in der AWS CLI Befehlsreferenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_get_item
#
# This function gets an item from a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 get.
#       [-q query]  -- Optional JMESPath query expression.
#
#  Returns:
#       The item as text output.
#  And:
#       0 - If successful.
```

```
#       1 - If it fails.
##############################################################################
function dynamodb_get_item() {
  local table_name keys query response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_get_item"
    echo "Get an item from a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k keys  -- Path to json file containing the keys that identify the item
 to get."
    echo " [-q query]  -- Optional JMESPath query expression."
    echo ""
  }
  query=""
  while getopts "n:k:q:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) keys="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$keys" ]]; then
```

```
    errecho "ERROR: You must provide a keys json file path the -k parameter."
    usage
    return 1
  fi

  if [[ -n "$query" ]]; then
    response=$(aws dynamodb get-item \
      --table-name "$table_name" \
      --key file://"$keys" \
      --output text \
      --query "$query")
  else
    response=$(
      aws dynamodb get-item \
        --table-name "$table_name" \
        --key file://"$keys" \
        --output text
    )
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports get-item operation failed.$response"
    return 1
  fi

  if [[ -n "$query" ]]; then
    echo "$response" | sed "/^\t/s/\t//1" # Remove initial tab that the JMSEPath
 query inserts on some strings.
  else
    echo "$response"
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
#############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie GetItem in der AWS CLI Befehlsreferenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_list_tables
#
# This function lists all the tables in a DynamoDB.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function dynamodb_list_tables() {
  response=$(aws dynamodb list-tables \
    --output text \
    --query "TableNames")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports batch-write-item operation failed.$response"
    return 1
  fi

  echo "$response" | tr -s "[:space:]" "\n"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
```

```
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie ListTables in der AWS CLI Befehlsreferenz.

PutItem

Das folgende Codebeispiel zeigt die VerwendungPutItem.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_put_item
#
# This function puts an item into a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -i item  -- Path to json file containing the item values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function dynamodb_put_item() {
  local table_name item response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
  ######################################
```

```
function usage() {
  echo "function dynamodb_put_item"
  echo "Put an item into a DynamoDB table."
  echo " -n table_name  -- The name of the table."
  echo " -i item  -- Path to json file containing the item values."
  echo ""
}

while getopts "n:i:h" option; do
  case "${option}" in
    n) table_name="${OPTARG}" ;;
    i) item="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$table_name" ]]; then
  errecho "ERROR: You must provide a table name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$item" ]]; then
  errecho "ERROR: You must provide an item with the -i parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    table_name:   $table_name"
iecho "    item:   $item"
iecho ""
iecho ""

response=$(aws dynamodb put-item \
```

```
    --table-name "$table_name" \
    --item file://"$item")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports put-item operation failed.$response"
    return 1
  fi

  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie PutItem in der AWS CLI Befehlsreferenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_query
#
# This function queries a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k key_condition_expression -- The key condition expression.
#       -a attribute_names -- Path to JSON file containing the attribute names.
#       -v attribute_values -- Path to JSON file containing the attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function dynamodb_query() {
  local table_name key_condition_expression attribute_names attribute_values
 projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_query"
    echo "Query a DynamoDB table."
    echo " -n table_name  -- The name of the table."
    echo " -k key_condition_expression -- The key condition expression."
    echo " -a attribute_names -- Path to JSON file containing the attribute names."
    echo " -v attribute_values -- Path to JSON file containing the attribute
 values."
```

```
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:k:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      k) key_condition_expression="${OPTARG}" ;;
      a) attribute_names="${OPTARG}" ;;
      v) attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$key_condition_expression" ]]; then
    errecho "ERROR: You must provide a key condition expression with the -k
parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_names" ]]; then
    errecho "ERROR: You must provide a attribute names with the -a parameter."
    usage
    return 1
  fi

  if [[ -z "$attribute_values" ]]; then
```

```
    errecho "ERROR: You must provide a attribute values with the -v parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values")
  else
    response=$(aws dynamodb query \
      --table-name "$table_name" \
      --key-condition-expression "$key_condition_expression" \
      --expression-attribute-names file://"$attribute_names" \
      --expression-attribute-values file://"$attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports query operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
```

```
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS CLI -Befehlsreferenz.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function dynamodb_scan
#
# This function scans a DynamoDB table.
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -f filter_expression  -- The filter expression.
#       -a expression_attribute_names -- Path to JSON file containing the expression
 attribute names.
#       -v expression_attribute_values -- Path to JSON file containing the
 expression attribute values.
#       [-p projection_expression]  -- Optional projection expression.
#
#  Returns:
#       The items as json output.
#  And:
#       0 - If successful.
#       1 - If it fails.
##########################################################################
function dynamodb_scan() {
  local table_name filter_expression expression_attribute_names
 expression_attribute_values projection_expression response
  local option OPTARG # Required to use getopts command in a function.

  # ######################################
  # Function usage explanation
  ######################################
  function usage() {
    echo "function dynamodb_scan"
    echo "Scan a DynamoDB table."
```

```
    echo " -n table_name  -- The name of the table."
    echo " -f filter_expression  -- The filter expression."
    echo " -a expression_attribute_names -- Path to JSON file containing the
expression attribute names."
    echo " -v expression_attribute_values -- Path to JSON file containing the
expression attribute values."
    echo " [-p projection_expression]  -- Optional projection expression."
    echo ""
  }

  while getopts "n:f:a:v:p:h" option; do
    case "${option}" in
      n) table_name="${OPTARG}" ;;
      f) filter_expression="${OPTARG}" ;;
      a) expression_attribute_names="${OPTARG}" ;;
      v) expression_attribute_values="${OPTARG}" ;;
      p) projection_expression="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$table_name" ]]; then
    errecho "ERROR: You must provide a table name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$filter_expression" ]]; then
    errecho "ERROR: You must provide a filter expression with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_names" ]]; then
```

```
    errecho "ERROR: You must provide expression attribute names with the -a
 parameter."
    usage
    return 1
  fi

  if [[ -z "$expression_attribute_values" ]]; then
    errecho "ERROR: You must provide expression attribute values with the -v
 parameter."
    usage
    return 1
  fi

  if [[ -z "$projection_expression" ]]; then
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values")
  else
    response=$(aws dynamodb scan \
      --table-name "$table_name" \
      --filter-expression "$filter_expression" \
      --expression-attribute-names file://"$expression_attribute_names" \
      --expression-attribute-values file://"$expression_attribute_values" \
      --projection-expression "$projection_expression")
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports scan operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS CLI -Befehlsreferenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function dynamodb_update_item
#
# This function updates an item in a DynamoDB table.
#
#
# Parameters:
#       -n table_name  -- The name of the table.
#       -k keys  -- Path to json file containing the keys that identify the item to
 update.
#       -e update expression  -- An expression that defines one or more attributes
 to be updated.
#       -v values  -- Path to json file containing the update values.
#
#  Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function dynamodb_update_item() {
  local table_name keys update_expression values response
  local option OPTARG # Required to use getopts command in a function.

  ######################################
  # Function usage explanation
```

```
 ######################################
 function usage() {
   echo "function dynamodb_update_item"
   echo "Update an item in a DynamoDB table."
   echo " -n table_name  -- The name of the table."
   echo " -k keys  -- Path to json file containing the keys that identify the item
to update."
   echo " -e update expression  -- An expression that defines one or more
attributes to be updated."
   echo " -v values  -- Path to json file containing the update values."
   echo ""
 }

 while getopts "n:k:e:v:h" option; do
   case "${option}" in
     n) table_name="${OPTARG}" ;;
     k) keys="${OPTARG}" ;;
     e) update_expression="${OPTARG}" ;;
     v) values="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$table_name" ]]; then
   errecho "ERROR: You must provide a table name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$keys" ]]; then
   errecho "ERROR: You must provide a keys json file path the -k parameter."
   usage
   return 1
 fi
 if [[ -z "$update_expression" ]]; then
```

```
    errecho "ERROR: You must provide an update expression with the -e parameter."
    usage
    return 1
  fi

  if [[ -z "$values" ]]; then
    errecho "ERROR: You must provide a values json file path the -v parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    table_name:   $table_name"
  iecho "    keys:   $keys"
  iecho "    update_expression:   $update_expression"
  iecho "    values:   $values"

  response=$(aws dynamodb update-item \
    --table-name "$table_name" \
    --key file://"$keys" \
    --update-expression "$update_expression" \
    --expression-attribute-values file://"$values")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports update-item operation failed.$response"
    return 1
  fi

  return 0

}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
```

```
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# See https://docs.aws.amazon.com/cli/latest/topic/return-codes.html#cli-aws-help-
return-codes.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
```

```
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie UpdateItem in der AWS CLI Befehlsreferenz.

Szenarien

Erweiterte Szenarien für globale sekundäre Indizes

Das folgende Codebeispiel zeigt, wie Sie mit erweiterten Konfigurationen für globale sekundäre Indizes arbeiten.

- Erstellen Sie eine Tabelle mit mehreren GSIs.
- Erstellen Sie eine Tabelle mit On-Demand-Kapazität und einem GSI.
- Fügen Sie Elemente in eine Tabelle mit mehreren GSIs ein.
- Fragen Sie mehrere GSIs mit unterschiedlichen Bedingungen ab.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit mehreren GSIs.

```
# Create a table with multiple GSIs
aws dynamodb create-table \
    --table-name MusicLibrary \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
        AttributeName=Year,AttributeType=N \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
```

```
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"AlbumIndex\",
                \"KeySchema\": [{\"AttributeName\":\"AlbumTitle\",\"KeyType\":\"HASH
\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            },
            {
                \"IndexName\": \"GenreYearIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Year\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",\"NonKeyAttributes
\":[\"Artist\",\"SongTitle\"]}
            }
        ]"
```

Erstellen Sie eine Tabelle mit On-Demand-Kapazität und einem GSI.

```
# Create a table with on-demand capacity and GSI
aws dynamodb create-table \
    --table-name MusicOnDemand \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=Genre,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "[
            {
                \"IndexName\": \"GenreIndex\",
                \"KeySchema\": [{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],
                \"Projection\": {\"ProjectionType\":\"ALL\"}
            }
        ]"
```

Fügen Sie Elemente in eine Tabelle mit mehreren GSIs ein.

```
# Add items to MusicLibrary table
aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "The Beatles"},
        "SongTitle": {"S": "Hey Jude"},
        "AlbumTitle": {"S": "Past Masters"},
        "Genre": {"S": "Rock"},
        "Year": {"N": "1968"}
    }'

aws dynamodb put-item \
    --table-name MusicLibrary \
    --item '{
        "Artist": {"S": "Miles Davis"},
        "SongTitle": {"S": "So What"},
        "AlbumTitle": {"S": "Kind of Blue"},
        "Genre": {"S": "Jazz"},
        "Year": {"N": "1959"}
    }'
```

Fragen Sie Elemente aus einer Tabelle mit mehreren GSIs ab.

```
# Query the AlbumIndex GSI
echo "Querying AlbumIndex GSI:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Kind of Blue"}}'

# Query the GenreYearIndex GSI with a range condition
echo "Querying GenreYearIndex GSI with range condition:"
aws dynamodb query \
    --table-name MusicLibrary \
    --index-name GenreYearIndex \
    --key-condition-expression "Genre = :genre AND #yr > :year" \
    --expression-attribute-names '{"#yr": "Year"}' \
    --expression-attribute-values '{":genre":{"S":"Rock"},":year":{"N":"1965"}}'
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - PutItem
  - Query

Erstellen und Verwalten von globalen MRSC-Tabellen

Das folgende Codebeispiel zeigt, wie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency (MRSC) erstellt und verwaltet werden.

- Erstellen Sie eine Tabelle mit Multi-Region starker Konsistenz.
- Überprüfen Sie die MRSC-Konfiguration und den Replikatstatus.
- Testen Sie die starke Konsistenz in allen Regionen mit sofortigen Lesevorgängen.
- Führen Sie bedingte Schreibvorgänge mit MRSC-Garantien durch.
- Bereinigen Sie die globalen MRSC-Tabellenressourcen.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Multi-Region starker Konsistenz.

```
# Step 1: Create a new table in us-east-2 (primary region for MRSC)
# Note: Table must be empty when enabling MRSC
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --region us-east-2

# Wait for table to become active
aws dynamodb wait table-exists --table-name MusicTable --region us-east-2

# Step 2: Add replica and witness with Multi-Region Strong Consistency
# MRSC requires exactly three replicas in supported regions
aws dynamodb update-table \
```

```
    --table-name MusicTable \
    --replica-updates '[{"Create": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Create": {"RegionName": "us-west-2"}}]' \
    --multi-region-consistency STRONG \
    --region us-east-2
```

Überprüfen Sie die MRSC-Konfiguration und den Replikatstatus.

```
# Verify the global table configuration and MRSC setting
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistency,R
{Region:RegionName,Status:ReplicaStatus}}'
```

Testen Sie die starke Konsistenz mit sofortigen Lesevorgängen in allen Regionen.

```
# Write an item to the primary region
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"},"Album":
 {"S":"The Beatles 1967-1970"},"Year": {"N":"1968"}}' \
    --region us-east-2

# Read the item from replica region to verify strong consistency (cannot read or
 write to witness)
# No wait time needed - MRSC provides immediate consistency
echo "Reading from us-east-1 (immediate consistency):"
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --consistent-read \
    --region us-east-1
```

Führen Sie bedingte Schreibvorgänge mit MRSC-Garantien durch.

```
# Perform a conditional update from a different region
# This demonstrates that conditions work consistently across all regions
```

```
aws dynamodb update-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"The Beatles"},"SongTitle": {"S":"Hey Jude"}}' \
    --update-expression "SET #rating = :rating" \
    --condition-expression "attribute_exists(Artist)" \
    --expression-attribute-names '{"#rating": "Rating"}' \
    --expression-attribute-values '{":rating": {"N":"5"}}' \
    --region us-east-1
```

Bereinigen Sie die globalen MRSC-Tabellenressourcen.

```
# Remove replica tables (must be done before deleting the primary table)
aws dynamodb update-table \
    --table-name MusicTable \
    --replica-updates '[{"Delete": {"RegionName": "us-east-1"}}]' \
    --global-table-witness-updates '[{"Delete": {"RegionName": "us-west-2"}}]' \
    --region us-east-2

# Wait for replicas to be deleted
echo "Waiting for replicas to be deleted..."
sleep 30

# Delete the primary table
aws dynamodb delete-table \
    --table-name MusicTable \
    --region us-east-2
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Erste Schritte mit NoSQL-Datenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Tabelle in DynamoDB
- Aktualisieren Sie Daten in einer DynamoDB-Tabelle
- Daten in einer DynamoDB-Tabelle abfragen
- Löschen Sie Ihre DynamoDB-Tabelle

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# DynamoDB Getting Started Tutorial Script
# This script demonstrates basic operations with Amazon DynamoDB:
# - Creating a table
# - Writing data to the table
# - Reading data from the table
# - Updating data in the table
# - Querying data in the table
# - Deleting the table (cleanup)

set -uo pipefail

# Set up logging with secure permissions
LOG_DIR="${XDG_STATE_HOME:-.}/dynamodb-tutorial-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/dynamodb-tutorial-$(date +%Y%m%d-%H%M%S).log"
chmod 700 "$LOG_DIR"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting DynamoDB Getting Started Tutorial at $(date)"
```

```
echo "Logging to $LOG_FILE"

# Validate AWS CLI is configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH"
    exit 1
fi

# Check AWS credentials are available
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd_name=$2

    if echo "$output" | grep -qi "error\|failed"; then
        echo "ERROR detected in $cmd_name command:" >&2
        echo "$output" >&2
        return 1
    fi
    return 0
}

# Function to wait for table to be in ACTIVE state
wait_for_table_active() {
    local table_name=$1
    local max_attempts=60
    local attempt=0
    local status=""

    echo "Waiting for table $table_name to become ACTIVE..."

    while [[ "$status" != "ACTIVE" && $attempt -lt $max_attempts ]]; do
        sleep 5
        status=$(aws dynamodb describe-table --table-name "$table_name" --query
 "Table.TableStatus" --output text 2>/dev/null || echo "UNKNOWN")
        echo "Current status: $status"
        ((attempt++))
    done
```

```
    if [[ "$status" != "ACTIVE" ]]; then
        echo "ERROR: Table $table_name did not become ACTIVE within timeout period"
 >&2
        return 1
    fi

    echo "Table $table_name is now ACTIVE"
    return 0
}

# Track created resources for cleanup
declare -a RESOURCES=()

# Cleanup function
cleanup() {
    local exit_code=$?

    if [[ $exit_code -ne 0 ]]; then
        echo "Script encountered an error (exit code: $exit_code)" >&2
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP"
    echo "============================================"
    echo "Resources to clean up:"
    for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
        echo "- $resource"
    done
    echo ""

    if [[ ${#RESOURCES[@]} -gt 0 ]]; then
        echo "Proceeding with cleanup of all created resources..."

        for resource in "${RESOURCES[@]+"${RESOURCES[@]}"}"; do
            if [[ "$resource" == Table:* ]]; then
                local table_name="${resource#Table:}"
                echo "Deleting table: $table_name"
                if aws dynamodb delete-table --table-name "$table_name" 2>/dev/null;
 then
                    echo "Waiting for table deletion to complete..."
                    aws dynamodb wait table-not-exists --table-name "$table_name"
 2>/dev/null || true
                else
```

```
                    echo "Warning: Failed to delete table $table_name" >&2
                fi
            fi
        done

        echo "Cleanup completed."
    fi

    return $exit_code
}

trap cleanup EXIT

# Validate table name
validate_table_name() {
    local name=$1
    if [[ ! $name =~ ^[a-zA-Z0-9._-]+$ ]] || [[ ${#name} -gt 255 ]]; then
        echo "ERROR: Invalid table name: $name" >&2
        return 1
    fi
    return 0
}

# Step 1: Create a table in DynamoDB
echo "Step 1: Creating Music table in DynamoDB..."

TABLE_NAME="Music"
validate_table_name "$TABLE_NAME"

CREATE_TABLE_OUTPUT=$(aws dynamodb create-table \
    --table-name "$TABLE_NAME" \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH
 AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --table-class STANDARD \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-dynamodb-gs 2>&1)
 || {
    echo "ERROR: Failed to create table" >&2
    exit 1
}
```

```
check_error "$CREATE_TABLE_OUTPUT" "create-table"
echo "$CREATE_TABLE_OUTPUT"

# Add table to resources list
RESOURCES+=("Table:$TABLE_NAME")

# Wait for table to be active
wait_for_table_active "$TABLE_NAME"

# Enable point-in-time recovery (best practice)
echo "Enabling point-in-time recovery for the $TABLE_NAME table..."

PITR_OUTPUT=$(aws dynamodb update-continuous-backups \
    --table-name "$TABLE_NAME" \
    --point-in-time-recovery-specification PointInTimeRecoveryEnabled=true 2>&1) ||
 {
    echo "ERROR: Failed to enable PITR" >&2
    exit 1
}

check_error "$PITR_OUTPUT" "update-continuous-backups"
echo "$PITR_OUTPUT"

# Step 2: Write data to the DynamoDB table
echo "Step 2: Writing data to the $TABLE_NAME table..."

# Use a temporary file for item data
ITEMS_TEMP=$(mktemp)
trap "rm -f '$ITEMS_TEMP'" EXIT

cat > "$ITEMS_TEMP" << 'EOF'
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Call Me Today"},
 "AlbumTitle": {"S": "Somewhat Famous"}, "Awards": {"N": "1"}}
{"Artist": {"S": "No One You Know"}, "SongTitle": {"S": "Howdy"}, "AlbumTitle":
 {"S": "Somewhat Famous"}, "Awards": {"N": "2"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}, "AlbumTitle": {"S":
 "Songs About Life"}, "Awards": {"N": "10"}}
{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "PartiQL Rocks"}, "AlbumTitle":
 {"S": "Another Album Title"}, "Awards": {"N": "8"}}
EOF

declare -i item_num=0
while IFS= read -r item_data; do
    ((item_num++))
```

```
    ITEM_OUTPUT=$(aws dynamodb put-item \
        --table-name "$TABLE_NAME" \
        --item "$item_data" 2>&1) || {
        echo "ERROR: Failed to put item $item_num" >&2
        exit 1
    }
    check_error "$ITEM_OUTPUT" "put-item (item $item_num)"
    echo "Item $item_num added successfully"
done < "$ITEMS_TEMP"

# Step 3: Read data from the DynamoDB table
echo "Step 3: Reading data from the $TABLE_NAME table..."

# Get a specific item
GET_ITEM_OUTPUT=$(aws dynamodb get-item --consistent-read \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' 2>&1) ||
 {
    echo "ERROR: Failed to get item" >&2
    exit 1
}

check_error "$GET_ITEM_OUTPUT" "get-item"
echo "Retrieved item:"
echo "$GET_ITEM_OUTPUT"

# Step 4: Update data in the DynamoDB table
echo "Step 4: Updating data in the $TABLE_NAME table..."

# Update an item
UPDATE_ITEM_OUTPUT=$(aws dynamodb update-item \
    --table-name "$TABLE_NAME" \
    --key '{"Artist": {"S": "Acme Band"}, "SongTitle": {"S": "Happy Day"}}' \
    --update-expression "SET AlbumTitle = :newval" \
    --expression-attribute-values '{":newval": {"S": "Updated Album Title"}}' \
    --return-values ALL_NEW 2>&1) || {
    echo "ERROR: Failed to update item" >&2
    exit 1
}

check_error "$UPDATE_ITEM_OUTPUT" "update-item"
echo "Updated item:"
echo "$UPDATE_ITEM_OUTPUT"
```

```
# Step 5: Query data in the DynamoDB table
echo "Step 5: Querying data in the $TABLE_NAME table..."

# Query items by Artist
QUERY_OUTPUT=$(aws dynamodb query \
    --table-name "$TABLE_NAME" \
    --key-condition-expression "Artist = :name" \
    --expression-attribute-values '{":name": {"S": "Acme Band"}}' 2>&1) || {
    echo "ERROR: Failed to query table" >&2
    exit 1
}

check_error "$QUERY_OUTPUT" "query"
echo "Query results:"
echo "$QUERY_OUTPUT"

echo "DynamoDB Getting Started Tutorial completed successfully at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - UpdateContinuousBackups
  - UpdateItem
  - Wait

Verwalten globaler sekundärer Indizes

Im folgenden Codebeispiel wird gezeigt, wie Sie den gesamten Lebenszyklus globaler sekundärer Indizes verwalten.

- Erstellen Sie eine Tabelle mit einem globalen sekundären Index.
- Fügen Sie einer neuen GSI zu einer vorhandenen Tabelle.
- Aktualisieren (erhöhen) Sie den GSI-Warmdurchsatz.

- Fragen Sie Daten mithilfe von GSIs ab.
- Löschen Sie einen GSI.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einem globalen sekundären Index.

```
# Create a table with a GSI
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
        AttributeName=AlbumTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST \
    --global-secondary-indexes \
        "IndexName=AlbumIndex,\
        KeySchema=[{AttributeName=AlbumTitle,KeyType=HASH}],\
        Projection={ProjectionType=ALL}"
```

Fügen Sie einer vorhandenen Tabelle einen neuen (On-Demand-)GSI hinzu.

```
# Add a new GSI to an existing table
aws dynamodb update-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Genre,AttributeType=S \
    --global-secondary-index-updates \
        "[{\"Create\":{\"IndexName\":\"GenreIndex\",\
        \"KeySchema\":[{\"AttributeName\":\"Genre\",\"KeyType\":\"HASH\"}],\
        \"Projection\":{\"ProjectionType\":\"ALL\"}}}]"
```

Aktualisieren (erhöhen) Sie den GSI-Warmdurchsatz.

```
# Increase the warm throughput of a GSI (default values are 12k reads, 4k writes)
aws dynamodb update-table \
```

```
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Update\":{\"IndexName\":\"AlbumIndex\",\
        \"WarmThroughput\":{\"ReadUnitsPerSecond\":15000,\"WriteUnitsPerSecond
\":6000}}}]"
```

Fragen Sie Daten mithilfe von GSIs ab.

```
# Query the AlbumIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name AlbumIndex \
    --key-condition-expression "AlbumTitle = :album" \
    --expression-attribute-values '{":album":{"S":"Let It Be"}}'

# Query the GenreIndex GSI
aws dynamodb query \
    --table-name MusicCollection \
    --index-name GenreIndex \
    --key-condition-expression "Genre = :genre" \
    --expression-attribute-values '{":genre":{"S":"Jazz"}}'
```

Löschen Sie einen GSI.

```
# Delete a GSI from a table
aws dynamodb update-table \
    --table-name MusicCollection \
    --global-secondary-index-updates \
        "[{\"Delete\":{\"IndexName\":\"GenreIndex\"}}]"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - Query
  - UpdateTable

Verwalten ressourcenbasierter Richtlinien

Im folgenden Codebeispiel wird gezeigt, wie Sie den gesamten Lebenszyklus ressourcenbasierter Richtlinien für DynamoDB-Tabellen verwalten.

- Erstellen einer Tabelle mit einer Ressourcenrichtlinie
- Rufen Sie eine Ressourcenrichtlinie ab.
- Aktualisieren Sie eine Ressourcenrichtlinie.
- Löschen Sie eine Ressourcenrichtlinie.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einer Ressourcenrichtlinie.

```
# Step 1: Create a DynamoDB table
aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
    --billing-mode PAY_PER_REQUEST

# Step 2: Create a resource-based policy document
cat > policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/DynamoDBReadOnly"
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
```

```
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
}
EOF

# Step 3: Attach the resource-based policy to the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
    --policy file://policy.json
```

Rufen Sie eine Ressourcenrichtlinie ab.

```
# Get the resource-based policy attached to a table
aws dynamodb get-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

Aktualisieren Sie eine Ressourcenrichtlinie.

```
# Step 1: Create an updated policy document
cat > updated-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:role/DynamoDBReadOnly",
          "arn:aws:iam::123456789012:role/DynamoDBAnalytics"
        ]
      },
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection"
    }
  ]
```

```
}
EOF

# Step 2: Update the resource-based policy on the table
aws dynamodb put-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection \
    --policy file://updated-policy.json
```

Löschen Sie eine Ressourcenrichtlinie.

```
# Delete the resource-based policy from a table
aws dynamodb delete-resource-policy \
    --resource-arn arn:aws:dynamodb:us-west-2:123456789012:table/MusicCollection
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - DeleteResourcePolicy
  - GetResourcePolicy
  - PutResourcePolicy

Richten Sie die Attribute-Based Zugriffskontrolle ein

Das folgende Codebeispiel zeigt, wie Attribute-Based Access Control (ABAC) für DynamoDB implementiert wird.

- Erstellen Sie eine IAM-Richtlinie für ABAC.
- Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.
- Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

AWS CLI mit Bash-Skript

Erstellen Sie eine IAM-Richtlinie für ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
```

```
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreatePolicy
  - CreateTable
  - ListTables

Arbeiten mit lokalen sekundären Indizes

Im folgenden Codebeispiel wird gezeigt, wie Sie Tabellen mit lokalen sekundären Indizes erstellen und abfragen.

- Erstellen einer Tabelle mit einem lokalen sekundären Index (LSI)
- Erstellen einer Tabelle mit mehreren LSIs mit unterschiedlichen Projektionstypen
- Abfragen von Daten mithilfe von LSIs

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit einem lokalen sekundären Index.

```
# Create a table with a Local Secondary Index
aws dynamodb create-table \
    --table-name CustomerOrders \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=OrderID,AttributeType=S \
        AttributeName=OrderDate,AttributeType=S \
    --key-schema \
```

```
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=OrderID,KeyType=RANGE \
    --local-secondary-indexes \
        "IndexName=OrderDateIndex,\
        KeySchema=[{AttributeName=CustomerID,KeyType=HASH},
{AttributeName=OrderDate,KeyType=RANGE}],\
        Projection={ProjectionType=ALL}" \
    --billing-mode PAY_PER_REQUEST
```

Erstellen Sie eine Tabelle mit mehreren LSIs.

```
# Create a table with multiple Local Secondary Indexes
aws dynamodb create-table \
    --table-name CustomerDetails \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
        AttributeName=Name,AttributeType=S \
        AttributeName=Email,AttributeType=S \
        AttributeName=RegistrationDate,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
        AttributeName=Name,KeyType=RANGE \
    --local-secondary-indexes \
        "[
            {
                \"IndexName\": \"EmailIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"Email\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"INCLUDE\",\"NonKeyAttributes
\":[\"Address\",\"Phone\"]}
            },
            {
                \"IndexName\": \"RegistrationIndex\",
                \"KeySchema\": [
                    {\"AttributeName\":\"CustomerID\",\"KeyType\":\"HASH\"},
                    {\"AttributeName\":\"RegistrationDate\",\"KeyType\":\"RANGE\"}
                ],
                \"Projection\": {\"ProjectionType\":\"KEYS_ONLY\"}
            }
        ]" \
```

```
    --billing-mode PAY_PER_REQUEST
```

Fragen Sie Daten mithilfe von LSIs ab.

```
# Query the OrderDateIndex LSI
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId AND OrderDate BETWEEN :date1
 AND :date2" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":date1": {"S": "2023-01-01"},
        ":date2": {"S": "2023-02-01"}
    }'

# Query with a filter expression
aws dynamodb query \
    --table-name CustomerOrders \
    --index-name OrderDateIndex \
    --key-condition-expression "CustomerID = :custId" \
    --filter-expression "Amount > :amount" \
    --expression-attribute-values '{
        ":custId": {"S": "C1"},
        ":amount": {"N": "150"}
    }'
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - Query

Arbeite mit Streams und Time-to-Live

Das folgende Codebeispiel zeigt, wie DynamoDB-Streams und Time-to-Live Funktionen verwaltet werden.

- Erstellen Sie eine Tabelle mit aktiviertem Streams.
- Beschreiben Sie Streams.
- Erstellen Sie eine Lambda-Funktion für die Verarbeitung von Streams.

- Aktivieren Sie TTL für eine Tabelle.
- Fügen Sie Elemente mit TTL-Attributen hinzu.
- Beschreiben Sie TTL-Einstellungen.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit aktiviertem Streams.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Beschreiben Sie Streams.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Erstellen Sie eine Lambda-Funktion für Streams.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Aktivieren Sie TTL für eine Tabelle.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
```

```
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Fügen Sie Elemente mit TTL-Attributen hinzu.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Beschreiben Sie TTL-Einstellungen.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
```

```
    --table-name TTLDemo
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

Arbeiten mit globalen Tabellen und letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC)

Das folgende Codebeispiel zeigt, wie globale DynamoDB-Tabellen mit letztendlicher multiregionaler Konsistenz (Multi-Region Eventual Consistency, MREC) verwaltet werden.

- Erstellen Sie eine Tabelle mit multiregionaler Replikation (MREC).
- Fügen Sie Elemente in eine Replikattabelle ein und rufen Sie sie ab.
- Entfernen Sie die Replikate nacheinander.
- Führen Sie eine Bereinigung durch Löschen der Tabelle aus.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit multiregionaler Replikation.

```
# Step 1: Create a new table (MusicTable) in US East (Ohio), with DynamoDB Streams
 enabled (NEW_AND_OLD_IMAGES)
aws dynamodb create-table \
    --table-name MusicTable \
    --attribute-definitions \
        AttributeName=Artist,AttributeType=S \
        AttributeName=SongTitle,AttributeType=S \
    --key-schema \
        AttributeName=Artist,KeyType=HASH \
        AttributeName=SongTitle,KeyType=RANGE \
```

```
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES \
    --region us-east-2

# Step 2: Create an identical MusicTable table in US East (N. Virginia)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2

# Step 3: Create a table in Europe (Ireland)
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Create": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2
```

Beschreiben Sie die multiregionale Tabelle.

```
# Step 4: View the list of replicas created using describe-table
aws dynamodb describe-table \
    --table-name MusicTable \
    --region us-east-2 \
    --query 'Table.
{TableName:TableName,TableStatus:TableStatus,MultiRegionConsistency:MultiRegionConsistency,R
{Region:RegionName,Status:ReplicaStatus}}'
```

Fügen Sie Elemente in eine Replikattabelle ein.

```
# Step 5: To verify that replication is working, add a new item to the Music table
 in US East (Ohio)
aws dynamodb put-item \
    --table-name MusicTable \
    --item '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-2
```

Rufen Sie Elemente aus Replikattabellen ab.

```
# Step 6: Wait for a few seconds, and then check to see whether the item has been
# successfully replicated to US East (N. Virginia) and Europe (Ireland)
aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region us-east-1

aws dynamodb get-item \
    --table-name MusicTable \
    --key '{"Artist": {"S":"item_1"},"SongTitle": {"S":"Song Value 1"}}' \
    --region eu-west-1
```

Entfernen Sie Replikate.

```
# Step 7: Delete the replica table in Europe (Ireland) Region
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "eu-west-1"
      }
    }
  ]
}' \
--region us-east-2

# Delete the replica table in US East (N. Virginia) Region
```

```
aws dynamodb update-table --table-name MusicTable --cli-input-json \
'{
  "ReplicaUpdates":
  [
    {
      "Delete": {
        "RegionName": "us-east-1"
      }
    }
  ]
}' \
--region us-east-2
```

Führen Sie eine Bereinigung durch Löschen der Tabelle aus.

```
# Clean up: Delete the primary table
aws dynamodb delete-table --table-name MusicTable --region us-east-2

echo "Global table demonstration complete."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateTable

Arbeiten mit Ressourcen-Tagging

Im folgenden Codebeispiel wird gezeigt, wie Sie Tags für DynamoDB-Ressourcen verwalten.

- Erstellen Sie eine Tabelle mit Tags.
- Listen Sie Tags für eine Ressource auf.
- Fügen Sie Tags zu einer Ressource hinzu.
- Entfernen Sie Tags aus einer Ressource.

- Filtern Sie Tabellen nach Tags.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Tags.

```
# Create a table with tags
aws dynamodb create-table \
    --table-name TaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Production \
        Key=Project,Value=Analytics \
        Key=Owner,Value=DataTeam
```

Listen Sie Tags für eine Ressource auf.

```
# Get the table ARN
TABLE_ARN=$(aws dynamodb describe-table \
    --table-name TaggedTable \
    --query "Table.TableArn" \
    --output text)

# List tags for the table
aws dynamodb list-tags-of-resource \
    --resource-arn $TABLE_ARN
```

Fügen Sie Tags zu einer Ressource hinzu.

```
# Add tags to an existing table
aws dynamodb tag-resource \
    --resource-arn $TABLE_ARN \
    --tags \
        Key=CostCenter,Value=12345 \
        Key=BackupSchedule,Value=Daily
```

Entfernen Sie Tags aus einer Ressource.

```
# Remove tags from a table
aws dynamodb untag-resource \
    --resource-arn $TABLE_ARN \
    --tag-keys Owner BackupSchedule
```

Filtern Sie Tabellen nach Tags.

```
# Create another table with different tags
aws dynamodb create-table \
    --table-name AnotherTaggedTable \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Environment,Value=Development \
        Key=Project,Value=Testing

# Wait for table to become active
aws dynamodb wait table-exists --table-name AnotherTaggedTable

# List all tables
echo "All tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nFiltering tables by Environment=Production tag:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# Find tables with specific tag
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
```

```
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTable
  - ListTagsOfResource
  - TagResource
  - UntagResource

Arbeiten mit der Tabellenverschlüsselung

Im folgenden Codebeispiel wird gezeigt, wie Sie Verschlüsselungsoptionen für DynamoDB-Tabellen verwalten.

- Erstellen Sie eine Tabelle mit Standardverschlüsselung.
- Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.
- Aktualisieren Sie die Einstellungen für die Tabellenverschlüsselung.
- Beschreiben Sie die Tabellenverschlüsselung.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit Standardverschlüsselung.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Aktualisieren Sie die Tabellenverschlüsselung.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Beschreiben Sie die Tabellenverschlüsselung.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateKey
  - CreateTable
  - DescribeTable

- UpdateTable

Amazon EC2-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
###############################################################################
# function get_started_with_ec2_instances
#
# Runs an interactive scenario that shows how to get started using EC2 instances.
#
#     "EC2 access" permissions are needed to run this code.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function get_started_with_ec2_instances() {
  # Requires version 4 for mapfile.
  local required_version=4.0

  # Get the current Bash version
  # Check if BASH_VERSION is set
  local current_version
  if [[ -n "$BASH_VERSION" ]]; then
    # Convert BASH_VERSION to a number for comparison
    current_version=$BASH_VERSION
  else
    # Get the current Bash version using the bash command
    current_version=$(bash --version | head -n 1 | awk '{ print $4 }')
  fi

  # Convert version strings to numbers for comparison
  local required_version_num current_version_num
  required_version_num=$(echo "$required_version" | awk -F. '{ print ($1 * 10000) +
 ($2 * 100) + $3 }')
  current_version_num=$(echo "$current_version" | awk -F. '{ print ($1 * 10000) +
 ($2 * 100) + $3 }')
```

```
  # Compare versions
  if ((current_version_num < required_version_num)); then
    echo "Error: This script requires Bash version $required_version or higher."
    echo "Your current Bash version is number is $current_version."
    exit 1
  fi

  {
    if [ "$EC2_OPERATIONS_SOURCED" != "True" ]; then

      source ./ec2_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
instances demo."
  echo_repeat "*" 88
  echo

  echo "Let's create an RSA key pair that you can be use to securely connect to "
  echo "your EC2 instance."

  echo -n "Enter a unique name for your key: "
  get_input
  local key_name
  key_name=$get_input_result

  local temp_dir
  temp_dir=$(mktemp -d)
  local key_file_name="$temp_dir/${key_name}.pem"

  if ec2_create_keypair -n "${key_name}" -f "${key_file_name}"; then
    echo "Created a key pair $key_name and saved the private key to $key_file_name"
    echo
  else
    errecho "The key pair failed to create. This demo will exit."
    return 1
  fi

  chmod 400 "${key_file_name}"

  if yes_no_input "Do you want to list some of your key pairs? (y/n) "; then
    local keys_and_fingerprints
```

```
    keys_and_fingerprints="$(ec2_describe_key_pairs)" && {
      local image_name_and_id
      while IFS=$'\n' read -r image_name_and_id; do
        local entries
        IFS=$'\t' read -ra entries <<<"$image_name_and_id"
        echo "Found rsa key ${entries[0]} with fingerprint:"
        echo "     ${entries[1]}"
      done <<<"$keys_and_fingerprints"

    }
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create a security group to manage access to your instance."
  echo -n "Enter a unique name for your security group: "
  get_input
  local security_group_name
  security_group_name=$get_input_result
  local security_group_id
  security_group_id=$(ec2_create_security_group -n "$security_group_name" \
    -d "Security group for EC2 instance") || {
    errecho "The security failed to create. This demo will exit."
    clean_up "$key_name" "$key_file_name"
    return 1
  }

  echo "Security group created with ID $security_group_id"
  echo

  local public_ip
  public_ip=$(curl -s http://checkip.amazonaws.com)

  echo "Let's add a rule to allow SSH only from your current IP address."
  echo "Your public IP address is $public_ip"
  echo -n "press return to add this rule to your security group."
  get_input

  if ! ec2_authorize_security_group_ingress -g "$security_group_id" -i "$public_ip"
-p tcp -f 22 -t 22; then
    errecho "The security group rules failed to update. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
```

```
  fi

  echo "Security group rules updated"

  local security_group_description
  security_group_description="$(ec2_describe_security_groups -g
 "${security_group_id}")" || {
    errecho "Failed to describe security groups. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  mapfile -t parameters <<<"$security_group_description"
  IFS=$'\t' read -ra entries <<<"${parameters[0]}"
  echo "Security group: ${entries[0]}"
  echo "    ID: ${entries[1]}"
  echo "    VPC: ${entries[2]}"
  echo "Inbound permissions:"
  IFS=$'\t' read -ra entries <<<"${parameters[1]}"
  echo "    IpProtocol: ${entries[0]}"
  echo "    FromPort: ${entries[1]}"
  echo "    ToPort: ${entries[2]}"
  echo "    CidrIp: ${parameters[2]}"

  local parameters
  parameters="$(ssm_get_parameters_by_path -p "/aws/service/ami-amazon-linux-
latest")" || {
    errecho "Failed to get parameters. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  local image_ids=""
  mapfile -t parameters <<<"$parameters"
  for image_name_and_id in "${parameters[@]}"; do
    IFS=$'\t' read -ra values <<<"$image_name_and_id"
    if [[ "${values[0]}" == *"amzn2"* ]]; then
      image_ids+="${values[1]} "
    fi
  done

  local images
  images="$(ec2_describe_images -i "$image_ids")" || {
```

```
    errecho "Failed to describe images. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1

  }

  new_line_and_tab_to_list "$images"
  local images=("${list_result[@]}")

  # Get the size of the array
  local images_count=${#images[@]}

  if ((images_count == 0)); then
    errecho "No images found. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  fi

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's create an instance from an Amazon Linux 2 AMI. Here are some options:"
  for ((i = 0; i < images_count; i += 3)); do
    echo "$(((i / 3) + 1)) - ${images[$i]}"
  done

  integer_input "Please enter the number of the AMI you want to use: " 1
"$((images_count / 3))"
  local choice=$get_input_result
  choice=$(((choice - 1) * 3))

  echo "Great choice."
  echo

  local architecture=${images[$((choice + 1))]}
  local image_id=${images[$((choice + 2))]}
  echo "Here are some instance types that support the ${architecture} architecture
of the image:"
  response="$(ec2_describe_instance_types -a "${architecture}" -t
"*.micro,*.small")" || {
    errecho "Failed to describe instance types. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }
```

```
  local instance_types
  mapfile -t instance_types <<<"$response"

  # Get the size of the array
  local instance_types_count=${#instance_types[@]}

  echo "Here are some options:"
  for ((i = 0; i < instance_types_count; i++)); do
    echo "$((i + 1)) - ${instance_types[$i]}"
  done

  integer_input "Which one do you want to use? " 1 "${#instance_types[@]}
"
  choice=$get_input_result
  local instance_type=${instance_types[$((choice - 1))]}
  echo "Another great choice."
  echo

  echo "Creating your instance and waiting for it to start..."
  local instance_id
  instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k "$key_name"
 -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
  }

  ec2_wait_for_instance_running -i "$instance_id"
  echo "Your instance is ready:"
  echo

  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  echo
  print_instance_details "${instance_details}"

  local public_ip
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  echo
  echo "You can use SSH to connect to your instance"
  echo "If the connection attempt times out, you might have to manually update the
 SSH ingress rule"
```

```
echo "for your IP address in the AWS Management Console."
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's stop and start your instance to see what changes."
echo "Stopping your instance and waiting until it's stopped..."
ec2_stop_instances -i "$instance_id"
ec2_wait_for_instance_stopped -i "$instance_id"

echo "Your instance is stopped. Restarting..."

ec2_start_instances -i "$instance_id"
ec2_wait_for_instance_running -i "$instance_id"

echo "Your instance is running again."
local instance_details
instance_details="$(ec2_describe_instances -i "${instance_id}")"

print_instance_details "${instance_details}"

public_ip=$(echo "${instance_details}" | awk '{print $6}')

echo "Every time your instance is restarted, its public IP address changes"
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "You can allocate an Elastic IP address and associate it with your instance"
echo "to keep a consistent IP address even when your instance restarts."

local result
result=$(ec2_allocate_address -d vpc) || {
  errecho "Failed to allocate an address. This demo will exit."
  clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
  return 1
```

```
  }

  local elastic_ip allocation_id
  elastic_ip=$(echo "$result" | awk '{print $1}')
  allocation_id=$(echo "$result" | awk '{print $2}')

  echo "Allocated static Elastic IP address: $elastic_ip"

  local association_id
  association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id") || {
    errecho "Failed to associate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
"$allocation_id"
    return 1
  }

  echo "Associated your Elastic IP with your instance."
  echo "You can now use SSH to connect to your instance by using the Elastic IP."
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  echo "Let's stop and start your instance to see what changes."
  echo "Stopping your instance and waiting until it's stopped..."
  ec2_stop_instances -i "$instance_id"
  ec2_wait_for_instance_stopped -i "$instance_id"

  echo "Your instance is stopped. Restarting..."

  ec2_start_instances -i "$instance_id"
  ec2_wait_for_instance_running -i "$instance_id"

  echo "Your instance is running again."
  local instance_details
  instance_details="$(ec2_describe_instances -i "${instance_id}")"

  print_instance_details "${instance_details}"

  echo "Because you have associated an Elastic IP with your instance, you can"
  echo "connect by using a consistent IP address after the instance restarts."
```

```
  connect_to_instance "$key_file_name" "$elastic_ip"

  echo -n "Press Enter when you're ready to continue the demo: "
  get_input

  echo_repeat "*" 88
  echo_repeat "*" 88

  if yes_no_input "Do you want to delete the resources created in this demo: (y/n)
 "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
      "$allocation_id" "$association_id"
  else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
  fi
}

###############################################################################
# function clean_up
#
# This function cleans up the created resources.
#     $1 - The name of the ec2 key pair to delete.
#     $2 - The name of the key file to delete.
#     $3 - The ID of the security group to delete.
#     $4 - The ID of the instance to terminate.
#     $5 - The ID of the elastic IP address to release.
#     $6 - The ID of the elastic IP address to disassociate.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function clean_up() {
  local result=0
  local key_pair_name=$1
  local key_file_name=$2
  local security_group_id=$3
  local instance_id=$4
  local allocation_id=$5
```

```
  local association_id=$6

  if [ -n "$association_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_disassociate_address -a "$association_id"); then
      echo "Disassociated elastic IP address with ID $association_id"
    else
      errecho "The elastic IP address disassociation failed."
      result=1
    fi
  fi

  if [ -n "$allocation_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_release_address -a "$allocation_id"); then
      echo "Released elastic IP address with ID $allocation_id"
    else
      errecho "The elastic IP address release failed."
      result=1
    fi
  fi

  if [ -n "$instance_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_terminate_instances -i "$instance_id"); then
      echo "Started terminating instance with ID $instance_id"

      ec2_wait_for_instance_terminated -i "$instance_id"
    else
      errecho "The instance terminate failed."
      result=1
    fi
  fi

  if [ -n "$security_group_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_security_group -i "$security_group_id"); then
      echo "Deleted security group with ID $security_group_id"
    else
      errecho "The security group delete failed."
      result=1
    fi
  fi
```

```
  if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
      echo "Deleted key pair named $key_pair_name"
    else
      errecho "The key pair delete failed."
      result=1
    fi
  fi

  if [ -n "$key_file_name" ]; then
    rm -f "$key_file_name"
  fi

  return $result
}

###############################################################################
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
 Parameter Store
# by specifying a parameter path.
#
# Parameters:
#       -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ssm_get_parameters_by_path() {
  local parameter_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ssm_get_parameters_by_path"
    echo "Retrieves one or more parameters from the AWS Systems Manager Parameter
 Store by specifying a parameter path."
    echo "  -p parameter_path - The path of the parameter(s) to retrieve."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "p:h" option; do
    case "${option}" in
      p) parameter_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$parameter_path" ]]; then
    errecho "ERROR: You must provide a parameter path with the -p parameter."
    usage
    return 1
  fi

  response=$(aws ssm get-parameters-by-path \
    --path "$parameter_path" \
    --query "Parameters[*].[Name, Value]" \
    --output text) || {
    aws_cli_error_log $?
    errecho "ERROR: AWS reports get-parameters-by-path operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function print_instance_details
#
# This function prints the details of an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
#
# Parameters:
```

```
#       instance_details - The instance details in the format "InstanceId ImageId
 InstanceType KeyName VpcId PublicIpAddress State.Name".
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function print_instance_details() {
  local instance_details="$1"

  if [[ -z "${instance_details}" ]]; then
    echo "Error: Missing required instance details argument."
    return 1
  fi

  local instance_id image_id instance_type key_name vpc_id public_ip state
  instance_id=$(echo "${instance_details}" | awk '{print $1}')
  image_id=$(echo "${instance_details}" | awk '{print $2}')
  instance_type=$(echo "${instance_details}" | awk '{print $3}')
  key_name=$(echo "${instance_details}" | awk '{print $4}')
  vpc_id=$(echo "${instance_details}" | awk '{print $5}')
  public_ip=$(echo "${instance_details}" | awk '{print $6}')
  state=$(echo "${instance_details}" | awk '{print $7}')

  echo "    ID: ${instance_id}"
  echo "    Image ID: ${image_id}"
  echo "    Instance type: ${instance_type}"
  echo "    Key name: ${key_name}"
  echo "    VPC ID: ${vpc_id}"
  echo "    Public IP: ${public_ip}"
  echo "    State: ${state}"

  return 0
}

##############################################################################
# function connect_to_instance
#
# This function displays the public IP address of an Amazon Elastic Compute Cloud
 (Amazon EC2) instance and prompts the user to connect to the instance via SSH.
#
# Parameters:
#       $1 - The name of the key file used to connect to the instance.
#       $2 - The public IP address of the instance.
```

```
#
# Returns:
#       None
###############################################################################
function connect_to_instance() {
  local key_file_name="$1"
  local public_ip="$2"

  # Validate the input parameters
  if [[ -z "$key_file_name" ]]; then
    echo "ERROR: You must provide a key file name as the first argument." >&2
    return 1
  fi

  if [[ -z "$public_ip" ]]; then
    echo "ERROR: You must provide a public IP address as the second argument." >&2
    return 1
  fi

  # Display the public IP address and connection command
  echo "To connect, run the following command:"
  echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

  # Prompt the user to connect to the instance
  if yes_no_input "Do you want to connect now? (y/n) "; then
    echo "After you have connected, you can return to this example by typing 'exit'"
    ssh -i "${key_file_name}" ec2-user@"${public_ip}"
  fi
}

###############################################################################
# function get_input
#
# This function gets user input from the command line.
#
# Outputs:
#   User input to stdout.
#
# Returns:
#       0
###############################################################################
function get_input() {

  if [ -z "${mock_input+x}" ]; then
```

```
    read -r get_input_result
  else

    if [ "$mock_input_array_index" -lt ${#mock_input_array[@]} ]; then
      get_input_result="${mock_input_array[$mock_input_array_index]}"
      # bashsupport disable=BP2001
      # shellcheck disable=SC2206
      ((mock_input_array_index++))
      echo -n "$get_input_result"
    else
      echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
      return 1
    fi
  fi

  return 0
}

##############################################################################
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#       $1 - The prompt.
#
# Returns:
#       0 - If yes.
#       1 - If no.
##############################################################################
function yes_no_input() {
  if [ -z "$1" ]; then
    echo "Internal error yes_no_input"
    return 1
  fi

  local index=0
  local response="N"
  while [[ $index -lt 10 ]]; do
    index=$((index + 1))
    echo -n "$1"
    if ! get_input; then
      return 1
    fi
```

```
    response=$(echo "$get_input_result" | tr '[:upper:]' '[:lower:]')
    if [ "$response" = "y" ] || [ "$response" = "n" ]; then
      break
    else
      echo -e "\nPlease enter or 'y' or 'n'."
    fi
  done

  echo

  if [ "$response" = "y" ]; then
    return 0
  else
    return 1
  fi
}

###############################################################################
# function integer_input
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#       $1 - The prompt message to display to the user.
#       $2 - The minimum value of the accepted range.
#       $3 - The maximum value of the accepted range.
#
# Returns:
#       The valid integer input from the user.
#       If the input is invalid or out of range, the function will continue
#       prompting the user until a valid input is provided.
###############################################################################
function integer_input() {
  local prompt="$1"
  local min_value="$2"
  local max_value="$3"
  local input=""

  while true; do
    # Display the prompt message and wait for user input
    echo -n "$prompt"

    if ! get_input; then
```

```
      return 1
    fi

    input="$get_input_result"

    # Check if the input is a valid integer
    if [[ "$input" =~ ^-?[0-9]+$ ]]; then
      # Check if the input is within the specified range
      if ((input >= min_value && input <= max_value)); then
        return 0
      else
        echo "Error: Input, $input, must be between $min_value and $max_value."
      fi
    else
      echo "Error: Invalid input- $input. Please enter an integer."
    fi
  done
}
##############################################################################
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#       $1 - The input string containing newlines and tabs.
#
# Returns:
#       The resulting list (array) is stored in the global variable
#       'list_result'.
##############################################################################
function new_line_and_tab_to_list() {
  local input=$1
  export list_result

  list_result=()
  mapfile -t lines <<<"$input"
  local line
  for line in "${lines[@]}"; do
    IFS=$'\t' read -ra parameters <<<"$line"
    list_result+=("${parameters[@]}")
  done
}
```

```
##############################################################################
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#       $1 - The string.
#       $2 - Number of times to print the string.
#
# Outputs:
#   String 'n' times to stdout.
#
# Returns:
#       0
##############################################################################
function echo_repeat() {
  local end=$2
  for ((i = 0; i < end; i++)); do
    echo -n "$1"
  done
  echo
}
```

Die in diesem Szenario verwendeten „DynamoDB“-Funktionen.

```
##############################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_create_keypair() {
  local key_pair_name file_path response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-bit
RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi

  response=$(aws ec2 create-key-pair \
```

```
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}

###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
#
# Returns:
#       The ID of the created security group, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
```

```
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n parameter."
    return 1
  fi

  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
```

```
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) security
 groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:h" option; do
    case "${option}" in
```

```
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id" --
query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
#
# Parameters:
```

```
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
 parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation failed.
$response"
```

```
    return 1
  }

  return 0
}

###############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g., x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
###############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--
type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance types
 (e.g., t2.micro)"
    echo "  -h, --help                       Show this help message"
  }

  while [[ $# -gt 0 ]]; do
    case "$1" in
      -a | --architecture)
        architecture="$2"
        shift 2
        ;;
      -t | --type)
        instance_types="$2"
        shift 2
        ;;
      -h | --help)
        usage
        return 0
        ;;
      *)
        echo "Unknown argument: $1"
        return 1
        ;;
    esac
  done

  if [[ -z "$architecture" ]]; then
    errecho "Error: Architecture not specified."
    usage
    return 1
  fi

  if [[ -z "$instance_types" ]]; then
    errecho "Error: Instance type not specified."
    usage
    return 1
  fi
```

```
  local tmp_json_file="temp_ec2.json"
  echo -n '[
    {
      "Name": "processor-info.supported-architecture",
      "Values": [' >"$tmp_json_file"

  local items
  IFS=',' read -ra items <<<"$architecture"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
```

```
    return 1
  fi

  echo "$response"
  return 0
}

###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:t:k:s:c:h" option; do
```

```
    case "${option}" in
      i) image_id="${OPTARG}" ;;
      t) instance_type="${OPTARG}" ;;
      k) key_pair_name="${OPTARG}" ;;
      s) security_group_id="${OPTARG}" ;;
      c) count="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$image_id" ]]; then
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
parameter."
    usage
    return 1
  fi

  if [[ -z "$instance_type" ]]; then
    errecho "ERROR: You must provide an instance type with the -t parameter."
    usage
    return 1
  fi

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -k parameter."
    usage
    return 1
  fi

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
```

```
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
```

```
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}

###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}

###############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }
```

```
  return 0
}

##############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_allocate_address() {
  local domain response

  # Function to display usage information
  function usage() {
    echo "function ec2_allocate_address"
    echo "Allocates an Elastic IP address for use with Amazon Elastic Compute Cloud
 (Amazon EC2) instances in a specific AWS Region."
    echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard')."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "d:h" option; do
    case "${option}" in
      d) domain="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc' or
 'standard')."
    return 1
  fi

  if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
  fi

  # Allocate the Elastic IP address
  response=$(aws ec2 allocate-address \
    --domain "$domain" \
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to associate.
```

```
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
```

```
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}

###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
```

```
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the association
 of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_release_address
```

```
#
# This function releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
```

```
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}

###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
```

```
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --output
 text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
```

```
  }

  return 0
}

###############################################################################
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Szenario verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
```

```
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress

- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_allocate_address
#
# This function allocates an Elastic IP address for use with Amazon Elastic Compute
 Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#       -d domain - The domain for the Elastic IP address (either 'vpc' or
 'standard').
#
# Returns:
#       The allocated Elastic IP address, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
################################################################################
function ec2_allocate_address() {
  local domain response
```

```
# Function to display usage information
function usage() {
  echo "function ec2_allocate_address"
  echo "Allocates an Elastic IP address for use with Amazon Elastic Compute Cloud
(Amazon EC2) instances in a specific AWS Region."
  echo "  -d domain - The domain for the Elastic IP address (either 'vpc' or
'standard')."
  echo ""
}

# Parse the command-line arguments
while getopts "d:h" option; do
  case "${option}" in
    d) domain="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$domain" ]]; then
  errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc' or
'standard')."
  return 1
fi

if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
  errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
  return 1
fi

# Allocate the Elastic IP address
response=$(aws ec2 allocate-address \
  --domain "$domain" \
  --query "[PublicIp,AllocationId]" \
```

```
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
```

```
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie AllocateAddress in der AWS CLI Befehlsreferenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to associate.
#       -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
#
##############################################################################
function ec2_associate_address() {
  local allocation_id instance_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 associate."
    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic IP
 address with."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:i:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      i) instance_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
```

```
    return 1
  fi

  # Associate the Elastic IP address
  response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
```

```
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie AssociateAddress in der AWS CLI Befehlsreferenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
 (Amazon EC2) security group.
```

```
#
# Parameters:
#       -g security_group_id - The ID of the security group.
#       -i ip_address - The IP address or CIDR block to authorize.
#       -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#       -f from_port - The start of the port range to authorize.
#       -t to_port - The end of the port range to authorize.
#
# And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_authorize_security_group_ingress() {
  local security_group_id ip_address protocol from_port to_port response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
 EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      i) ip_address="${OPTARG}" ;;
      p) protocol="${OPTARG}" ;;
      f) from_port="${OPTARG}" ;;
      t) to_port="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
  fi

  if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
    usage
    return 1
  fi

  if [[ -z "$protocol" ]]; then
    errecho "ERROR: You must provide a protocol with the -p parameter."
    usage
    return 1
  fi

  if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
  fi

  if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
  fi

  response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
```

```
    errecho "ERROR: AWS reports authorize-security-group-ingress operation failed.
$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
```

```
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS CLI Befehlsreferenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#       -f file_path - File to store the key pair.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function ec2_create_keypair() {
```

```
  local key_pair_name file_path response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_create_keypair"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-bit
RSA key pair"
    echo " and writes it to a file."
    echo "  -n key_pair_name - A key pair name."
    echo "  -f file_path - File to store the key pair."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:f:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      f) file_path="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
  fi
```

```
  response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  }

  if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
  fi

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
```

```
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie CreateKeyPair in der AWS CLI Befehlsreferenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -n security_group_name - The name of the security group.
#       -d security_group_description - The description of the security group.
```

```
#
# Returns:
#       The ID of the created security group, or an error message if the operation
 fails.
# And:
#       0 - If successful.
#       1 - If it fails.
#
##############################################################################
function ec2_create_security_group() {
  local security_group_name security_group_description response

  # Function to display usage information
  function usage() {
    echo "function ec2_create_security_group"
    echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -n security_group_name - The name of the security group."
    echo "  -d security_group_description - The description of the security group."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "n:d:h" option; do
    case "${option}" in
      n) security_group_name="${OPTARG}" ;;
      d) security_group_description="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$security_group_name" ]]; then
    errecho "ERROR: You must provide a security group name with the -n parameter."
    return 1
  fi
```

```
  if [[ -z "$security_group_description" ]]; then
    errecho "ERROR: You must provide a security group description with the -d
 parameter."
    return 1
  fi

  # Create the security group
  response=$(aws ec2 create-security-group \
    --group-name "$security_group_name" \
    --description "$security_group_description" \
    --query "GroupId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-security-group operation failed."
    errecho "$response"
    return 1
  }

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
```

```
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie CreateSecurityGroup in der AWS CLI Befehlsreferenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
```

```
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#       -n key_pair_name - A key pair name.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_delete_keypair() {
  local key_pair_name response

  local option OPTARG # Required to use getopts command in a function.
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) key_pair_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
```

```
  fi

  response=$(aws ec2 delete-key-pair \
    --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
```

```
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DeleteKeyPair in der AWS CLI Befehlsreferenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Parameters:
#       -i security_group_id - The ID of the security group to delete.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function ec2_delete_security_group() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws ec2 delete-security-group --group-id "$security_group_id" --output
 text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi
```

```
  return 0
}
```

- API-Details finden Sie DeleteSecurityGroup in der AWS CLI Befehlsreferenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 images.
#
# Parameters:
#       -i image_ids - A space-separated  list of image IDs (optional).
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_images() {
  local image_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
```

```
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) image_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
  fi

  response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeImages in der AWS CLI Befehlsreferenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g., x86_64)
# -t, --type INSTANCE_TYPE         Comma-separated list of instance types (e.g.,
 t2.micro)
# -h, --help                       Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
###############################################################################
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--
type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture
 (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE         Comma-separated list of instance types
 (e.g., t2.micro)"
```

```
  echo "  -h, --help                      Show this help message"
}

while [[ $# -gt 0 ]]; do
  case "$1" in
    -a | --architecture)
      architecture="$2"
      shift 2
      ;;
    -t | --type)
      instance_types="$2"
      shift 2
      ;;
    -h | --help)
      usage
      return 0
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
```

```
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done
  echo -n ']},
    {
    "Name": "instance-type",
      "Values": [' >>"$tmp_json_file"
  IFS=',' read -ra items <<<"$instance_types"
  local array_size
  array_size=${#items[@]}
  for i in $(seq 0 $((array_size - 1))); do
    echo -n '"'"${items[$i]}"'"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
      echo -n ',' >>"$tmp_json_file"
    fi
  done

  echo -n ']}]' >>"$tmp_json_file"

  local response
  response=$(aws ec2 describe-instance-types --filters file://"$tmp_json_file" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

  local error_code=$?

  rm "$tmp_json_file"

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
  fi

  echo "$response"
  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeInstanceTypes in der AWS CLI Befehlsreferenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID of the instance to describe (optional).
#       -q query - The query to filter the response (optional).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_instances() {
  local instance_id query response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_instances"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_id - The ID of the instance to describe (optional)."
    echo "  -q query - The query to filter the response (optional)."
```

```
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:q:h" option; do
    case "${option}" in
      i) instance_id="${OPTARG}" ;;
      q) query="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local aws_cli_args=()

  if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
  fi

  local query_arg=""
  if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
  else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
```

```
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
```

```
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeInstances in der AWS CLI Befehlsreferenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs.
#
# Parameters:
#       -h - Display help.
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_key_pairs() {
  local option OPTARG # Required to use getopts command in a function.
```

```
  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_key_pairs"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
 pairs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
##############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeKeyPairs in der AWS CLI Befehlsreferenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
 security groups.
#
# Parameters:
#       -g security_group_id - The ID of the security group to describe (optional).
#
# And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_describe_security_groups() {
  local security_group_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_describe_security_groups"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) security
 groups."
    echo "  -g security_group_id - The ID of the security group to describe
 (optional)."
    echo ""
  }

  # Retrieve the calling parameters.
```

```
  while getopts "g:h" option; do
    case "${option}" in
      g) security_group_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

  if [[ -n "$security_group_id" ]]; then
    response=$(aws ec2 describe-security-groups --group-ids "$security_group_id" --
query "${query}" --output text)
  else
    response=$(aws ec2 describe-security-groups --query "${query}" --output text)
  fi

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports describe-security-groups operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
```

```
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS CLI Befehlsreferenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic Compute
 Cloud (Amazon EC2) instance.
#
# Parameters:
#       -a association_id - The association ID that represents the association of
 the Elastic IP address with an instance.
#
# And:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the association
 of the Elastic IP address with an instance."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
```

```
      a) association_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie DisassociateAddress in der AWS CLI Befehlsreferenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance.
#
# Parameters:
#       -a allocation_id - The allocation ID of the Elastic IP address to release.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#
###############################################################################
function ec2_release_address() {
  local allocation_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_release_address"
    echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
 (Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
 release."
    echo ""
  }

  # Parse the command-line arguments
  while getopts "a:h" option; do
    case "${option}" in
      a) allocation_id="${OPTARG}" ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Validate the input parameters
  if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
  fi

  response=$(aws ec2 release-address \
    --allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
```

```
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie ReleaseAddress in der AWS CLI Befehlsreferenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#       -t instance_type - The instance type to use (e.g., t2.micro).
#       -k key_pair_name - The name of the key pair to use.
#       -s security_group_id - The ID of the security group to use.
#       -c count - The number of instances to launch (default: 1).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function ec2_run_instances() {
  local image_id instance_type key_pair_name security_group_id count response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_run_instances"
    echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
    echo "  -k key_pair_name - The name of the key pair to use."
    echo "  -s security_group_id - The ID of the security group to use."
    echo "  -c count - The number of instances to launch (default: 1)."
    echo "  -h - Display help."
    echo ""
  }
```

```
# Retrieve the calling parameters.
while getopts "i:t:k:s:c:h" option; do
  case "${option}" in
    i) image_id="${OPTARG}" ;;
    t) instance_type="${OPTARG}" ;;
    k) key_pair_name="${OPTARG}" ;;
    s) security_group_id="${OPTARG}" ;;
    c) count="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$image_id" ]]; then
  errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
parameter."
  usage
  return 1
fi

if [[ -z "$instance_type" ]]; then
  errecho "ERROR: You must provide an instance type with the -t parameter."
  usage
  return 1
fi

if [[ -z "$key_pair_name" ]]; then
  errecho "ERROR: You must provide a key pair name with the -k parameter."
  usage
  return 1
fi

if [[ -z "$security_group_id" ]]; then
  errecho "ERROR: You must provide a security group ID with the -s parameter."
  usage
```

```
    return 1
  fi

  if [[ -z "$count" ]]; then
    count=1
  fi

  response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
  }

  echo "$response"

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
```

```
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
###############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie RunInstances in der AWS CLI Befehlsreferenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_start_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
    return 1
  fi

  response=$(aws ec2 start-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports start-instances operation failed with $response."
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
```

```
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie StartInstances in der AWS CLI Befehlsreferenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances.
```

```
#
# Parameters:
#       -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_stop_instances() {
  local instance_ids
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_stop_instances"
    echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
 parameter."
    usage
```

```
    return 1
  fi

  response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
#############################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
```

```
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie StopInstances in der AWS CLI Befehlsreferenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###########################################################################
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#       -i instance_ids - A space-separated list of instance IDs.
#       -h - Display help.
#
# Returns:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function ec2_terminate_instances() {
  local instance_ids response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_terminate_instances"
    echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
 instances."
    echo "  -i instance_ids - A space-separated list of instance IDs."
    echo "  -h - Display help."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) instance_ids="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  # Check if instance ID is provided
  if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
```

```
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
  }

  return 0
}
```

Die in diesem Beispiel verwendeten Hilfsfunktionen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

################################################################################
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#         $1 - The error code returned by the AWS CLI.
#
#  Returns:
#          0: - Success.
#
################################################################################
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho "  One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then
    errecho "  Command line failed to parse."
  elif [ "$err_code" == 130 ]; then
    errecho "  Process received SIGINT."
```

```
  elif [ "$err_code" == 252 ]; then
    errecho "  Command syntax invalid."
  elif [ "$err_code" == 253 ]; then
    errecho "  The system environment or configuration was invalid."
  elif [ "$err_code" == 254 ]; then
    errecho "  The service returned an error."
  elif [ "$err_code" == 255 ]; then
    errecho "  255 is a catch-all error."
  fi

  return 0
}
```

- API-Details finden Sie TerminateInstances in der AWS CLI Befehlsreferenz.

Szenarien

Konfigurieren Sie die Container-Dienstkonnektivität

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR
```

```
# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
```

```
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
```

```
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
```

```
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
```

```
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
```

```
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
```

```
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
```

```
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)
```

```
    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "==========================================="
    echo "CREATED RESOURCES"
```

```
    echo "==========================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "==========================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
```

```
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi
```

```
                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service
```

```
    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteLogGroup
- DeleteNamespace
- DeleteRole
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeClusters
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie ein einfaches virtuelles privates Netzwerk

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer VPC
- Erstellen von Subnetzen
- Konfigurieren der Internetverbindung

- Erstellen eines NAT-Gateways
- Konfigurieren der Subnetz-Einstellungen
- Erstellen von Sicherheitsgruppen
- Ihre VPC-Konfiguration überprüfen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Creation Script
# This script creates a VPC with public and private subnets, internet gateway, NAT
 gateway, and security groups

# Set up logging
LOG_FILE="vpc_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to handle errors
handle_error() {
  echo "ERROR: $1"
  echo "Resources created before error:"
  for resource in "${CREATED_RESOURCES[@]}"
  do
    echo "- $resource"
  done

  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to clean up resources
cleanup_resources() {
```

```
 echo "Cleaning up resources in reverse order..."

 # Reverse the array to delete in reverse order of creation
 for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--))
 do
   resource="${CREATED_RESOURCES[$i]}"
   resource_type=$(echo "$resource" | cut -d':' -f1)
   resource_id=$(echo "$resource" | cut -d':' -f2)

   case "$resource_type" in
     "INSTANCE")
       echo "Terminating EC2 instance: $resource_id"
       aws ec2 terminate-instances --instance-ids "$resource_id" || echo "Failed to
terminate instance: $resource_id"
       # Wait for instance to terminate
       echo "Waiting for instance to terminate..."
       aws ec2 wait instance-terminated --instance-ids "$resource_id" || echo
"Failed to wait for instance termination: $resource_id"
       ;;
     "KEY_PAIR")
       echo "Deleting key pair: $resource_id"
       aws ec2 delete-key-pair --key-name "$resource_id" || echo "Failed to delete
key pair: $resource_id"
       # Remove the .pem file if it exists
       if [ -f "${resource_id}.pem" ]; then
         rm -f "${resource_id}.pem"
       fi
       ;;
     "NAT_GATEWAY")
       echo "Deleting NAT Gateway: $resource_id"
       aws ec2 delete-nat-gateway --nat-gateway-id "$resource_id" || echo "Failed
to delete NAT Gateway: $resource_id"
       # NAT Gateway deletion takes time, wait for it to complete
       echo "Waiting for NAT Gateway to be deleted..."
       aws ec2 wait nat-gateway-deleted --nat-gateway-ids "$resource_id" || echo
"Failed to wait for NAT Gateway deletion: $resource_id"
       ;;
     "EIP")
       echo "Releasing Elastic IP: $resource_id"
       aws ec2 release-address --allocation-id "$resource_id" || echo "Failed to
release Elastic IP: $resource_id"
       ;;
     "ROUTE_TABLE_ASSOCIATION")
       echo "Disassociating Route Table: $resource_id"
```

```
        aws ec2 disassociate-route-table --association-id "$resource_id" || echo
 "Failed to disassociate Route Table: $resource_id"
        ;;
      "ROUTE_TABLE")
        echo "Deleting Route Table: $resource_id"
        aws ec2 delete-route-table --route-table-id "$resource_id" || echo "Failed
 to delete Route Table: $resource_id"
        ;;
      "INTERNET_GATEWAY")
        echo "Detaching Internet Gateway: $resource_id from VPC: $VPC_ID"
        aws ec2 detach-internet-gateway --internet-gateway-id "$resource_id" --vpc-
id "$VPC_ID" || echo "Failed to detach Internet Gateway: $resource_id"
        echo "Deleting Internet Gateway: $resource_id"
        aws ec2 delete-internet-gateway --internet-gateway-id "$resource_id" || echo
 "Failed to delete Internet Gateway: $resource_id"
        ;;
      "SECURITY_GROUP")
        echo "Deleting Security Group: $resource_id"
        aws ec2 delete-security-group --group-id "$resource_id" || echo "Failed to
 delete Security Group: $resource_id"
        ;;
      "SUBNET")
        echo "Deleting Subnet: $resource_id"
        aws ec2 delete-subnet --subnet-id "$resource_id" || echo "Failed to delete
 Subnet: $resource_id"
        ;;
      "VPC")
        echo "Deleting VPC: $resource_id"
        aws ec2 delete-vpc --vpc-id "$resource_id" || echo "Failed to delete VPC:
 $resource_id"
        ;;
    esac
  done
}

# Initialize array to track created resources
CREATED_RESOURCES=()

echo "Starting VPC creation script at $(date)"

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
aws configure list || handle_error "AWS CLI is not properly configured"
```

```
# Verify identity and permissions
echo "Verifying identity and permissions..."
if ! aws sts get-caller-identity; then
  echo "ERROR: Unable to verify AWS identity. This could be due to:"
  echo "  - Expired credentials"
  echo "  - Missing or invalid AWS credentials"
  echo "  - Insufficient permissions"
  echo ""
  echo "Please run 'aws configure' to update your credentials or check your IAM
 permissions."
  exit 1
fi

# Create VPC
echo "Creating VPC with CIDR block 10.0.0.0/16..."
VPC_ID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'Vpc.VpcId' --output text)

if [ -z "$VPC_ID" ]; then
  handle_error "Failed to create VPC"
fi

CREATED_RESOURCES+=("VPC:$VPC_ID")
echo "VPC created with ID: $VPC_ID"

# Enable DNS support and hostnames
echo "Enabling DNS support and hostnames for VPC..."
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-support || handle_error
 "Failed to enable DNS support"
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-hostnames ||
 handle_error "Failed to enable DNS hostnames"

# Get available Availability Zones
echo "Getting available Availability Zones..."
AZ1=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[0].ZoneName' --
output text)
AZ2=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[1].ZoneName' --
output text)

if [ -z "$AZ1" ] || [ -z "$AZ2" ]; then
  handle_error "Failed to get Availability Zones"
fi
```

```
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create public subnets
echo "Creating public subnet in $AZ1..."
PUBLIC_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.0.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ1" ]; then
  handle_error "Failed to create public subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ1")
echo "Public subnet created in $AZ1 with ID: $PUBLIC_SUBNET_AZ1"

echo "Creating public subnet in $AZ2..."
PUBLIC_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.1.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ2" ]; then
  handle_error "Failed to create public subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ2")
echo "Public subnet created in $AZ2 with ID: $PUBLIC_SUBNET_AZ2"

# Create private subnets
echo "Creating private subnet in $AZ1..."
PRIVATE_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.2.0/24 \
  --availability-zone "$AZ1" \
```

```
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ1},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ1" ]; then
  handle_error "Failed to create private subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ1")
echo "Private subnet created in $AZ1 with ID: $PRIVATE_SUBNET_AZ1"

echo "Creating private subnet in $AZ2..."
PRIVATE_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.3.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ2},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PRIVATE_SUBNET_AZ2" ]; then
  handle_error "Failed to create private subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ2")
echo "Private subnet created in $AZ2 with ID: $PRIVATE_SUBNET_AZ2"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
  --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name,Value=MyIGW},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'InternetGateway.InternetGatewayId' \
  --output text)

if [ -z "$IGW_ID" ]; then
  handle_error "Failed to create Internet Gateway"
fi

CREATED_RESOURCES+=("INTERNET_GATEWAY:$IGW_ID")
echo "Internet Gateway created with ID: $IGW_ID"
```

```
# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id "$VPC_ID"
 || handle_error "Failed to attach Internet Gateway to VPC"

# Create public route table
echo "Creating public route table..."
PUBLIC_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-RT},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PUBLIC_RT" ]; then
  handle_error "Failed to create public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PUBLIC_RT")
echo "Public route table created with ID: $PUBLIC_RT"

# Add route to Internet Gateway
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID" || handle_error "Failed to add route to Internet
 Gateway"

# Associate public subnets with public route table
echo "Associating public subnet in $AZ1 with public route table..."
PUBLIC_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT" --
subnet-id "$PUBLIC_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_1" ]; then
  handle_error "Failed to associate public subnet in AZ1 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_1")

echo "Associating public subnet in $AZ2 with public route table..."
PUBLIC_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT" --
subnet-id "$PUBLIC_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_2" ]; then
  handle_error "Failed to associate public subnet in AZ2 with public route table"
```

```
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_2")

# Create private route table
echo "Creating private route table..."
PRIVATE_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-RT},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PRIVATE_RT" ]; then
  handle_error "Failed to create private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PRIVATE_RT")
echo "Private route table created with ID: $PRIVATE_RT"

# Associate private subnets with private route table
echo "Associating private subnet in $AZ1 with private route table..."
PRIVATE_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT" --
subnet-id "$PRIVATE_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_1" ]; then
  handle_error "Failed to associate private subnet in AZ1 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_1")

echo "Associating private subnet in $AZ2 with private route table..."
PRIVATE_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT" --
subnet-id "$PRIVATE_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_2" ]; then
  handle_error "Failed to associate private subnet in AZ2 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_2")

# Allocate Elastic IP for NAT Gateway
echo "Allocating Elastic IP for NAT Gateway..."
```

```
EIP_ALLOC=$(aws ec2 allocate-address --domain vpc --tag-specifications
 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-gs}]' --query 'AllocationId' --output text)

if [ -z "$EIP_ALLOC" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

CREATED_RESOURCES+=("EIP:$EIP_ALLOC")
echo "Elastic IP allocated with ID: $EIP_ALLOC"

# Create NAT Gateway
echo "Creating NAT Gateway in public subnet in $AZ1..."
NAT_GW=$(aws ec2 create-nat-gateway \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --allocation-id "$EIP_ALLOC" \
  --tag-specifications 'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'NatGateway.NatGatewayId' \
  --output text)

if [ -z "$NAT_GW" ]; then
  handle_error "Failed to create NAT Gateway"
fi

CREATED_RESOURCES+=("NAT_GATEWAY:$NAT_GW")
echo "NAT Gateway created with ID: $NAT_GW"

# Wait for NAT Gateway to be available
echo "Waiting for NAT Gateway to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW" || handle_error "NAT
 Gateway did not become available"

# Add route to NAT Gateway in private route table
echo "Adding route to NAT Gateway in private route table..."
aws ec2 create-route --route-table-id "$PRIVATE_RT" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW" || handle_error "Failed to add route to NAT
 Gateway"

# Enable auto-assign public IP for instances in public subnets
echo "Enabling auto-assign public IP for instances in public subnet in $AZ1..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ1" --map-public-ip-on-
launch || handle_error "Failed to enable auto-assign public IP for public subnet in
 AZ1"
```

```
echo "Enabling auto-assign public IP for instances in public subnet in $AZ2..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ2" --map-public-ip-on-
launch || handle_error "Failed to enable auto-assign public IP for public subnet in
 AZ2"

# Create security group for web servers
echo "Creating security group for web servers..."
WEB_SG=$(aws ec2 create-security-group \
  --group-name "WebServerSG-$(date +%s)" \
  --description "Security group for web servers" \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$WEB_SG" ]; then
  handle_error "Failed to create security group for web servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$WEB_SG")
echo "Security group for web servers created with ID: $WEB_SG"

# Allow HTTP and HTTPS traffic
echo "Allowing HTTP traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --port
 80 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTP traffic"

echo "Allowing HTTPS traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --port
 443 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTPS traffic"

# Note: In a production environment, you should restrict the source IP ranges for
 security
echo "NOTE: In a production environment, you should restrict the source IP ranges
 for HTTP and HTTPS traffic"

# Create security group for database servers
echo "Creating security group for database servers..."
DB_SG=$(aws ec2 create-security-group \
  --group-name "DBServerSG-$(date +%s)" \
  --description "Security group for database servers" \
  --vpc-id "$VPC_ID" \
```

```
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'GroupId' \
  --output text)

if [ -z "$DB_SG" ]; then
  handle_error "Failed to create security group for database servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$DB_SG")
echo "Security group for database servers created with ID: $DB_SG"

# Allow MySQL/Aurora traffic from web servers only
echo "Allowing MySQL/Aurora traffic from web servers to database servers..."
aws ec2 authorize-security-group-ingress --group-id "$DB_SG" --protocol tcp --
port 3306 --source-group "$WEB_SG" || handle_error "Failed to allow MySQL/Aurora
 traffic"

# Verify VPC configuration
echo "Verifying VPC configuration..."
echo "VPC:"
aws ec2 describe-vpcs --vpc-id "$VPC_ID" || handle_error "Failed to describe VPC"

echo "Subnets:"
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe subnets"

echo "Route tables:"
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe route tables"

echo "Internet gateway:"
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=$VPC_ID"
 || handle_error "Failed to describe Internet Gateway"

echo "NAT gateway:"
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=$VPC_ID" || handle_error
 "Failed to describe NAT Gateway"

echo "Security groups:"
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$VPC_ID" ||
 handle_error "Failed to describe security groups"

echo ""
```

```
# Summary of created resources
echo "VPC creation completed successfully!"
echo "Summary of created resources:"
echo "- VPC: $VPC_ID"
echo "- Public Subnet in $AZ1: $PUBLIC_SUBNET_AZ1"
echo "- Public Subnet in $AZ2: $PUBLIC_SUBNET_AZ2"
echo "- Private Subnet in $AZ1: $PRIVATE_SUBNET_AZ1"
echo "- Private Subnet in $AZ2: $PRIVATE_SUBNET_AZ2"
echo "- Internet Gateway: $IGW_ID"
echo "- Public Route Table: $PUBLIC_RT"
echo "- Private Route Table: $PRIVATE_RT"
echo "- Elastic IP: $EIP_ALLOC"
echo "- NAT Gateway: $NAT_GW"
echo "- Web Servers Security Group: $WEB_SG"
echo "- Database Servers Security Group: $DB_SG"

# Deploy EC2 instances
echo ""
echo "Deploying EC2 instances..."

# Create key pair for SSH access
KEY_NAME="vpc-tutorial-key-$(date +%s)"
echo "Creating key pair $KEY_NAME..."
aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-specifications
 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
gs}]' --query 'KeyMaterial' --output text > "${KEY_NAME}.pem" || handle_error
 "Failed to create key pair"
chmod 400 "${KEY_NAME}.pem"
echo "Key pair saved to ${KEY_NAME}.pem"
CREATED_RESOURCES+=("KEY_PAIR:$KEY_NAME")

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text) ||
 handle_error "Failed to get AMI"
echo "Using AMI: $AMI_ID"

# Launch web server in public subnet
echo "Launching web server in public subnet..."
WEB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
```

```
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$WEB_SG" \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --associate-public-ip-address \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y httpd
    systemctl start httpd
    systemctl enable httpd
    echo "<h1>Hello from $(hostname -f) in the public subnet</h1>" > /var/www/html/
index.html' \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch web server"
echo "Web server instance created with ID: $WEB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$WEB_INSTANCE")

# Wait for web server to be running
echo "Waiting for web server to be running..."
aws ec2 wait instance-running --instance-ids "$WEB_INSTANCE"

# Get web server public IP
WEB_PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$WEB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "Web server public IP: $WEB_PUBLIC_IP"
echo "You can access the web server at: http://$WEB_PUBLIC_IP"

# Launch database server in private subnet
echo "Launching database server in private subnet..."
DB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$DB_SG" \
  --subnet-id "$PRIVATE_SUBNET_AZ1" \
  --user-data '#!/bin/bash
    yum update -y
    yum install -y mariadb-server
    systemctl start mariadb
    systemctl enable mariadb' \
```

```
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-gs}]' \
  --query 'Instances[0].InstanceId' \
  --output text) || handle_error "Failed to launch database server"
echo "Database server instance created with ID: $DB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$DB_INSTANCE")

# Wait for database server to be running
echo "Waiting for database server to be running..."
aws ec2 wait instance-running --instance-ids "$DB_INSTANCE"

# Get database server private IP
DB_PRIVATE_IP=$(aws ec2 describe-instances --instance-ids "$DB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PrivateIpAddress' --output text)
echo "Database server private IP: $DB_PRIVATE_IP"

echo "EC2 instances deployed successfully!"
echo "- Web Server (Public): $WEB_INSTANCE ($WEB_PUBLIC_IP)"
echo "- Database Server (Private): $DB_INSTANCE ($DB_PRIVATE_IP)"
echo ""
echo "Note: To connect to the web server: ssh -i ${KEY_NAME}.pem ec2-user@
$WEB_PUBLIC_IP"
echo "To connect to the database server, you must first connect to the web server,
 then use it as a bastion host."
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  echo "Cleaning up resources..."
  cleanup_resources
  echo "All resources have been cleaned up."
else
  echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AllocateAddress
  - AssociateRouteTable

- AttachInternetGateway
- AuthorizeSecurityGroupIngress
- CreateInternetGateway
- CreateKeyPair
- CreateNatGateway
- CreateRoute
- CreateRouteTable
- CreateSecurityGroup
- CreateSubnet
- CreateVpc
- DeleteInternetGateway
- DeleteKeyPair
- DeleteNatGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeImages
- DescribeInstances
- DescribeInternetGateways
- DescribeNatGateways
- DescribeRouteTables
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachInternetGateway
- DisassociateRouteTable
- ModifySubnetAttribute

- ModifyVpcAttribute

- ReleaseAddress
- RunInstances
- TerminateInstances

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
```

```
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}
```

```
# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."
```

```
        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"
```

```
        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
```

```
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi
```

```
        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
```

```
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
```

```
# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
```

```
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"
```

```
# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
```

```
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
```

```
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
```

```
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition

- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Erstellen und überwachen Sie einen Dienst
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
```

```
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
```

```
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
 "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
```

```
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
```

```
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
for details."
    fi
```

```
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}
```

```
# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
```

```
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
```

```
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole
```

```
        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
```

```
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
```

```
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
```

```
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
```

```
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T
```

```
    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
```

```
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
```

```
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances

- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Eine relationale Datenbankinstanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Netzwerkkomponenten ein
- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
```

```
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi
```

```
echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
```

```
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
```

```
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
```

```
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
```

```
    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - Wait

Erstellung und Verwaltung eines Service-Netzwerk-Mesh

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ein Servicenetzwerk erstellen
- Einen Service erstellen
- Verfügbare VPCs auflisten
- Listet Sicherheitsgruppen für die ausgewählte VPC auf
- Listet die Dienstzuordnungen auf
- VPC-Zuordnungen auflisten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
```

```
    if [ "$1" -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""
```

```
   validate_input "$resource_type" "resource_type" || return 1
   validate_input "$resource_id" "resource_id" || return 1
   validate_input "$desired_status" "desired_status" || return 1

   echo "Waiting for $resource_type $resource_id to be in state $desired_status..."
| tee -a "$LOG_FILE"

   while [ "$attempt" -le "$max_attempts" ]; do
       echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

       # Run the command to get the status and capture the output
       status_output=$(eval "$command" 2>&1) || true
       echo "$status_output" >> "$LOG_FILE"

       # For service networks, they do not have a status field in the output
       # We'll consider them active if we can retrieve them
       if [[ "$resource_type" == "Service Network" ]]; then
           if [[ "$status_output" == *"$resource_id"* ]]; then
               echo "$resource_type $resource_id is now active" | tee -a
"$LOG_FILE"
               return 0
           fi
       else
           # For other resources, extract the status field
           status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
$4}' || true)
           echo "Current status: $status" >> "$LOG_FILE"

           if [[ "$status" == "$desired_status" ]]; then
               echo "$resource_type $resource_id is now in state $desired_status" |
tee -a "$LOG_FILE"
               return 0
           elif [[ "$status" == *"FAIL"* ]]; then
               echo "ERROR: $resource_type $resource_id failed to reach desired
state. Current status: $status" | tee -a "$LOG_FILE"
               return 1
           fi
       fi

       echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
"$LOG_FILE"
       sleep 10
       ((attempt++))
   done
```

```
    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-lattice
 get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --output
 json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name '$SERVICE_NAME'
 --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-vpc-lattice-gs --
output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service --
service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"
```

```
SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-service-
association --service-identifier '$SERVICE_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"

VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
```

```
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?
```

```
        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-vpc-association --service-network-vpc-association-
identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee -a
 "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --output
 json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
```

```
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --service-
network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --service-
network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'" ||
 true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"
```

```
echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation
  - ListServiceNetworkServiceAssociations
  - ListServiceNetworkVpcAssociations

Blockspeicher-Volumes erstellen und verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EBS-Volume
- Überprüfen Sie den Status des Volumes
- Erstellen Sie eine EC2-Instanz (optional)
- Hängen Sie ein Volume an eine EC2-Instance an
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create and manage Amazon EBS volumes
# This script demonstrates how to create an EBS volume and attach it to an EC2
 instance
# It can also create a test EC2 instance if needed

# Set up logging
LOG_FILE="ebs-volume-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS volume creation script at $(date)"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "$VOLUME_ID" ]; then
        echo "- EBS Volume: $VOLUME_ID"
    fi
    if [ -n "$INSTANCE_ID" ]; then
        echo "- EC2 Instance: $INSTANCE_ID"
    fi
    if [ -n "$SG_ID" ]; then
        echo "- Security Group: $SG_ID"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Do you want to clean up created resources? (y/n): "
    read -r CLEANUP_CHOICE
```

```
    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
    else
        echo "Resources were not cleaned up. You will need to delete them manually."
    fi

    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "$VOLUME_ID" ] && [ "$ATTACHED" = true ]; then
        echo "Detaching volume $VOLUME_ID..."
        aws ec2 detach-volume --volume-id "$VOLUME_ID"

        # Wait for volume to be detached
        echo "Waiting for volume to be detached..."
        aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
    fi

    if [ -n "$VOLUME_ID" ]; then
        echo "Deleting volume $VOLUME_ID..."
        aws ec2 delete-volume --volume-id "$VOLUME_ID"
    fi

    if [ -n "$INSTANCE_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Terminating instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
    fi

    # Clean up security group if created
    if [ -n "$SG_ID" ] && [ "$CREATED_INSTANCE" = true ]; then
        echo "Deleting security group $SG_ID..."
        # Wait a bit for instance termination to complete
        sleep 10
        aws ec2 delete-security-group --group-id "$SG_ID" 2>/dev/null || echo
 "Security group may have dependencies, delete manually if needed"
    fi
```

```
    echo "Cleanup completed."
}

# Function to get available instance type
get_available_instance_type() {
    local region=$1

    # Try instance types in order of preference (cheapest first)
    local instance_types=("t3.nano" "t3.micro" "t2.micro" "t2.nano")

    for instance_type in "${instance_types[@]}"; do
        local available=$(aws ec2 describe-instance-type-offerings \
            --region "$region" \
            --filters "Name=instance-type,Values=$instance_type" \
            --query "length(InstanceTypeOfferings)" \
            --output text)

        if [ "$available" -gt 0 ]; then
            echo "$instance_type"
            return 0
        fi
    done

    # If none of the preferred types are available, get any available type
    local fallback_type=$(aws ec2 describe-instance-type-offerings \
        --region "$region" \
        --query "InstanceTypeOfferings[0].InstanceType" \
        --output text)

    if [ "$fallback_type" != "None" ] && [ -n "$fallback_type" ]; then
        echo "$fallback_type"
        return 0
    fi

    return 1
}

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 "AvailabilityZones[0].RegionName" --output text)
fi
```

```
echo "Using region: $REGION"

# Get available Availability Zones
echo "Retrieving available Availability Zones..."
AZ=$(aws ec2 describe-availability-zones --filters "Name=state,Values=available" --
query "AvailabilityZones[0].ZoneName" --output text)
if [ -z "$AZ" ]; then
    handle_error "Failed to retrieve Availability Zones"
fi
echo "Using Availability Zone: $AZ"

# Create a gp3 volume
echo "Creating a 10 GiB gp3 volume in $AZ..."
VOLUME_ID=$(aws ec2 create-volume \
    --volume-type gp3 \
    --size 10 \
    --availability-zone "$AZ" \
    --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=EBSTutorialVolume},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-
gs-volumes}]' \
    --query 'VolumeId' \
    --output text)

if [ -z "$VOLUME_ID" ]; then
    handle_error "Failed to create EBS volume"
fi

echo "Volume created with ID: $VOLUME_ID"

# Wait for volume to become available
echo "Waiting for volume to become available..."
aws ec2 wait volume-available --volume-ids "$VOLUME_ID"
if [ $? -ne 0 ]; then
    handle_error "Volume did not become available"
fi

# Check volume details
echo "Retrieving volume details..."
aws ec2 describe-volumes --volume-ids "$VOLUME_ID"

# Ask if user wants to attach the volume to an instance
echo ""
echo "==========================================="
```

```
echo "VOLUME ATTACHMENT"
echo "==========================================="
echo "Do you want to attach this volume to an EC2 instance? (y/n): "
read -r ATTACH_CHOICE

ATTACHED=false
CREATED_INSTANCE=false
INSTANCE_ID=""
SG_ID=""

if [[ "$ATTACH_CHOICE" =~ ^[Yy]$ ]]; then
    # List available instances in the same AZ
    echo "Retrieving EC2 instances in $AZ..."
    INSTANCES_COUNT=$(aws ec2 describe-instances \
        --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
        --query "length(Reservations[].Instances[])" \
        --output text)

    # Check if there are any running instances in the AZ
    if [ "$INSTANCES_COUNT" -eq 0 ]; then
        echo "No running instances found in $AZ."
        echo ""
        echo "Would you like to create a test EC2 instance? (y/n): "
        read -r CREATE_INSTANCE_CHOICE

        if [[ "$CREATE_INSTANCE_CHOICE" =~ ^[Yy]$ ]]; then
            # Get available instance type
            echo "Finding available instance type for region $REGION..."
            INSTANCE_TYPE=$(get_available_instance_type "$REGION")
            if [ $? -ne 0 ] || [ -z "$INSTANCE_TYPE" ]; then
                handle_error "No suitable instance type found in region $REGION"
            fi
            echo "Using instance type: $INSTANCE_TYPE"

            # Get the latest Amazon Linux 2 AMI
            echo "Finding the latest Amazon Linux 2 AMI..."
            AMI_ID=$(aws ec2 describe-images \
                --owners amazon \
                --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
                --query "sort_by(Images, &CreationDate)[-1].ImageId" \
                --output text)
```

```
            if [ -z "$AMI_ID" ]; then
                handle_error "Failed to find a suitable AMI"
            fi

            echo "Using AMI: $AMI_ID"

            # Check if a default VPC exists
            DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
                --filters "Name=isDefault,Values=true" \
                --query "Vpcs[0].VpcId" \
                --output text)

            if [ "$DEFAULT_VPC_ID" = "None" ] || [ -z "$DEFAULT_VPC_ID" ]; then
                handle_error "No default VPC found. Please create a VPC and subnet
 before running this script."
            fi

            # Get a subnet in the selected AZ
            SUBNET_ID=$(aws ec2 describe-subnets \
                --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=availability-
zone,Values=$AZ" \
                --query "Subnets[0].SubnetId" \
                --output text)

            if [ -z "$SUBNET_ID" ] || [ "$SUBNET_ID" = "None" ]; then
                handle_error "No subnet found in $AZ. Please create a subnet before
 running this script."
            fi

            echo "Using subnet: $SUBNET_ID"

            # Create a security group that allows SSH
            SG_NAME="EBSTutorialSG-$(date +%s)"
            SG_ID=$(aws ec2 create-security-group \
                --group-name "$SG_NAME" \
                --description "Security group for EBS tutorial" \
                --vpc-id "$DEFAULT_VPC_ID" \
                --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-gs-volumes}]' \
                --query "GroupId" \
                --output text)

            if [ -z "$SG_ID" ]; then
                handle_error "Failed to create security group"
```

```
            fi

            echo "Created security group: $SG_ID"

            # Add a rule to allow SSH
            aws ec2 authorize-security-group-ingress \
                --group-id "$SG_ID" \
                --protocol tcp \
                --port 22 \
                --cidr 0.0.0.0/0

            echo "Added SSH rule to security group"

            # Create the instance
            echo "Creating EC2 instance in $AZ with instance type $INSTANCE_TYPE..."
            INSTANCE_ID=$(aws ec2 run-instances \
                --image-id "$AMI_ID" \
                --instance-type "$INSTANCE_TYPE" \
                --subnet-id "$SUBNET_ID" \
                --security-group-ids "$SG_ID" \
                --tag-specifications
 'ResourceType=instance,Tags=[{Key=Name,Value=EBSTutorialInstance},
{Key=Purpose,Value=Tutorial},{Key=project,Value=doc-smith},{Key=tutorial,Value=ebs-
gs-volumes}]' \
                --query "Instances[0].InstanceId" \
                --output text)

            if [ -z "$INSTANCE_ID" ]; then
                handle_error "Failed to create EC2 instance"
            fi

            CREATED_INSTANCE=true
            echo "Instance created with ID: $INSTANCE_ID"

            # Wait for the instance to be running
            echo "Waiting for instance to be running..."
            aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

            # Wait a bit more for the instance to initialize
            echo "Waiting for instance initialization (30 seconds)..."
            sleep 30
        else
            echo "Skipping instance creation and volume attachment."
            INSTANCE_ID=""
```

```
        fi
    else
        # Display available instances
        echo "Available instances in $AZ:"
        aws ec2 describe-instances \
            --filters "Name=availability-zone,Values=$AZ" "Name=instance-state-
name,Values=running" \
            --query "Reservations[*].Instances[*].[InstanceId,Tags[?
Key=='Name'].Value|[0],InstanceType]" \
            --output table

        # Ask for instance ID
        echo ""
        echo "Enter the instance ID to attach the volume to (or press Enter to
 skip): "
        read -r INSTANCE_ID
    fi

    if [ -n "$INSTANCE_ID" ]; then
        # Attach volume to the instance
        echo "Attaching volume $VOLUME_ID to instance $INSTANCE_ID..."
        ATTACH_RESULT=$(aws ec2 attach-volume \
            --volume-id "$VOLUME_ID" \
            --instance-id "$INSTANCE_ID" \
            --device "/dev/sdf" \
            --query 'State' \
            --output text)

        if [ $? -ne 0 ] || [ -z "$ATTACH_RESULT" ]; then
            handle_error "Failed to attach volume to instance"
        fi

        ATTACHED=true
        echo "Volume attached successfully. Device: /dev/sdf"

        # Verify attachment
        echo "Verifying attachment..."
        aws ec2 describe-volumes \
            --volume-ids "$VOLUME_ID" \
            --query "Volumes[0].Attachments"
    else
        echo "Skipping volume attachment."
    fi
else
```

```
    echo "Skipping volume attachment."
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "Created resources:"
echo "- EBS Volume: $VOLUME_ID"
if [ "$ATTACHED" = true ]; then
    echo "  - Attached to: $INSTANCE_ID as /dev/sdf"
fi
if [ "$CREATED_INSTANCE" = true ]; then
    echo "- EC2 Instance: $INSTANCE_ID (type: $INSTANCE_TYPE)"
fi
if [ -n "$SG_ID" ]; then
    echo "- Security Group: $SG_ID"
fi

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
else
    echo ""
    echo "Resources were not cleaned up. You can manually delete them later."
    if [ -n "$VOLUME_ID" ]; then
        if [ "$ATTACHED" = true ]; then
            echo "To detach the volume:"
            echo "  aws ec2 detach-volume --volume-id $VOLUME_ID"
        fi
        echo "To delete the volume:"
        echo "  aws ec2 delete-volume --volume-id $VOLUME_ID"
    fi
    if [ "$CREATED_INSTANCE" = true ]; then
        echo "To terminate the instance:"
        echo "  aws ec2 terminate-instances --instance-ids $INSTANCE_ID"
```

```
    fi
    if [ -n "$SG_ID" ]; then
        echo "To delete the security group (after instance termination):"
        echo "  aws ec2 delete-security-group --group-id $SG_ID"
    fi
fi

echo ""
echo "Script completed at $(date)"
echo "================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachVolume
  - AuthorizeSecurityGroupIngress
  - CreateSecurityGroup
  - CreateVolume
  - DeleteSecurityGroup
  - DeleteVolume
  - DescribeAvailabilityZones
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeSubnets
  - DescribeVolumes
  - DescribeVpcs
  - DetachVolume
  - RunInstances
  - TerminateInstances
  - Wait

Beginnen Sie mit dedizierten Netzwerkverbindungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 CreateVpnGateway

- Benutze ec2 DeleteVpnGateway
- Benutze ec2 DescribeVpnGateways
- Verwenden Sie DirectConnect CreateConnection
- Verwenden Sie DirectConnect CreatePrivateVirtualInterface
- Verwenden Sie DirectConnect DeleteConnection

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Direct Connect Connection Management Script - Version 6
# This script demonstrates how to create and manage AWS Direct Connect connections
 using the AWS CLI
# This version includes fixes for user input handling and better error reporting

# Set up logging
LOG_FILE="directconnect-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting AWS Direct Connect script v6"

# Function to check for errors in command output
check_error() {
    local output=$1
    local command=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $command"
        echo "Output: $output"
        cleanup_resources
        exit 1
    fi
}
```

```
# Function to wait for VGW to be available
wait_for_vgw() {
    local vgw_id=$1
    local max_attempts=30
    local attempt=1

    echo "Waiting for virtual private gateway $vgw_id to become available..."

    while [ $attempt -le $max_attempts ]; do
        VGW_STATE=$(aws ec2 describe-vpn-gateways --vpn-gateway-ids "$vgw_id" --
query 'VpnGateways[0].State' --output text)

        if [ "$VGW_STATE" == "available" ]; then
            echo "Virtual private gateway is now available"
            return 0
        elif [ "$VGW_STATE" == "failed" ]; then
            echo "Virtual private gateway failed to become available"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: VGW state is $VGW_STATE, waiting 10
 seconds..."
        sleep 10
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for VGW to become available"
    return 1
}

# Function to wait for connection to be available
wait_for_connection() {
    local connection_id=$1
    local max_attempts=60
    local attempt=1

    echo "Waiting for connection $connection_id to become available..."
    echo "Note: This can take 30+ minutes in production as AWS provisions the
 physical connection"

    while [ $attempt -le $max_attempts ]; do
        CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$connection_id" --query 'connections[0].connectionState' --output text)
```

```
        if [ "$CONNECTION_STATE" == "available" ]; then
            echo "Connection is now available"
            return 0
        elif [ "$CONNECTION_STATE" == "rejected" ] || [ "$CONNECTION_STATE" ==
 "deleted" ]; then
            echo "Connection failed with state: $CONNECTION_STATE"
            return 1
        fi

        echo "Attempt $attempt/$max_attempts: Connection state is $CONNECTION_STATE,
 waiting 30 seconds..."
        sleep 30
        attempt=$((attempt + 1))
    done

    echo "Timeout waiting for connection to become available"
    return 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete virtual interfaces if they exist
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "Deleting private virtual interface: $PRIVATE_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PRIVATE_VIF_ID"
    fi

    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "Deleting public virtual interface: $PUBLIC_VIF_ID"
        aws directconnect delete-virtual-interface --virtual-interface-id
 "$PUBLIC_VIF_ID"
    fi

    # Delete connection if it exists
    if [ -n "$CONNECTION_ID" ]; then
        echo "Deleting connection: $CONNECTION_ID"
        aws directconnect delete-connection --connection-id "$CONNECTION_ID"
    fi

    # Delete VGW if it exists
    if [ -n "$VGW_ID" ]; then
```

```
        echo "Deleting virtual private gateway: $VGW_ID"
        aws ec2 delete-vpn-gateway --vpn-gateway-id "$VGW_ID"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 6)
CONNECTION_NAME="DxConn-${RANDOM_ID}"

# Step 1: List available Direct Connect locations
echo "Listing available Direct Connect locations..."
LOCATIONS_OUTPUT=$(aws directconnect describe-locations)
check_error "$LOCATIONS_OUTPUT" "describe-locations"
echo "$LOCATIONS_OUTPUT"

# Extract the first location code for demonstration purposes
LOCATION_CODE=$(aws directconnect describe-locations --query
 'locations[0].locationCode' --output text)
if [ -z "$LOCATION_CODE" ] || [ "$LOCATION_CODE" == "None" ]; then
    echo "Error: Could not extract location code from the output."
    exit 1
fi
echo "Using location: $LOCATION_CODE"

# Step 2: Create a dedicated connection
echo "Creating a dedicated connection at location $LOCATION_CODE with bandwidth
 1Gbps..."
CONNECTION_OUTPUT=$(aws directconnect create-connection \
    --location "$LOCATION_CODE" \
    --bandwidth "1Gbps" \
    --connection-name "$CONNECTION_NAME" \
    --tags key=project,value=doc-smith key=tutorial,value=aws-direct-connect-gs)
check_error "$CONNECTION_OUTPUT" "create-connection"
echo "$CONNECTION_OUTPUT"

# Extract connection ID directly from the output
CONNECTION_ID=$(echo "$CONNECTION_OUTPUT" | grep -o '"connectionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$CONNECTION_ID" ]; then
    echo "Error: Could not extract connection ID from the output."
    exit 1
fi
echo "Connection created with ID: $CONNECTION_ID"
```

```
# Step 3: Describe the connection
echo "Retrieving connection details..."
DESCRIBE_OUTPUT=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID")
check_error "$DESCRIBE_OUTPUT" "describe-connections"
echo "$DESCRIBE_OUTPUT"

# Step 4: Update the connection name
NEW_CONNECTION_NAME="${CONNECTION_NAME}-updated"
echo "Updating connection name to $NEW_CONNECTION_NAME..."
UPDATE_OUTPUT=$(aws directconnect update-connection \
    --connection-id "$CONNECTION_ID" \
    --connection-name "$NEW_CONNECTION_NAME")
check_error "$UPDATE_OUTPUT" "update-connection"
echo "$UPDATE_OUTPUT"

# Step 5: Check if we can download the LOA-CFA
# Note: In a real scenario, the LOA-CFA might not be immediately available
echo "Attempting to download the LOA-CFA (this may not be available yet)..."
LOA_OUTPUT=$(aws directconnect describe-loa --connection-id "$CONNECTION_ID" 2>&1)
if echo "$LOA_OUTPUT" | grep -i "error" > /dev/null; then
    echo "LOA-CFA not available yet. This is expected for newly created
 connections."
    echo "The LOA-CFA will be available once AWS begins provisioning your
 connection."
else
    LOA_CONTENT=$(echo "$LOA_OUTPUT" | grep -o '"loaContent": "[^"]*' | cut -d'"' -
f4)
    echo "$LOA_CONTENT" | base64 --decode > "loa-cfa-${CONNECTION_ID}.pdf"
    echo "LOA-CFA downloaded to loa-cfa-${CONNECTION_ID}.pdf"
fi

# Step 6: Create a virtual private gateway (required for private virtual interface)
echo "Creating a virtual private gateway..."
VGW_OUTPUT=$(aws ec2 create-vpn-gateway --type ipsec.1 \
    --tag-specifications 'ResourceType=vpn-gateway,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=aws-direct-connect-gs}]')
check_error "$VGW_OUTPUT" "create-vpn-gateway"
echo "$VGW_OUTPUT"

# Extract VGW ID directly from the output
VGW_ID=$(echo "$VGW_OUTPUT" | grep -o '"VpnGatewayId": "[^"]*' | cut -d'"' -f4)
if [ -z "$VGW_ID" ]; then
    echo "Error: Could not extract VPN gateway ID from the output."
```

```
    exit 1
fi
echo "Virtual private gateway created with ID: $VGW_ID"

# Wait for VGW to become available
if ! wait_for_vgw "$VGW_ID"; then
    echo "Failed to wait for VGW to become available. Skipping virtual interface
 creation."
    VIF_CREATION_SKIPPED=true
else
    VIF_CREATION_SKIPPED=false
fi

# Step 7: Create a private virtual interface (only if VGW is available)
if [ "$VIF_CREATION_SKIPPED" = false ]; then
    echo "Creating a private virtual interface..."
    PRIVATE_VIF_OUTPUT=$(aws directconnect create-private-virtual-interface \
        --connection-id "$CONNECTION_ID" \
        --new-private-virtual-interface '{
            "virtualInterfaceName": "PrivateVIF-'"$RANDOM_ID"'",
            "vlan": 100,
            "asn": 65000,
            "authKey": "'"$RANDOM_ID"'key",
            "amazonAddress": "192.168.1.1/30",
            "customerAddress": "192.168.1.2/30",
            "addressFamily": "ipv4",
            "virtualGatewayId": "'"$VGW_ID"'"
        }' 2>&1)

    if echo "$PRIVATE_VIF_OUTPUT" | grep -i "error" > /dev/null; then
        echo "Could not create private virtual interface. This is expected if the
 connection is not yet available."
        echo "Error: $PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=""
    else
        echo "$PRIVATE_VIF_OUTPUT"
        PRIVATE_VIF_ID=$(echo "$PRIVATE_VIF_OUTPUT" | grep -o '"virtualInterfaceId":
 "[^"]*' | cut -d'"' -f4)
        echo "Private virtual interface created with ID: $PRIVATE_VIF_ID"
    fi
else
    echo "Skipping private virtual interface creation due to VGW not being
 available"
    PRIVATE_VIF_ID=""
```

```
fi

# Step 8: Check connection state and provide guidance for public virtual interface
CONNECTION_STATE=$(aws directconnect describe-connections --connection-id
 "$CONNECTION_ID" --query 'connections[0].connectionState' --output text)
echo "Current connection state: $CONNECTION_STATE"

if [ "$CONNECTION_STATE" != "available" ]; then
    echo ""
    echo "============================================"
    echo "CONNECTION NOT YET AVAILABLE"
    echo "============================================"
    echo "The connection is in '$CONNECTION_STATE' state."
    echo "In production, you would:"
    echo "1. Wait for AWS to provision the connection (can take 30+ minutes)"
    echo "2. Download the LOA-CFA when available"
    echo "3. Provide the LOA-CFA to your network provider for cross-connect"
    echo "4. Create virtual interfaces once connection is 'available'"
    echo ""

    # Ask if user wants to wait for connection to become available
    echo ""
    echo "============================================"
    echo "CONNECTION WAIT CONFIRMATION"
    echo "============================================"
    echo -n "Do you want to wait for the connection to become available? (y/n): "
    read -r WAIT_CHOICE

    if [[ "$WAIT_CHOICE" =~ ^[Yy]$ ]]; then
        if wait_for_connection "$CONNECTION_ID"; then
            echo "Connection is now available! You could now create virtual
 interfaces."
        else
            echo "Connection did not become available within the timeout period."
        fi
    else
        echo "Skipping wait for connection availability."
    fi
else
    echo "Connection is available! Virtual interfaces can be created."
fi

# Step 9: List all virtual interfaces
echo "Listing all virtual interfaces..."
```

```
VIF_LIST_OUTPUT=$(aws directconnect describe-virtual-interfaces)
check_error "$VIF_LIST_OUTPUT" "describe-virtual-interfaces"
echo "$VIF_LIST_OUTPUT"

# Step 10: Display important information about production usage
echo ""
echo "============================================"
echo "IMPORTANT PRODUCTION NOTES"
echo "============================================"
echo "1. Direct Connect connections take time to be provisioned by AWS"
echo "2. You cannot create virtual interfaces until the connection is 'available'"
echo "3. For public virtual interfaces, you must own the public IP addresses"
echo "4. LOA-CFA (Letter of Authorization) is needed for cross-connect at the
 facility"
echo "5. This demo creates resources that incur costs (~\$300/month for 1Gbps)"
echo "6. Always test connectivity before putting into production"
echo ""

# Step 11: Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo -n "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually delete them later."
    echo ""
    echo "Created resources:"
    echo "- Connection ID: $CONNECTION_ID"
    if [ -n "$PRIVATE_VIF_ID" ]; then
        echo "- Private Virtual Interface ID: $PRIVATE_VIF_ID"
    fi
    if [ -n "$PUBLIC_VIF_ID" ]; then
        echo "- Public Virtual Interface ID: $PUBLIC_VIF_ID"
    fi
    echo "- Virtual Private Gateway ID: $VGW_ID"
    echo ""
    echo "Manual cleanup commands:"
    echo "aws directconnect delete-connection --connection-id $CONNECTION_ID"
```

```
    echo "aws ec2 delete-vpn-gateway --vpn-gateway-id $VGW_ID"
    echo ""
    echo "Remember: Direct Connect resources incur ongoing costs!"
fi

echo "$(date): Script completed"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateConnection
  - CreatePrivateVirtualInterface
  - CreateVpnGateway
  - DeleteConnection
  - DeleteVirtualInterface
  - DeleteVpnGateway
  - DescribeConnections
  - DescribeLoa
  - DescribeLocations
  - DescribeVirtualInterfaces
  - DescribeVpnGateways
  - UpdateConnection

Beginnen Sie mit dem Kauf auf dem Software-Marktplatz

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie ec2 AuthorizeSecurityGroupIngress
- Benutze ec2 CreateKeyPair
- Benutze ec2 CreateSecurityGroup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Marketplace Buyer Getting Started Script
# This script demonstrates how to search for products in AWS Marketplace,
# launch an EC2 instance with a product AMI, and manage subscriptions.

set -euo pipefail

# Setup logging with secure permissions
LOG_FILE="marketplace-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS Marketplace Buyer Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. List available products in AWS Marketplace"
echo "2. Create resources needed to launch an EC2 instance"
echo "3. Launch an EC2 instance with an Amazon Linux 2 AMI"
echo "4. Show how to manage and terminate the instance"
echo "====================================================="
echo ""

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it first."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured. Please configure them first."
```

```
    exit 1
fi

# Validate jq is installed
if ! command -v jq &> /dev/null; then
    echo "ERROR: jq is not installed. Please install jq for safe JSON parsing."
    exit 1
fi

# Function to safely extract JSON values using jq
extract_json_value() {
    local json=$1
    local query=$2

    echo "$json" | jq -r "$query" 2>/dev/null || {
        echo "ERROR: Failed to parse JSON with query: $query" >&2
        return 1
    }
}

# Function to validate AWS permissions
validate_aws_permissions() {
    echo "Validating AWS permissions..."

    local identity
    identity=$(aws sts get-caller-identity --output json)
    local account_id
    account_id=$(extract_json_value "$identity" '.Account') || return 1
    local arn
    arn=$(extract_json_value "$identity" '.Arn') || return 1

    echo "AWS Account ID: $account_id"
    echo "AWS Principal ARN: $arn"
    echo "Note: This script requires EC2 permissions for key pair, security group,
 and instance management."
    echo ""
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="
```

```
    if [ -n "${INSTANCE_ID:-}" ]; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true

        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" --region
 "$AWS_REGION" 2>/dev/null || true
        echo "Instance terminated successfully."
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting before deleting security group..."
        sleep 5
        echo "Deleting security group: $SECURITY_GROUP_ID"
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID" --region
 "$AWS_REGION" > /dev/null 2>&1 || true
        echo "Security group deleted."
    fi

    if [ -n "${KEY_NAME:-}" ]; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" --region "$AWS_REGION" > /
dev/null 2>&1 || true

        # Remove the local key file if it exists with secure deletion
        if [ -f "${KEY_NAME}.pem" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "${KEY_NAME}.pem" 2>/dev/null || rm -f
 "${KEY_NAME}.pem"
            else
                rm -f "${KEY_NAME}.pem"
            fi
            echo "Local key file securely deleted."
        fi
    fi

    echo "Cleanup completed."
}

# Set trap to ensure cleanup on script exit
trap cleanup_resources EXIT
```

```
# Get the current AWS region
AWS_REGION=$(aws configure get region || echo "us-east-1")
if [ -z "$AWS_REGION" ] || [ "$AWS_REGION" = "None" ]; then
    AWS_REGION="us-east-1"
fi
echo "Using AWS Region: $AWS_REGION"
echo ""

# Validate permissions
validate_aws_permissions

# Generate random identifier for resource names using cryptographically secure
 method
RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
KEY_NAME="marketplace-key-${RANDOM_ID}"
SECURITY_GROUP_NAME="marketplace-sg-${RANDOM_ID}"

# Initialize variables to track created resources
INSTANCE_ID=""
SECURITY_GROUP_ID=""
AMI_ID=""

# Step 1: List available products in AWS Marketplace
echo "Listing available products in AWS Marketplace..."
echo "Note: In a real scenario, you would use marketplace-catalog commands to list
 and search for products."
echo "However, this requires specific permissions and product knowledge."
echo ""
echo "For this tutorial, we'll use a public Amazon Linux 2 AMI instead of an actual
 marketplace product."
echo "This is because subscribing to marketplace products requires accepting terms
 via the console."
echo ""

# Step 2: Create a key pair for SSH access
echo "Creating key pair: $KEY_NAME"
KEY_OUTPUT=$(aws ec2 create-key-pair \
  --key-name "$KEY_NAME" \
  --region "$AWS_REGION" \
  --query 'KeyMaterial' \
  --output text) || {
    echo "ERROR: Failed to create key pair" >&2
    exit 1
}
```

```
# Securely save the key with restricted permissions
if ! echo "$KEY_OUTPUT" > "${KEY_NAME}.pem" 2>/dev/null; then
    echo "ERROR: Failed to write key file ${KEY_NAME}.pem" >&2
    exit 1
fi
chmod 600 "${KEY_NAME}.pem" || {
    echo "ERROR: Failed to set permissions on key file" >&2
    rm -f "${KEY_NAME}.pem"
    exit 1
}
echo "Key pair created and saved to ${KEY_NAME}.pem with secure permissions (600)"

# Step 3: Create a security group
echo "Creating security group: $SECURITY_GROUP_NAME"
SG_OUTPUT=$(aws ec2 create-security-group \
  --group-name "$SECURITY_GROUP_NAME" \
  --description "Security group for AWS Marketplace tutorial" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to create security group" >&2
    exit 1
}

# Extract security group ID using jq for safe parsing
SECURITY_GROUP_ID=$(extract_json_value "$SG_OUTPUT" '.GroupId') || exit 1
if [ -z "$SECURITY_GROUP_ID" ] || [ "$SECURITY_GROUP_ID" = "null" ]; then
    echo "ERROR: Could not extract security group ID" >&2
    exit 1
fi
echo "Security group created with ID: $SECURITY_GROUP_ID"

# Add inbound rules for SSH and HTTP in parallel for better performance
echo "Configuring security group rules..."
{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 22 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
SSH_PID=$!
```

```
{
    aws ec2 authorize-security-group-ingress \
      --group-id "$SECURITY_GROUP_ID" \
      --protocol tcp \
      --port 80 \
      --cidr 0.0.0.0/0 \
      --region "$AWS_REGION" > /dev/null 2>&1
} &
HTTP_PID=$!

# Wait for both operations to complete
wait $SSH_PID || {
    echo "ERROR: Failed to add SSH ingress rule" >&2
    exit 1
}
wait $HTTP_PID || {
    echo "ERROR: Failed to add HTTP ingress rule" >&2
    exit 1
}

echo "Security group configured with SSH and HTTP access."
echo "WARNING: SSH rule allows access from any IP (0.0.0.0/0). Restrict this in
 production."
echo "WARNING: In a production environment, you should restrict access to specific
 IP ranges."
echo ""

# Step 4: Get the latest Amazon Linux 2 AMI ID - Use pagination to optimize costs
echo "Getting the latest Amazon Linux 2 AMI ID..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-2.0.*-x86_64-gp2"
 "Name=state,Values=available" \
  --region "$AWS_REGION" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text \
  --max-results 50) || {
    echo "ERROR: Failed to describe images" >&2
    exit 1
}

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    echo "ERROR: Could not find a suitable AMI ID" >&2
    exit 1
```

```
fi

echo "Using AMI ID: $AMI_ID"
echo "Note: In a real marketplace scenario, you would use the AMI ID from your
 subscribed product."
echo ""

# Step 5: Launch an EC2 instance with cost optimization
echo "Launching EC2 instance with the AMI..."
echo "Using t2.micro (eligible for AWS Free Tier if applicable)"
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --count 1 \
  --region "$AWS_REGION" \
  --monitoring Enabled=false \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=marketplace-buyer-gs}]' \
  --output json) || {
    echo "ERROR: Failed to launch instance" >&2
    exit 1
}

# Extract instance ID using jq for safe parsing
INSTANCE_ID=$(extract_json_value "$INSTANCE_OUTPUT" '.Instances[0].InstanceId') ||
 exit 1
if [ -z "$INSTANCE_ID" ] || [ "$INSTANCE_ID" = "null" ]; then
    echo "ERROR: Could not extract instance ID" >&2
    exit 1
fi
echo "Instance launched with ID: $INSTANCE_ID"

# Wait for the instance to be running
echo "Waiting for instance to be in running state..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID" --region "$AWS_REGION"
 || {
    echo "ERROR: Instance failed to reach running state" >&2
    exit 1
}
echo "Instance is now running."
echo ""
```

```
# Step 6: Get instance details
echo "Getting instance details..."
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --region "$AWS_REGION" \
  --output json) || {
    echo "ERROR: Failed to describe instance" >&2
    exit 1
}

echo "Instance details:"
extract_json_value "$INSTANCE_DETAILS" '.Reservations[0].Instances[0] | {InstanceId,
 State: .State.Name, PublicDnsName, PrivateIpAddress, LaunchTime, InstanceType}' ||
 exit 1

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Key Pair: $KEY_NAME"
echo "Security Group: $SECURITY_GROUP_NAME (ID: $SECURITY_GROUP_ID)"
echo "EC2 Instance: $INSTANCE_ID"
echo "Instance Type: t2.micro (cost-optimized)"
echo "AMI ID: $AMI_ID"
echo "Region: $AWS_REGION"
echo ""
echo "COST OPTIMIZATION NOTES:"
echo "- t2.micro instances are eligible for AWS Free Tier (750 hours/month for 12
 months)"
echo "- Detailed monitoring is disabled to reduce costs"
echo "- Consider using Spot Instances for non-production workloads"
echo "- Review AWS Pricing Calculator: https://calculator.aws/"
echo ""
echo "To connect to your instance (once it's fully initialized):"
echo "ssh -i ${KEY_NAME}.pem ec2-user@<public-dns-name>"
echo "Replace <public-dns-name> with the PublicDnsName from the instance details
 above."
echo ""

# Auto-confirm cleanup of resources
echo "===================================================="
echo "CLEANUP CONFIRMATION"
echo "===================================================="
```

```
echo "Cleaning up all created resources..."

echo ""
echo "Script completed. See $LOG_FILE for the complete log."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - RunInstances
  - TerminateInstances
  - Wait

Erste Schritte mit großen Datenverarbeitungsclustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
```

```
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
```

```
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
```

```
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession
```

```
def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
```

```
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
```

```
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
```

```
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
```

```
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"
```

```
# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup
```

```
echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - SSH
  - TerminateClusters
  - Wait

Erste Schritte mit Dokumentdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

###############################################################################
# Configuration
###############################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

###############################################################################
# Logging
###############################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""
```

```
##############################################################################
# Region pre-check
##############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
```

```
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Wait for deletion
```

```
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
```

```
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
 rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
 delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
```

```
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi
```

```
SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "============================================"
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
```

```
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi
```

```
CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "==========================================="
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""
```

```
################################################################################
# Step 5: Create DocumentDB instance
################################################################################
echo "============================================="
echo "Step 5: Create DocumentDB instance"
echo "============================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

################################################################################
# Step 6: Get cluster endpoint and security group
################################################################################
echo "============================================="
echo "Step 6: Get cluster endpoint and security group"
echo "============================================="
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
```

```
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

################################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
################################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
```

```
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "============================================"
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================"
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
```

```
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo ""
echo "Resources created:"
```

```
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Erste Schritte mit Graphdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihre Neptune-Datenbank

- Subnetze in mehreren Verfügbarkeitszonen erstellen
- Konfigurieren Sie die Sicherheit für Ihre Neptune-Datenbank
- Erstellen Sie eine Neptune DB-Subnetzgruppe
- Erstellen Sie einen Neptune-DB-Cluster und eine Neptune-DB-Instance
- Fügen Sie Ihrer Graphdatenbank Daten hinzu
- Fragen Sie Ihre Graphdatenbank ab

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi

    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
```

```
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"

# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"
```

```
log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2 |
 tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3 |
 tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
```

```
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' | cut
 -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --
protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
```

```
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
```

```
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name\\\",
 \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or setting
 up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
```

```
    log_cmd "aws neptune delete-db-instance --db-instance-identifier $DB_INSTANCE_ID
 --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID --
skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so we'll
 sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
```

```
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later." |
 tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteInternetGateway
  - DeleteRouteTable
  - DeleteSecurityGroup
  - DeleteSubnet
  - DeleteVpc

- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

Erste Schritte mit In-Memory-Caching

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine Sicherheitsgruppe für den Zugriff ein ElastiCache
- Erstellen Sie einen serverlosen Valkey-Cache
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
```

```
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"
```

```
# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
```

```
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from any
 IP address (0.0.0.0/0)."
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs 2>&1);
 then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
```

```
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' | awk -
F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"

    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi
```

```
if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
```

```
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
        echo "Cache deletion initiated. This may take several minutes to complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
```

```
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Erste Schritte mit Load Balancing

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Application Load Balancers
- Erstellen einer Zielgruppe
- Erstellen eines Listeners
- Überprüfen Ihrer Konfiguration
- Fügen Sie einen HTTPS-Listener hinzu (optional)
- Fügen Sie pfadbasiertes Routing hinzu (optional)
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group
```

```
set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
```

```
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null || true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN" 2>/
dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
```

```
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/null
 && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30 seconds...
 (Attempt $retry_count of $max_retries)"
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID" >&2
            echo "You may need to delete it manually using: aws ec2 delete-security-
group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"
```

```
# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=elastic-load-balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict to
 specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
```

```
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-name,Values=running"
 \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null ||
 echo "")

# Convert space-separated list to array
```

```
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
```

```
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/null
 || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi
```

```
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateListener
  - CreateLoadBalancer
  - CreateSecurityGroup
  - CreateTargetGroup
  - DeleteListener
  - DeleteLoadBalancer
  - DeleteSecurityGroup
  - DeleteTargetGroup
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetHealth
  - DescribeVpcs
  - Hilfe
  - RegisterTargets
  - Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "==============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"
```

```
    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
```

```
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
```

```
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi
```

```
    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."
```

```
    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"
```

```
echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
```

```
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
```

```
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "================================================"
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"
```

```
# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi
```

```
echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
```

```
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"
```

```
# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30
```

```
# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
```

```
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
```

```
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config
```

```
# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT
```

```
chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "================================================"

# Display summary of created resources
echo ""
echo "================================================"
echo "RESOURCE SUMMARY"
echo "================================================"
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "================================================"
echo ""

# Instructions for connecting to the instance and setting up the client
```

```
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
```

```
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi
```

```
echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeCluster
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DescribeSubnets
  - DescribeVpcs
  - DetachRolePolicy
  - GetBootstrapBrokers
  - GetCallerIdentity

- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit Netzwerk-Transit-Gateways

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Transit-Gateways
- Verbinden Sie Ihre VPCs mit Ihrem Transit-Gateway
- Fügen Sie Routen zwischen dem Transit-Gateway und Ihren VPCs hinzu
- Testen Sie das Transit-Gateway
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon VPC Transit Gateway CLI Script
# This script demonstrates how to create a transit gateway and connect two VPCs
# Modified to work with older AWS CLI versions that don't support transit gateway
 wait commands
# Security improved: Added input validation, error handling, and credential checks
```

```
# Error handling
set -euo pipefail
LOG_FILE="transit-gateway-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Security: Check for required AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
  echo "ERROR: AWS credentials not configured or invalid. Please configure AWS
 credentials."
  exit 1
fi

# Function to validate AWS CLI output
validate_aws_output() {
  local output=$1
  local context=$2

  if [ -z "$output" ] || [ "$output" = "None" ]; then
    echo "ERROR: Failed to retrieve $context from AWS API"
    return 1
  fi
}

# Function to validate CIDR blocks
validate_cidr() {
  local cidr=$1
  if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
    echo "ERROR: Invalid CIDR block format: $cidr"
    return 1
  fi
}

# Function to wait for transit gateway to be available
wait_for_tgw() {
  local tgw_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway $tgw_id to become available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateways \
      --transit-gateway-ids "$tgw_id" \
```

```
      --query "TransitGateways[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway is now available"
      return 0
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway creation failed"
      return 1
    fi

    echo "Waiting for transit gateway to become available. Current state: $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway to become available"
  return 1
}

# Function to wait for transit gateway attachment to be available
wait_for_tgw_attachment() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to become
 available..."

  while [ $attempt -lt $max_attempts ]; do
    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "failed")

    echo "Current status: $status"

    if [ "$status" = "available" ]; then
      echo "Transit Gateway Attachment is now available"
      return 0
```

```
    fi

    if [ "$status" = "failed" ]; then
      echo "ERROR: Transit Gateway Attachment creation failed"
      return 1
    fi

    echo "Waiting for transit gateway attachment to become available. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "ERROR: Timeout waiting for transit gateway attachment to become available"
  return 1
}

# Function to wait for transit gateway attachment to be deleted
wait_for_tgw_attachment_deleted() {
  local attachment_id=$1
  local max_attempts=60
  local attempt=0

  echo "Waiting for Transit Gateway Attachment $attachment_id to be deleted..."

  while [ $attempt -lt $max_attempts ]; do
    count=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --filters "Name=transit-gateway-attachment-id,Values=$attachment_id" \
      --query "length(TransitGatewayVpcAttachments)" \
      --output text 2>/dev/null || echo "0")

    if [ "$count" = "0" ]; then
      echo "Transit Gateway Attachment has been deleted"
      return 0
    fi

    status=$(aws ec2 describe-transit-gateway-vpc-attachments \
      --transit-gateway-attachment-ids "$attachment_id" \
      --query "TransitGatewayVpcAttachments[0].State" \
      --output text 2>/dev/null || echo "deleted")

    if [ "$status" = "deleted" ] || [ "$status" = "deleting" ]; then
      echo "Transit Gateway Attachment is being deleted. Current state: $status"
    fi
```

```
    echo "Waiting for transit gateway attachment to be deleted. Current state:
 $status"
    sleep 10
    ((attempt++))
  done

  echo "WARNING: Timeout waiting for transit gateway attachment to be deleted"
  return 0
}

# Function to clean up resources
cleanup() {
  local exit_code=$?
  echo "Error occurred (exit code: $exit_code). Cleaning up resources..."

  # Delete resources in reverse order
  if [ -n "${TGW_ATTACHMENT_1_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 1: $TGW_ATTACHMENT_1_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
  fi

  if [ -n "${TGW_ATTACHMENT_2_ID:-}" ]; then
    echo "Deleting Transit Gateway VPC Attachment 2: $TGW_ATTACHMENT_2_ID"
    aws ec2 delete-transit-gateway-vpc-attachment \
      --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" &>/dev/null || true
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
  fi

  if [ -n "${TGW_ID:-}" ]; then
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" &>/dev/null ||
 true
  fi

  exit "$exit_code"
}

# Set up trap for error handling
trap cleanup EXIT

echo "=== Amazon VPC Transit Gateway Tutorial ==="
```

```
echo "This script will create a transit gateway and connect two VPCs"
echo ""

# Get a valid availability zone dynamically
echo "Getting available AZ in current region..."
AZ=$(aws ec2 describe-availability-zones \
  --query "AvailabilityZones[0].ZoneName" \
  --output text)
validate_aws_output "$AZ" "availability zone" || exit 1
echo "Using availability zone: $AZ"

# Check if VPCs exist
echo "Checking for existing VPCs..."
VPC1_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC1" \
  --query "Vpcs[0].VpcId" \
  --output text)
VPC2_ID=$(aws ec2 describe-vpcs \
  --filters "Name=tag:Name,Values=VPC2" \
  --query "Vpcs[0].VpcId" \
  --output text)

if [ "$VPC1_ID" = "None" ] || [ -z "$VPC1_ID" ]; then
  echo "Creating VPC1..."
  VPC1_ID=$(aws ec2 create-vpc \
    --cidr-block 10.1.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC1_ID" "VPC1" || exit 1
  echo "Created VPC1: $VPC1_ID"

  # Create a subnet in VPC1
  echo "Creating subnet in VPC1..."
  SUBNET1_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC1_ID" \
    --cidr-block 10.1.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Subnet.SubnetId \
    --output text)
  validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
```

```
  echo "Created subnet in VPC1: $SUBNET1_ID"
else
  echo "Using existing VPC1: $VPC1_ID"
  SUBNET1_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC1_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET1_ID" = "None" ] || [ -z "$SUBNET1_ID" ]; then
    echo "Creating subnet in VPC1..."
    SUBNET1_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC1_ID" \
      --cidr-block 10.1.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET1_ID" "VPC1 subnet" || exit 1
    echo "Created subnet in VPC1: $SUBNET1_ID"
  else
    echo "Using existing subnet in VPC1: $SUBNET1_ID"
  fi
fi

if [ "$VPC2_ID" = "None" ] || [ -z "$VPC2_ID" ]; then
  echo "Creating VPC2..."
  VPC2_ID=$(aws ec2 create-vpc \
    --cidr-block 10.2.0.0/16 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Vpc.VpcId \
    --output text)
  validate_aws_output "$VPC2_ID" "VPC2" || exit 1
  echo "Created VPC2: $VPC2_ID"

  # Create a subnet in VPC2
  echo "Creating subnet in VPC2..."
  SUBNET2_ID=$(aws ec2 create-subnet \
    --vpc-id "$VPC2_ID" \
    --cidr-block 10.2.0.0/24 \
    --availability-zone "$AZ" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
    --query Subnet.SubnetId \
```

```
    --output text)
  validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
  echo "Created subnet in VPC2: $SUBNET2_ID"
else
  echo "Using existing VPC2: $VPC2_ID"
  SUBNET2_ID=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$VPC2_ID" \
    --query "Subnets[0].SubnetId" \
    --output text)
  if [ "$SUBNET2_ID" = "None" ] || [ -z "$SUBNET2_ID" ]; then
    echo "Creating subnet in VPC2..."
    SUBNET2_ID=$(aws ec2 create-subnet \
      --vpc-id "$VPC2_ID" \
      --cidr-block 10.2.0.0/24 \
      --availability-zone "$AZ" \
      --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=transitgateway-gettingstarted}]' \
      --query Subnet.SubnetId \
      --output text)
    validate_aws_output "$SUBNET2_ID" "VPC2 subnet" || exit 1
    echo "Created subnet in VPC2: $SUBNET2_ID"
  else
    echo "Using existing subnet in VPC2: $SUBNET2_ID"
  fi
fi

# Get route tables for each VPC
RTB1_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC1_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB1_ID" "VPC1 route table" || exit 1

RTB2_ID=$(aws ec2 describe-route-tables \
  --filters "Name=vpc-id,Values=$VPC2_ID" \
  --query "RouteTables[0].RouteTableId" \
  --output text)
validate_aws_output "$RTB2_ID" "VPC2 route table" || exit 1

echo "Route table for VPC1: $RTB1_ID"
echo "Route table for VPC2: $RTB2_ID"

# Step 1: Create the transit gateway
echo "Creating Transit Gateway..."
```

```
TGW_ID=$(aws ec2 create-transit-gateway \
  --description "My Transit Gateway" \
  --options
 "AmazonSideAsn=64512,AutoAcceptSharedAttachments=disable,DefaultRouteTableAssociation=enabl
 \
  --tag-specifications 'ResourceType=transit-
gateway,Tags=[{Key=Name,Value=MyTransitGateway},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGateway.TransitGatewayId \
  --output text)
validate_aws_output "$TGW_ID" "Transit Gateway" || exit 1
echo "Created Transit Gateway: $TGW_ID"

# Wait for the transit gateway to become available
wait_for_tgw "$TGW_ID" || exit 1

# Step 2: Attach VPCs to the transit gateway
echo "Attaching VPC1 to Transit Gateway..."
TGW_ATTACHMENT_1_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC1_ID" \
  --subnet-ids "$SUBNET1_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC1-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_1_ID" "Transit Gateway VPC1 Attachment" || exit
 1
echo "Created Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"

echo "Attaching VPC2 to Transit Gateway..."
TGW_ATTACHMENT_2_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC2_ID" \
  --subnet-ids "$SUBNET2_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC2-Attachment},{Key=project,Value=doc-smith},
{Key=tutorial,Value=transitgateway-gettingstarted}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)
validate_aws_output "$TGW_ATTACHMENT_2_ID" "Transit Gateway VPC2 Attachment" || exit
 1
echo "Created Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
```

```
# Wait for the attachments to become available
wait_for_tgw_attachment "$TGW_ATTACHMENT_1_ID" || exit 1
wait_for_tgw_attachment "$TGW_ATTACHMENT_2_ID" || exit 1

# Step 3: Add routes between the transit gateway and VPCs
echo "Adding route from VPC1 to VPC2 via Transit Gateway..."
validate_cidr "10.2.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB1_ID" \
  --destination-cidr-block 10.2.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Adding route from VPC2 to VPC1 via Transit Gateway..."
validate_cidr "10.1.0.0/16" || exit 1
aws ec2 create-route \
  --route-table-id "$RTB2_ID" \
  --destination-cidr-block 10.1.0.0/16 \
  --transit-gateway-id "$TGW_ID" || exit 1

echo "Routes added successfully"

# Step 4: Display information for testing
echo ""
echo "=== Transit Gateway Setup Complete ==="
echo "Transit Gateway ID: $TGW_ID"
echo "VPC1 ID: $VPC1_ID"
echo "VPC2 ID: $VPC2_ID"
echo ""
echo "To test connectivity:"
echo "1. Launch an EC2 instance in each VPC"
echo "2. Configure security groups to allow ICMP traffic"
echo "3. Connect to one instance and ping the other instance's private IP"
echo ""

echo "Viewing created resources..."

echo ""
echo "=== Resources Created ==="
echo "Transit Gateway: $TGW_ID"
echo "VPC1: $VPC1_ID"
echo "VPC2: $VPC2_ID"
echo "Subnet in VPC1: $SUBNET1_ID"
echo "Subnet in VPC2: $SUBNET2_ID"
```

```
echo "Transit Gateway Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
echo "Transit Gateway Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
echo ""

echo "Starting cleanup..."

# Delete routes
echo "Deleting routes..."
aws ec2 delete-route --route-table-id "$RTB1_ID" --destination-cidr-block
 10.2.0.0/16 || true
aws ec2 delete-route --route-table-id "$RTB2_ID" --destination-cidr-block
 10.1.0.0/16 || true

# Delete transit gateway attachments
echo "Deleting Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_1_ID" || true

echo "Deleting Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
aws ec2 delete-transit-gateway-vpc-attachment \
  --transit-gateway-attachment-id "$TGW_ATTACHMENT_2_ID" || true

# Wait for attachments to be deleted
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true

# Delete transit gateway
echo "Deleting Transit Gateway: $TGW_ID"
aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" || true

echo "Cleanup completed successfully"

echo "Tutorial completed. See $LOG_FILE for detailed logs."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateRoute
  - CreateSubnet
  - CreateTransitGateway
  - CreateTransitGatewayVpcAttachment
  - CreateVpc
  - DeleteRoute

- DeleteTransitGateway
- DeleteTransitGatewayVpcAttachment
- DescribeAvailabilityZones
- DescribeRouteTables
- DescribeSubnets
- DescribeTransitGatewayAttachments
- DescribeTransitGatewayVpcAttachments
- DescribeTransitGateways
- DescribeVpcs

Erste Schritte mit dem Streaming von Videoübertragungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Zugriff auf Elemental MediaConnect
- Erstellen Sie einen Flow
- Fügen Sie eine Ausgabe hinzu
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
```

```
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/dev/
null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
```

```
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"' -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
```

```
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output" ==
 "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
```

```
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi
```

```
# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi

# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1 |
 cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
```

```
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-Z0-9:-]+
$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/dev/
null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
```

```
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o '"EntitlementArn":
 "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN" --
query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
```

```
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Erste Schritte mit virtuellen Maschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Schlüsselpaares
- Eine Sicherheitsgruppe erstellen
- Anhalten und Starten Ihrer Instance
- Testen Sie die Elastic IP-Persistenz
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}

# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi
```

```
  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
```

```
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
resources may not have been deleted properly."
```

```
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial' --
output text)

if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
```

```
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value" \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)
```

```
if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' 'ResourceType=volume,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
```

```
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
```

```
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi
```

```
log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
```

```
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstances
  - DisassociateAddress
  - GetParametersByPath
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - Wait

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Versuchsvorlage
- Führen Sie das Experiment aus
- Überprüfe die Ergebnisse
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
```

```
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true
```

```
        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
```

```
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
```

```
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
```

```
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"
```

```
# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
```

```
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
```

```
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""
```

```
while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
```

```
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
```

```
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
```

```
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
```

```
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances
  - Wait

Virtuelles privates Netzwerk mit IP-Adressverwaltung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Amazon VPC IP Address Manager (IPAM) mit der CLI ein und konfigurieren Sie ihn.
- Erstellen Sie einen IPAM mit Betriebsregionen (z. B. us-east-1, us-west-2).
- Rufen Sie die private Bereichs-ID für den IPAM ab.
- Erstellen Sie eine hierarchische Struktur von IPv4-Pools (Pools der obersten Ebene, regionale Pools und Entwicklungspools).
- Stellen Sie CIDR-Blöcke für jeden Pool bereit (z. B. 10.0.0. 0/8, 10.0.0. 0/16, 10.0.0. 0/24).
- Erstellen Sie eine VPC mit einem CIDR, das aus einem IPAM-Pool zugewiesen wird.
- Überprüfen Sie die IPAM-Poolzuweisungen und die VPC-Erstellung.
- Beheben Sie häufig auftretende Probleme wie Berechtigungsfehler, Fehler bei der CIDR-Zuweisung und Verstöße gegen Abhängigkeiten.
- Bereinigen Sie die IPAM-Ressourcen (VPC, Pools, CIDRs und IPAM), um unnötige Gebühren zu vermeiden.
- Erkunden Sie die nächsten Schritte für erweiterte IPAM-Features.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# IPAM Getting Started CLI Script - Version 7
# This script creates an IPAM, creates a hierarchy of IP address pools, and
 allocates a CIDR to a VPC
# Fixed to correctly identify the private scope ID, wait for resources to be
 available, add locale to development pool,
# use the correct parameter names for VPC creation, and wait for CIDR provisioning
 to complete
```

```
# Set up logging
LOG_FILE="ipam_script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting IPAM setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED:"
    echo "============================================"

    if [ -n "$VPC_ID" ]; then
        echo "VPC: $VPC_ID"
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Development Pool: $DEV_POOL_ID"
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Regional Pool: $REGIONAL_POOL_ID"
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Top-level Pool: $TOP_POOL_ID"
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "IPAM: $IPAM_ID"
    fi

    echo ""
    echo "============================================"
```

```
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."

        # Delete resources in reverse order of creation to handle dependencies

        if [ -n "$VPC_ID" ]; then
            echo "Deleting VPC: $VPC_ID"
            aws ec2 delete-vpc --vpc-id "$VPC_ID" || echo "Failed to delete VPC"
            echo "Waiting for VPC to be deleted..."
            sleep 10
        fi

        if [ -n "$DEV_POOL_ID" ]; then
            echo "Deleting Development Pool: $DEV_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$DEV_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Development Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$DEV_POOL_ID" --
cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$DEV_POOL_ID" || echo "Failed
 to delete Development Pool"
            echo "Waiting for Development Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$REGIONAL_POOL_ID" ]; then
            echo "Deleting Regional Pool: $REGIONAL_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$REGIONAL_POOL_ID"
 --query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Regional Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id
 "$REGIONAL_POOL_ID" --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
```

```
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$REGIONAL_POOL_ID" || echo
 "Failed to delete Regional Pool"
            echo "Waiting for Regional Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$TOP_POOL_ID" ]; then
            echo "Deleting Top-level Pool: $TOP_POOL_ID"
            # First deprovision any CIDRs from the pool
            CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$TOP_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
            for CIDR in $CIDRS; do
                echo "Deprovisioning CIDR $CIDR from Top-level Pool"
                aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$TOP_POOL_ID" --
cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
                sleep 5
            done
            aws ec2 delete-ipam-pool --ipam-pool-id "$TOP_POOL_ID" || echo "Failed
 to delete Top-level Pool"
            echo "Waiting for Top-level Pool to be deleted..."
            sleep 10
        fi

        if [ -n "$IPAM_ID" ]; then
            echo "Deleting IPAM: $IPAM_ID"
            aws ec2 delete-ipam --ipam-id "$IPAM_ID" || echo "Failed to delete IPAM"
        fi

        echo "Cleanup completed."
    else
        echo "Cleanup skipped. Resources will remain in your account."
    fi
}

# Function to wait for a pool to be in the 'create-complete' state
wait_for_pool() {
    local pool_id=$1
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for pool $pool_id to be available..."
```

```
    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 describe-ipam-pools --ipam-pool-ids "$pool_id" --query
 'IpamPools[0].State' --output text)

        if [ "$state" = "create-complete" ]; then
            echo "Pool $pool_id is now available (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: Pool $pool_id is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for pool $pool_id to be available"
    return 1
}

# Function to wait for a CIDR to be fully provisioned
wait_for_cidr_provisioning() {
    local pool_id=$1
    local cidr=$2
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for CIDR $cidr to be fully provisioned in pool $pool_id..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$pool_id" --query
 "IpamPoolCidrs[?Cidr=='$cidr'].State" --output text)

        if [ "$state" = "provisioned" ]; then
            echo "CIDR $cidr is now fully provisioned (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: CIDR $cidr is in state: $state.
 Waiting..."
        sleep 10
        ((attempt++))
    done
```

```
    echo "Timed out waiting for CIDR $cidr to be provisioned"
    return 1
}

# Step 1: Create an IPAM
echo "Creating IPAM..."
IPAM_RESULT=$(aws ec2 create-ipam \
    --description "My IPAM" \
    --operating-regions RegionName=us-east-1 RegionName=us-west-2 \
    --tag-specifications 'ResourceType=ipam,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create IPAM"
fi

IPAM_ID=$(echo "$IPAM_RESULT" | grep -o '"IpamId": "[^"]*' | cut -d'"' -f4)
echo "IPAM created with ID: $IPAM_ID"

# Wait for IPAM to be created and available
echo "Waiting for IPAM to be available..."
sleep 20

# Step 2: Get the IPAM Scope ID - FIXED to correctly identify the private scope
echo "Getting IPAM Scope ID..."
SCOPE_RESULT=$(aws ec2 describe-ipams --ipam-id "$IPAM_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to get IPAM details"
fi

# Extract the private scope ID directly from the IPAM details
PRIVATE_SCOPE_ID=$(echo "$SCOPE_RESULT" | grep -o '"PrivateDefaultScopeId": "[^"]*'
 | cut -d'"' -f4)
echo "Private Scope ID: $PRIVATE_SCOPE_ID"

if [ -z "$PRIVATE_SCOPE_ID" ]; then
    handle_error "Failed to get Private Scope ID"
fi

# Step 3: Create a Top-Level IPv4 Pool
echo "Creating Top-level IPv4 Pool..."
TOP_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
```

```
    --address-family ipv4 \
    --description "Top-level pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Top-level Pool"
fi

TOP_POOL_ID=$(echo "$TOP_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -
f4)
echo "Top-level Pool created with ID: $TOP_POOL_ID"

# Wait for the top-level pool to be available
if ! wait_for_pool "$TOP_POOL_ID"; then
    handle_error "Top-level Pool did not become available in time"
fi

# Provision CIDR to the top-level pool
echo "Provisioning CIDR to Top-level Pool..."
TOP_POOL_CIDR="10.0.0.0/8"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$TOP_POOL_ID" \
    --cidr "$TOP_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Top-level Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$TOP_POOL_ID" "$TOP_POOL_CIDR"; then
    handle_error "CIDR provisioning to Top-level Pool did not complete in time"
fi

# Step 4: Create a Regional IPv4 Pool
echo "Creating Regional IPv4 Pool..."
REGIONAL_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$TOP_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Regional pool in us-east-1" \
```

```
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create Regional Pool"
fi

REGIONAL_POOL_ID=$(echo "$REGIONAL_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' |
 cut -d'"' -f4)
echo "Regional Pool created with ID: $REGIONAL_POOL_ID"

# Wait for the regional pool to be available
if ! wait_for_pool "$REGIONAL_POOL_ID"; then
    handle_error "Regional Pool did not become available in time"
fi

# Provision CIDR to the regional pool
echo "Provisioning CIDR to Regional Pool..."
REGIONAL_POOL_CIDR="10.0.0.0/16"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$REGIONAL_POOL_ID" \
    --cidr "$REGIONAL_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Regional Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$REGIONAL_POOL_ID" "$REGIONAL_POOL_CIDR"; then
    handle_error "CIDR provisioning to Regional Pool did not complete in time"
fi

# Step 5: Create a Development IPv4 Pool - FIXED to include locale
echo "Creating Development IPv4 Pool..."
DEV_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$REGIONAL_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Development pool" \
    --tag-specifications 'ResourceType=ipam-pool,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-ipam-gs}]')
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create Development Pool"
fi

DEV_POOL_ID=$(echo "$DEV_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -
f4)
echo "Development Pool created with ID: $DEV_POOL_ID"

# Wait for the development pool to be available
if ! wait_for_pool "$DEV_POOL_ID"; then
    handle_error "Development Pool did not become available in time"
fi

# Provision CIDR to the development pool
echo "Provisioning CIDR to Development Pool..."
DEV_POOL_CIDR="10.0.0.0/24"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$DEV_POOL_ID" \
    --cidr "$DEV_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Development Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$DEV_POOL_ID" "$DEV_POOL_CIDR"; then
    handle_error "CIDR provisioning to Development Pool did not complete in time"
fi

# Step 6: Create a VPC Using an IPAM Pool CIDR - FIXED to use the correct parameter
 names and a smaller netmask length
echo "Creating VPC using IPAM Pool CIDR..."
VPC_RESULT=$(aws ec2 create-vpc \
    --ipv4-ipam-pool-id "$DEV_POOL_ID" \
    --ipv4-netmask-length 26 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=IPAM-VPC},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-ipam-gs}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create VPC"
fi
```

```
VPC_ID=$(echo "$VPC_RESULT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID"

# Step 7: Verify the IPAM Pool Allocation
echo "Verifying IPAM Pool Allocation..."
ALLOCATION_RESULT=$(aws ec2 get-ipam-pool-allocations \
    --ipam-pool-id "$DEV_POOL_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to verify IPAM Pool Allocation"
fi

echo "IPAM Pool Allocation verified:"
echo "$ALLOCATION_RESULT" | grep -A 5 "Allocations"

echo ""
echo "IPAM setup completed successfully!"
echo ""

# Prompt for cleanup
cleanup_resources

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateIpam
  - CreateIpamPool
  - CreateVpc
  - DeleteIpam
  - DeleteIpamPool
  - DeleteVpc
  - DeprovisionIpamPoolCidr
  - DescribeIpamPools
  - DescribeIpams
  - DescribeVpcs
  - GetIpamPoolAllocations

- GetIpamPoolCidrs
- ProvisionIpamPoolCidr

Virtuelles privates Netzwerk mit privaten Servern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC mit privaten Subnetzen und NAT-Gateways mit der CLI.
- Richten Sie die erforderlichen Komponenten wie VPC, Subnetze, Routing-Tabellen und NAT-Gateways ein.
- Konfigurieren Sie Sicherheitsgruppen und IAM-Rollen für ordnungsgemäßen Zugriff und Sicherheit.
- Verwenden Sie CLI-Befehle, um die Erstellung und Konfiguration dieser Ressourcen zu automatisieren.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC with Private Subnets and NAT Gateways (IMDSv2 Compliant Version)
# This script creates a VPC with public and private subnets in two Availability
 Zones,
# NAT gateways, an internet gateway, route tables, a VPC endpoint for S3,
# security groups, a launch template, an Auto Scaling group, and an Application Load
 Balancer.

# Set up logging
LOG_FILE="vpc-private-subnets-nat.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Cleanup function to delete all created resources
cleanup_resources() {
  echo "Cleaning up resources..."
```

```
# Delete Auto Scaling group if it exists
if [ -n "${ASG_NAME:-}" ]; then
  echo "Deleting Auto Scaling group: $ASG_NAME"
  aws autoscaling delete-auto-scaling-group --auto-scaling-group-name "$ASG_NAME"
--force-delete
  echo "Waiting for Auto Scaling group to be deleted..."
  aws autoscaling wait auto-scaling-groups-deleted --auto-scaling-group-names
"$ASG_NAME"
fi

# Delete load balancer if it exists
if [ -n "${LB_ARN:-}" ]; then
  echo "Deleting load balancer: $LB_ARN"
  aws elbv2 delete-load-balancer --load-balancer-arn "$LB_ARN"
  # Wait for load balancer to be deleted
  sleep 30
fi

# Delete target group if it exists
if [ -n "${TARGET_GROUP_ARN:-}" ]; then
  echo "Deleting target group: $TARGET_GROUP_ARN"
  aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN"
fi

# Delete launch template if it exists
if [ -n "${LAUNCH_TEMPLATE_NAME:-}" ]; then
  echo "Deleting launch template: $LAUNCH_TEMPLATE_NAME"
  aws ec2 delete-launch-template --launch-template-name "$LAUNCH_TEMPLATE_NAME"
fi

# Delete NAT Gateways if they exist
if [ -n "${NAT_GW1_ID:-}" ]; then
  echo "Deleting NAT Gateway 1: $NAT_GW1_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW1_ID"
fi

if [ -n "${NAT_GW2_ID:-}" ]; then
  echo "Deleting NAT Gateway 2: $NAT_GW2_ID"
  aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW2_ID"
fi

# Wait for NAT Gateways to be deleted
if [ -n "${NAT_GW1_ID:-}" ] || [ -n "${NAT_GW2_ID:-}" ]; then
```

```
    echo "Waiting for NAT Gateways to be deleted..."
    sleep 60
  fi

  # Release Elastic IPs if they exist
  if [ -n "${EIP1_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 1: $EIP1_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP1_ALLOC_ID"
  fi

  if [ -n "${EIP2_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 2: $EIP2_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP2_ALLOC_ID"
  fi

  # Delete VPC endpoint if it exists
  if [ -n "${VPC_ENDPOINT_ID:-}" ]; then
    echo "Deleting VPC endpoint: $VPC_ENDPOINT_ID"
    aws ec2 delete-vpc-endpoints --vpc-endpoint-ids "$VPC_ENDPOINT_ID"
  fi

  # Delete security groups if they exist
  if [ -n "${APP_SG_ID:-}" ]; then
    echo "Deleting application security group: $APP_SG_ID"
    aws ec2 delete-security-group --group-id "$APP_SG_ID"
  fi

  if [ -n "${LB_SG_ID:-}" ]; then
    echo "Deleting load balancer security group: $LB_SG_ID"
    aws ec2 delete-security-group --group-id "$LB_SG_ID"
  fi

  # Detach and delete Internet Gateway if it exists
  if [ -n "${IGW_ID:-}" ] && [ -n "${VPC_ID:-}" ]; then
    echo "Detaching Internet Gateway: $IGW_ID from VPC: $VPC_ID"
    aws ec2 detach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID"
    echo "Deleting Internet Gateway: $IGW_ID"
    aws ec2 delete-internet-gateway --internet-gateway-id "$IGW_ID"
  fi

  # Delete route table associations and route tables if they exist
  if [ -n "${PUBLIC_RT_ASSOC1_ID:-}" ]; then
    echo "Disassociating public route table from subnet 1: $PUBLIC_RT_ASSOC1_ID"
```

```
  aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC1_ID"
fi

if [ -n "${PUBLIC_RT_ASSOC2_ID:-}" ]; then
  echo "Disassociating public route table from subnet 2: $PUBLIC_RT_ASSOC2_ID"
  aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC2_ID"
fi

if [ -n "${PRIVATE_RT1_ASSOC_ID:-}" ]; then
  echo "Disassociating private route table 1: $PRIVATE_RT1_ASSOC_ID"
  aws ec2 disassociate-route-table --association-id "$PRIVATE_RT1_ASSOC_ID"
fi

if [ -n "${PRIVATE_RT2_ASSOC_ID:-}" ]; then
  echo "Disassociating private route table 2: $PRIVATE_RT2_ASSOC_ID"
  aws ec2 disassociate-route-table --association-id "$PRIVATE_RT2_ASSOC_ID"
fi

if [ -n "${PUBLIC_RT_ID:-}" ]; then
  echo "Deleting public route table: $PUBLIC_RT_ID"
  aws ec2 delete-route-table --route-table-id "$PUBLIC_RT_ID"
fi

if [ -n "${PRIVATE_RT1_ID:-}" ]; then
  echo "Deleting private route table 1: $PRIVATE_RT1_ID"
  aws ec2 delete-route-table --route-table-id "$PRIVATE_RT1_ID"
fi

if [ -n "${PRIVATE_RT2_ID:-}" ]; then
  echo "Deleting private route table 2: $PRIVATE_RT2_ID"
  aws ec2 delete-route-table --route-table-id "$PRIVATE_RT2_ID"
fi

# Delete subnets if they exist
if [ -n "${PUBLIC_SUBNET1_ID:-}" ]; then
  echo "Deleting public subnet 1: $PUBLIC_SUBNET1_ID"
  aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET1_ID"
fi

if [ -n "${PUBLIC_SUBNET2_ID:-}" ]; then
  echo "Deleting public subnet 2: $PUBLIC_SUBNET2_ID"
  aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET2_ID"
fi
```

```
  if [ -n "${PRIVATE_SUBNET1_ID:-}" ]; then
    echo "Deleting private subnet 1: $PRIVATE_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET1_ID"
  fi

  if [ -n "${PRIVATE_SUBNET2_ID:-}" ]; then
    echo "Deleting private subnet 2: $PRIVATE_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET2_ID"
  fi

  # Delete VPC if it exists
  if [ -n "${VPC_ID:-}" ]; then
    echo "Deleting VPC: $VPC_ID"
    aws ec2 delete-vpc --vpc-id "$VPC_ID"
  fi

  echo "Cleanup completed."
}

# Error handling function
handle_error() {
  echo "ERROR: $1"
  echo "Attempting to clean up resources..."
  cleanup_resources
  exit 1
}

# Function to check command success
check_command() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Create VPC
echo "Creating VPC..."
VPC_RESULT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC"
```

```
VPC_ID=$(echo "$VPC_RESULT" | jq -r '.Vpc.VpcId')
echo "VPC created with ID: $VPC_ID"

# Get Availability Zones
echo "Getting Availability Zones..."
AZ_RESULT=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0:2].ZoneName' --output text)
check_command "Failed to get Availability Zones"

# Convert space-separated output to array
read -r -a AZS <<< "$AZ_RESULT"
AZ1=${AZS[0]}
AZ2=${AZS[1]}
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create subnets
echo "Creating subnets..."
PUBLIC_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.0.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 1"
PUBLIC_SUBNET1_ID=$(echo "$PUBLIC_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.1.0/24 --availability-zone "$AZ1" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 1"
PRIVATE_SUBNET1_ID=$(echo "$PRIVATE_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PUBLIC_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.2.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create public subnet 2"
PUBLIC_SUBNET2_ID=$(echo "$PUBLIC_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.3.0/24 --availability-zone "$AZ2" --tag-specifications
 "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create private subnet 2"
```

```
PRIVATE_SUBNET2_ID=$(echo "$PRIVATE_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

echo "Subnets created with IDs:"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_RESULT=$(aws ec2 create-internet-gateway --tag-specifications
 "ResourceType=internet-gateway,Tags=[{Key=Name,Value=ProductionIGW-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create Internet Gateway"
IGW_ID=$(echo "$IGW_RESULT" | jq -r '.InternetGateway.InternetGatewayId')
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id "$VPC_ID"
check_command "Failed to attach Internet Gateway to VPC"

# Create route tables
echo "Creating route tables..."
PUBLIC_RT_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create public route table"
PUBLIC_RT_ID=$(echo "$PUBLIC_RT_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT1_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable1-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create private route table 1"
PRIVATE_RT1_ID=$(echo "$PRIVATE_RT1_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT2_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PrivateRouteTable2-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create private route table 2"
PRIVATE_RT2_ID=$(echo "$PRIVATE_RT2_RESULT" | jq -r '.RouteTable.RouteTableId')
```

```
echo "Route tables created with IDs:"
echo "Public Route Table: $PUBLIC_RT_ID"
echo "Private Route Table 1: $PRIVATE_RT1_ID"
echo "Private Route Table 2: $PRIVATE_RT2_ID"

# Add route to Internet Gateway in public route table
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT_ID" --destination-cidr-block
 0.0.0.0/0 --gateway-id "$IGW_ID"
check_command "Failed to add route to Internet Gateway"

# Associate subnets with route tables
echo "Associating subnets with route tables..."
PUBLIC_RT_ASSOC1_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET1_ID")
check_command "Failed to associate public subnet 1 with route table"
PUBLIC_RT_ASSOC1_ID=$(echo "$PUBLIC_RT_ASSOC1_RESULT" | jq -r '.AssociationId')

PUBLIC_RT_ASSOC2_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET2_ID")
check_command "Failed to associate public subnet 2 with route table"
PUBLIC_RT_ASSOC2_ID=$(echo "$PUBLIC_RT_ASSOC2_RESULT" | jq -r '.AssociationId')

PRIVATE_RT1_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT1_ID" --subnet-id "$PRIVATE_SUBNET1_ID")
check_command "Failed to associate private subnet 1 with route table"
PRIVATE_RT1_ASSOC_ID=$(echo "$PRIVATE_RT1_ASSOC_RESULT" | jq -r '.AssociationId')

PRIVATE_RT2_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
 "$PRIVATE_RT2_ID" --subnet-id "$PRIVATE_SUBNET2_ID")
check_command "Failed to associate private subnet 2 with route table"
PRIVATE_RT2_ASSOC_ID=$(echo "$PRIVATE_RT2_ASSOC_RESULT" | jq -r '.AssociationId')

echo "Route table associations created with IDs:"
echo "Public Subnet 1 Association: $PUBLIC_RT_ASSOC1_ID"
echo "Public Subnet 2 Association: $PUBLIC_RT_ASSOC2_ID"
echo "Private Subnet 1 Association: $PRIVATE_RT1_ASSOC_ID"
echo "Private Subnet 2 Association: $PRIVATE_RT2_ASSOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateways..."

# Allocate Elastic IPs for NAT Gateways
```

```
echo "Allocating Elastic IPs for NAT Gateways..."
EIP1_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 1"
EIP1_ALLOC_ID=$(echo "$EIP1_RESULT" | jq -r '.AllocationId')

EIP2_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
 "ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to allocate Elastic IP 2"
EIP2_ALLOC_ID=$(echo "$EIP2_RESULT" | jq -r '.AllocationId')

echo "Elastic IPs allocated with IDs:"
echo "EIP 1 Allocation ID: $EIP1_ALLOC_ID"
echo "EIP 2 Allocation ID: $EIP2_ALLOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateway in public subnet 1..."
NAT_GW1_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET1_ID" --allocation-id "$EIP1_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 1"
NAT_GW1_ID=$(echo "$NAT_GW1_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "Creating NAT Gateway in public subnet 2..."
NAT_GW2_RESULT=$(aws ec2 create-nat-gateway --subnet-id
 "$PUBLIC_SUBNET2_ID" --allocation-id "$EIP2_ALLOC_ID" --tag-specifications
 "ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2-$RANDOM_ID},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create NAT Gateway 2"
NAT_GW2_ID=$(echo "$NAT_GW2_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "NAT Gateways created with IDs:"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"

# Wait for NAT Gateways to be available
echo "Waiting for NAT Gateways to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW1_ID"
check_command "NAT Gateway 1 did not become available"
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW2_ID"
check_command "NAT Gateway 2 did not become available"
```

```
echo "NAT Gateways are now available"

# Add routes to NAT Gateways in private route tables
echo "Adding routes to NAT Gateways in private route tables..."
aws ec2 create-route --route-table-id "$PRIVATE_RT1_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW1_ID"
check_command "Failed to add route to NAT Gateway 1"

aws ec2 create-route --route-table-id "$PRIVATE_RT2_ID" --destination-cidr-block
 0.0.0.0/0 --nat-gateway-id "$NAT_GW2_ID"
check_command "Failed to add route to NAT Gateway 2"

# Create VPC Endpoint for S3
echo "Creating VPC Endpoint for S3..."
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
 'PrefixLists[0].PrefixListId' --output text)
check_command "Failed to get S3 prefix list ID"

VPC_ENDPOINT_RESULT=$(aws ec2 create-vpc-endpoint --vpc-id "$VPC_ID" --
service-name "com.amazonaws.$(aws configure get region).s3" --route-table-ids
 "$PRIVATE_RT1_ID" "$PRIVATE_RT2_ID" --tag-specifications "ResourceType=vpc-
endpoint,Tags=[{Key=Name,Value=S3-Endpoint-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create VPC endpoint for S3"
VPC_ENDPOINT_ID=$(echo "$VPC_ENDPOINT_RESULT" | jq -r '.VpcEndpoint.VpcEndpointId')
echo "VPC Endpoint created with ID: $VPC_ENDPOINT_ID"

# Create security groups
echo "Creating security groups..."
LB_SG_RESULT=$(aws ec2 create-security-group --group-name "LoadBalancerSG-
$RANDOM_ID" --description "Security group for the load balancer"
 --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG-$RANDOM_ID},{Key=project,Value=doc-
smith},{Key=tutorial,Value=vpc-private-servers-gs}]")
check_command "Failed to create load balancer security group"
LB_SG_ID=$(echo "$LB_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from anywhere to the load balancer
aws ec2 authorize-security-group-ingress --group-id "$LB_SG_ID" --protocol tcp --
port 80 --cidr 0.0.0.0/0
check_command "Failed to authorize ingress to load balancer security group"
```

```
APP_SG_RESULT=$(aws ec2 create-security-group --group-name "AppServerSG-$RANDOM_ID"
 --description "Security group for the application servers" --vpc-id "$VPC_ID" --
tag-specifications "ResourceType=security-group,Tags=[{Key=Name,Value=AppServerSG-
$RANDOM_ID},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-private-servers-
gs}]")
check_command "Failed to create application server security group"
APP_SG_ID=$(echo "$APP_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from the load balancer security group to the
 application servers
aws ec2 authorize-security-group-ingress --group-id "$APP_SG_ID" --protocol tcp --
port 80 --source-group "$LB_SG_ID"
check_command "Failed to authorize ingress to application server security group"

echo "Security groups created with IDs:"
echo "Load Balancer Security Group: $LB_SG_ID"
echo "Application Server Security Group: $APP_SG_ID"

# Create a launch template
echo "Creating launch template..."

# Create user data script with IMDSv2 support
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd

# Use IMDSv2 with session token
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
AZ=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/
meta-data/placement/availability-zone)
HOSTNAME=$(hostname -f)

echo "<h1>Hello from $HOSTNAME in $AZ</h1>" > /var/www/html/index.html
EOF

# Encode user data
USER_DATA=$(base64 -w 0 user-data.sh)

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
```

```
AMI_ID=$(aws ec2 describe-images --owners amazon --filters "Name=name,Values=amzn2-
ami-hvm-*-x86_64-gp2" "Name=state,Values=available" --query 'sort_by(Images,
 &CreationDate)[-1].ImageId' --output text)
check_command "Failed to get latest Amazon Linux 2 AMI"
echo "Using AMI: $AMI_ID"

# Create launch template with IMDSv2 required
LAUNCH_TEMPLATE_NAME="AppServerTemplate-$RANDOM_ID"
echo "Creating launch template: $LAUNCH_TEMPLATE_NAME"

aws ec2 create-launch-template \
  --launch-template-name "$LAUNCH_TEMPLATE_NAME" \
  --version-description "Initial version" \
  --tag-specifications "ResourceType=launch-template,Tags=[{Key=Name,Value=
$LAUNCH_TEMPLATE_NAME},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-
private-servers-gs}]" \
  --launch-template-data "{
    \"NetworkInterfaces\": [{
      \"DeviceIndex\": 0,
      \"Groups\": [\"$APP_SG_ID\"],
      \"DeleteOnTermination\": true
    }],
    \"ImageId\": \"$AMI_ID\",
    \"InstanceType\": \"t3.micro\",
    \"UserData\": \"$USER_DATA\",
    \"MetadataOptions\": {
      \"HttpTokens\": \"required\",
      \"HttpEndpoint\": \"enabled\"
    },
    \"TagSpecifications\": [{
      \"ResourceType\": \"instance\",
      \"Tags\": [{
        \"Key\": \"Name\",
        \"Value\": \"AppServer-$RANDOM_ID\"
      }]
    }]
  }"
check_command "Failed to create launch template"

# Create target group
echo "Creating target group..."
TARGET_GROUP_NAME="AppTargetGroup-$RANDOM_ID"
TARGET_GROUP_RESULT=$(aws elbv2 create-target-group \
  --name "$TARGET_GROUP_NAME" \
```

```
  --protocol HTTP \
  --port 80 \
  --vpc-id "$VPC_ID" \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path "/" \
  --health-check-port traffic-port \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create target group"
TARGET_GROUP_ARN=$(echo "$TARGET_GROUP_RESULT" | jq -r
 '.TargetGroups[0].TargetGroupArn')
echo "Target group created with ARN: $TARGET_GROUP_ARN"

# Create load balancer
echo "Creating load balancer..."
LB_NAME="AppLoadBalancer-$RANDOM_ID"
LB_RESULT=$(aws elbv2 create-load-balancer \
  --name "$LB_NAME" \
  --subnets "$PUBLIC_SUBNET1_ID" "$PUBLIC_SUBNET2_ID" \
  --security-groups "$LB_SG_ID" \
  --tags "Key=Name,Value=$LB_NAME" Key=project,Value=doc-smith
 Key=tutorial,Value=vpc-private-servers-gs)
check_command "Failed to create load balancer"
LB_ARN=$(echo "$LB_RESULT" | jq -r '.LoadBalancers[0].LoadBalancerArn')
echo "Load balancer created with ARN: $LB_ARN"

# Wait for load balancer to be active
echo "Waiting for load balancer to be active..."
aws elbv2 wait load-balancer-available --load-balancer-arns "$LB_ARN"
check_command "Load balancer did not become available"

# Create listener
echo "Creating listener..."
LISTENER_RESULT=$(aws elbv2 create-listener \
  --load-balancer-arn "$LB_ARN" \
  --protocol HTTP \
  --port 80 \
  --default-actions "Type=forward,TargetGroupArn=$TARGET_GROUP_ARN")
check_command "Failed to create listener"
LISTENER_ARN=$(echo "$LISTENER_RESULT" | jq -r '.Listeners[0].ListenerArn')
echo "Listener created with ARN: $LISTENER_ARN"

# Create Auto Scaling group
echo "Creating Auto Scaling group..."
```

```
ASG_NAME="AppAutoScalingGroup-$RANDOM_ID"
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name "$ASG_NAME" \
  --launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=\$Latest" \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "$PRIVATE_SUBNET1_ID,$PRIVATE_SUBNET2_ID" \
  --target-group-arns "$TARGET_GROUP_ARN" \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags "Key=Name,Value=AppServer-$RANDOM_ID,PropagateAtLaunch=true"
 Key=project,Value=doc-smith,PropagateAtLaunch=true Key=tutorial,Value=vpc-private-
servers-gs,PropagateAtLaunch=true
check_command "Failed to create Auto Scaling group"
echo "Auto Scaling group created with name: $ASG_NAME"

# Get load balancer DNS name
LB_DNS_NAME=$(aws elbv2 describe-load-balancers --load-balancer-arns "$LB_ARN" --
query 'LoadBalancers[0].DNSName' --output text)
check_command "Failed to get load balancer DNS name"

echo ""
echo "============================================"
echo "DEPLOYMENT COMPLETE"
echo "============================================"
echo "VPC ID: $VPC_ID"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID (AZ: $AZ1)"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID (AZ: $AZ1)"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID (AZ: $AZ2)"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID (AZ: $AZ2)"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"
echo "Load Balancer: $LB_NAME"
echo "Auto Scaling Group: $ASG_NAME"
echo ""
echo "Your application will be available at: http://$LB_DNS_NAME"
echo "It may take a few minutes for the instances to launch and pass health checks."
echo ""

# Add health check monitoring
echo "============================================"
echo "MONITORING INSTANCE HEALTH AND LOAD BALANCER"
echo "============================================"
```

```
echo "Waiting for instances to launch and pass health checks..."
echo "This may take 3-5 minutes. Checking every 30 seconds..."

# Monitor instance health and load balancer accessibility
MAX_ATTEMPTS=10
ATTEMPT=1
HEALTHY_INSTANCES=0

while [ $ATTEMPT -le $MAX_ATTEMPTS ] && [ $HEALTHY_INSTANCES -lt 2 ]; do
  echo "Check attempt $ATTEMPT of $MAX_ATTEMPTS..."

  # Check Auto Scaling group instances
  echo "Checking Auto Scaling group instances..."
  ASG_INSTANCES=$(aws autoscaling describe-auto-scaling-groups --auto-
scaling-group-names "$ASG_NAME" --query 'AutoScalingGroups[0].Instances[*].
[InstanceId,HealthStatus]' --output json)
  echo "ASG Instances status:"
  echo "$ASG_INSTANCES" | jq -r '.[] | "Instance: \(.[0]), Health: \(.[1])"'

  # Check target group health
  echo "Checking target group health..."
  TARGET_HEALTH=$(aws elbv2 describe-target-health --target-group-arn
 "$TARGET_GROUP_ARN" --output json)
  echo "Target health status:"
  echo "$TARGET_HEALTH" | jq -r '.TargetHealthDescriptions[] | "Instance:
 \(.Target.Id), State: \(.TargetHealth.State), Reason: \(.TargetHealth.Reason // "N/
A"), Description: \(.TargetHealth.Description // "N/A")"'

  # Count healthy instances
  HEALTHY_INSTANCES=$(echo "$TARGET_HEALTH" | jq -r '[.TargetHealthDescriptions[] |
 select(.TargetHealth.State=="healthy")] | length')
  echo "Number of healthy instances: $HEALTHY_INSTANCES of 2 expected"

  # Check if we have healthy instances
  if [ $HEALTHY_INSTANCES -ge 2 ]; then
    echo "All instances are healthy!"

    # Test load balancer accessibility
    echo "Testing load balancer accessibility..."
    HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" "http://$LB_DNS_NAME")

    if [ "$HTTP_STATUS" = "200" ]; then
      echo "Load balancer is accessible! HTTP Status: $HTTP_STATUS"
      echo "You can access your application at: http://$LB_DNS_NAME"
```

```
      # Try to get the content to verify IMDSv2 is working
      echo "Fetching content to verify IMDSv2 functionality..."
      CONTENT=$(curl -s "http://$LB_DNS_NAME")
      echo "Response from server:"
      echo "$CONTENT"

      # Check if the content contains the expected pattern
      if [[ "$CONTENT" == *"Hello from"* && "$CONTENT" == *"in"* ]]; then
        echo "IMDSv2 is working correctly! The instance was able to access metadata
 using the token-based approach."
      else
        echo "Warning: Content doesn't match expected pattern. IMDSv2 functionality
 could not be verified."
      fi

      break
    else
      echo "Load balancer returned HTTP status: $HTTP_STATUS"
      echo "Will try again in 30 seconds..."
    fi
  else
    echo "Waiting for instances to become healthy..."
    echo "Will check again in 30 seconds..."
  fi

  ATTEMPT=$((ATTEMPT+1))

  if [ $ATTEMPT -le $MAX_ATTEMPTS ]; then
    sleep 30
  fi
done

if [ $HEALTHY_INSTANCES -lt 2 ]; then
  echo "Warning: Not all instances are healthy after maximum attempts."
  echo "You may need to wait longer or check for configuration issues."
fi

echo "To test your application, run:"
echo "curl http://$LB_DNS_NAME"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  cleanup_resources
  echo "All resources have been deleted."
else
  echo "Resources will not be deleted. You can manually delete them later."
  echo "To delete resources, run this script again and choose to clean up."
fi
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AllocateAddress
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateInternetGateway
  - CreateLaunchTemplate
  - CreateNatGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet
  - CreateVpc
  - CreateVpcEndpoint
  - DeleteAutoScalingGroup
  - DeleteInternetGateway
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteNatGateway
  - DeleteRouteTable
  - DeleteSecurityGroup
  - DeleteSubnet

- DeleteTargetGroup
- DeleteVpc
- DeleteVpcEndpoints
- DescribeAvailabilityZones
- DescribeImages
- DescribePrefixLists
- DetachInternetGateway
- ReleaseAddress

Arbeiten mit Blockspeicherverschlüsselung, Snapshots und Volume-Initialisierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Aktivieren Sie standardmäßig die Amazon EBS-Verschlüsselung
- Erstellen Sie einen EBS-Snapshot
- Erstellen und initialisieren Sie ein Volume aus einem Snapshot
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script for EBS operations: encryption, snapshots, and volume initialization
# This script demonstrates:
# 1. Enabling EBS encryption by default
# 2. Creating an EBS snapshot
# 3. Creating a volume from a snapshot
# Cost optimizations:
# - Reduced volume size from 1 GiB to 100 MiB for testing
# - Changed volume type to gp3 with cost-optimized IOPS/throughput
```

```
# - Added early cleanup to minimize storage duration
# - Removed unnecessary API calls for KMS key retrieval

set -euo pipefail

# Security: Restrict file permissions for log files
umask 0077

# Setup logging with secure permissions
LOG_FILE="ebs-operations-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting EBS operations script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Function to check command status
check_status() {
    local exit_code=$?
    if [ $exit_code -ne 0 ]; then
        echo "ERROR: $1 failed with exit code $exit_code. Exiting."
        cleanup_resources
        exit 1
    fi
}

# Function to cleanup resources with retry logic
cleanup_resources() {
    echo "Attempting to clean up resources..."
    local retry_count=0
    local max_retries=3

    if [ -n "${NEW_VOLUME_ID:-}" ]; then
        echo "Deleting new volume $NEW_VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$NEW_VOLUME_ID" --region
 "$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted new volume $NEW_VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
 $NEW_VOLUME_ID..."
                    sleep 2
                else
```

```
                    echo "WARNING: Could not delete new volume $NEW_VOLUME_ID after
$max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${VOLUME_ID:-}" ]; then
        echo "Deleting original volume $VOLUME_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-volume --volume-id "$VOLUME_ID" --region "$AWS_REGION"
2>/dev/null; then
                echo "Successfully deleted original volume $VOLUME_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
$VOLUME_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete original volume $VOLUME_ID after
$max_retries attempts"
                fi
            fi
        done
    fi

    if [ -n "${SNAPSHOT_ID:-}" ]; then
        echo "Deleting snapshot $SNAPSHOT_ID..."
        for ((retry_count=0; retry_count<max_retries; retry_count++)); do
            if aws ec2 delete-snapshot --snapshot-id "$SNAPSHOT_ID" --region
"$AWS_REGION" 2>/dev/null; then
                echo "Successfully deleted snapshot $SNAPSHOT_ID"
                break
            else
                if [ $retry_count -lt $((max_retries-1)) ]; then
                    echo "Retry $((retry_count+1))/$max_retries for deleting
$SNAPSHOT_ID..."
                    sleep 2
                else
                    echo "WARNING: Could not delete snapshot $SNAPSHOT_ID after
$max_retries attempts"
                fi
            fi
```

```
        done
    fi

    if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
        echo "Restoring original encryption setting..."
        if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
            aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION" 2>/dev/
null || echo "WARNING: Could not restore encryption setting"
        fi
    fi

    echo "Cleanup completed."
}

# Set trap for cleanup on exit
trap cleanup_resources EXIT

# Track created resources
VOLUME_ID=""
NEW_VOLUME_ID=""
SNAPSHOT_ID=""
ENCRYPTION_MODIFIED=false
ORIGINAL_ENCRYPTION=""

# Input validation function
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS credentials are not properly configured"
        exit 1
    fi
}

validate_aws_cli

# Security: Validate AWS region format
validate_region() {
    local region="$1"
    if [[ ! "$region" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
```

```
        echo "ERROR: Invalid AWS region format: $region"
        exit 1
    fi
}

# Get the current AWS region
AWS_REGION="${AWS_REGION:-$(aws configure get region)}"
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config. Using default: $AWS_REGION"
fi

validate_region "$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Security: Validate volume ID format before use
validate_volume_id() {
    local volume_id="$1"
    if [[ ! "$volume_id" =~ ^vol-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid volume ID format: $volume_id"
        exit 1
    fi
}

# Security: Validate snapshot ID format before use
validate_snapshot_id() {
    local snapshot_id="$1"
    if [[ ! "$snapshot_id" =~ ^snap-[a-z0-9]{17}$ ]]; then
        echo "ERROR: Invalid snapshot ID format: $snapshot_id"
        exit 1
    fi
}

# Get availability zones in the region with caching
AVAILABILITY_ZONE=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --
query 'AvailabilityZones[0].ZoneName' --output text)
check_status "Getting availability zone"

# Security: Validate AZ format
if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}[a-z]$ ]]; then
    echo "ERROR: Invalid availability zone format: $AVAILABILITY_ZONE"
    exit 1
fi
echo "Using availability zone: $AVAILABILITY_ZONE"
```

```
# Step 1: Check and enable EBS encryption by default
echo "Step 1: Checking current EBS encryption by default setting..."
ORIGINAL_ENCRYPTION=$(aws ec2 get-ebs-encryption-by-default --region "$AWS_REGION"
 --query 'EbsEncryptionByDefault' --output text)
check_status "Checking encryption status"
echo "Current encryption by default setting: $ORIGINAL_ENCRYPTION"

if [ "$ORIGINAL_ENCRYPTION" = "False" ]; then
    echo "Enabling EBS encryption by default..."
    aws ec2 enable-ebs-encryption-by-default --region "$AWS_REGION"
    check_status "Enabling encryption by default"
    ENCRYPTION_MODIFIED=true
    echo "Updated encryption by default setting: True"
else
    echo "EBS encryption by default is already enabled."
fi

# Step 2: Create a test volume for snapshot with minimal size for cost optimization
echo "Step 2: Creating a test volume (1 GiB for testing)..."
VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --availability-zone
 "$AVAILABILITY_ZONE" --size 1 --volume-type gp3 --iops 3000 --throughput 125
 --tag-specifications 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-
volume},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating test volume"

# Security: Validate volume ID
validate_volume_id "$VOLUME_ID"
echo "Created test volume: $VOLUME_ID"

# Wait for volume to become available with timeout
echo "Waiting for volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$VOLUME_ID" || {
    echo "ERROR: Volume did not become available within timeout"
    exit 1
}
check_status "Waiting for volume"

# Step 3: Create a snapshot of the volume
echo "Step 3: Creating snapshot of the volume..."
SNAPSHOT_ID=$(aws ec2 create-snapshot --region "$AWS_REGION" --volume-id
 "$VOLUME_ID" --description "Snapshot for EBS tutorial - $(date +%Y-%m-%d)" --
```

```
tag-specifications 'ResourceType=snapshot,Tags=[{Key=Name,Value=ebs-tutorial-
snapshot},{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-
smith},{Key=tutorial,Value=ebs-intermediate}]' --query 'SnapshotId' --output text)
check_status "Creating snapshot"

# Security: Validate snapshot ID
validate_snapshot_id "$SNAPSHOT_ID"
echo "Created snapshot: $SNAPSHOT_ID"

# Wait for snapshot to complete with progress indication and timeout
echo "Waiting for snapshot to complete (this may take several minutes)..."
timeout 1800 aws ec2 wait snapshot-completed --region "$AWS_REGION" --snapshot-ids
 "$SNAPSHOT_ID" || {
    echo "ERROR: Snapshot did not complete within timeout"
    exit 1
}
check_status "Waiting for snapshot"
echo "Snapshot completed."

# Step 4: Create a new volume from the snapshot
echo "Step 4: Creating a new volume from the snapshot..."
NEW_VOLUME_ID=$(aws ec2 create-volume --region "$AWS_REGION" --
snapshot-id "$SNAPSHOT_ID" --availability-zone "$AVAILABILITY_ZONE"
 --volume-type gp3 --iops 3000 --throughput 125 --tag-specifications
 'ResourceType=volume,Tags=[{Key=Name,Value=ebs-tutorial-volume-from-snapshot},
{Key=ManagedBy,Value=ebs-intermediate-script},{Key=project,Value=doc-smith},
{Key=tutorial,Value=ebs-intermediate}]' --query 'VolumeId' --output text)
check_status "Creating new volume from snapshot"

# Security: Validate new volume ID
validate_volume_id "$NEW_VOLUME_ID"
echo "Created new volume from snapshot: $NEW_VOLUME_ID"

# Wait for new volume to become available with timeout
echo "Waiting for new volume to become available..."
timeout 300 aws ec2 wait volume-available --region "$AWS_REGION" --volume-ids
 "$NEW_VOLUME_ID" || {
    echo "ERROR: New volume did not become available within timeout"
    exit 1
}
check_status "Waiting for new volume"

# Display created resources
echo ""
```

```
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Original Volume: $VOLUME_ID"
echo "Snapshot: $SNAPSHOT_ID"
echo "New Volume: $NEW_VOLUME_ID"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup process to minimize storage costs..."

# Delete the new volume immediately to reduce storage duration
echo "Deleting new volume $NEW_VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$NEW_VOLUME_ID"
check_status "Deleting new volume"

# Delete the original volume
echo "Deleting original volume $VOLUME_ID..."
aws ec2 delete-volume --region "$AWS_REGION" --volume-id "$VOLUME_ID"
check_status "Deleting original volume"

# Delete the snapshot
echo "Deleting snapshot $SNAPSHOT_ID..."
aws ec2 delete-snapshot --region "$AWS_REGION" --snapshot-id "$SNAPSHOT_ID"
check_status "Deleting snapshot"

# Restore original encryption setting if modified
if [ "${ENCRYPTION_MODIFIED:-false}" = true ]; then
    echo "Restoring original encryption setting..."
    if [ "${ORIGINAL_ENCRYPTION:-}" = "False" ]; then
        aws ec2 disable-ebs-encryption-by-default --region "$AWS_REGION"
        check_status "Disabling encryption by default"
    fi
fi

echo "Cleanup completed successfully."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- CreateSnapshot
- CreateVolume
- DeleteSnapshot
- DeleteVolume
- DescribeAvailabilityZones
- DescribeVolumes
- DetachVolume
- DisableEbsEncryptionByDefault
- EnableEbsEncryptionByDefault
- GetEbsDefaultKmsKeyId
- GetEbsEncryptionByDefault
- Wait

Arbeiten mit Netzwerk-Peering-Verbindungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie VPCs für Peering
- Erstellen einer VPC-Peering-Verbindung
- Aktualisieren von Routing-Tabellen
- Überprüfen Sie die VPC-Peering-Verbindung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Peering Connection Script - Version 6 (Security Enhanced)
```

```
# This script establishes a VPC peering connection between two VPCs,
# creates subnets if needed, and configures the necessary route tables.
# It will use existing VPCs if available, or create new ones if needed.

set -euo pipefail

# Security: Set strict umask
umask 0077

# Initialize log file with restricted permissions
LOG_FILE="./vpc-peering-script-v6.log"
touch "$LOG_FILE"
chmod 0600 "$LOG_FILE"
echo "Starting VPC Peering script at $(date)" > "$LOG_FILE"

# Configuration
declare -r AWS_REGION="${AWS_REGION:-us-east-1}"
declare -r MAX_RETRIES=3
declare -r RETRY_DELAY=5

# Validate script is run from secure location
if [[ "$LOG_FILE" != /* ]] && [[ "$LOG_FILE" != ./* ]]; then
    echo "ERROR: Log file path must be absolute or relative starting with ./" >&2
    exit 1
fi

# Function to sanitize variable for safe command execution
sanitize_var() {
    local var="$1"
    if [[ ! "$var" =~ ^[a-zA-Z0-9_/.-]+$ ]]; then
        echo "ERROR: Invalid characters in variable: $var" | tee -a "$LOG_FILE"
        return 1
    fi
    echo "$var"
    return 0
}

# Function to escape string for safe use in commands
escape_string() {
    local string="$1"
    printf '%s\n' "$string" | sed -e 's/[\/&]/\\&/g'
}

# Function to log commands and their output securely
```

```
log_cmd() {
    local cmd="$1"

    # Validate command doesn't contain suspicious patterns
    if [[ "$cmd" =~ (\$\(|\`|;.*rm|;.*mv|;.*cp) ]]; then
        echo "ERROR: Suspicious command pattern detected" | tee -a "$LOG_FILE"
        return 1
    fi

    echo "$(date): COMMAND: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    local exit_code="$1"
    local error_msg="${2:-Command failed}"
    if [ "$exit_code" -ne 0 ]; then
        echo "ERROR: $error_msg (exit code: $exit_code)" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        cleanup_on_error
        exit "$exit_code"
    fi
}

# Function to validate AWS CLI is available and configured
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
        exit 1
    fi

    # Check AWS CLI version
    local aws_version
    aws_version=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2)
    echo "AWS CLI version: $aws_version" >> "$LOG_FILE"

    if ! aws sts get-caller-identity --region "$AWS_REGION" &>/dev/null; then
        echo "ERROR: AWS CLI is not properly configured or credentials are invalid"
 | tee -a "$LOG_FILE"
        exit 1
    fi
```

```
    # Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null)
    if [[ ! "$account_id" =~ ^[0-9]{12}$ ]]; then
        echo "ERROR: Invalid AWS account ID" | tee -a "$LOG_FILE"
        exit 1
    fi
    echo "Authenticated as AWS Account: $account_id" | tee -a "$LOG_FILE"
}

# Function to validate CIDR blocks
validate_cidr() {
    local cidr="$1"
    if ! [[ "$cidr" =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}/[0-9]{1,2}$ ]]; then
        echo "ERROR: Invalid CIDR block format: $cidr" | tee -a "$LOG_FILE"
        return 1
    fi

    # Additional validation for IP octets
    local ip_part="${cidr%/*}"
    local mask_part="${cidr#*/}"

    IFS='.' read -r -a octets <<< "$ip_part"
    for octet in "${octets[@]}"; do
        if (( octet > 255 )); then
            echo "ERROR: Invalid octet value in CIDR: $cidr" | tee -a "$LOG_FILE"
            return 1
        fi
    done

    if (( mask_part > 32 || mask_part < 0 )); then
        echo "ERROR: Invalid CIDR mask value: $mask_part" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 "$LOG_FILE"
```

```
    # List created resources
    echo "Resources created:" | tee -a "$LOG_FILE"
    for resource in "${CREATED_RESOURCES[@]:-}"; do
        echo "- $resource" | tee -a "$LOG_FILE"
    done

    # Clean up in reverse order with retry logic
    for ((i=${#CLEANUP_COMMANDS[@]}-1; i>=0; i--)); do
        echo "Executing cleanup: ${CLEANUP_COMMANDS[$i]}" >> "$LOG_FILE"
        local retry_count=0
        while [ $retry_count -lt $MAX_RETRIES ]; do
            if eval "${CLEANUP_COMMANDS[$i]}" 2>&1 >> "$LOG_FILE"; then
                break
            else
                retry_count=$((retry_count + 1))
                if [ $retry_count -lt $MAX_RETRIES ]; then
                    echo "Cleanup command failed, retrying in ${RETRY_DELAY}s..." >>
 "$LOG_FILE"
                    sleep "$RETRY_DELAY"
                fi
            fi
        done
    done
}

# Array to store created resources and cleanup commands
declare -a CREATED_RESOURCES=()
declare -a CLEANUP_COMMANDS=()

# Trap errors and cleanup
trap cleanup_on_error EXIT

echo "Setting up VPC peering connection..."

# Validate AWS CLI
validate_aws_cli

# Check for existing VPCs
echo "Checking for existing VPCs..."
EXISTING_VPCS=$(aws ec2 describe-vpcs --region "$AWS_REGION" --query 'Vpcs[?
State==`available`].[VpcId,CidrBlock]' --output text 2>/dev/null || echo "")

if [ -z "$EXISTING_VPCS" ]; then
    echo "No existing VPCs found. Creating new VPCs..."
```

```
    CREATE_VPCS=true
else
    echo "Found existing VPCs:"
    echo "$EXISTING_VPCS"
    echo ""
    echo "Using existing VPCs..."
    CREATE_VPCS=false
    # Get the first two available VPCs
    VPC1_INFO=$(echo "$EXISTING_VPCS" | head -n 1)
    VPC2_INFO=$(echo "$EXISTING_VPCS" | head -n 2 | tail -n 1)

    if [ -z "$VPC2_INFO" ]; then
        echo "Only one VPC found. Creating a second VPC..."
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid VPC1_CIDR
 format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        CREATE_VPC2_ONLY=true
    else
        VPC1_ID=$(echo "$VPC1_INFO" | awk '{print $1}')
        VPC1_CIDR=$(echo "$VPC1_INFO" | awk '{print $2}')
        VPC2_ID=$(echo "$VPC2_INFO" | awk '{print $1}')
        VPC2_CIDR=$(echo "$VPC2_INFO" | awk '{print $2}')

        # Sanitize extracted values
        VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID format"
        VPC1_CIDR=$(sanitize_var "$VPC1_CIDR") || check_error 1 "Invalid VPC1_CIDR
 format"
        VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID format"
        VPC2_CIDR=$(sanitize_var "$VPC2_CIDR") || check_error 1 "Invalid VPC2_CIDR
 format"

        validate_cidr "$VPC1_CIDR" || check_error 1 "Invalid VPC1 CIDR"
        validate_cidr "$VPC2_CIDR" || check_error 1 "Invalid VPC2 CIDR"
        CREATE_VPC2_ONLY=false
    fi
fi

# Create VPCs if needed
```

```
if [ "$CREATE_VPCS" = true ]; then
    echo "Creating VPC1..."
    VPC1_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.1.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC1"
    VPC1_ID=$(sanitize_var "$VPC1_ID") || check_error 1 "Invalid VPC1_ID returned"
    VPC1_CIDR="10.1.0.0/16"
    CREATED_RESOURCES+=("VPC1: $VPC1_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC1_ID'")
    echo "VPC1 created with ID: $VPC1_ID"

    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPCs to be available
    echo "Waiting for VPCs to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID'"
    check_error $? "Timeout waiting for VPCs to become available"

elif [ "$CREATE_VPC2_ONLY" = true ]; then
    echo "Creating VPC2..."
    VPC2_ID=$(log_cmd "aws ec2 create-vpc --region '$AWS_REGION' --cidr-block
 10.2.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2-
Peering-Demo},{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --
query 'Vpc.VpcId' --output text")
    check_error $? "Failed to create VPC2"
    VPC2_ID=$(sanitize_var "$VPC2_ID") || check_error 1 "Invalid VPC2_ID returned"
    VPC2_CIDR="10.2.0.0/16"
    CREATED_RESOURCES+=("VPC2: $VPC2_ID")
```

```
    CLEANUP_COMMANDS+=("aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id
 '$VPC2_ID'")
    echo "VPC2 created with ID: $VPC2_ID"

    # Wait for VPC2 to be available
    echo "Waiting for VPC2 to be available..."
    log_cmd "aws ec2 wait vpc-available --region '$AWS_REGION' --vpc-ids '$VPC2_ID'"
    check_error $? "Timeout waiting for VPC2 to become available"
fi

echo "Using the following VPCs:"
echo "VPC1: $VPC1_ID ($VPC1_CIDR)"
echo "VPC2: $VPC2_ID ($VPC2_CIDR)"

# Verify the VPCs exist and are available
echo "Verifying VPCs..."
log_cmd "aws ec2 describe-vpcs --region '$AWS_REGION' --vpc-ids '$VPC1_ID'
 '$VPC2_ID' --query 'Vpcs[*].[VpcId,State,CidrBlock]' --output table"
check_error $? "Failed to verify VPCs"

# Determine subnet CIDR blocks based on VPC CIDR blocks
VPC1_SUBNET_CIDR=$(echo "$VPC1_CIDR" | sed 's/0\.0\/16/1.0\/24/')
VPC2_SUBNET_CIDR=$(echo "$VPC2_CIDR" | sed 's/0\.0\/16/1.0\/24/')

# Sanitize subnet CIDR blocks
VPC1_SUBNET_CIDR=$(sanitize_var "$VPC1_SUBNET_CIDR") || check_error 1 "Invalid
 VPC1_SUBNET_CIDR format"
VPC2_SUBNET_CIDR=$(sanitize_var "$VPC2_SUBNET_CIDR") || check_error 1 "Invalid
 VPC2_SUBNET_CIDR format"

validate_cidr "$VPC1_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC1"
validate_cidr "$VPC2_SUBNET_CIDR" || check_error 1 "Invalid subnet CIDR for VPC2"

# Create subnets in both VPCs
echo "Creating subnet in VPC1..."
SUBNET1_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC1_ID' --cidr-block '$VPC1_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC1"
SUBNET1_ID=$(sanitize_var "$SUBNET1_ID") || check_error 1 "Invalid SUBNET1_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC1: $SUBNET1_ID")
```

```
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET1_ID'")
echo "Subnet created in VPC1 with ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"

echo "Creating subnet in VPC2..."
SUBNET2_ID=$(log_cmd "aws ec2 create-subnet --region '$AWS_REGION' --
vpc-id '$VPC2_ID' --cidr-block '$VPC2_SUBNET_CIDR' --tag-specifications
 'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Peering-Subnet},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'Subnet.SubnetId' --output text")
check_error $? "Failed to create subnet in VPC2"
SUBNET2_ID=$(sanitize_var "$SUBNET2_ID") || check_error 1 "Invalid SUBNET2_ID
 returned"
CREATED_RESOURCES+=("Subnet in VPC2: $SUBNET2_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id
 '$SUBNET2_ID'")
echo "Subnet created in VPC2 with ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"

# Create a VPC peering connection
echo "Creating VPC peering connection..."
PEERING_ID=$(log_cmd "aws ec2 create-vpc-peering-connection --region
 '$AWS_REGION' --vpc-id '$VPC1_ID' --peer-vpc-id '$VPC2_ID' --tag-specifications
 'ResourceType=vpc-peering-connection,Tags=[{Key=Name,Value=VPC1-VPC2-Peering},
{Key=project,Value=doc-smith},{Key=tutorial,Value=vpc-peering}]' --query
 'VpcPeeringConnection.VpcPeeringConnectionId' --output text")
check_error $? "Failed to create VPC peering connection"
PEERING_ID=$(sanitize_var "$PEERING_ID") || check_error 1 "Invalid PEERING_ID
 returned"
CREATED_RESOURCES+=("VPC Peering Connection: $PEERING_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --
vpc-peering-connection-id '$PEERING_ID'")
echo "VPC Peering Connection created with ID: $PEERING_ID"

# Accept the VPC peering connection
echo "Accepting VPC peering connection..."
log_cmd "aws ec2 accept-vpc-peering-connection --region '$AWS_REGION' --vpc-peering-
connection-id '$PEERING_ID'"
check_error $? "Failed to accept VPC peering connection"
echo "VPC Peering Connection accepted"

# Wait for the peering connection to become active
echo "Waiting for peering connection to become active..."
log_cmd "aws ec2 wait vpc-peering-connection-exists --region '$AWS_REGION' --vpc-
peering-connection-ids '$PEERING_ID'"
```

```
check_error $? "Timeout waiting for peering connection to become active"

# Create a route table for VPC1
echo "Creating route table for VPC1..."
RTB1_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC1_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC1-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output text")
check_error $? "Failed to create route table for VPC1"
RTB1_ID=$(sanitize_var "$RTB1_ID") || check_error 1 "Invalid RTB1_ID returned"
CREATED_RESOURCES+=("Route Table for VPC1: $RTB1_ID")
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-table-
id '$RTB1_ID'")
echo "Route table created for VPC1 with ID: $RTB1_ID"

# Create a route from VPC1 to VPC2
echo "Creating route from VPC1 to VPC2..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB1_ID' --
destination-cidr-block '$VPC2_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC1 to VPC2"
echo "Route created from VPC1 to VPC2"

# Associate the route table with the subnet in VPC1
echo "Associating route table with subnet in VPC1..."
RTB1_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB1_ID' --subnet-id '$SUBNET1_ID' --query 'AssociationId' --output
 text")
check_error $? "Failed to associate route table with subnet in VPC1"
RTB1_ASSOC_ID=$(sanitize_var "$RTB1_ASSOC_ID") || check_error 1 "Invalid
 RTB1_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC1: $RTB1_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB1_ASSOC_ID'")
echo "Route table associated with subnet in VPC1"

# Create a route table for VPC2
echo "Creating route table for VPC2..."
RTB2_ID=$(log_cmd "aws ec2 create-route-table --region '$AWS_REGION'
 --vpc-id '$VPC2_ID' --tag-specifications 'ResourceType=route-
table,Tags=[{Key=Name,Value=VPC2-RouteTable},{Key=project,Value=doc-smith},
{Key=tutorial,Value=vpc-peering}]' --query 'RouteTable.RouteTableId' --output text")
check_error $? "Failed to create route table for VPC2"
RTB2_ID=$(sanitize_var "$RTB2_ID") || check_error 1 "Invalid RTB2_ID returned"
CREATED_RESOURCES+=("Route Table for VPC2: $RTB2_ID")
```

```
CLEANUP_COMMANDS+=("aws ec2 delete-route-table --region '$AWS_REGION' --route-table-
id '$RTB2_ID'")
echo "Route table created for VPC2 with ID: $RTB2_ID"

# Create a route from VPC2 to VPC1
echo "Creating route from VPC2 to VPC1..."
log_cmd "aws ec2 create-route --region '$AWS_REGION' --route-table-id '$RTB2_ID' --
destination-cidr-block '$VPC1_CIDR' --vpc-peering-connection-id '$PEERING_ID'"
check_error $? "Failed to create route from VPC2 to VPC1"
echo "Route created from VPC2 to VPC1"

# Associate the route table with the subnet in VPC2
echo "Associating route table with subnet in VPC2..."
RTB2_ASSOC_ID=$(log_cmd "aws ec2 associate-route-table --region '$AWS_REGION' --
route-table-id '$RTB2_ID' --subnet-id '$SUBNET2_ID' --query 'AssociationId' --output
 text")
check_error $? "Failed to associate route table with subnet in VPC2"
RTB2_ASSOC_ID=$(sanitize_var "$RTB2_ASSOC_ID") || check_error 1 "Invalid
 RTB2_ASSOC_ID returned"
CREATED_RESOURCES+=("Route Table Association for VPC2: $RTB2_ASSOC_ID")
CLEANUP_COMMANDS+=("aws ec2 disassociate-route-table --region '$AWS_REGION' --
association-id '$RTB2_ASSOC_ID'")
echo "Route table associated with subnet in VPC2"

# Verify the VPC peering connection
echo "Verifying VPC peering connection..."
log_cmd "aws ec2 describe-vpc-peering-connections --region '$AWS_REGION' --
vpc-peering-connection-ids '$PEERING_ID' --query 'VpcPeeringConnections[0].
[VpcPeeringConnectionId,Status.Code,AccepterVpcInfo.VpcId,RequesterVpcInfo.VpcId]'
 --output table"
check_error $? "Failed to verify VPC peering connection"
echo "VPC peering connection verified"

# Display summary of created resources
echo ""
echo "================================================"
echo "SUMMARY OF CREATED RESOURCES"
echo "================================================"
echo "VPC1 ID: $VPC1_ID"
echo "VPC1 CIDR: $VPC1_CIDR"
echo "Subnet1 ID: $SUBNET1_ID (CIDR: $VPC1_SUBNET_CIDR)"
echo "VPC2 ID: $VPC2_ID"
echo "VPC2 CIDR: $VPC2_CIDR"
echo "Subnet2 ID: $SUBNET2_ID (CIDR: $VPC2_SUBNET_CIDR)"
```

```
echo "Peering Connection ID: $PEERING_ID"
echo "Route Table 1 ID: $RTB1_ID"
echo "Route Table 1 Association ID: $RTB1_ASSOC_ID"
echo "Route Table 2 ID: $RTB2_ID"
echo "Route Table 2 Association ID: $RTB2_ASSOC_ID"
echo ""
echo "Created resources:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "================================================"
echo ""

# Test connectivity (optional)
echo "================================================"
echo "CONNECTIVITY TEST"
echo "================================================"
echo "To test connectivity between VPCs, you would need to:"
echo "1. Launch EC2 instances in each subnet"
echo "2. Configure security groups to allow traffic"
echo "3. Test ping or other network connectivity"
echo ""

# Automatic cleanup
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Auto-confirming cleanup of all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Clean up in reverse order
    echo "Disassociating route table from subnet in VPC2..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --association-
id '$RTB2_ASSOC_ID'" || true

    echo "Disassociating route table from subnet in VPC1..."
    log_cmd "aws ec2 disassociate-route-table --region '$AWS_REGION' --association-
id '$RTB1_ASSOC_ID'" || true

    echo "Deleting route table for VPC2..."
```

```
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB2_ID'" || true

    echo "Deleting route table for VPC1..."
    log_cmd "aws ec2 delete-route-table --region '$AWS_REGION' --route-table-id
 '$RTB1_ID'" || true

    echo "Deleting VPC peering connection..."
    log_cmd "aws ec2 delete-vpc-peering-connection --region '$AWS_REGION' --vpc-
peering-connection-id '$PEERING_ID'" || true

    echo "Deleting subnet in VPC2..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id '$SUBNET2_ID'"
 || true

    echo "Deleting subnet in VPC1..."
    log_cmd "aws ec2 delete-subnet --region '$AWS_REGION' --subnet-id '$SUBNET1_ID'"
 || true

    # Delete VPCs if they were created by this script
    if [ "$CREATE_VPCS" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'" ||
 true

        echo "Deleting VPC1..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC1_ID'" ||
 true
    elif [ "$CREATE_VPC2_ONLY" = true ]; then
        echo "Deleting VPC2..."
        log_cmd "aws ec2 delete-vpc --region '$AWS_REGION' --vpc-id '$VPC2_ID'" ||
 true
    fi

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo ""
    echo "To manually clean up later, you can delete resources in this order:"
    echo "1. Route table associations"
    echo "2. Route tables"
    echo "3. VPC peering connection"
    echo "4. Subnets"
    if [ "$CREATE_VPCS" = true ] || [ "$CREATE_VPC2_ONLY" = true ]; then
```

```
        echo "5. VPCs (if created by this script)"
    fi
fi

echo "Script execution completed. See $LOG_FILE for detailed logs."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AcceptVpcPeeringConnection
  - AssociateRouteTable
  - CreateRoute
  - CreateRouteTable
  - CreateSubnet
  - CreateVpc
  - CreateVpcPeeringConnection
  - DeleteRouteTable
  - DeleteSubnet
  - DeleteVpc
  - DeleteVpcPeeringConnection
  - DescribeVpcPeeringConnections
  - DescribeVpcs
  - DisassociateRouteTable
  - Wait

Amazon ECR-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Container-Registrierungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Docker-Image
- Erstellen Sie ein Amazon ECR-Repository
- Löschen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "====================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "====================================================="
```

```
# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
 for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure' to
 set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running this
 script."
    echo "Visit https://docs.docker.com/get-docker/ for installation instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"
```

```
    echo "==================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "==================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "==================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force || echo
 "Failed to delete repository, it may not exist or may have already been deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local image,
 it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove tagged
 image, it may not exist or may have already been deleted."
    fi
```

```
    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build Docker
 image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list Docker
 images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"
```

```
# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials are
 configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --tags
 key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-registry-
gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username AWS --
password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" || handle_error
 "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"
```

```
# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image to
 ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- BatchDeleteImage
- CreateRepository
- DeleteRepository
- GetCallerIdentity
- GetLoginPassword

Amazon ECS-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Amazon ECS Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Konfigurieren Sie die Konnektivität des Container-Service

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
```

```
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
```

```
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
```

```
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
```

```
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"
```

```
    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi
```

```
    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
```

```
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}
```

```
# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
```

```
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
```

```
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
```

```
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
```

```
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")
```

```
                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster
```

```
    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNamespaces
  - ListServices
  - ListTaskDefinitions
  - ListTasks
  - RegisterTaskDefinition
  - UpdateService
  - Wait

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"
```

```
# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"
```

```
    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
```

```
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
```

```
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
```

```
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
```

```
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
```

```
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"
```

```
# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
```

```
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
```

```
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi
```

```
echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
```

```
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices

- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Erstellen und überwachen Sie einen Dienst
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service
```

```
set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
```

```
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
 "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
```

```
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
```

```
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
 retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
 attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."
```

```
    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"
```

```
    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)
```

```
    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."
```

```
        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole
```

```
        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
```

```
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}
```

```
# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
```

```
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)
```

```
    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table
```

```
    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================"
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================"
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
```

```
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
```

```
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Amazon EKS-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon EKS Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster

- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
```

```
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
```

```
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
```

```
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM
```

```
# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
```

```
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
```

```
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."
```

```
    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."
```

```
# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
```

```
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi
```

```
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Beispiele für Elastic Load Balancing — Version 2 mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Load Balancing

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Application Load Balancers
- Erstellen einer Zielgruppe
- Erstellen eines Listeners
- Überprüfen Ihrer Konfiguration
- Fügen Sie einen HTTPS-Listener hinzu (optional)
- Fügen Sie pfadbasiertes Routing hinzu (optional)
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Elastic Load Balancing Getting Started Script - v2
# This script creates an Application Load Balancer with HTTP listener and target
 group

set -euo pipefail

# Set up logging
LOG_FILE="elb-script-v2.log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Elastic Load Balancing setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check AWS CLI command success
check_command() {
    local output="$1"
    if [[ -z "$output" ]] || [[ "$output" == "None" ]]; then
        handle_error "AWS CLI command returned empty or invalid output"
    fi
}

# Function to validate ARN format
validate_arn() {
    local arn="$1"
    if [[ ! "$arn" =~ ^arn:aws:[a-z0-9-]+:[a-z0-9-]*:[0-9]{12}:.+$ ]]; then
        handle_error "Invalid ARN format: $arn"
    fi
}

# Function to validate security group ID
validate_security_group_id() {
    local sg_id="$1"
    if [[ ! "$sg_id" =~ ^sg-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid security group ID format: $sg_id"
    fi
}

# Function to validate VPC ID
validate_vpc_id() {
    local vpc_id="$1"
    if [[ ! "$vpc_id" =~ ^vpc-[a-f0-9]{8,17}$ ]]; then
        handle_error "Invalid VPC ID format: $vpc_id"
    fi
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    if [ -n "${LISTENER_ARN:-}" ]; then
        echo "Deleting listener: $LISTENER_ARN"
        aws elbv2 delete-listener --listener-arn "$LISTENER_ARN" 2>/dev/null || true
    fi

    if [ -n "${LOAD_BALANCER_ARN:-}" ]; then
        echo "Deleting load balancer: $LOAD_BALANCER_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LOAD_BALANCER_ARN" 2>/
dev/null || true

        # Wait for load balancer to be deleted before deleting target group
        echo "Waiting for load balancer to be deleted..."
        aws elbv2 wait load-balancers-deleted --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null || true
    fi

    if [ -n "${TARGET_GROUP_ARN:-}" ]; then
        echo "Deleting target group: $TARGET_GROUP_ARN"
        aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/
null || true
    fi

    if [ -n "${SECURITY_GROUP_ID:-}" ]; then
        echo "Waiting 30 seconds before deleting security group to ensure all
 dependencies are removed..."
        sleep 30

        echo "Deleting security group: $SECURITY_GROUP_ID"
        local sg_delete_output
        sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)

        local retry_count=0
        local max_retries=5
        while echo "$sg_delete_output" | grep -i "DependencyViolation" > /dev/null
 && [ $retry_count -lt $max_retries ]; do
            retry_count=$((retry_count+1))
            echo "Security group still has dependencies. Retrying in 30 seconds...
 (Attempt $retry_count of $max_retries)"
```

```
            sleep 30
            sg_delete_output=$(aws ec2 delete-security-group --group-id
 "$SECURITY_GROUP_ID" 2>&1 || true)
        done

        if echo "$sg_delete_output" | grep -i "error" > /dev/null; then
            echo "WARNING: Could not delete security group: $SECURITY_GROUP_ID" >&2
            echo "You may need to delete it manually using: aws ec2 delete-security-
group --group-id $SECURITY_GROUP_ID"
        else
            echo "Security group deleted successfully."
        fi
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
RESOURCE_PREFIX="elb-demo-${RANDOM_ID}"

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Step 1: Verify AWS CLI support for Elastic Load Balancing
echo "Verifying AWS CLI support for Elastic Load Balancing..."
if ! aws elbv2 help > /dev/null 2>&1; then
    handle_error "AWS CLI does not support elbv2 commands. Please update your AWS
 CLI."
fi

# Step 2: Get VPC ID and subnet information
echo "Retrieving VPC information..."
VPC_INFO=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 "Vpcs[0].VpcId" --output text 2>/dev/null || echo "")
check_command "$VPC_INFO"
VPC_ID="$VPC_INFO"
validate_vpc_id "$VPC_ID"
echo "Using VPC: $VPC_ID"

# Get two subnets from different Availability Zones
echo "Retrieving subnet information..."
SUBNET_INFO=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[0:2].SubnetId" --output text 2>/dev/null || echo "")
```

```
check_command "$SUBNET_INFO"

# Convert space-separated list to array
read -r -a SUBNETS <<< "$SUBNET_INFO"
if [ ${#SUBNETS[@]} -lt 2 ]; then
    handle_error "Need at least 2 subnets in different Availability Zones. Found:
 ${#SUBNETS[@]}"
fi

echo "Using subnets: ${SUBNETS[0]} and ${SUBNETS[1]}"

# Step 3: Create a security group for the load balancer
echo "Creating security group for the load balancer..."
SG_INFO=$(aws ec2 create-security-group \
    --group-name "${RESOURCE_PREFIX}-sg" \
    --description "Security group for ELB demo" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=elastic-load-balancing-gs}]' \
    --query "GroupId" --output text 2>/dev/null || echo "")
check_command "$SG_INFO"
SECURITY_GROUP_ID="$SG_INFO"
validate_security_group_id "$SECURITY_GROUP_ID"
echo "Created security group: $SECURITY_GROUP_ID"

# Add inbound rule to allow HTTP traffic
echo "Adding inbound rule to allow HTTP traffic..."
aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 80 \
    --cidr "0.0.0.0/0" > /dev/null 2>&1 || handle_error "Failed to authorize
 security group ingress"

echo "WARNING: Security group allows HTTP from 0.0.0.0/0. In production, restrict to
 specific IP addresses." >&2

# Step 4: Create the load balancer
echo "Creating Application Load Balancer..."
LB_INFO=$(aws elbv2 create-load-balancer \
    --name "${RESOURCE_PREFIX}-lb" \
    --subnets "${SUBNETS[0]}" "${SUBNETS[1]}" \
    --security-groups "$SECURITY_GROUP_ID" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "LoadBalancers[0].LoadBalancerArn" --output text 2>/dev/null || echo "")
check_command "$LB_INFO"
LOAD_BALANCER_ARN="$LB_INFO"
validate_arn "$LOAD_BALANCER_ARN"
echo "Created load balancer: $LOAD_BALANCER_ARN"

# Wait for the load balancer to be active
echo "Waiting for load balancer to become active..."
if ! aws elbv2 wait load-balancer-available --load-balancer-arns
 "$LOAD_BALANCER_ARN" 2>/dev/null; then
    handle_error "Load balancer did not reach active state within timeout period"
fi

# Step 5: Create a target group
echo "Creating target group..."
TG_INFO=$(aws elbv2 create-target-group \
    --name "${RESOURCE_PREFIX}-targets" \
    --protocol HTTP \
    --port 80 \
    --vpc-id "$VPC_ID" \
    --target-type instance \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "TargetGroups[0].TargetGroupArn" --output text 2>/dev/null || echo "")
check_command "$TG_INFO"
TARGET_GROUP_ARN="$TG_INFO"
validate_arn "$TARGET_GROUP_ARN"
echo "Created target group: $TARGET_GROUP_ARN"

# Step 6: Find EC2 instances to register as targets
echo "Looking for available EC2 instances to register as targets..."
INSTANCES=$(aws ec2 describe-instances \
    --filters "Name=vpc-id,Values=$VPC_ID" "Name=instance-state-name,Values=running"
 \
    --query "Reservations[*].Instances[*].InstanceId" --output text 2>/dev/null ||
 echo "")

# Convert space-separated list to array
read -r -a INSTANCE_IDS <<< "$INSTANCES"

if [ ${#INSTANCE_IDS[@]} -eq 0 ]; then
    echo "No running instances found in VPC $VPC_ID."
```

```
    echo "You will need to register targets manually after launching instances."
else
    # Step 7: Register targets with the target group (up to 2 instances)
    echo "Registering targets with the target group..."
    target_args=()
    for i in "${!INSTANCE_IDS[@]}"; do
        if [ "$i" -lt 2 ]; then
            target_args+=("Id=${INSTANCE_IDS[$i]}")
        fi
    done

    if [ ${#target_args[@]} -gt 0 ]; then
        if aws elbv2 register-targets \
            --target-group-arn "$TARGET_GROUP_ARN" \
            --targets "${target_args[@]}" 2>/dev/null; then
            echo "Registered instances: ${target_args[*]}"
        else
            handle_error "Failed to register targets"
        fi
    fi
fi

# Step 8: Create a listener
echo "Creating HTTP listener..."
LISTENER_INFO=$(aws elbv2 create-listener \
    --load-balancer-arn "$LOAD_BALANCER_ARN" \
    --protocol HTTP \
    --port 80 \
    --default-actions Type=forward,TargetGroupArn="$TARGET_GROUP_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=elastic-load-balancing-gs
 \
    --query "Listeners[0].ListenerArn" --output text 2>/dev/null || echo "")
check_command "$LISTENER_INFO"
LISTENER_ARN="$LISTENER_INFO"
validate_arn "$LISTENER_ARN"
echo "Created listener: $LISTENER_ARN"

# Step 9: Verify target health
echo "Verifying target health..."
aws elbv2 describe-target-health --target-group-arn "$TARGET_GROUP_ARN" 2>/dev/null
 || true

# Display load balancer DNS name
LB_DNS=$(aws elbv2 describe-load-balancers \
```

```
    --load-balancer-arns "$LOAD_BALANCER_ARN" \
    --query "LoadBalancers[0].DNSName" --output text 2>/dev/null || echo "")
check_command "$LB_DNS"

echo ""
echo "================================================"
echo "SETUP COMPLETE"
echo "================================================"
echo "Load Balancer DNS Name: $LB_DNS"
echo ""
echo "Resources created:"
echo "- Load Balancer: $LOAD_BALANCER_ARN"
echo "- Target Group: $TARGET_GROUP_ARN"
echo "- Listener: $LISTENER_ARN"
echo "- Security Group: $SECURITY_GROUP_ID"
echo ""

# Prompt for cleanup confirmation
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
read -p "Do you want to clean up all created resources? (y/n): " -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
    echo "To clean up later, run the following commands:"
    echo "aws elbv2 delete-listener --listener-arn $LISTENER_ARN"
    echo "aws elbv2 delete-load-balancer --load-balancer-arn $LOAD_BALANCER_ARN"
    echo "aws elbv2 wait load-balancers-deleted --load-balancer-arns
 $LOAD_BALANCER_ARN"
    echo "aws elbv2 delete-target-group --target-group-arn $TARGET_GROUP_ARN"
    echo "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateListener

- CreateLoadBalancer
- CreateSecurityGroup
- CreateTargetGroup
- DeleteListener
- DeleteLoadBalancer
- DeleteSecurityGroup
- DeleteTargetGroup
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetHealth
- DescribeVpcs
- Hilfe
- RegisterTargets
- Wait

ElastiCache Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. ElastiCache

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit In-Memory-Caching

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine Sicherheitsgruppe für den Zugriff ein ElastiCache
- Erstellen Sie einen serverlosen Valkey-Cache
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ElastiCache Getting Started Script
# This script creates a Valkey serverless cache, configures security groups,
# and demonstrates how to connect to and use the cache.

set -uo pipefail

# Set up logging
LOG_FILE="elasticache_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting ElastiCache tutorial script. Logging to $LOG_FILE"
echo "============================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${CACHE_NAME:-}" ]; then
        echo "- ElastiCache serverless cache: $CACHE_NAME"
    fi
    if [ -n "${SG_RULE_6379:-}" ] || [ -n "${SG_RULE_6380:-}" ]; then
        echo "- Security group rules for ports 6379 and 6380"
    fi
    echo "Please clean up these resources manually."
    exit 1
}
```

```
# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

# Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CACHE_NAME="valkey-cache-${RANDOM_ID}"

echo "Using cache name: $CACHE_NAME"

# Step 1: Set up security group for ElastiCache access
echo "Step 1: Setting up security group for ElastiCache access..."

# Get default security group ID
echo "Getting default security group ID..."
SG_ID=$(aws ec2 describe-security-groups \
  --filters Name=group-name,Values=default \
  --query "SecurityGroups[0].GroupId" \
  --output text 2>/dev/null || echo "")

if [[ -z "$SG_ID" || "$SG_ID" == "None" ]]; then
    handle_error "Failed to get default security group ID"
fi

echo "Default security group ID: $SG_ID"

# Add inbound rule for port 6379
echo "Adding inbound rule for port 6379..."
SG_RULE_6379=""
if SG_RULE_6379=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6379 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
```

```
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    fi
else
    if [[ "$SG_RULE_6379" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6379" ==
 *"already exists"* ]]; then
        echo "Rule for port 6379 already exists, continuing..."
        SG_RULE_6379="existing"
    else
        handle_error "Failed to add security group rule for port 6379:
 $SG_RULE_6379"
    fi
fi

# Add inbound rule for port 6380
echo "Adding inbound rule for port 6380..."
SG_RULE_6380=""
if SG_RULE_6380=$(aws ec2 authorize-security-group-ingress \
  --group-id "$SG_ID" \
  --protocol tcp \
  --port 6380 \
  --cidr 0.0.0.0/0 \
  --query "SecurityGroupRules[0].SecurityGroupRuleId" \
  --output text 2>&1); then
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    fi
else
    if [[ "$SG_RULE_6380" == *"InvalidGroup.Duplicate"* ]] || [[ "$SG_RULE_6380" ==
 *"already exists"* ]]; then
        echo "Rule for port 6380 already exists, continuing..."
        SG_RULE_6380="existing"
    else
        handle_error "Failed to add security group rule for port 6380:
 $SG_RULE_6380"
    fi
fi

echo "Security group rules added successfully."
echo ""
echo "##  SECURITY WARNING: The security group rules created allow access from any
 IP address (0.0.0.0/0)."
```

```
echo "This is NOT RECOMMENDED for production environments. For production,"
echo "you should restrict access to specific IP ranges or security groups."
echo "Update the CIDR blocks in this script before using in production."
echo ""

# Step 2: Create a Valkey serverless cache
echo "Step 2: Creating Valkey serverless cache..."
if ! CREATE_RESULT=$(aws elasticache create-serverless-cache \
  --serverless-cache-name "$CACHE_NAME" \
  --engine valkey \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-elasticache-gs 2>&1);
 then
    handle_error "Failed to create serverless cache: $CREATE_RESULT"
fi

echo "Cache creation initiated. Waiting for cache to become available..."

# Step 3: Check the status of the cache creation
echo "Step 3: Checking cache status..."

# Wait for the cache to become active
MAX_ATTEMPTS=30
ATTEMPT=1
CACHE_STATUS=""

while [[ $ATTEMPT -le $MAX_ATTEMPTS ]]; do
    echo "Checking cache status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    if ! DESCRIBE_RESULT=$(aws elasticache describe-serverless-caches \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        handle_error "Failed to describe serverless cache: $DESCRIBE_RESULT"
    fi

    # Extract status using jq for reliable JSON parsing
    if command -v jq &> /dev/null; then
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | jq -r
 '.ServerlessCaches[0].Status // "UNKNOWN"' 2>/dev/null || echo "")
    else
        CACHE_STATUS=$(echo "$DESCRIBE_RESULT" | grep -o '"Status": "[^"]*"' | awk -
F'"' '{print $4}' | head -n 1)
    fi

    echo "Current status: $CACHE_STATUS"
```

```
    if [[ "${CACHE_STATUS,,}" == "available" ]]; then
        echo "Cache is now available!"
        break
    elif [[ "${CACHE_STATUS,,}" == "create-failed" ]]; then
        handle_error "Cache creation failed. Please check the AWS console for
 details."
    fi

    if [[ $ATTEMPT -lt $MAX_ATTEMPTS ]]; then
        echo "Waiting 30 seconds..."
        sleep 30
    fi

    ((ATTEMPT++))
done

if [[ "${CACHE_STATUS,,}" != "available" ]]; then
    handle_error "Cache did not become available within the expected time. Last
 status: $CACHE_STATUS"
fi

# Step 4: Find your cache endpoint
echo "Step 4: Getting cache endpoint..."
if ! ENDPOINT=$(aws elasticache describe-serverless-caches \
  --serverless-cache-name "$CACHE_NAME" \
  --query "ServerlessCaches[0].Endpoint.Address" \
  --output text 2>&1); then
    handle_error "Failed to get cache endpoint: $ENDPOINT"
fi

if [[ -z "$ENDPOINT" || "$ENDPOINT" == "None" ]]; then
    handle_error "Failed to get cache endpoint"
fi

echo "Cache endpoint: $ENDPOINT"

# Step 5: Instructions for connecting to the cache
echo ""
echo "============================================================"
echo "Your Valkey serverless cache has been successfully created!"
echo "Cache Name: $CACHE_NAME"
echo "Endpoint: $ENDPOINT"
echo "============================================================"
echo ""
```

```
echo "To connect to your cache from an EC2 instance, follow these steps:"
echo ""
echo "1. Install valkey-cli on your EC2 instance:"
echo "   sudo amazon-linux-extras install epel -y"
echo "   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y"
echo "   wget https://github.com/valkey-io/valkey/archive/refs/tags/8.0.0.tar.gz"
echo "   tar xvzf 8.0.0.tar.gz"
echo "   cd valkey-8.0.0"
echo "   make BUILD_TLS=yes"
echo ""
echo "2. Connect to your cache using valkey-cli:"
echo "   src/valkey-cli -h $ENDPOINT --tls -p 6379"
echo ""
echo "3. Once connected, you can run commands like:"
echo "   set mykey \"Hello ElastiCache\""
echo "   get mykey"
echo ""

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- ElastiCache serverless cache: $CACHE_NAME"
if [[ "${SG_RULE_6379:-}" != "existing" ]] || [[ "${SG_RULE_6380:-}" !=
 "existing" ]]; then
    echo "- Security group rules for ports 6379 and 6380"
fi
echo ""
echo "Proceeding with cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Starting cleanup process..."

    # Step 7: Delete the cache
    echo "Deleting serverless cache $CACHE_NAME..."
    if ! DELETE_RESULT=$(aws elasticache delete-serverless-cache \
      --serverless-cache-name "$CACHE_NAME" 2>&1); then
        echo "WARNING: Failed to delete serverless cache: $DELETE_RESULT"
        echo "Please delete the cache manually from the AWS console."
    else
```

```
        echo "Cache deletion initiated. This may take several minutes to complete."
    fi

    # Only attempt to remove security group rules if we created them
    if [[ "${SG_RULE_6379:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6379..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6379 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6379"
        fi
    fi

    if [[ "${SG_RULE_6380:-}" != "existing" ]]; then
        echo "Removing security group rule for port 6380..."
        if ! aws ec2 revoke-security-group-ingress \
          --group-id "$SG_ID" \
          --protocol tcp \
          --port 6380 \
          --cidr 0.0.0.0/0 2>&1; then
            echo "WARNING: Failed to remove security group rule for port 6380"
        fi
    fi

    echo "Cleanup completed."
fi

echo ""
echo "Script completed. See $LOG_FILE for the full log."
echo "============================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateServerlessCache
  - DeleteServerlessCache
  - DescribeSecurityGroups
  - DescribeServerlessCaches
  - RevokeSecurityGroupIngress

Amazon EMR-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon EMR Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit großen Datenverarbeitungsclustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail
```

```
# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
```

```
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
```

```
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
```

```
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
```

```
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)
```

```
if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi
```

```
if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
```

```
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"

# Download results
echo "Downloading results file"
```

```
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup
```

```
echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - SSH
  - TerminateClusters
  - Wait

OpenSearch Service-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Bash-Skripts AWS Command Line Interface mit OpenSearch Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Such- und Analysesystemen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Service-Domain
- Laden Sie Daten auf Ihre Domain hoch
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
```

```
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/null
 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
```

```
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
```

```
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
# This policy restricts access to specific actions and should be further restricted
 to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack creation
 failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120
```

```
# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo "No";
 fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel", "Malahide,
 Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas", "Stellate, Nick",
 "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese", "Spielbauer, Bruce",
```

```
 "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett, Charlie"], "title": "U.S.
 Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch, Paul",
 "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson", "Williams,
 Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon, David",
 "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
```

```
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }

    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d:
-f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
$HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
$HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
then
                echo "Error: Request is being treated as anonymous (authentication
not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
```

```
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential backoff..."

            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT+1))/
$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)
```

```
                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" |
cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
(HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding with
data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
'"result"'; then
            echo "# Single document uploaded successfully! (HTTP $UPLOAD_HTTP_CODE)"
        else
```

```
            echo "# Warning: Single document upload may have failed (HTTP
$UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
'"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
$BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
```

```
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-name "$DOMAIN_NAME"
 --query 'DomainStatus.{AdvancedSecurityOptions: AdvancedSecurityOptions,
 AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
```

```
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
```

```
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
```

```
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

AWS FIS Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS FIS

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Versuchsvorlage
- Führen Sie das Experiment aus
- Überprüfe die Ergebnisse
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
```

```
    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
```

```
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
```

```
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
```

```
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
```

```
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
```

```
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
```

```
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
```

```
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
```

```
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")
```

```
# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
```

```
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
```

```
    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi
```

```
echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances

- Wait

AWS Glue Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS Glue

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit dem Datenkatalog

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Glue Data Catalog Tutorial Script
```

```
# This script demonstrates how to create and manage AWS Glue Data Catalog resources
 using the AWS CLI
# Cost improvements: Reduced API calls, optimized queries, eliminated redundant
 operations
# Reliability improvements: Enhanced error handling, input validation, resource
 tracking

set -euo pipefail

# Setup logging
LOG_FILE="glue-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Glue Data Catalog tutorial script at $(date)"
echo "All operations will be logged to $LOG_FILE"

# Generate a unique identifier for resource names
UNIQUE_ID=$(openssl rand -hex 4)
DB_NAME="tutorial-db-${UNIQUE_ID}"
TABLE_NAME="flights-data-${UNIQUE_ID}"
TABLE_INPUT_FILE="table-input-${UNIQUE_ID}.json"

# Track created resources
declare -a CREATED_RESOURCES=()

# Set default region if not provided
AWS_REGION="${AWS_REGION:-us-east-1}"

# Flag to track if database was successfully created
DATABASE_CREATED=false

# Trap to ensure cleanup on exit
trap cleanup_resources EXIT

# Function to check command status
check_status() {
    if [ $? -ne 0 ]; then
        echo "ERROR: $1 failed." >&2
        exit 1
    fi
}

# Function to cleanup resources
cleanup_resources() {
```

```
    local exit_code=$?
    echo "Attempting to clean up resources..."

    # Delete resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource=${CREATED_RESOURCES[$i]}
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_name=$(echo "$resource" | cut -d':' -f2)

        echo "Deleting $resource_type: $resource_name"

        case $resource_type in
            "table")
                if [ "$DATABASE_CREATED" = true ]; then
                    aws glue delete-table \
                        --database-name "$DB_NAME" \
                        --name "$resource_name" \
                        --region "$AWS_REGION" \
                        2>/dev/null || echo "Warning: Failed to delete table
$resource_name"
                fi
                ;;
            "database")
                aws glue delete-database \
                    --name "$resource_name" \
                    --region "$AWS_REGION" \
                    2>/dev/null || echo "Warning: Failed to delete database
$resource_name"
                ;;
            *)
                echo "Unknown resource type: $resource_type" >&2
                ;;
        esac
    done

    # Clean up temporary files securely
    if [ -f "$TABLE_INPUT_FILE" ]; then
        if command -v shred &> /dev/null; then
            shred -vfz -n 3 "$TABLE_INPUT_FILE" 2>/dev/null || rm -f
"$TABLE_INPUT_FILE"
        else
            rm -f "$TABLE_INPUT_FILE"
        fi
    fi
```

```
    echo "Cleanup completed."
    exit $exit_code
}

# Function to validate prerequisites
validate_prerequisites() {
    # Validate AWS CLI is available
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS CLI version
    local AWS_CLI_VERSION
    AWS_CLI_VERSION=$(aws --version 2>&1 | cut -d' ' -f1 | cut -d'/' -f2 | cut -d'.'
 -f1)
    if [ "$AWS_CLI_VERSION" -lt 1 ]; then
        echo "ERROR: AWS CLI is required" >&2
        exit 1
    fi

    # Validate jq is available for JSON validation
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH" >&2
        exit 1
    fi

    # Validate AWS credentials and get account identity in single call (cost
 optimization)
    local CALLER_IDENTITY
    CALLER_IDENTITY=$(aws sts get-caller-identity --region "$AWS_REGION" --query
 'Account' --output text 2>/dev/null) || {
        echo "ERROR: Failed to get AWS caller identity. Check credentials and
 permissions." >&2
        exit 1
    }

    if [ -z "$CALLER_IDENTITY" ] || [ "$CALLER_IDENTITY" == "None" ]; then
        echo "ERROR: Unable to determine AWS account identity" >&2
        exit 1
    fi
    echo "Using AWS Account: $CALLER_IDENTITY"
    echo "Using Region: $AWS_REGION"
```

```
}

# Function to create database with verification
create_database() {
    echo "Step 1: Creating a database named $DB_NAME"

    if ! aws glue create-database \
        --database-input "Name=$DB_NAME,Description=Database for AWS Glue tutorial"
 \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create database $DB_NAME" >&2
        exit 1
    fi

    ACCOUNT_ID=$(aws sts get-caller-identity --query 'Account' --output text)
    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:database/
${DB_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    DATABASE_CREATED=true
    CREATED_RESOURCES+=("database:$DB_NAME")
    echo "Database $DB_NAME created successfully."
}

# Function to prepare table input JSON
prepare_table_input() {
    # Create a temporary JSON file for table input with restricted permissions
    if ! touch "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to create temporary file $TABLE_INPUT_FILE" >&2
        exit 1
    fi

    if ! chmod 600 "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Failed to set permissions on $TABLE_INPUT_FILE" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    cat > "$TABLE_INPUT_FILE" << 'EOF'
{
```

```
  "Name": "TABLE_NAME_PLACEHOLDER",
  "StorageDescriptor": {
    "Columns": [
      {
        "Name": "year",
        "Type": "bigint"
      },
      {
        "Name": "quarter",
        "Type": "bigint"
      }
    ],
    "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
    "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
    "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
    "Compressed": false,
    "NumberOfBuckets": -1,
    "SerdeInfo": {
      "SerializationLibrary": "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
      "Parameters": {
        "field.delim": ",",
        "serialization.format": ","
      }
    }
  },
  "PartitionKeys": [
    {
      "Name": "mon",
      "Type": "string"
    }
  ],
  "TableType": "EXTERNAL_TABLE",
  "Parameters": {
    "EXTERNAL": "TRUE",
    "classification": "csv",
    "columnsOrdered": "true",
    "compressionType": "none",
    "delimiter": ",",
    "skip.header.line.count": "1",
    "typeOfData": "file"
  }
}
EOF
```

```
    # Replace placeholder with actual table name
    if ! sed -i "s/TABLE_NAME_PLACEHOLDER/$TABLE_NAME/g" "$TABLE_INPUT_FILE" 2>/dev/
null; then
        echo "ERROR: Failed to substitute table name in JSON file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    # Validate JSON syntax before using it
    if ! jq empty "$TABLE_INPUT_FILE" 2>/dev/null; then
        echo "ERROR: Invalid JSON in table input file" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi
}

# Function to create table
create_table() {
    echo "Step 2: Creating a table named $TABLE_NAME in database $DB_NAME"

    prepare_table_input

    local TABLE_ARN
    if ! aws glue create-table \
        --database-name "$DB_NAME" \
        --table-input "file://${TABLE_INPUT_FILE}" \
        --region "$AWS_REGION" \
        --output json > /dev/null 2>&1; then
        echo "ERROR: Failed to create table $TABLE_NAME" >&2
        rm -f "$TABLE_INPUT_FILE"
        exit 1
    fi

    aws glue tag-resource \
        --resource-arn "arn:aws:glue:${AWS_REGION}:${ACCOUNT_ID}:table/${DB_NAME}/
${TABLE_NAME}" \
        --tags-to-add Key=project,Value=doc-smith Key=tutorial,Value=glue-gs \
        --region "$AWS_REGION" \
        2>/dev/null || true

    CREATED_RESOURCES+=("table:$TABLE_NAME")
    echo "Table $TABLE_NAME created successfully."
}
```

```
# Function to get and display table details
display_table_details() {
    echo "Step 3: Getting details of table $TABLE_NAME"

    if ! aws glue get-table \
        --database-name "$DB_NAME" \
        --name "$TABLE_NAME" \
        --region "$AWS_REGION" \
        --output json; then
        echo "ERROR: Failed to retrieve table details" >&2
        exit 1
    fi
}

# Function to display summary
display_summary() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="
    echo "Database: $DB_NAME"
    echo "Table: $TABLE_NAME"
    echo "==========================================="
}

# Main execution flow
validate_prerequisites
create_database
create_table
display_table_details
display_summary

echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Starting cleanup process..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateDatabase
  - CreateTable

- DeleteDatabase
- DeleteTable
- GetDatabase
- GetTable

HealthImaging Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. HealthImaging

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDatastore`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatastore`.

AWS CLI mit Bash-Skript

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function imaging_create_datastore
```

```
#
# This function creates an AWS HealthImaging data store for importing DICOM P10
 files.
#
# Parameters:
#       -n data_store_name - The name of the data store.
#
# Returns:
#       The datastore ID.
#    And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function imaging_create_datastore() {
  local datastore_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_create_datastore"
    echo "Creates an AWS HealthImaging data store for importing DICOM P10 files."
    echo "  -n data_store_name - The name of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) datastore_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$datastore_name" ]]; then
    errecho "ERROR: You must provide a data store name with the -n parameter."
```

```
    usage
    return 1
  fi

  response=$(aws medical-imaging create-datastore \
    --datastore-name "$datastore_name" \
    --output text \
    --query 'datastoreId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging create-datastore operation failed.
$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie CreateDatastore in der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatastore`.

AWS CLI mit Bash-Skript

```
###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
```

```
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_delete_datastore
#
# This function deletes an AWS HealthImaging data store.
#
# Parameters:
#       -i datastore_id - The ID of the data store.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_delete_datastore() {
  local datastore_id response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_delete_datastore"
    echo "Deletes an AWS HealthImaging data store."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  response=$(aws medical-imaging delete-datastore \
    --datastore-id "$datastore_id")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports medical-imaging delete-datastore operation failed.
$response"
    return 1
  fi

  return 0
}
```

- API-Details finden Sie DeleteDatastore in der AWS CLI Befehlsreferenz.

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDatastore`

Das folgende Codebeispiel zeigt die Verwendung`GetDatastore`.

AWS CLI mit Bash-Skript

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_get_datastore
#
# Get a data store's properties.
#
# Parameters:
#       -i data_store_id - The ID of the data store.
#
# Returns:
#       [datastore_name, datastore_id, datastore_status, datastore_arn,  created_at,
 updated_at]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_get_datastore() {
  local datastore_id option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_get_datastore"
    echo "Gets a data store's properties."
    echo "  -i datastore_id - The ID of the data store."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "i:h" option; do
    case "${option}" in
      i) datastore_id="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$datastore_id" ]]; then
    errecho "ERROR: You must provide a data store ID with the -i parameter."
    usage
    return 1
  fi

  local response

  response=$(
    aws medical-imaging get-datastore \
      --datastore-id "$datastore_id" \
      --output text \
      --query "[ datastoreProperties.datastoreName,
 datastoreProperties.datastoreId, datastoreProperties.datastoreStatus,
 datastoreProperties.datastoreArn,  datastoreProperties.createdAt,
 datastoreProperties.updatedAt]"
  )
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie GetDatastore in der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

ListDatastores

Das folgende Codebeispiel zeigt die VerwendungListDatastores.

AWS CLI mit Bash-Skript

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function imaging_list_datastores
#
# List the HealthImaging data stores in the account.
#
# Returns:
#       [[datastore_name, datastore_id, datastore_status]]
#    And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function imaging_list_datastores() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function imaging_list_datastores"
    echo "Lists the AWS HealthImaging data stores in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response
  response=$(aws medical-imaging list-datastores \
    --output text \
    --query "datastoreSummaries[*][datastoreName, datastoreId, datastoreStatus]")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-datastores operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie ListDatastores in der AWS CLI Befehlsreferenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role to
 the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result
```

```
  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
```

```
  }"

local role_arn
role_arn=$(iam_create_role -n "$iam_role_name" -p "$assume_role_policy_document")

# shellcheck disable=SC2181
if [ ${?} == 0 ]; then
  echo "Created IAM role named $iam_role_name"
else
  errecho "The role failed to create. This demo will exit."
  clean_up "$user_name" "$key_name"
  return 1
fi

local policy_name
policy_name=$(generate_random_name "test-policy")
local policy_document="{
              \"Version\": \"2012-10-17\",
              \"Statement\": [{
                  \"Effect\": \"Allow\",
                  \"Action\": \"s3:ListAllMyBuckets\",
                  \"Resource\": \"arn:aws:s3:::*\"}]}"

local policy_arn
policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
# shellcheck disable=SC2181
if [[ ${?} == 0 ]]; then
  echo "Created  IAM policy named $policy_name"
else
  errecho "The policy failed to create."
  clean_up "$user_name" "$key_name" "$iam_role_name"
  return 1
fi

if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
  echo "Attached policy $policy_arn to role $iam_role_name"
else
  errecho "The policy failed to attach."
  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
  return 1
fi

local assume_role_policy_document="{
              \"Version\": \"2012-10-17\",
```

```
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

 local assume_role_policy_name
 assume_role_policy_name=$(generate_random_name "test-assume-role-")

 # shellcheck disable=SC2181
 local assume_role_policy_arn
 assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
 sleep 10
 echo_repeat "*" 88
 echo

 echo "Try to list buckets without the new user assuming the role."
 echo_repeat "*" 88
 echo

 # Set the environment variables for the created user.
 # bashsupport disable=BP2001
 export AWS_ACCESS_KEY_ID=$key_name
 # bashsupport disable=BP2001
 export AWS_SECRET_ACCESS_KEY=$key_secret

 local buckets
 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
```

```
    echo "There are $bucket_count buckets in the account. This should not have
happened."
  else
    errecho "Because the role with permissions has not been assumed, listing buckets
failed."
  fi

  echo
  echo_repeat "*" 88
  echo "Now assume the role $iam_role_name and list the buckets."
  echo_repeat "*" 88
  echo

  local credentials

  credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
  else
    errecho "Failed to assume role."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
    return 1
  fi

  IFS=$'\t ' read -r -a credentials <<<"$credentials"

  export AWS_ACCESS_KEY_ID=${credentials[0]}
  export AWS_SECRET_ACCESS_KEY=${credentials[1]}
  # bashsupport disable=BP2001
  export AWS_SESSION_TOKEN=${credentials[2]}

  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets succeeded
because of "
    echo "the assumed role."
```

```
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

Die in diesem Szenario verwendeten IAM-Funktionen.

```
###############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
```

```
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
```

```
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
```

```
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
```

```
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi
```

```
  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

```
###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
```

```
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
```

```
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

```
##############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
```

```
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
```

```
  echo "Deletes an AWS Identity and Access Management (IAM) role"
  echo "  -n role_name -- The name of the IAM role."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

echo "role_name:$role_name"
if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Role name:  $role_name"
iecho ""

response=$(aws iam delete-role \
  --role-name "$role_name")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-role operation failed.\n$response"
  return 1
fi
```

```
  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
```

```
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
```

```
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
```

```
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
```

```
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie AttachRolePolicy in der AWS CLI Befehlsreferenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user_access_key
```

```
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- API-Details finden Sie CreateAccessKey in der AWS CLI Befehlsreferenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }
```

```
# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
  --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-policy operation failed.\n$response"
  return 1
fi
```

```
  echo "$response"
}
```

- API-Details finden Sie CreatePolicy in der AWS CLI Befehlsreferenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
```

```
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi
```

```
  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie CreateRole in der AWS CLI Befehlsreferenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
```

```
    echo "$@"
  fi
}

###########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###########################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###########################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
```

```
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    User name:   $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
  errecho "ERROR: A user with that name already exists in the account."
  return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
  --output text \
  --query 'User.Arn')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-user operation failed.$response"
  return 1
fi

echo "$response"
```

```
  return 0
}
```

- API-Details finden Sie CreateUser in der AWS CLI Befehlsreferenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
```

```
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- API-Details finden Sie DeleteAccessKey in der AWS CLI Befehlsreferenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
```

```
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
```

```
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}
```

- API-Details finden Sie DeletePolicy in der AWS CLI Befehlsreferenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- API-Details finden Sie DeleteRole in der AWS CLI Befehlsreferenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
################################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}
```

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must supply
 a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- API-Details finden Sie DeleteUser in der AWS CLI Befehlsreferenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
```

```
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam detach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie DetachRolePolicy in der AWS CLI Befehlsreferenz.

`GetUser`

Das folgende Codebeispiel zeigt die Verwendung`GetUser`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
```

```
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###########################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

- API-Details finden Sie GetUser in der AWS CLI Befehlsreferenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for the
 specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie ListAccessKeys in der AWS CLI Befehlsreferenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie ListUsers in der AWS CLI Befehlsreferenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iam_update_access_key
#
# This function can activate or deactivate an IAM access key for the specified IAM
 user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to update.
#       -a            -- Activate the selected access key.
#       -d            -- Deactivate the selected access key.
#
# Example:
#       # To deactivate the selected access key for IAM user Bob
#       iam_update_access_key -u Bob -k AKIAIOSFODNN7EXAMPLE -d
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_update_access_key() {
  local user_name access_key status response
  local option OPTARG # Required to use getopts command in a function.
  local activate_flag=false deactivate_flag=false

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_update_access_key"
    echo "Updates the status of an AWS Identity and Access Management (IAM) access
 key for the specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to update."
    echo "  -a              Activate the access key."
    echo "  -d              Deactivate the access key."
```

```
  echo ""
}

# Retrieve the calling parameters.
  while getopts "u:k:adh" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      a) activate_flag=true ;;
      d) deactivate_flag=true ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

 # Validate input parameters
  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  # Ensure that only -a or -d is specified
  if [[ "$activate_flag" == true && "$deactivate_flag" == true ]]; then
    errecho "ERROR: You cannot specify both -a (activate) and -d (deactivate) at
the same time."
    usage
    return 1
  fi
```

```
    # If neither -a nor -d is provided, return an error
    if [[ "$activate_flag" == false && "$deactivate_flag" == false ]]; then
      errecho "ERROR: You must specify either -a (activate) or -d (deactivate)."
      usage
      return 1
    fi

    # Determine the status based on the flag
    if [[ "$activate_flag" == true ]]; then
      status="Active"
    elif [[ "$deactivate_flag" == true ]]; then
      status="Inactive"
    fi

    iecho "Parameters:\n"
    iecho "    Username:   $user_name"
    iecho "    Access key: $access_key"
    iecho "    New status: $status"
    iecho ""

    # Update the access key status
    response=$(aws iam update-access-key \
      --user-name "$user_name" \
      --access-key-id "$access_key" \
      --status "$status" 2>&1)

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log $error_code
      errecho "ERROR: AWS reports update-access-key operation failed.\n$response"
      return 1
    fi

    iecho "update-access-key response: $response"
    iecho

    return 0
}
```

- API-Details finden Sie UpdateAccessKey in der AWS CLI Befehlsreferenz.

Szenarien

Konfigurieren Sie die Container-Dienstkonnektivität

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
```

```
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
```

```
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}
```

```
# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"
```

```
    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}
```

```
# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
```

```
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null
```

```
        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
```

```
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
```

```
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
```

```
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi
```

```
    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)
```

```
        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
```

```
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
```

```
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
```

```
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks

- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
```

```
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error
```

```
    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1
```

```
    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
```

```
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
```

```
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi
```

```
        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
```

```
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
```

```
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
```

```
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
```

```
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
```

```
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
```

```
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")
```

```
# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
```

```
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- AttachRolePolicy
- AuthorizeSecurityGroupIngress
- CreateCluster
- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Erstellen Sie eine Lambda-Funktion und stellen Sie sie bereit
- Erstellen Sie eine REST-API
- Konfigurieren Sie die Lambda-Proxy-Integration

- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
```

```
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")
```

```
        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."
```

```
# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
```

```
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"
```

```
# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2
```

```
aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Erstellen und überwachen Sie einen Dienst
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
```

```
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
```

```
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
```

```
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
 2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
 retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
 attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
```

```
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
```

```
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."
```

```
    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
```

```
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
```

```
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
```

```
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
```

```
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair

- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Einen verwalteten Überwachungsbereich erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstelle eine IAM-Rolle für deinen Workspace

- Erstellen Sie einen Grafana-Workspace
- Konfigurieren Sie die Authentifizierung
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
```

```
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
```

```
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"
```

```
echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
```

```
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"
```

```
    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
```

```
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Erstellen eines Monitoring-Dashboards mit einem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}
```

```
# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
```

```
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
```

```
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"
```

```
# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
```

```
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi
```

```
# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
```

```
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi
```

```
        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Lambda
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

##############################################################################
# Helper functions
##############################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
```

```
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
 detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR
```

```
wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

##############################################################################
# Region pre-check
##############################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
```

```
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "============================================"
echo "AWS Lambda - Create Your First Function"
echo "============================================"
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
```

```
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

###############################################################################
# Step 1: Create IAM execution role
###############################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
```

```
echo "Waiting for IAM role to propagate..."
sleep 10

################################################################################
# Step 2: Create Lambda function
################################################################################

echo ""
echo "==========================================="
echo "Step 2: Create Lambda function"
echo "==========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
```

```
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

##############################################################################
# Step 3: Invoke the function
##############################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi
```

```
RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

################################################################################
# Step 4: View CloudWatch logs
################################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
```

```
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

##############################################################################
# Summary and cleanup
##############################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteFunction
- DeleteLogGroup
- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

Erste Schritte mit Serverless-Data Warehouses

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Benutze den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace
- Verwenden Sie Rotverschiebung CreateWorkgroup
- Verwenden Sie Rotverschiebung DeleteNamespace
- Benutze iam CreateRole

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
```

```
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
```

```
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
```

```
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}
```

```
# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "==========================================="
  echo "CLEANUP CONFIRMATION"
  echo "==========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"
```

```
    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi
```

```
# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
```

```
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
```

```
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
```

```
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
```

```
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
```

```
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
```

```
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
```

```
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
```

```
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup

- GetNamespace
- GetRole
- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config-Konfigurationsrekorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
```

```
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi
```

```
    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
```

```
    echo "CREATED RESOURCES"
    echo "==========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "==========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
```

```
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"
```

```
# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
```

```
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
```

```
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
```

```
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
```

```
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders

- DescribeDeliveryChannels
- DetachRolePolicy
- GetCallerIdentity
- PutConfigurationRecorder
- PutDeliveryChannel
- PutPublicAccessBlock
- PutRolePolicy
- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Erste Schritte mit dem Geräteschutz im Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- Aktivieren Sie die Audit-Checks von IoT Device Defender
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Schadensbegrenzungsmaßnahme
- Wenden Sie Abhilfemaßnahmen auf Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- Aktivieren Sie die IoT-Protokollierung

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
```

```
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
```

```
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
```

```
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
```

```
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi
```

```
    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "==================================================="
echo "Step 1: Creating required IAM roles"
echo "==================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
```

```
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true
```

```
# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true
```

```
if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
```

```
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
```

```
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
```

```
else
    echo ""
    echo "======================================================"
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "======================================================"
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "======================================================"
echo "Step 6: Setting up SNS notifications"
echo "======================================================"

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."
```

```
    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."
```

```
    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
```

```
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
```

```
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
```

```
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions

- GetRole
- GetV2LoggingOptions
- ListAuditFindings
- ListAuditMitigationActionsTasks
- ListMitigationActions
- ListRolePolicies
- ListTopics
- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Erste Schritte mit Feature Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
```

```
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
```

```
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
```

```
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
```

```
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
```

```
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
```

```
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
```

```
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"
```

```
ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
```

```
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
```

```
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)
```

```
if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
```

```
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
```

```
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters

- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
```

```
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}
```

```
# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi
```

```
    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
```

```
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
```

```
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
```

```
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
```

```
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
```

```
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
```

```
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
```

```
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups
  - UpdateKubeconfig
  - Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten

- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
```

```
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi
```

```
    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done
```

```
                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi
```

```
   # Delete MSK cluster first (to remove dependencies on security group)
   if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
       echo "Deleting MSK cluster: $CLUSTER_ARN"
       aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
delete cluster"

       # Wait a bit for the cluster deletion to start
       echo "Waiting 30 seconds for cluster deletion to begin..."
       sleep 30
   fi

   # Remove security group references before attempting deletion
   if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
then
       remove_security_group_references "$CLIENT_SG_ID"

       echo "Deleting security group: $CLIENT_SG_ID"
       # Try multiple times with longer delays to ensure dependencies are removed
       for i in {1..10}; do
           if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
               echo "Security group deleted successfully"
               break
           fi
           echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
           sleep 30
       done
   fi

   # Delete key pair if it exists
   if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
       echo "Deleting key pair: $KEY_NAME"
       aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
   fi

   # Remove role from instance profile
   if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
       echo "Removing role from instance profile"
       aws iam remove-role-from-instance-profile \
           --instance-profile-name "$INSTANCE_PROFILE_NAME" \
           --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
```

```
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
 "$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
 delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
 "$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")
```

```
    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
```

```
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
```

```
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."
```

```
# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "==============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
```

```
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
```

```
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
```

```
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi
```

```
# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
```

```
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi
```

```
echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
```

```
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
```

```
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
```

```
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"
```

```
# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
```

```
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
```

```
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
```

```
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair
  - DeletePolicy
  - DeleteRole
  - DeleteSecurityGroup
  - DescribeAvailabilityZones
  - DescribeCluster
  - DescribeImages
  - DescribeInstanceTypeOfferings
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DescribeSubnets
  - DescribeVpcs
  - DetachRolePolicy

- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Redshift-Cluster
- Erstellen Sie eine IAM-Rolle für den S3-Zugriff
- Tabellen erstellen und Daten laden
- Führen Sie Beispielabfragen aus
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
```

```
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-final-
cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/dev/
null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true
```

```
        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --query
 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id "$statement_id"
 --query 'Error' --output text 2>/dev/null || echo "Unknown error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done
```

```
    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus" \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and try
 again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
```

```
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
```

```
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
```

```
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed to
 attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
```

```
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer not
 null, eventid integer not null, dateid smallint not null sortkey, qtysold smallint
 not null, pricepaid decimal(8,2), commission decimal(8,2), saletime timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
```

```
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint not
 null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt' DELIMITER
 '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
```

```
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid AND
 caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"
```

```
echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Erste Schritte mit der Workflow-Orchestrierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Step Functions
- Erstellen Sie Ihre erste State-Machine
- Starten Sie die Ausführung Ihrer Zustandsmaschine

- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS Step
 Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
```

```
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be less
 reliable."
    echo "Consider installing jq for better error handling: brew install jq (macOS)
 or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
```

```
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
```

```
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] || [ "$wait_time"
 -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"
```

```
    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
```

```
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
$ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed to
delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
```

```
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
```

```
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
```

```
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)
```

```
check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
```

```
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
```

```
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
```

```
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
```

```
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
```

```
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
```

```
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
```

```
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
```

```
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
```

```
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
```

```
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "==========================================="

# Cleanup
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy

- StartExecution
- UpdateStateMachine

Fest codierte Geheimnisse in einen sicheren Geheimspeicher verschieben

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisiere das Geheimnis
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
```

```
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
```

```
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"
```

```
echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
```

```
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
```

```
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"
```

```
# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
```

```
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
```

```
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue
  - PutResourcePolicy
  - UpdateSecret

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Versuchsvorlage
- Führen Sie das Experiment aus
- Überprüfen Sie die Ergebnisse
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi
```

```
    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
"$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
"$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
"$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
"arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
```

```
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
```

```
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"
```

```
# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
```

```
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
```

```
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
```

```
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
```

```
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done
```

```
if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
```

```
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi
```

```
# Delete experiment template
echo "Deleting experiment template $TEMPLATE_ID..."
DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

# Delete CloudWatch alarm
echo "Deleting CloudWatch alarm $ALARM_NAME..."
DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

# Terminate EC2 instance
echo "Terminating EC2 instance $INSTANCE_ID..."
TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
echo "Waiting for instance to terminate..."
aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

# Clean up IAM resources
echo "Removing role from instance profile..."
REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

echo "Deleting instance profile..."
DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

echo "Deleting FIS role policy..."
DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME")
check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

echo "Detaching policy from EC2 role..."
DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

echo "Deleting FIS role..."
DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
  --role-name "$FIS_ROLE_NAME")
```

```
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy

- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Richten Sie die Attribute-Based Zugriffskontrolle ein

Das folgende Codebeispiel zeigt, wie Attribute-Based Access Control (ABAC) für DynamoDB implementiert wird.

- Erstellen Sie eine IAM-Richtlinie für ABAC.
- Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.
- Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

AWS CLI mit Bash-Skript

Erstellen Sie eine IAM-Richtlinie für ABAC.

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
```

```
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

Erstellen Sie Tabellen mit Tags für verschiedene Abteilungen.

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
```

```
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

Listen Sie Tabellen auf der Grundlage von Tags auf und filtern Sie sie.

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text | xargs
 -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn" --output
 text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
```

```
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreatePolicy
  - CreateTable
  - ListTables

Systemverwaltung einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Berechtigungen für Systems Manager
- Erstellen einer IAM-Rolle für Systems Manager
- Konfigurieren Sie Systems Manager
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
```

```
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
```

```
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
```

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmDocument",
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument",
         "ssm:DescribeDocument"
       ],
       "Resource": [
         "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
         "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
       ]
     },
     {
       "Sid": "SsmEnableExplorer",
       "Effect": "Allow",
       "Action": "ssm:StartAutomationExecution",
```

```
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
```

```
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
```

```
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
```

```
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
```

```
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Lambda-Funktionen für die Überwachung
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
```

```
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
```

```
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
```

```
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
```

```
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
```

```
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
```

```
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Arbeite mit Streams und Time-to-Live

Das folgende Codebeispiel zeigt, wie DynamoDB-Streams und Time-to-Live Funktionen verwaltet werden.

- Erstellen Sie eine Tabelle mit aktiviertem Streams.
- Beschreiben Sie Streams.
- Erstellen Sie eine Lambda-Funktion für die Verarbeitung von Streams.
- Aktivieren Sie TTL für eine Tabelle.
- Fügen Sie Elemente mit TTL-Attributen hinzu.
- Beschreiben Sie TTL-Einstellungen.

AWS CLI mit Bash-Skript

Erstellen Sie eine Tabelle mit aktiviertem Streams.

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

Beschreiben Sie Streams.

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

Erstellen Sie eine Lambda-Funktion für Streams.

```
# Step 1: Create an IAM role for the Lambda function
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

Aktivieren Sie TTL für eine Tabelle.

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
```

```
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

Fügen Sie Elemente mit TTL-Attributen hinzu.

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

Beschreiben Sie TTL-Einstellungen.

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
    --table-name TTLDemo
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateTable

- DescribeTable
- DescribeTimeToLive
- PutItem
- UpdateTimeToLive

AWS IoT Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS IoT

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit dem Geräteschutz im Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- Aktivieren Sie die Audit-Checks von IoT Device Defender
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Schadensbegrenzungsmaßnahme
- Wenden Sie Abhilfemaßnahmen auf Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- Aktivieren Sie die IoT-Protokollierung

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
```

```
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"
```

```
    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi
```

```
    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
```

```
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
```

```
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
```

```
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "==================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "==================================================="
```

```
# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "====================================================="
echo "Step 3: Running an on-demand audit"
echo "====================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi
```

```
TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
```

```
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="
```

```
# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
```

```
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "==================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "==================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
```

```
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true
```

```
    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi
```

```
# Step 7: Enable AWS IoT logging
echo ""
echo "======================================================"
echo "Step 7: Enabling AWS IoT logging"
echo "======================================================"

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
```

```
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
```

```
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
```

```
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction

- CreateRole
- CreateTopic
- DeleteAccountAuditConfiguration
- DeleteMitigationAction
- DeleteRole
- DeleteRolePolicy
- DeleteTopic
- DescribeAccountAuditConfiguration
- DescribeAuditTask
- DetachRolePolicy
- GetLoggingOptions
- GetRole
- GetV2LoggingOptions
- ListAuditFindings
- ListAuditMitigationActionsTasks
- ListMitigationActions
- ListRolePolicies
- ListTopics
- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Erste Schritte mit Nachrichten im Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IoT-Ressourcen
- Konfigurieren Ihres Geräts
- Ausführen der Beispielanwendungen

- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
```

```
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --
new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
```

```
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your region."
 | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-document file://
iot-policy.json --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-core-
gs"
check_error $?

# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn' --
output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
```

```
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active --
certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile $CERTS_DIR/
public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." | tee
 -a $LOG_FILE
```

```
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-python-
v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
```

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?
```

```
# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE
```

```
    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-status
 INACTIVE" | tee -a $LOG_FILE
```

```
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint
  - DetachPolicy
  - DetachThingPrincipal
  - GetCallerIdentity
  - UpdateCertificate

AWS KMS Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS KMS

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Arbeiten mit der Tabellenverschlüsselung

Im folgenden Codebeispiel wird gezeigt, wie Sie Verschlüsselungsoptionen für DynamoDB-Tabellen verwalten.

- Erstellen Sie eine Tabelle mit Standardverschlüsselung.
- Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.
- Aktualisieren Sie die Einstellungen für die Tabellenverschlüsselung.
- Beschreiben Sie die Tabellenverschlüsselung.

AWS CLI mit dem Bash-Skript

Erstellen Sie eine Tabelle mit Standardverschlüsselung.

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
    --table-name CustomerData \
    --attribute-definitions \
        AttributeName=CustomerID,AttributeType=S \
    --key-schema \
        AttributeName=CustomerID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS
```

Erstellen Sie eine Tabelle mit einem kundenverwalteten CMK.

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
    --description "Key for DynamoDB table encryption" \
    --key-usage ENCRYPT_DECRYPT \
    --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
```

```
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
 DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
    --table-name SensitiveData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Aktualisieren Sie die Tabellenverschlüsselung.

```
# Update a table to use a different KMS key
aws dynamodb update-table \
    --table-name CustomerData \
    --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

Beschreiben Sie die Tabellenverschlüsselung.

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
    --table-name CustomerData \
    --query "Table.SSEDescription"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateKey
  - CreateTable
  - DescribeTable
  - UpdateTable

Lambda-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Erstellen Sie eine Lambda-Funktion und stellen Sie sie bereit
- Erstellen Sie eine REST-API
- Konfigurieren Sie die Lambda-Proxy-Integration
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
```

```
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)
```

```
        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF
```

```
# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}
```

```
# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
```

```
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
```

```
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
```

```
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi

- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetResources
- GetRestApis
- PutIntegration
- PutMethod

Erstellen eines Monitoring-Dashboards mit einem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"
```

```
# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
```

```
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
```

```
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
```

```
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
```

```
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
```

```
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
```

```
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
```

```
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"
```

```
if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
```

```
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Erstellen Sie Ihre erste serverlose Funktion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Lambda
- Funktionscode erstellen
- Erstellen einer Lambda-Funktion
- Testen Sie Ihre Lambda-Funktion
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"

exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

################################################################################
# Helper functions
################################################################################
```

```
cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function deletion."
                        break
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
```

```
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could not
 detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "============================================"
    echo "ERROR: Script failed at $1"
    echo "============================================"
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5
```

```
    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################
```

```
echo ""
echo "==========================================="
echo "AWS Lambda - Create Your First Function"
echo "==========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
```

```
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !== 'number')
 {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

################################################################################
# Step 1: Create IAM execution role
################################################################################

echo ""
echo "============================================"
echo "Step 1: Create IAM execution role"
echo "============================================"
echo ""

TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/service-
role/AWSLambdaBasicExecutionRole")
echo "Policy attached."

# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

##############################################################################
# Step 2: Create Lambda function
##############################################################################

echo ""
```

```
echo "============================================"
echo "Step 2: Create Lambda function"
echo "============================================"
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)

if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
```

```
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --query
 State --output text" \
    "Active"

################################################################################
# Step 3: Invoke the function
################################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
```

```
fi

################################################################################
# Step 4: View CloudWatch logs
################################################################################

echo ""
echo "==========================================="
echo "Step 4: View CloudWatch Logs"
echo "==========================================="
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
```

```
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

###############################################################################
# Summary and cleanup
###############################################################################

echo ""
echo "============================================"
echo "SUMMARY"
echo "============================================"
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup

- DeleteRole
- DescribeLogStreams
- DetachRolePolicy
- GetFunction
- GetFunctionConfiguration
- GetLogEvents
- Invoke

Verwendung von Eigenschaftsvariablen in Überwachungs-Dashboards zur Überwachung mehrerer serverloser Funktionen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Lambda-Funktionen für die Überwachung
- Erstellen Sie ein Dashboard CloudWatch
- Fügen Sie dem Dashboard eine Eigenschaftsvariable hinzu
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077
```

```
# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
```

```
    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/dev/
null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
```

```
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json

# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"
```

```
# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
```

```
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /dev/
null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /dev/
null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a "$LOG_FILE"

# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
```

```
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name "$DASHBOARD_NAME"
 --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
 | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME" --
output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"
```

```
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

Lightsail-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Lightsail Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit einfachen virtuellen privaten Servern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer -Instance
- Fügen Sie Ihrer Instance Speicher hinzu
- Snapshot erstellen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Lightsail Getting Started CLI Script
# This script demonstrates how to create and manage Lightsail resources using the
 AWS CLI
```

```
set -euo pipefail

# Set AWS region
export AWS_REGION="${AWS_REGION:-us-west-2}"
echo "Using AWS region: $AWS_REGION"

# Set up logging
LOG_FILE="lightsail-script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Lightsail Getting Started script at $(date)"

# Error handling function
handle_error() {
  echo "ERROR: $1" >&2
  echo "Attempting to clean up resources..."
  cleanup_resources || true
  exit 1
}

# Function to check if a command succeeded
check_status() {
  if [ $? -ne 0 ]; then
    handle_error "$1"
  fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
INSTANCE_NAME="LightsailInstance-${RANDOM_ID}"
DISK_NAME="LightsailDisk-${RANDOM_ID}"
SNAPSHOT_NAME="LightsailSnapshot-${RANDOM_ID}"

# Array to track created resources
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
  CREATED_RESOURCES+=("$1:$2")
  echo "Created $1: $2"
}

# Function to clean up resources
cleanup_resources() {
```

```
  echo "Resources created by this script:"
  for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "  $resource"
  done

  # Reverse the array to delete resources in reverse order
  for (( idx=${#CREATED_RESOURCES[@]}-1 ; idx>=0 ; idx-- )); do
    IFS=':' read -r type name <<< "${CREATED_RESOURCES[idx]}"

    case "$type" in
      "instance_snapshot")
        echo "Deleting instance snapshot: $name"
        aws lightsail delete-instance-snapshot --instance-snapshot-name "$name" --
region "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk_snapshot")
        echo "Deleting disk snapshot: $name"
        aws lightsail delete-disk-snapshot --disk-snapshot-name "$name" --region
 "$AWS_REGION" 2>/dev/null || true
        ;;
      "disk")
        echo "Detaching disk: $name"
        aws lightsail detach-disk --disk-name "$name" --region "$AWS_REGION" 2>/dev/
null || true
        sleep 10
        echo "Deleting disk: $name"
        aws lightsail delete-disk --disk-name "$name" --region "$AWS_REGION" 2>/dev/
null || true
        ;;
      "instance")
        echo "Deleting instance: $name"
        INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name "$name"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
        if [ "$INSTANCE_STATE" == "pending" ]; then
          echo "Instance is in pending state. Waiting for it to be ready before
 deleting..."
          MAX_WAIT=30
          WAITED=0
          while [ "$INSTANCE_STATE" == "pending" ] && [ $WAITED -lt $MAX_WAIT ]; do
            sleep 10
            WAITED=$((WAITED+1))
```

```
            INSTANCE_STATE=$(aws lightsail get-instance-state --instance-name
 "$name" --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null ||
 echo "unknown")
            echo "Instance state: $INSTANCE_STATE"
          done
        fi
        aws lightsail delete-instance --instance-name "$name" --region "$AWS_REGION"
 2>/dev/null || true
        ;;
    esac
  done

  echo "Cleanup completed"
}

# Step 1: Verify AWS CLI configuration
echo "Step 1: Verifying AWS CLI configuration"
aws configure list
check_status "Failed to verify AWS CLI configuration"

# Step 2: Get available blueprints and bundles
echo "Step 2: Getting available blueprints and bundles"
echo "Available blueprints (showing first 5):"
aws lightsail get-blueprints --region "$AWS_REGION" --query 'blueprints[0:5].
[blueprintId,name]' --output table
check_status "Failed to get blueprints"

echo "Available bundles (showing first 5):"
aws lightsail get-bundles --region "$AWS_REGION" --query 'bundles[0:5].
[bundleId,name,price]' --output table
check_status "Failed to get bundles"

# Get available regions and availability zones
echo "Getting available regions and availability zones"
# Use a specific availability zone in us-west-2 region
AVAILABILITY_ZONE="us-west-2a"
echo "Using availability zone: $AVAILABILITY_ZONE"

# Step 3: Create a Lightsail instance
echo "Step 3: Creating Lightsail instance: $INSTANCE_NAME"
aws lightsail create-instances \
  --instance-names "$INSTANCE_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --blueprint-id amazon_linux_2023 \
```

```
  --bundle-id nano_3_0 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create Lightsail instance"
track_resource "instance" "$INSTANCE_NAME"

# Wait for the instance to be in a running state
echo "Waiting for instance to be in running state..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  STATUS=$(aws lightsail get-instance-state --instance-name "$INSTANCE_NAME"
 --region "$AWS_REGION" --query 'state.name' --output text 2>/dev/null || echo
 "unknown")
  if [ "$STATUS" == "running" ]; then
    echo "Instance is now running"
    break
  fi
  echo "Instance status: $STATUS. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Instance failed to reach running state after 5 minutes"
fi

# Get instance details
echo "Getting instance details"
INSTANCE_IP=$(aws lightsail get-instance --instance-name "$INSTANCE_NAME" --region
 "$AWS_REGION" --query 'instance.publicIpAddress' --output text)
check_status "Failed to get instance IP address"
echo "Instance IP address: $INSTANCE_IP"

# Validate IP address format
if ! [[ "$INSTANCE_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
  handle_error "Invalid IP address format: $INSTANCE_IP"
fi

# Step 4: Download the default key pair
echo "Step 4: Downloading default key pair"
KEY_FILE="lightsail_key_${RANDOM_ID}.pem"
umask 077
```

```
aws lightsail download-default-key-pair --region "$AWS_REGION" --output text >
 "$KEY_FILE"
check_status "Failed to download key pair"

if [ ! -f "$KEY_FILE" ] || [ ! -s "$KEY_FILE" ]; then
  handle_error "Key pair file was not created or is empty"
fi

chmod 400 "$KEY_FILE"
check_status "Failed to set permissions on key pair"
echo "Key pair downloaded to $KEY_FILE"

echo "To connect to your instance, use:"
echo "ssh -i $KEY_FILE ec2-user@$INSTANCE_IP"

# Step 5: Create a block storage disk
echo "Step 5: Creating block storage disk: $DISK_NAME"
aws lightsail create-disk \
  --disk-name "$DISK_NAME" \
  --availability-zone "$AVAILABILITY_ZONE" \
  --size-in-gb 8 \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create disk"
track_resource "disk" "$DISK_NAME"

# Wait for the disk to be available using polling
echo "Waiting for disk to be available..."
MAX_ATTEMPTS=30
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  DISK_STATE=$(aws lightsail get-disk --disk-name "$DISK_NAME" --region
 "$AWS_REGION" --query 'disk.state' --output text 2>/dev/null || echo "unknown")
  if [ "$DISK_STATE" == "available" ]; then
    echo "Disk is now available"
    break
  fi
  echo "Disk status: $DISK_STATE. Waiting..."
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done

if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  handle_error "Disk failed to become available after 5 minutes"
```

```
fi

# Attach the disk to the instance
echo "Attaching disk to instance"
aws lightsail attach-disk \
  --disk-name "$DISK_NAME" \
  --instance-name "$INSTANCE_NAME" \
  --disk-path /dev/xvdf \
  --region "$AWS_REGION"
check_status "Failed to attach disk to instance"

echo "Disk attached. To format and mount the disk, connect to your instance and
 run:"
echo "sudo mkfs -t ext4 /dev/xvdf"
echo "sudo mkdir -p /mnt/my-data"
echo "sudo mount /dev/xvdf /mnt/my-data"
echo "sudo chown ec2-user:ec2-user /mnt/my-data"

# Step 6: Create a snapshot of the instance
echo "Step 6: Creating snapshot of the instance: $SNAPSHOT_NAME"
aws lightsail create-instance-snapshot \
  --instance-name "$INSTANCE_NAME" \
  --instance-snapshot-name "$SNAPSHOT_NAME" \
  --region "$AWS_REGION" \
  --tags key=project,value=doc-smith key=tutorial,value=lightsail-gs
check_status "Failed to create instance snapshot"
track_resource "instance_snapshot" "$SNAPSHOT_NAME"

# Wait for the snapshot to complete using polling
echo "Waiting for snapshot to complete... (this may take several minutes)"
MAX_ATTEMPTS=60
ATTEMPTS=0
while [ $ATTEMPTS -lt $MAX_ATTEMPTS ]; do
  SNAPSHOT_STATE=$(aws lightsail get-instance-snapshot --instance-snapshot-name
 "$SNAPSHOT_NAME" --region "$AWS_REGION" --query 'instanceSnapshot.state' --output
 text 2>/dev/null || echo "unknown")
  if [ "$SNAPSHOT_STATE" == "available" ]; then
    echo "Snapshot creation completed"
    break
  fi
  echo "Snapshot status: $SNAPSHOT_STATE. Waiting... ($ATTEMPTS/$MAX_ATTEMPTS)"
  ATTEMPTS=$((ATTEMPTS+1))
  sleep 10
done
```

```
if [ $ATTEMPTS -eq $MAX_ATTEMPTS ]; then
  echo "Warning: Snapshot creation is taking longer than expected but will continue
 in the background."
  echo "You can check its status later with: aws lightsail get-instance-snapshot --
instance-snapshot-name $SNAPSHOT_NAME --region $AWS_REGION"
fi

# Step 7: Clean up resources
echo "Step 7: Clean up resources"
echo "The script has created the following resources:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
  echo "  $resource"
done

echo "Cleaning up these resources automatically..."
cleanup_resources

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachDisk
  - CreateDisk
  - CreateInstanceSnapshot
  - CreateInstances
  - DeleteDisk
  - DeleteInstance
  - DeleteInstanceSnapshot
  - DetachDisk
  - DownloadDefaultKeyPair
  - GetBlueprints
  - GetBundles
  - GetDisk
  - GetInstance
  - GetInstanceSnapshot
  - GetInstanceState

Amazon Managed Grafana-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie das with Bash-Skript AWS Command Line Interface mit Amazon Managed Grafana verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Einen verwalteten Überwachungsbereich erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Ihren Workspace
- Erstellen Sie einen Grafana-Workspace
- Authentifizierung konfigurieren
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
```

```
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
```

```
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
```

```
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
```

```
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
```

```
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
```

```
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "==========================================="

# Step 6: Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
```

```
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace

- DetachRolePolicy
- GetCallerIdentity

MediaConnect Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. MediaConnect

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit dem Streaming von Videoübertragungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Zugriff auf Elemental MediaConnect
- Erstellen Sie einen Flow
- Fügen Sie eine Ausgabe hinzu
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
# AWS Elemental MediaConnect Getting Started Tutorial Script
# This script creates a MediaConnect flow, adds an output, grants an entitlement,
# and then cleans up the resources.

set -euo pipefail

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="mediaconnect-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Elemental MediaConnect tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Tags for all resources
TAGS_ARRAY=("Key=project,Value=doc-smith" "Key=tutorial,Value=aws-elemental-
mediaconnect-gs")

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI is available
validate_aws_cli() {
    if ! command -v aws &> /dev/null; then
        handle_error "AWS CLI is not installed or not in PATH"
    fi

    # Security: Verify AWS CLI version is recent
    local aws_version
    aws_version=$(aws --version 2>&1 | head -1)
    echo "AWS CLI version: $aws_version"

    if ! aws sts get-caller-identity &> /dev/null; then
        handle_error "AWS credentials are not configured or invalid"
```

```
    fi

    # Security: Validate caller identity
    local account_id
    account_id=$(aws sts get-caller-identity --query Account --output text 2>/dev/
null)
    if [ -z "$account_id" ]; then
        handle_error "Failed to retrieve AWS account ID"
    fi
    echo "AWS Account ID: $account_id"
}

# Function to safely extract JSON values using jq (preferred) or fallback
extract_json_value() {
    local json_output="$1"
    local key="$2"

    if [ -z "$json_output" ]; then
        return 1
    fi

    # Security: Use jq if available for safer JSON parsing
    if command -v jq &> /dev/null; then
        echo "$json_output" | jq -r ".${key} // empty" 2>/dev/null || return 1
    else
        # Fallback with additional validation
        if ! echo "$json_output" | grep -q "\"$key\""; then
            return 1
        fi
        echo "$json_output" | grep -o "\"$key\": \"[^\"]*" | head -1 | cut -d'"' -f4
    fi
}

# Function to tag MediaConnect resource
tag_mediaconnect_resource() {
    local resource_arn="$1"
    echo "Tagging resource: $resource_arn"

    for tag in "${TAGS_ARRAY[@]}"; do
        if ! aws mediaconnect tag-resource --resource-arn "$resource_arn" --tags
 "$tag" 2>&1; then
            echo "WARNING: Failed to apply tag $tag to resource"
        fi
    done
```

```
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [ -n "${FLOW_ARN:-}" ]; then
        # Security: Validate ARN format before using it
        if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-
Z0-9:-]+$ ]]; then
            echo "WARNING: Invalid Flow ARN format, skipping cleanup: $FLOW_ARN"
            return 1
        fi

        # Check flow status before attempting to stop
        echo "Checking flow status..."
        local flow_status_output
        if flow_status_output=$(aws mediaconnect describe-flow --flow-arn
 "$FLOW_ARN" --query "Flow.Status" --output text 2>&1); then
            echo "Current flow status: $flow_status_output"

            if [ "$flow_status_output" == "ACTIVE" ] || [ "$flow_status_output" ==
 "UPDATING" ]; then
                echo "Stopping flow: $FLOW_ARN"
                if aws mediaconnect stop-flow --flow-arn "$FLOW_ARN" 2>&1; then
                    # Wait for flow to stop before deleting
                    echo "Waiting for flow to stop..."
                    sleep 10
                else
                    echo "WARNING: Failed to stop flow. Attempting to delete
 anyway..."
                fi
            else
                echo "Flow is not in ACTIVE or UPDATING state, skipping stop
 operation."
            fi

            # Delete the flow
            echo "Deleting flow: $FLOW_ARN"
            if aws mediaconnect delete-flow --flow-arn "$FLOW_ARN" 2>&1; then
                echo "Flow deleted successfully"
            else
                echo "WARNING: Failed to delete flow. You may need to manually
 delete it from the AWS console."
```

```
            fi
        else
            echo "WARNING: Could not check flow status"
        fi
    fi
}

# Set trap to cleanup on script exit
trap cleanup_resources EXIT

# Validate AWS CLI setup
validate_aws_cli

# Get the current AWS region
aws_region=""
if aws_region=$(aws configure get region 2>/dev/null); then
    if [ -z "$aws_region" ]; then
        handle_error "Failed to get AWS region. Please make sure AWS CLI is
 configured."
    fi
else
    handle_error "Failed to retrieve AWS region configuration"
fi

# Security: Validate region format
if [[ ! "$aws_region" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $aws_region"
fi

AWS_REGION="$aws_region"
echo "Using AWS Region: $AWS_REGION"

# Get available availability zones in the current region
echo "Getting available availability zones in region $AWS_REGION..."
az_output=""
if az_output=$(aws ec2 describe-availability-zones --region "$AWS_REGION" --query
 "AvailabilityZones[0].ZoneName" --output text 2>&1); then
    AVAILABILITY_ZONE="$az_output"
    if [ -z "$AVAILABILITY_ZONE" ]; then
        handle_error "Failed to retrieve availability zones"
    fi

    # Security: Validate AZ format
    if [[ ! "$AVAILABILITY_ZONE" =~ ^[a-z]{2}-[a-z]+-[0-9][a-z]$ ]]; then
```

```
        handle_error "Invalid availability zone format: $AVAILABILITY_ZONE"
    fi

    echo "Using availability zone: $AVAILABILITY_ZONE"
else
    handle_error "Failed to get availability zones"
fi

# Generate a unique suffix for resource names
SUFFIX=$(date +%s | cut -c 6-10)
FLOW_NAME="AwardsNYCShow-${SUFFIX}"
SOURCE_NAME="AwardsNYCSource-${SUFFIX}"
OUTPUT_NAME="AwardsNYCOutput-${SUFFIX}"
ENTITLEMENT_NAME="PhillyTeam-${SUFFIX}"

echo "Using the following resource names:"
echo "Flow name: $FLOW_NAME"
echo "Source name: $SOURCE_NAME"
echo "Output name: $OUTPUT_NAME"
echo "Entitlement name: $ENTITLEMENT_NAME"

# Step 1: Verify access to MediaConnect
echo "Step 1: Verifying access to AWS Elemental MediaConnect..."
list_flows_output=""
if list_flows_output=$(aws mediaconnect list-flows 2>&1); then
    echo "$list_flows_output"
else
    handle_error "Failed to list flows. Please check your AWS credentials and
 permissions."
fi

# Step 2: Create a flow
echo "Step 2: Creating a flow..."
create_flow_output=""
if create_flow_output=$(aws mediaconnect create-flow \
    --availability-zone "$AVAILABILITY_ZONE" \
    --name "$FLOW_NAME" \
    --source "Name=$SOURCE_NAME,Protocol=zixi-
push,WhitelistCidr=10.24.34.0/23,StreamId=ZixiAwardsNYCFeed" 2>&1); then
    echo "$create_flow_output"
else
    handle_error "Failed to create flow"
fi
```

```
# Extract the flow ARN from the output
FLOW_ARN=$(echo "$create_flow_output" | jq -r '.Flow.FlowArn // empty' 2>/dev/null)
if [ -z "$FLOW_ARN" ]; then
    FLOW_ARN=$(echo "$create_flow_output" | grep -o '"FlowArn": "[^"]*' | head -1 |
 cut -d'"' -f4)
fi
if [ -z "$FLOW_ARN" ]; then
    handle_error "Failed to extract flow ARN from output"
fi
echo "Flow ARN: $FLOW_ARN"

# Validate flow ARN format
if [[ ! "$FLOW_ARN" =~ ^arn:aws:mediaconnect:[a-z0-9-]+:[0-9]+:flow:[a-zA-Z0-9:-]+
$ ]]; then
    handle_error "Invalid Flow ARN format: $FLOW_ARN"
fi

# Tag the flow
tag_mediaconnect_resource "$FLOW_ARN"

# Step 3: Add an output
echo "Step 3: Adding an output to the flow..."
add_output_output=""
if add_output_output=$(aws mediaconnect add-flow-outputs \
    --flow-arn "$FLOW_ARN" \
    --outputs "Name=$OUTPUT_NAME,Protocol=zixi-
push,Destination=198.51.100.11,Port=1024,StreamId=ZixiAwardsOutput" 2>&1); then
    echo "$add_output_output"
else
    handle_error "Failed to add output to flow"
fi

# Extract the output ARN
output_arn=""
output_arn=$(echo "$add_output_output" | jq -r ".Output.OutputArn // empty" 2>/dev/
null)
if [ -z "$output_arn" ]; then output_arn=$(echo "$add_output_output" | grep -o
 '"OutputArn": "[^"]*' | head -1 | cut -d'"' -f4); fi
if [ -z "$output_arn" ]; then
    echo "WARNING: Failed to extract output ARN from output"
else
    OUTPUT_ARN="$output_arn"
    echo "Output ARN: $OUTPUT_ARN"
    tag_mediaconnect_resource "$OUTPUT_ARN"
```

```
fi

# Step 4: Grant an entitlement
echo "Step 4: Granting an entitlement..."
grant_entitlement_output=""
if grant_entitlement_output=$(aws mediaconnect grant-flow-entitlements \
    --flow-arn "$FLOW_ARN" \
    --entitlements "Name=$ENTITLEMENT_NAME,Subscribers=222233334444" 2>&1); then
    echo "$grant_entitlement_output"
else
    handle_error "Failed to grant entitlement"
fi

# Extract the entitlement ARN
entitlement_arn=""
entitlement_arn=$(echo "$grant_entitlement_output" | jq -r
 '.Entitlement.EntitlementArn // empty' 2>/dev/null)
if [ -z "$entitlement_arn" ]; then
    entitlement_arn=$(echo "$grant_entitlement_output" | grep -o '"EntitlementArn":
 "[^"]*' | head -1 | cut -d'"' -f4)
fi
if [ -z "$entitlement_arn" ]; then
    echo "WARNING: Failed to extract entitlement ARN from output"
else
    ENTITLEMENT_ARN="$entitlement_arn"
    echo "Entitlement ARN: $ENTITLEMENT_ARN"
    tag_mediaconnect_resource "$ENTITLEMENT_ARN"
fi

# Step 5: List entitlements to share with affiliates
echo "Step 5: Listing entitlements for the flow..."
describe_flow_output=""
if describe_flow_output=$(aws mediaconnect describe-flow --flow-arn "$FLOW_ARN" --
query "Flow.Entitlements" 2>&1); then
    echo "Entitlements for the flow:"
    echo "$describe_flow_output"
else
    handle_error "Failed to describe flow"
fi

# Display information to share with affiliates
echo ""
echo "Information to share with your Philadelphia affiliate:"
echo "Entitlement ARN: ${ENTITLEMENT_ARN:-N/A}"
```

```
echo "AWS Region: $AWS_REGION"

# Display resource summary
echo ""
echo "============================================"
echo "RESOURCE SUMMARY"
echo "============================================"
echo "The following resources were created:"
echo "1. Flow: $FLOW_NAME (ARN: $FLOW_ARN)"
echo "2. Output: $OUTPUT_NAME (ARN: ${OUTPUT_ARN:-N/A})"
echo "3. Entitlement: $ENTITLEMENT_NAME (ARN: ${ENTITLEMENT_ARN:-N/A})"
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Automatically cleaning up all created resources..."

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddFlowOutputs
  - CreateFlow
  - DeleteFlow
  - DescribeAvailabilityZones
  - DescribeFlow
  - GrantFlowEntitlements
  - ListFlows
  - StopFlow

Amazon MSK-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
```

```
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
```

```
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
```

```
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
```

```
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
 then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
 seconds..."
```

```
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi
```

```
    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
```

```
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi
```

```
echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"

# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
```

```
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
```

```
done

echo "================================================"
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
```

```
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"
```

```
ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)
```

```
# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)
```

```
if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi
```

```
echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi
```

```
# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi
```

```
# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
```

```
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done
```

```
echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
```

```
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
```

```
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="
```

```
# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
```

```
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
```

```
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "==============================================="
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreatePolicy
  - CreateRole
  - CreateSecurityGroup
  - DeleteCluster
  - DeleteInstanceProfile
  - DeleteKeyPair

- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

Neptun-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Neptune Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Graphdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihre Neptune-Datenbank
- Erstellen Sie Subnetze in mehreren Verfügbarkeitszonen
- Konfigurieren Sie die Sicherheit für Ihre Neptune-Datenbank
- Erstellen Sie eine Neptune DB-Subnetzgruppe
- Erstellen Sie einen Neptune-DB-Cluster und eine Neptune-DB-Instance
- Fügen Sie Ihrer Graphdatenbank Daten hinzu
- Fragen Sie Ihre Graphdatenbank ab

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Neptune Getting Started Script
# This script creates an Amazon Neptune database cluster and demonstrates basic
 operations

# Set up logging
LOG_FILE="neptune-setup.log"
echo "Starting Neptune setup at $(date)" > "$LOG_FILE"
```

```
# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..." | tee -a "$LOG_FILE"

    # Only attempt to delete resources that were successfully created
    if [[ -n "$DB_INSTANCE_ID" ]]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --skip-final-snapshot"
        log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"
    fi

    if [[ -n "$DB_CLUSTER_ID" ]]; then
        echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --skip-final-snapshot"
    fi

    if [[ -n "$DB_SUBNET_GROUP" ]]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
        log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"
    fi
```

```
    if [[ -n "$SECURITY_GROUP_ID" ]]; then
        echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"
    fi

    if [[ -n "$SUBNET_IDS" ]]; then
        for SUBNET_ID in $SUBNET_IDS; do
            echo "Deleting subnet $SUBNET_ID..." | tee -a "$LOG_FILE"
            log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET_ID"
        done
    fi

    if [[ -n "$VPC_ID" ]]; then
        echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
        log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"
    fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
VPC_NAME="neptune-vpc-$RANDOM_ID"
DB_SUBNET_GROUP="neptune-subnet-group-$RANDOM_ID"
DB_CLUSTER_ID="neptune-cluster-$RANDOM_ID"
DB_INSTANCE_ID="neptune-instance-$RANDOM_ID"
SG_NAME="neptune-sg-$RANDOM_ID"

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "VPC Name: $VPC_NAME" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "DB Cluster ID: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "DB Instance ID: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "Security Group Name: $SG_NAME" | tee -a "$LOG_FILE"

# Step 1: Create VPC
echo "Creating VPC..." | tee -a "$LOG_FILE"
VPC_OUTPUT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
 "ResourceType=vpc,Tags=[{Key=Name,Value=$VPC_NAME},{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]" --output json)
check_error "$VPC_OUTPUT" $? "Failed to create VPC"

VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID" | tee -a "$LOG_FILE"
```

```
# Enable DNS support for the VPC
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-support"
log_cmd "aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames"

# Step 2: Create Internet Gateway and attach to VPC
echo "Creating Internet Gateway..." | tee -a "$LOG_FILE"
IGW_OUTPUT=$(aws ec2 create-internet-gateway --tag-specifications
 'ResourceType=internet-gateway,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$IGW_OUTPUT" $? "Failed to create Internet Gateway"

IGW_ID=$(echo "$IGW_OUTPUT" | grep -o '"InternetGatewayId": "[^"]*' | cut -d'"' -f4)
echo "Internet Gateway created with ID: $IGW_ID" | tee -a "$LOG_FILE"

log_cmd "aws ec2 attach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"

# Step 3: Create subnets in different AZs
echo "Creating subnets..." | tee -a "$LOG_FILE"

# Get available AZs
AZ_OUTPUT=$(aws ec2 describe-availability-zones --output json)
check_error "$AZ_OUTPUT" $? "Failed to get availability zones"

# Extract first 3 AZ names
AZ1=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -1)
AZ2=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -2 |
 tail -1)
AZ3=$(echo "$AZ_OUTPUT" | grep -o '"ZoneName": "[^"]*' | cut -d'"' -f4 | head -3 |
 tail -1)

# Create 3 subnets in different AZs
SUBNET1_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.1.0/24 --availability-zone $AZ1 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET1_OUTPUT" $? "Failed to create subnet 1"
SUBNET1_ID=$(echo "$SUBNET1_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET2_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.2.0/24 --availability-zone $AZ2 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET2_OUTPUT" $? "Failed to create subnet 2"
```

```
SUBNET2_ID=$(echo "$SUBNET2_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET3_OUTPUT=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block 10.0.3.0/24 --availability-zone $AZ3 --tag-specifications
 'ResourceType=subnet,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SUBNET3_OUTPUT" $? "Failed to create subnet 3"
SUBNET3_ID=$(echo "$SUBNET3_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4)

SUBNET_IDS="$SUBNET1_ID $SUBNET2_ID $SUBNET3_ID"
echo "Created subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"

# Step 4: Create route table and add route to Internet Gateway
echo "Creating route table..." | tee -a "$LOG_FILE"
ROUTE_TABLE_OUTPUT=$(aws ec2 create-route-table --vpc-id $VPC_ID --tag-
specifications 'ResourceType=route-table,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$ROUTE_TABLE_OUTPUT" $? "Failed to create route table"

ROUTE_TABLE_ID=$(echo "$ROUTE_TABLE_OUTPUT" | grep -o '"RouteTableId": "[^"]*' | cut
 -d'"' -f4)
echo "Route table created with ID: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"

# Add route to Internet Gateway
log_cmd "aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $IGW_ID"

# Associate route table with subnets
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET1_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET2_ID"
log_cmd "aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id
 $SUBNET3_ID"

# Step 5: Create security group
echo "Creating security group..." | tee -a "$LOG_FILE"
SG_OUTPUT=$(aws ec2 create-security-group --group-name $SG_NAME --description
 "Security group for Neptune" --vpc-id $VPC_ID --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-neptune-gs}]' --output json)
check_error "$SG_OUTPUT" $? "Failed to create security group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
```

```
echo "Security group created with ID: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"

# Add inbound rule for Neptune port (8182)
# Note: In production, you should restrict this to specific IP ranges
echo "Adding security group rule for Neptune port 8182..." | tee -a "$LOG_FILE"
log_cmd "aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --
protocol tcp --port 8182 --cidr 10.0.0.0/16"

# Step 6: Create DB subnet group
echo "Creating DB subnet group..." | tee -a "$LOG_FILE"
DB_SUBNET_GROUP_OUTPUT=$(aws neptune create-db-subnet-group --db-subnet-group-
name $DB_SUBNET_GROUP --db-subnet-group-description "Subnet group for Neptune" --
subnet-ids $SUBNET1_ID $SUBNET2_ID $SUBNET3_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_SUBNET_GROUP_OUTPUT" $? "Failed to create DB subnet group"
echo "DB subnet group created: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"

# Step 7: Create Neptune DB cluster
echo "Creating Neptune DB cluster..." | tee -a "$LOG_FILE"
DB_CLUSTER_OUTPUT=$(aws neptune create-db-cluster --db-cluster-identifier
 $DB_CLUSTER_ID --engine neptune --vpc-security-group-ids $SECURITY_GROUP_ID
 --db-subnet-group-name $DB_SUBNET_GROUP --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_CLUSTER_OUTPUT" $? "Failed to create Neptune DB cluster"
echo "Neptune DB cluster created: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"

# Step 8: Create Neptune DB instance
echo "Creating Neptune DB instance..." | tee -a "$LOG_FILE"
DB_INSTANCE_OUTPUT=$(aws neptune create-db-instance --db-instance-identifier
 $DB_INSTANCE_ID --db-instance-class db.r5.large --engine neptune --db-
cluster-identifier $DB_CLUSTER_ID --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-neptune-gs --output json)
check_error "$DB_INSTANCE_OUTPUT" $? "Failed to create Neptune DB instance"
echo "Neptune DB instance created: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"

# Step 9: Wait for the DB instance to become available
echo "Waiting for Neptune DB instance to become available..." | tee -a "$LOG_FILE"
log_cmd "aws neptune wait db-instance-available --db-instance-identifier
 $DB_INSTANCE_ID"

# Step 10: Get the Neptune endpoint
echo "Getting Neptune endpoint..." | tee -a "$LOG_FILE"
ENDPOINT_OUTPUT=$(aws neptune describe-db-clusters --db-cluster-identifier
 $DB_CLUSTER_ID --output json)
```

```
check_error "$ENDPOINT_OUTPUT" $? "Failed to get Neptune endpoint"

NEPTUNE_ENDPOINT=$(echo "$ENDPOINT_OUTPUT" | grep -o '"Endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Neptune endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"

# Step 11: Display information about how to connect to Neptune
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "NEPTUNE SETUP COMPLETE" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Neptune Endpoint: $NEPTUNE_ENDPOINT" | tee -a "$LOG_FILE"
echo "Port: 8182" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To query your Neptune database using Gremlin, you can use curl:" | tee -a
 "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.V().limit(1)\"}' https://
$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "To add data to your graph:" | tee -a "$LOG_FILE"
echo "curl -X POST -d '{\"gremlin\":\"g.addV(\\\"person\\\").property(\\\"name\\\",
 \\\"Howard\\\")\"}' https://$NEPTUNE_ENDPOINT:8182/gremlin" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Note: You may need to configure your client machine to access the Neptune
 instance within the VPC." | tee -a "$LOG_FILE"
echo "For production use, consider using an EC2 instance in the same VPC or setting
 up a bastion host." | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"

# Step 12: List all created resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "RESOURCES CREATED" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "VPC: $VPC_ID" | tee -a "$LOG_FILE"
echo "Internet Gateway: $IGW_ID" | tee -a "$LOG_FILE"
echo "Subnets: $SUBNET1_ID, $SUBNET2_ID, $SUBNET3_ID" | tee -a "$LOG_FILE"
echo "Route Table: $ROUTE_TABLE_ID" | tee -a "$LOG_FILE"
echo "Security Group: $SECURITY_GROUP_ID" | tee -a "$LOG_FILE"
echo "DB Subnet Group: $DB_SUBNET_GROUP" | tee -a "$LOG_FILE"
echo "Neptune DB Cluster: $DB_CLUSTER_ID" | tee -a "$LOG_FILE"
echo "Neptune DB Instance: $DB_INSTANCE_ID" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
```

```
# Step 13: Ask if user wants to clean up resources
echo "" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "==============================================" | tee -a "$LOG_FILE"
echo "Do you want to clean up all created resources? (y/n): " | tee -a "$LOG_FILE"
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..." | tee -a "$LOG_FILE"

    # Delete DB instance
    echo "Deleting DB instance $DB_INSTANCE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-instance --db-instance-identifier $DB_INSTANCE_ID
 --skip-final-snapshot"

    # Wait for DB instance to be deleted
    echo "Waiting for DB instance to be deleted..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune wait db-instance-deleted --db-instance-identifier
 $DB_INSTANCE_ID"

    # Delete DB cluster
    echo "Deleting DB cluster $DB_CLUSTER_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-cluster --db-cluster-identifier $DB_CLUSTER_ID --
skip-final-snapshot"

    # Wait for DB cluster to be deleted (no specific wait command for this, so we'll
 sleep)
    echo "Waiting for DB cluster to be deleted..." | tee -a "$LOG_FILE"
    sleep 60

    # Delete DB subnet group
    echo "Deleting DB subnet group $DB_SUBNET_GROUP..." | tee -a "$LOG_FILE"
    log_cmd "aws neptune delete-db-subnet-group --db-subnet-group-name
 $DB_SUBNET_GROUP"

    # Delete security group
    echo "Deleting security group $SECURITY_GROUP_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-security-group --group-id $SECURITY_GROUP_ID"

    # Detach and delete internet gateway
    echo "Detaching and deleting internet gateway $IGW_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 detach-internet-gateway --internet-gateway-id $IGW_ID --vpc-id
 $VPC_ID"
```

```
    log_cmd "aws ec2 delete-internet-gateway --internet-gateway-id $IGW_ID"

    # Delete subnets
    echo "Deleting subnets..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET1_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET2_ID"
    log_cmd "aws ec2 delete-subnet --subnet-id $SUBNET3_ID"

    # Delete route table
    echo "Deleting route table $ROUTE_TABLE_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-route-table --route-table-id $ROUTE_TABLE_ID"

    # Delete VPC
    echo "Deleting VPC $VPC_ID..." | tee -a "$LOG_FILE"
    log_cmd "aws ec2 delete-vpc --vpc-id $VPC_ID"

    echo "Cleanup complete!" | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can delete them manually later." |
 tee -a "$LOG_FILE"
    echo "See the list of resources above for reference." | tee -a "$LOG_FILE"
fi

echo "Script completed. See $LOG_FILE for details." | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AssociateRouteTable
  - AttachInternetGateway
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateInternetGateway
  - CreateRoute
  - CreateRouteTable
  - CreateSecurityGroup
  - CreateSubnet

- CreateVpc
- DeleteDbCluster
- DeleteDbInstance
- DeleteDbSubnetGroup
- DeleteInternetGateway
- DeleteRouteTable
- DeleteSecurityGroup
- DeleteSubnet
- DeleteVpc
- DescribeAvailabilityZones
- DescribeDbClusters
- DetachInternetGateway
- ModifyVpcAttribute
- Wait

AWS Payment Cryptography Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS Payment Cryptography

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit der Kryptografie zur Sicherheit von Zahlungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines -Schlüssels
- Überprüfen Sie den CVV2-Wert
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Payment Cryptography Getting Started Script
# This script demonstrates how to use AWS Payment Cryptography to create a key,
# generate and verify CVV2 values, and clean up resources.

set -euo pipefail

# Initialize log file with secure permissions
LOG_FILE="payment-cryptography-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "AWS Payment Cryptography Tutorial - $(date)" > "$LOG_FILE"

# Function to log messages
log() {
    local message="$1"
    echo "$(date +"%Y-%m-%d %H:%M:%S") - $message" | tee -a "$LOG_FILE"
}

# Function to handle errors
handle_error() {
    local error_message="$1"
    log "ERROR: $error_message"
    log "Script failed. Please check the log file: $LOG_FILE"

    echo ""
    echo "=========================================="
    echo "ERROR ENCOUNTERED"
```

```
    echo "==========================================="
    echo "The script encountered an error: $error_message"
    echo "Resources created will be listed below."
    echo ""

    if [ -n "${KEY_ARN:-}" ]; then
        echo "Key ARN: $KEY_ARN"
    fi

    exit 1
}

# Function to check command output for errors
check_error() {
    local output="$1"
    local command="$2"

    if echo "$output" | grep -iq "error\|exception\|fail"; then
        handle_error "Command failed: $command"
    fi
}

# Validate AWS CLI is available and credentials are configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! aws sts get-caller-identity &> /dev/null; then
    handle_error "AWS credentials are not properly configured"
fi

log "Starting AWS Payment Cryptography tutorial"

# Step 1: Create a key
log "Step 1: Creating a card verification key (CVK)"
if ! KEY_OUTPUT=$(aws payment-cryptography create-key \
  --exportable \
  --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyMod
 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-payment-cryptography-gs
 2>&1); then
    handle_error "Failed to create key"
fi
```

```
echo "$KEY_OUTPUT"
check_error "$KEY_OUTPUT" "create-key"

# Extract the Key ARN from the output
KEY_ARN=$(echo "$KEY_OUTPUT" | grep -o '"KeyArn": "[^"]*' | cut -d'"' -f4)

if [ -z "$KEY_ARN" ]; then
    handle_error "Failed to extract Key ARN from output"
fi

log "Successfully created key with ARN: $KEY_ARN"

# Step 2: Generate a CVV2 value
log "Step 2: Generating a CVV2 value"
if ! CVV2_OUTPUT=$(aws payment-cryptography-data generate-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --generation-attributes CardVerificationValue2={CardExpiryDate=0123} 2>&1); then
    handle_error "Failed to generate CVV2 value"
fi

echo "$CVV2_OUTPUT"
check_error "$CVV2_OUTPUT" "generate-card-validation-data"

# Extract the CVV2 value from the output - updated to use ValidationData instead of
 CardDataValue
CVV2_VALUE=$(echo "$CVV2_OUTPUT" | grep -o '"ValidationData": "[^"]*' | cut -d'"' -
f4)

if [ -z "$CVV2_VALUE" ]; then
    handle_error "Failed to extract CVV2 value from output"
fi

log "Successfully generated CVV2 value"

# Step 3: Verify the CVV2 value
log "Step 3: Verifying the CVV2 value"
if ! VERIFY_OUTPUT=$(aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data "$CVV2_VALUE" 2>&1); then
    handle_error "Failed to verify CVV2 value"
```

```
fi

echo "$VERIFY_OUTPUT"
check_error "$VERIFY_OUTPUT" "verify-card-validation-data"

log "Successfully verified CVV2 value"

# Step 4: Perform a negative test
log "Step 4: Performing a negative test with incorrect CVV2"
if aws payment-cryptography-data verify-card-validation-data \
  --key-identifier "$KEY_ARN" \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 999 2>&1; then
    handle_error "Negative test did not fail as expected"
fi

log "Negative test completed successfully (verification failed as expected)"

# Display created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "Key ARN: $KEY_ARN"
echo ""

# Auto-confirm cleanup
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Proceeding with cleanup of all created resources..."

log "Step 5: Cleaning up resources"

# Delete the key
log "Deleting key: $KEY_ARN"
if ! DELETE_OUTPUT=$(aws payment-cryptography delete-key \
  --key-identifier "$KEY_ARN" 2>&1); then
    handle_error "Failed to delete key"
fi

echo "$DELETE_OUTPUT"
check_error "$DELETE_OUTPUT" "delete-key"
```

```
log "Key scheduled for deletion. Default waiting period is 7 days."
log "To cancel deletion before the waiting period ends, use:"
log "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

echo ""
echo "============================================"
echo "CLEANUP COMPLETE"
echo "============================================"
echo "The key has been scheduled for deletion after the default waiting period (7
 days)."
echo "To cancel deletion before the waiting period ends, use:"
echo "aws payment-cryptography restore-key --key-identifier $KEY_ARN"

log "Tutorial completed successfully"
echo ""
echo "Tutorial completed successfully. See $LOG_FILE for details."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateKey
  - DeleteKey
  - RestoreKey

Amazon Pinpoint-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Amazon Pinpoint Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Push-Benachrichtigungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Anwendung
- Aktivieren Sie Kanäle für Push-Benachrichtigungen
- Push-Benachrichtigung senden
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
```

```
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""
```

```
    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
```

```
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters" >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi
```

```
check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "============================================"
echo "ENABLING FCM (GCM) CHANNEL"
echo "============================================"
```

```
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-demo-
only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --apns-
channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
```

```
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-channel
 command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
```

```
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens with
 actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder tokens)
echo ""
```

```
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
```

```
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
```

```
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Beispiele für Amazon Polly unter Verwendung AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit der Text-to-Speech-Synthese

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Polly Getting Started Script
# This script demonstrates how to use Amazon Polly with the AWS CLI

set -euo pipefail

# Set up logging
LOG_FILE="polly-tutorial.log"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
WORK_DIR=$(mktemp -d)
trap 'cleanup_temp' EXIT

cleanup_temp() {
    rm -rf "$WORK_DIR"
```

```
}

echo "Starting Amazon Polly tutorial at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    # Use bash array to safely handle arguments
    bash -c "$1" 2>&1 | tee -a "$LOG_FILE" || return $?
}

# Function to check for errors
check_error() {
    if echo "$1" | grep -iq "error"; then
        echo "ERROR detected in output. Exiting script." | tee -a "$LOG_FILE"
        echo "$1" | tee -a "$LOG_FILE"
        return 1
    fi
    return 0
}

# Function to handle errors and cleanup
handle_error() {
    local line_number=$1
    echo "Error occurred at line $line_number. Attempting cleanup..." | tee -a
 "$LOG_FILE"
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "============================================================" | tee -a
 "$LOG_FILE"

    # Delete lexicon if it exists
    if [[ -n "${LEXICON_NAME:-}" ]]; then
        echo "Deleting lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
        if aws polly delete-lexicon --name "$LEXICON_NAME" 2>&1 | tee -a
 "$LOG_FILE"; then
```

```
            echo "Lexicon deleted successfully." | tee -a "$LOG_FILE"
        else
            echo "Warning: Failed to delete lexicon." | tee -a "$LOG_FILE"
        fi
    fi

    # Remove audio files
    for file in output.mp3 ssml-output.mp3 lexicon-output.mp3 example.pls; do
        if [[ -f "$file" ]]; then
            rm -f "$file"
            echo "Removed $file" | tee -a "$LOG_FILE"
        fi
    done

    echo "Cleanup complete." | tee -a "$LOG_FILE"
}

# Trap errors with line number
trap 'handle_error ${LINENO}' ERR

# Verify AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "AWS CLI is not installed. Please install it first." | tee -a "$LOG_FILE"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "AWS credentials are not configured. Please configure them first." | tee -a
 "$LOG_FILE"
    exit 1
fi

# Step 1: Verify Amazon Polly is available
echo "Step 1: Verifying Amazon Polly availability" | tee -a "$LOG_FILE"
if aws polly describe-voices --query 'Voices[0].Name' --output text &> /dev/null;
 then
    echo "Amazon Polly is available. Proceeding with tutorial." | tee -a "$LOG_FILE"
else
    echo "Amazon Polly is not available in your AWS CLI installation or region." |
 tee -a "$LOG_FILE"
    echo "Please update your AWS CLI to the latest version or check your region." |
 tee -a "$LOG_FILE"
    exit 1
```

```
fi

# Step 2: List available voices
echo "" | tee -a "$LOG_FILE"
echo "Step 2: Listing available voices" | tee -a "$LOG_FILE"
log_cmd "aws polly describe-voices --language-code en-US --output text --query
 'Voices[0:3].[Id, LanguageCode, Gender]'" || true

# Step 3: Basic text-to-speech conversion
echo "" | tee -a "$LOG_FILE"
echo "Step 3: Converting text to speech" | tee -a "$LOG_FILE"
OUTPUT_FILE="${WORK_DIR}/output.mp3"
POLLY_TEXT="Hello, welcome to Amazon Polly. This is a sample text to speech
 conversion."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --text
 '$POLLY_TEXT' '$OUTPUT_FILE'" || true

if [[ -f "$OUTPUT_FILE" ]]; then
    echo "Successfully created output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$OUTPUT_FILE" output.mp3
else
    echo "Failed to create output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Step 4: Using SSML for enhanced speech
echo "" | tee -a "$LOG_FILE"
echo "Step 4: Using SSML for enhanced speech" | tee -a "$LOG_FILE"
SSML_OUTPUT="${WORK_DIR}/ssml-output.mp3"
SSML_TEXT='<speak>Hello! <break time="1s"/> This is a sample of <emphasis>SSML
 enhanced speech</emphasis>.</speak>'
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Matthew --text-
type ssml --text '$SSML_TEXT' '$SSML_OUTPUT'" || true

if [[ -f "$SSML_OUTPUT" ]]; then
    echo "Successfully created ssml-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$SSML_OUTPUT" ssml-output.mp3
else
    echo "Failed to create ssml-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
```

```
fi

# Step 5: Working with lexicons
echo "" | tee -a "$LOG_FILE"
echo "Step 5: Working with lexicons" | tee -a "$LOG_FILE"

# Generate a random identifier for the lexicon (max 20 chars, alphanumeric only)
LEXICON_NAME="example$(openssl rand -hex 6 | cut -c 1-10)"
echo "Using lexicon name: $LEXICON_NAME" | tee -a "$LOG_FILE"

# Create a lexicon file
echo "Creating lexicon file..." | tee -a "$LOG_FILE"
LEXICON_FILE="${WORK_DIR}/example.pls"
cat > "$LEXICON_FILE" << 'EOF'
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa"
      xml:lang="en-US">
  <lexeme>
    <grapheme>AWS</grapheme>
    <alias>Amazon Web Services</alias>
  </lexeme>
</lexicon>
EOF

# Upload the lexicon
echo "Uploading lexicon..." | tee -a "$LOG_FILE"
LEXICON_ARN=$(aws polly put-lexicon --name "$LEXICON_NAME" --content
 file://"$LEXICON_FILE" --query 'LexiconArn' --output text 2>&1) || true
if [[ -n "$LEXICON_ARN" && "$LEXICON_ARN" != "" && ! "$LEXICON_ARN" =~ error ]];
 then
    echo "Lexicon uploaded with ARN: $LEXICON_ARN" | tee -a "$LOG_FILE"
    aws polly tag-resource --resource-arn "$LEXICON_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-polly-gs 2>&1 | tee -a
 "$LOG_FILE" || true
else
    echo "Lexicon uploaded." | tee -a "$LOG_FILE"
fi

# List available lexicons
```

```
echo "Listing available lexicons..." | tee -a "$LOG_FILE"
log_cmd "aws polly list-lexicons --output text --query 'Lexicons[*].[Name]'" || true

# Get details about the lexicon
echo "Getting details about the lexicon..." | tee -a "$LOG_FILE"
log_cmd "aws polly get-lexicon --name '$LEXICON_NAME' --output text --query
 'Lexicon.Name'" || true

# Use the lexicon when synthesizing speech
echo "Using the lexicon for speech synthesis..." | tee -a "$LOG_FILE"
LEXICON_OUTPUT="${WORK_DIR}/lexicon-output.mp3"
LEXICON_TEXT="I work with AWS every day."
log_cmd "aws polly synthesize-speech --output-format mp3 --voice-id Joanna --
lexicon-names '$LEXICON_NAME' --text '$LEXICON_TEXT' '$LEXICON_OUTPUT'" || true

if [[ -f "$LEXICON_OUTPUT" ]]; then
    echo "Successfully created lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    echo "You can play this file with your preferred audio player." | tee -a
 "$LOG_FILE"
    cp "$LEXICON_OUTPUT" lexicon-output.mp3
else
    echo "Failed to create lexicon-output.mp3 file." | tee -a "$LOG_FILE"
    exit 1
fi

# Summary of created resources
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "TUTORIAL SUMMARY" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Created resources:" | tee -a "$LOG_FILE"
echo "1. Lexicon: $LEXICON_NAME" | tee -a "$LOG_FILE"
echo "2. Audio files:" | tee -a "$LOG_FILE"
echo "   - output.mp3" | tee -a "$LOG_FILE"
echo "   - ssml-output.mp3" | tee -a "$LOG_FILE"
echo "   - lexicon-output.mp3" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Cleanup with auto-confirmation
echo "" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
```

```
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "============================================================" | tee -a
 "$LOG_FILE"
echo "Cleaning up all created resources..." | tee -a "$LOG_FILE"
cleanup

echo "" | tee -a "$LOG_FILE"
echo "Tutorial completed successfully!" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - DeleteLexicon
  - DescribeVoices
  - GetLexicon
  - Hilfe
  - ListLexicons
  - PutLexicon
  - SynthesizeSpeech

Amazon RDS-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Eine relationale Datenbank-Instance erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Netzwerkkomponenten ein
- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
```

```
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"
```

```
INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
```

```
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi
```

```
echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."
```

```
    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"

    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue

- Wait

Beispiele für Amazon Redshift mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Serverless-Data Warehouses

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace
- Verwenden Sie Rotverschiebung CreateWorkgroup
- Verwenden Sie Rotverschiebung DeleteNamespace
- Benutze iam CreateRole

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
```

```
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
```

```
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
```

```
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}
```

```
# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"
```

```
    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi
```

```
# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
```

```
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
```

```
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
```

```
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
```

```
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
```

```
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
```

```
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
```

```
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
```

```
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup

- GetNamespace
- GetRole
- GetSecretValue
- GetWorkgroup
- PutRolePolicy
- UpdateNamespace

Erste Schritte mit bereitgestellten Data Warehouse-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Redshift-Cluster
- Erstellen Sie eine IAM-Rolle für den S3-Zugriff
- Tabellen erstellen und Daten laden
- Führen Sie Beispielabfragen aus
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices

set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-final-
cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/dev/
null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json
```

```
    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --query
 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id "$statement_id"
 --query 'Error' --output text 2>/dev/null || echo "Unknown error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
```

```
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus" \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and try
 again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"
```

```
# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)
```

```
echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
```

```
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed to
 attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
```

```
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer not
 null, eventid integer not null, dateid smallint not null sortkey, qtysold smallint
 not null, pricepaid decimal(8,2), commission decimal(8,2), saletime timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint not
 null, holiday boolean default('N'));" \
  --query 'Id' --output text)
```

```
echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt' DELIMITER
 '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
```

```
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid AND
 caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "============================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

Amazon S3-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
```

```
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon S3 getting started demo."
  echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
  local bucket_name
  bucket_name=$(generate_random_name "$bucket_name_prefix")

  local region_code
  region_code=$(aws configure get region)

  if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
  else
    errecho "The bucket failed to create. This demo will exit."
    return 1
  fi

  local file_name
  while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
      echo "Could not find file $file_name. Are you sure it exists?"
      file_name=""
    fi
  done

  local key
  key="$(basename "$file_name")"
```

```
  local result=0
  if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
  else
    result=1
  fi

  local destination_file
  destination_file="$file_name.download"
  if yes_no_input "Would you like to download $key to the file $destination_file?
 (y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key"; then
      echo "Downloaded $key in the bucket $bucket_name to the file
 $destination_file."
    else
      result=1
    fi
  fi

  if yes_no_input "Would you like to copy $key a new object key in your bucket? (y/
n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
      echo "Copied $key in the bucket $bucket_name to the  $to_key."
    else
      result=1
    fi
  fi

  local bucket_items
  bucket_items=$(list_items_in_bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    result=1
  fi

  echo "Your bucket contains the following items."
  echo -e "Name\t\tSize"
  echo "$bucket_items"
```

```
  if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
 (y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

Die in diesem Szenario verwendeten Amazon-S3-Funktionen.

```
###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
```

```
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
```

```
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
```

```
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_item_in_bucket
#
```

```
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
```

```
##############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

##############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
```

```
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
```

```
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie CopyObject in der AWS CLI Befehlsreferenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
```

```
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
```

```
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
```

```
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie CreateBucket in der AWS CLI Befehlsreferenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
```

```
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie DeleteBucket in der AWS CLI Befehlsreferenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie DeleteObject in der AWS CLI Befehlsreferenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
```

```
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie DeleteObjects in der AWS CLI Befehlsreferenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
```

```
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie GetObject in der AWS CLI Befehlsreferenz.

`HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung`HeadBucket`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
##############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
##############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- API-Details finden Sie HeadBucket in der AWS CLI Befehlsreferenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie unter ListObjects V2 in der AWS CLI Befehlsreferenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3
```

```
  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- API-Details finden Sie PutObject in der AWS CLI Befehlsreferenz.

Szenarien

Erste Schritte mit großen Datenverarbeitungsclustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail
```

```
# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "============================================"
    echo "CLEANUP IN PROGRESS"
    echo "============================================"
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null || true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
 true
        echo "Cluster terminated successfully."
    fi
```

```
    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ]; then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
```

```
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
 "Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration
 Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 || true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
```

```
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://
emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-tutorial-
bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
 Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
 bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
```

```
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
 "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
 download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
 then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
 handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output text
 2>/dev/null || true)
```

```
if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:
[[:space:]]*"[^"]*' | grep -o 'j-[A-Z0-9]*' || true)
fi
```

```
if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
 failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
 "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
 "TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
  --cluster-id "$CLUSTER_ID" \
```

```
  --steps Type=Spark,Name="Health Violations
 Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
 handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID"
 --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output files"

# Download results
echo "Downloading results file"
```

```
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv" |
 head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
 CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv" |
 head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
 handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup
```

```
echo "Script completed successfully."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddSteps
  - Cp
  - CreateCluster
  - CreateDefaultRoles
  - CreateKeyPair
  - DescribeCluster
  - DescribeKeyPairs
  - DescribeStep
  - Ls
  - Mb
  - Rb
  - Rm
  - SSH
  - TerminateClusters
  - Wait

Erste Schritte mit der Konfigurationsverwaltung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config-Konfigurationsrekorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
```

```
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
```

```
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
```

```
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
```

```
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
```

```
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
```

```
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
```

```
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
```

```
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
```

```
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

Erste Schritte mit der Extraktion von Dokumententexten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines S3-Buckets
- Laden Sie ein Dokument auf S3 hoch
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "====================================================="
echo "This script will guide you through using Amazon Textract to analyze document
 text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete
 document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\).*/
\1=***REDACTED***/g'
    exit 1
fi
```

```
# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
```

```
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3:// 
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text 50,50
 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
```

```
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
```

```
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "==================================================="
echo "Analysis Summary"
echo "==================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk '{print
 $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l || echo
 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l || echo
 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l || echo
 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l || echo
 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"' |
 wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' | wc -
l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
```

```
echo "==================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-results.json)
 has been kept."

echo ""
echo "==================================================="
echo "Tutorial complete!"
echo "==================================================="
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AnalyzeDocument
  - Cp
  - Hilfe
  - Mb
  - Rb
  - Rm

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""
```

```
# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
```

```
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
```

```
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
```

```
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi
```

```
# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
```

```
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"
```

```
CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
```

```
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)
```

```
if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
```

```
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
```

```
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
```

```
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================="
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi
```

```
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

Erste Schritte mit Objektspeicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Ihren ersten S3-Bucket
- Hochladen eines Objekts
- Aktivieren von Versioning
- Konfigurieren Sie die Standardverschlüsselung
- Fügen Sie Ihrem Bucket Tags hinzu
- Objekte und Versionen auflisten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# ===============================================================================
# Prerequisites check
# ===============================================================================

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
 AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# ===============================================================================
# Setup: logging, temp directory, resource tracking
# ===============================================================================

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
```

```
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"
fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

echo "============================================="
echo "S3 Getting Started"
echo "============================================="
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# ============================================================================
# Helper functions
# ============================================================================

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0
```

```
    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
 then
        return 0
    fi

    echo "$versions" | jq -r '.[] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
 IFS=$'\t' read -r key version_id; do
        if [ -n "$key" ] && [ "$key" != "null" ]; then
            aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
 "$version_id" >/dev/null 2>&1 || true
        fi
    done

    return 0
}

# ============================================================================
# Error handling and cleanup functions
# ============================================================================

cleanup() {
    echo ""
    echo "============================================="
    echo "CLEANUP"
    echo "============================================="

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[].
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[].
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
```

```
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null;
 then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "============================================="
    echo "ERROR on line ${line_number}"
    echo "============================================="
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# ==============================================================================
# Step 1: Create a bucket
# ==============================================================================

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
```

```
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag bucket"
    fi
fi
echo ""

# ==============================================================================
# Step 2: Upload a sample text file
# ==============================================================================

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
```

```
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# ==============================================================================
# Step 3: Download the object
# ==============================================================================

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# ==============================================================================
# Step 4: Copy the object to a folder prefix
# ==============================================================================

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
```

```
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# ==============================================================================
# Step 5: Enable versioning and upload a second version
# ==============================================================================

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# ==============================================================================
# Step 6: Configure SSE-S3 encryption
# ==============================================================================

echo "Step 6: Configuring SSE-S3 default encryption..."
```

```
if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# ==============================================================================
# Step 7: Block all public access
# ==============================================================================

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# ==============================================================================
# Step 8: Configure bucket logging
# ==============================================================================

echo "Step 8: Configuring bucket logging..."
```

```
LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
 true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
 2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to tag log bucket"
    fi

    aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-write
 2>/dev/null || true

    if ! aws s3api put-bucket-logging \
        --bucket "$BUCKET_NAME" \
        --bucket-logging-status '{
            "LoggingEnabled": {
                "TargetBucket": "'$LOG_TARGET_BUCKET'",
                "TargetPrefix": "logs/"
            }
        }' >/dev/null 2>&1; then
        echo "WARNING: Failed to configure bucket logging"
```

```
    else
        echo "Bucket logging configured."
    fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# ==============================================================================
# Step 9: Tag the bucket
# ==============================================================================

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."
```

```
echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# ==============================================================================
# Step 10: List objects and versions
# ==============================================================================

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# ==============================================================================
# Step 11: Cleanup
# ==============================================================================

echo ""
echo "============================================"
echo "TUTORIAL COMPLETE"
echo "============================================"
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "============================================"
```

```
echo "CLEANUP"
echo "==========================================="
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - HeadObject
  - ListObjectVersions
  - ListObjectsV2
  - PutBucketEncryption
  - PutBucketTagging
  - PutBucketVersioning
  - PutObject
  - PutPublicAccessBlock

Erste Schritte mit Abfrageanalysen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen S3-Bucket für Abfrageergebnisse
- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Ausführen einer -Abfrage
- Erstellen und verwenden Sie benannte Abfragen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
```

```
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] || [ ${#bucket_name}
 -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi
```

```
# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
        --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
```

```
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-
z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION" 2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
athena-gs}]'

    # Security: Enable S3 bucket encryption with KMS validation
    echo "Enabling default encryption on S3 bucket..."
    if ! aws s3api put-bucket-encryption \
        --bucket "$S3_BUCKET" \
        --server-side-encryption-configuration '{
            "Rules": [{
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }]
        }' 2>&1; then
        echo "Warning: Could not enable encryption on bucket"
    fi

    # Security: Block public access
    echo "Blocking public access to S3 bucket..."
    if ! aws s3api put-public-access-block \
        --bucket "$S3_BUCKET" \
        --public-access-block-configuration \

 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1; then
        echo "Warning: Could not block public access on bucket"
    fi
```

```
    # Security: Enable versioning for data protection
    echo "Enabling versioning on S3 bucket..."
    if ! aws s3api put-bucket-versioning \
        --bucket "$S3_BUCKET" \
        --versioning-configuration Status=Enabled 2>&1; then
        echo "Warning: Could not enable versioning on bucket"
    fi

    echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
 || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
```

```
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
 "$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.$TABLE_NAME (
  \`Date\` DATE,
  Time STRING,
  Location STRING,
  Bytes INT,
  RequestIP STRING,
  Method STRING,
  Host STRING,
  Uri STRING,
  Status INT,
  Referrer STRING,
  os STRING,
  Browser STRING,
  BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  \"input.regex\" = \"^(?!#)([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s
+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+([^ ]+)\\\\s+[^\\\
\(]+[\\\\(]([^\\\\;]+).*\\\\%20([^\\\\/]+)[\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
```

```
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
```

```
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
```

```
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
 "$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
```

```
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/dev/
null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -d'"' -
f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
```

```
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null ||
 echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region "$AWS_REGION"
 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ]; then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
```

```
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "============================================"

# Auto-confirm cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
 "$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
 2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
 cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
```

```
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status: $QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \
        --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
 "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
 "$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
```

```
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
 "CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
 $QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
        if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
        else
            echo "S3 bucket deleted successfully."
        fi
    else
        echo "Skipping S3 bucket deletion (shared resource)"
    fi

    # Security: Remove downloaded query results
    if [ -f "./query-results.csv" ]; then
        if command -v shred &>/dev/null; then
            shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
        else
            rm -f "./query-results.csv"
        fi
        echo "Query results file securely removed."
```

```
    fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - Cp
  - CreateNamedQuery
  - DeleteNamedQuery
  - GetNamedQuery
  - GetQueryExecution
  - GetQueryResults
  - ListDatabases
  - ListNamedQueries
  - ListTableMetadata
  - Mb
  - Rb
  - Rm
  - StartQueryExecution

SageMaker KI-Beispiele unter Verwendung AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit SageMaker KI Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Feature Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"
```

```
        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
```

```
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
```

```
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"
```

```
# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
```

```
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
```

```
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
```

```
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"
```

```
CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
```

```
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
```

```
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
```

```
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup

- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Secrets Manager-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie das with Bash-Skript AWS Command Line Interface mit Secrets Manager verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Eine relationale Datenbankinstanz erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Netzwerkkomponenten ein
- Erstellen einer DB-Subnetzgruppe
- Erstellen einer DB-Instance

- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create an Amazon RDS DB instance
# This script follows the tutorial at https://docs.aws.amazon.com/AmazonRDS/latest/
UserGuide/USER_CreateDBInstance.html

# Set up logging
LOG_FILE="rds_creation_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting RDS DB instance creation script - $(date)"
echo "All actions will be logged to $LOG_FILE"
echo "==============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$DB_INSTANCE_ID" ]; then
```

```
        echo "Deleting DB instance $DB_INSTANCE_ID..."
        aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --
skip-final-snapshot
        echo "Waiting for DB instance to be deleted..."
        aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
    fi

    if [ -n "$DB_SUBNET_GROUP_NAME" ] && [ "$CREATED_SUBNET_GROUP" = "true" ]; then
        echo "Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
        aws rds delete-db-subnet-group --db-subnet-group-name
 "$DB_SUBNET_GROUP_NAME"
    fi

    if [ -n "$SECURITY_GROUP_ID" ] && [ "$CREATED_SECURITY_GROUP" = "true" ]; then
        echo "Deleting security group $SECURITY_GROUP_ID..."
        aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
    fi

    echo "Cleanup completed."
}

# Generate a random identifier for resources
RANDOM_ID=$(openssl rand -hex 4)
DB_INSTANCE_ID="mydb-${RANDOM_ID}"
DB_SUBNET_GROUP_NAME="mydbsubnet-${RANDOM_ID}"
SECURITY_GROUP_NAME="mydbsg-${RANDOM_ID}"

# Track created resources
CREATED_SECURITY_GROUP="false"
CREATED_SUBNET_GROUP="false"

# Array to store created resources for display
declare -a CREATED_RESOURCES

echo "Step 1: Checking for default VPC..."
VPC_OUTPUT=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true")
check_error "$VPC_OUTPUT" "aws ec2 describe-vpcs"

# Extract VPC ID
VPC_ID=$(echo "$VPC_OUTPUT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)

if [ -z "$VPC_ID" ]; then
    echo "No default VPC found. Please create a VPC before running this script."
    exit 1
```

```
fi

echo "Using VPC: $VPC_ID"

echo "Step 2: Getting subnets from the VPC..."
SUBNET_OUTPUT=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID")
check_error "$SUBNET_OUTPUT" "aws ec2 describe-subnets"

# Extract subnet IDs (we need at least 2 in different AZs)
SUBNET_IDS=($(echo "$SUBNET_OUTPUT" | grep -o '"SubnetId": "[^"]*' | cut -d'"' -f4))

if [ ${#SUBNET_IDS[@]} -lt 2 ]; then
    echo "Error: Need at least 2 subnets in different AZs. Found ${#SUBNET_IDS[@]}
 subnets."
    exit 1
fi

echo "Found ${#SUBNET_IDS[@]} subnets: ${SUBNET_IDS[*]}"

echo "Step 3: Creating security group for RDS..."
SG_OUTPUT=$(aws ec2 create-security-group \
    --group-name "$SECURITY_GROUP_NAME" \
    --description "Security group for RDS database access" \
    --vpc-id "$VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=rds-gs}]')
check_error "$SG_OUTPUT" "aws ec2 create-security-group"

SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*' | cut -d'"' -f4)
CREATED_SECURITY_GROUP="true"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID ($SECURITY_GROUP_NAME)")

echo "Created security group: $SECURITY_GROUP_ID"

echo "Step 4: Adding inbound rule to security group..."
# Note: In a production environment, you should restrict this to specific IP ranges
# We're using the local machine's IP address for this example
MY_IP=$(curl -s https://checkip.amazonaws.com)
check_error "$MY_IP" "curl -s https://checkip.amazonaws.com"

INGRESS_OUTPUT=$(aws ec2 authorize-security-group-ingress \
    --group-id "$SECURITY_GROUP_ID" \
    --protocol tcp \
    --port 3306 \
```

```
    --cidr "${MY_IP}/32")
check_error "$INGRESS_OUTPUT" "aws ec2 authorize-security-group-ingress"

echo "Added inbound rule to allow MySQL connections from ${MY_IP}/32"

echo "Step 5: Creating DB subnet group..."
# Select the first two subnets for the DB subnet group
SUBNET1=${SUBNET_IDS[0]}
SUBNET2=${SUBNET_IDS[1]}

SUBNET_GROUP_OUTPUT=$(aws rds create-db-subnet-group \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for RDS tutorial" \
    --subnet-ids "$SUBNET1" "$SUBNET2" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SUBNET_GROUP_OUTPUT" "aws rds create-db-subnet-group"

CREATED_SUBNET_GROUP="true"
CREATED_RESOURCES+=("DB Subnet Group: $DB_SUBNET_GROUP_NAME")

echo "Created DB subnet group: $DB_SUBNET_GROUP_NAME"

echo "Step 6: Creating a secure password in AWS Secrets Manager..."
SECRET_NAME="rds-db-credentials-${RANDOM_ID}"
SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "RDS DB credentials for $DB_INSTANCE_ID" \
    --secret-string '{"username":"adminuser","password":"'"$(openssl rand -base64
 16)"'"}' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$SECRET_OUTPUT" "aws secretsmanager create-secret"

SECRET_ARN=$(echo "$SECRET_OUTPUT" | grep -o '"ARN": "[^"]*' | cut -d'"' -f4)
CREATED_RESOURCES+=("Secret: $SECRET_ARN ($SECRET_NAME)")

echo "Created secret: $SECRET_NAME"

echo "Step 7: Retrieving the username and password from the secret..."
SECRET_VALUE_OUTPUT=$(aws secretsmanager get-secret-value --secret-id "$SECRET_NAME"
 --query 'SecretString' --output text)
check_error "$SECRET_VALUE_OUTPUT" "aws secretsmanager get-secret-value"

DB_USERNAME=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"username":"[^"]*' | cut -d'"'
 -f4)
```

```
DB_PASSWORD=$(echo "$SECRET_VALUE_OUTPUT" | grep -o '"password":"[^"]*' | cut -d'"'
 -f4)

echo "Retrieved database credentials"

echo "Step 8: Creating RDS DB instance..."
echo "This may take several minutes..."

DB_OUTPUT=$(aws rds create-db-instance \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --db-instance-class db.t3.micro \
    --engine mysql \
    --master-username "$DB_USERNAME" \
    --master-user-password "$DB_PASSWORD" \
    --allocated-storage 20 \
    --vpc-security-group-ids "$SECURITY_GROUP_ID" \
    --db-subnet-group-name "$DB_SUBNET_GROUP_NAME" \
    --backup-retention-period 7 \
    --no-publicly-accessible \
    --no-multi-az \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=rds-gs)
check_error "$DB_OUTPUT" "aws rds create-db-instance"

CREATED_RESOURCES+=("DB Instance: $DB_INSTANCE_ID")

echo "DB instance creation initiated: $DB_INSTANCE_ID"
echo "Waiting for DB instance to become available..."
echo "This may take 5-10 minutes..."

aws rds wait db-instance-available --db-instance-identifier "$DB_INSTANCE_ID"
DB_STATUS=$?

if [ $DB_STATUS -ne 0 ]; then
    echo "Error waiting for DB instance to become available"
    cleanup_on_error
    exit 1
fi

echo "DB instance is now available!"

echo "Step 9: Getting connection information..."
ENDPOINT_INFO=$(aws rds describe-db-instances \
    --db-instance-identifier "$DB_INSTANCE_ID" \
    --query 'DBInstances[0].[Endpoint.Address,Endpoint.Port,MasterUsername]' \
```

```
    --output text)
check_error "$ENDPOINT_INFO" "aws rds describe-db-instances"

DB_ENDPOINT=$(echo "$ENDPOINT_INFO" | awk '{print $1}')
DB_PORT=$(echo "$ENDPOINT_INFO" | awk '{print $2}')
DB_USER=$(echo "$ENDPOINT_INFO" | awk '{print $3}')

echo "================================================"
echo "DB Instance successfully created!"
echo "================================================"
echo "Connection Information:"
echo "  Endpoint: $DB_ENDPOINT"
echo "  Port: $DB_PORT"
echo "  Username: $DB_USER"
echo "  Password: [Stored in AWS Secrets Manager - $SECRET_NAME]"
echo ""
echo "To connect using the mysql client:"
echo "mysql -h $DB_ENDPOINT -P $DB_PORT -u $DB_USER -p"
echo "================================================"

echo ""
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done
echo ""

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy] ]]; then
    echo "Starting cleanup process..."

    echo "Step 1: Deleting DB instance $DB_INSTANCE_ID..."
    aws rds delete-db-instance --db-instance-identifier "$DB_INSTANCE_ID" --skip-
final-snapshot
    echo "Waiting for DB instance to be deleted..."
    aws rds wait db-instance-deleted --db-instance-identifier "$DB_INSTANCE_ID"
```

```
    echo "Step 2: Deleting secret $SECRET_NAME..."
    aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery

    echo "Step 3: Deleting DB subnet group $DB_SUBNET_GROUP_NAME..."
    aws rds delete-db-subnet-group --db-subnet-group-name "$DB_SUBNET_GROUP_NAME"

    echo "Step 4: Deleting security group $SECURITY_GROUP_ID..."
    aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"

    echo "Cleanup completed successfully!"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
    echo "To clean up later, you'll need to delete these resources manually."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateSecret
  - CreateSecurityGroup
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DeleteSecurityGroup
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - Wait

Erste Schritte mit Serverless-Data Warehouses

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Benutze den Secrets-Manager CreateSecret
- Verwenden Sie den Secrets-Manager DeleteSecret
- Verwenden Sie den Secrets-Manager GetSecretValue
- Benutze Redshift CreateNamespace
- Verwenden Sie Rotverschiebung CreateWorkgroup
- Verwenden Sie Rotverschiebung DeleteNamespace
- Benutze iam CreateRole

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq dependency)
# This script creates a Redshift Serverless environment, loads sample data, and runs
 queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
```

```
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done
```

```
  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}

# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
```

```
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}

# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status: $status.
 Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5
```

```
  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "============================================"
  echo "CLEANUP CONFIRMATION"
  echo "============================================"
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --workgroup-
name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
```

```
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --namespace-
name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
```

```
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin credentials
 for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"
```

```
# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"

# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless get-
namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless get-
workgroup --workgroup-name $WORKGROUP_NAME"
```

```
# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)
```

```
# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5

# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
```

```
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -
f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
```

```
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID" 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi

# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time total
 sales..."
```

```
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price desc)
 as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID" 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
```

```
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "============================================"
echo "TUTORIAL SUMMARY"
echo "============================================"
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources
```

```
echo "Tutorial completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

Erste Schritte mit Dokumentdatenbanken

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer DB-Subnetzgruppe
- Erstellen Sie einen DocumentDB-Cluster
- Erstellen Sie eine DocumentDB-Instanz
- Konfigurieren Sie Sicherheit und Konnektivität
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
# Amazon DocumentDB - Getting Started
# This script creates a DocumentDB cluster with encrypted storage, stores the
# master password in Secrets Manager, and displays connection information.

set -eE

################################################################################
# Configuration
################################################################################
SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' </dev/urandom | head -c 8)
CLUSTER_ID="docdb-gs-${SUFFIX}"
INSTANCE_ID="${CLUSTER_ID}-inst"
SUBNET_GROUP_NAME="docdb-subnet-${SUFFIX}"
SECRET_NAME="docdb-secret-${SUFFIX}"
MASTER_USER="docdbadmin"
ENGINE_VERSION="5.0.0"
INSTANCE_CLASS="db.t3.medium"
DOCDB_PORT=27017
WAIT_TIMEOUT=900

TEMP_DIR=$(mktemp -d)
LOG_FILE="${TEMP_DIR}/documentdb-gs.log"
trap 'rm -rf "$TEMP_DIR"' EXIT

CREATED_RESOURCES=()

################################################################################
# Logging
################################################################################
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Log file: $LOG_FILE"
echo ""
```

```
##############################################################################
# Region pre-check
##############################################################################
CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi
REGION="${AWS_REGION:-${AWS_DEFAULT_REGION:-$CONFIGURED_REGION}}"
echo "Using region: $REGION"
echo ""

##############################################################################
# Error handler
##############################################################################
handle_error() {
    echo ""
    echo "==========================================="
    echo "ERROR at $1"
    echo "==========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Resources created before error:"
        for r in "${CREATED_RESOURCES[@]}"; do
            echo "  - $r"
        done
        echo ""
        echo "Attempting cleanup..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR

##############################################################################
# Wait function
##############################################################################
wait_for_status() {
    local resource_type="$1"
    local resource_id="$2"
```

```
    local target_status="$3"
    local timeout="${4:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to reach '$target_status'..."

    while true; do
        local current_status=""
        if [ "$resource_type" = "cluster" ]; then
            current_status=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1)
        elif [ "$resource_type" = "instance" ]; then
            current_status=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1)
        fi

        if echo "$current_status" | grep -iq "error"; then
            echo "ERROR checking status: $current_status"
            return 1
        fi

        echo "  Status: $current_status ($elapsed/${timeout}s)"

        if [ "$current_status" = "$target_status" ]; then
            echo "  $resource_type '$resource_id' is now '$target_status'."
            return 0
        fi

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "ERROR: Timed out after ${timeout}s waiting for $resource_type
 '$resource_id'."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

###############################################################################
# Wait for deletion
```

```
##############################################################################
wait_for_deletion() {
    local resource_type="$1"
    local resource_id="$2"
    local timeout="${3:-$WAIT_TIMEOUT}"
    local elapsed=0
    local interval=30

    echo "Waiting for $resource_type '$resource_id' to be deleted..."

    while true; do
        local result=""
        if [ "$resource_type" = "cluster" ]; then
            result=$(aws docdb describe-db-clusters \
                --db-cluster-identifier "$resource_id" \
                --query "DBClusters[0].Status" --output text 2>&1) || true
        elif [ "$resource_type" = "instance" ]; then
            result=$(aws docdb describe-db-instances \
                --db-instance-identifier "$resource_id" \
                --query "DBInstances[0].DBInstanceStatus" --output text 2>&1) ||
 true
        fi

        if echo "$result" | grep -iq "DBClusterNotFoundFault\|DBInstanceNotFound\|
not found"; then
            echo "  $resource_type '$resource_id' deleted."
            return 0
        fi

        echo "  Still deleting... ($elapsed/${timeout}s)"

        if [ "$elapsed" -ge "$timeout" ]; then
            echo "WARNING: Timed out waiting for $resource_type '$resource_id'
 deletion."
            return 1
        fi

        sleep "$interval"
        elapsed=$((elapsed + interval))
    done
}

##############################################################################
# Cleanup
```

```
##############################################################################
cleanup_resources() {
    echo ""
    echo "Cleaning up resources..."
    echo ""

    # Revoke security group ingress rule
    if [ -n "${SG_ID:-}" ] && [ -n "${MY_IP:-}" ]; then
        echo "Revoking security group ingress rule..."
        aws ec2 revoke-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1 || echo "WARNING: Failed to revoke SG ingress
 rule."
    fi

    # Delete instance (must be deleted before cluster)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "instance:"; then
        echo "Deleting instance '${INSTANCE_ID}'..."
        aws docdb delete-db-instance \
            --db-instance-identifier "$INSTANCE_ID" 2>&1 || echo "WARNING: Failed to
 delete instance."
        wait_for_deletion "instance" "$INSTANCE_ID" || true
    fi

    # Delete cluster (skip final snapshot)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "cluster:"; then
        echo "Deleting cluster '${CLUSTER_ID}'..."
        aws docdb delete-db-cluster \
            --db-cluster-identifier "$CLUSTER_ID" \
            --skip-final-snapshot 2>&1 || echo "WARNING: Failed to delete cluster."
        wait_for_deletion "cluster" "$CLUSTER_ID" || true
    fi

    # Delete subnet group (must wait for cluster deletion)
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "subnet-group:"; then
        echo "Deleting subnet group '${SUBNET_GROUP_NAME}'..."
        aws docdb delete-db-subnet-group \
            --db-subnet-group-name "$SUBNET_GROUP_NAME" 2>&1 || echo "WARNING:
 Failed to delete subnet group."
    fi

    # Delete secret with scheduled deletion instead of immediate deletion
```

```
    if printf '%s\n' "${CREATED_RESOURCES[@]}" | grep -q "secret:"; then
        echo "Deleting secret '${SECRET_NAME}' (scheduled for 7 days)..."
        aws secretsmanager delete-secret \
            --secret-id "$SECRET_NAME" \
            --recovery-window-in-days 7 2>&1 || echo "WARNING: Failed to delete
 secret."
    fi

    echo ""
    echo "Cleanup complete."
}

################################################################################
# Step 1: Generate password and store in Secrets Manager
################################################################################
echo "==========================================="
echo "Step 1: Create master password in Secrets Manager"
echo "==========================================="
echo ""

# Generate a strong password using openssl for better randomness
# Meets DocumentDB requirements: 8-100 chars, alphanumeric + special chars
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d '/' | cut -c1-20)

# Securely store password in temporary file with restricted permissions
TEMP_PASS_FILE=$(mktemp)
chmod 600 "$TEMP_PASS_FILE"
echo -n "$MASTER_PASSWORD" > "$TEMP_PASS_FILE"
trap "rm -f '$TEMP_PASS_FILE'; rm -rf '$TEMP_DIR'" EXIT

SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "DocumentDB master password for ${CLUSTER_ID}" \
    --secret-string file://"$TEMP_PASS_FILE" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --output text --query "ARN" 2>&1)

# Securely clear password from memory
MASTER_PASSWORD=""

if echo "$SECRET_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating secret: $SECRET_OUTPUT"
    exit 1
fi
```

```
SECRET_ARN="$SECRET_OUTPUT"
CREATED_RESOURCES+=("secret:${SECRET_NAME}")
echo "Secret created: $SECRET_NAME"
echo "Secret ARN: $SECRET_ARN"
echo ""

##############################################################################
# Step 2: Find default VPC and subnets
##############################################################################
echo "============================================"
echo "Step 2: Find default VPC and subnets"
echo "============================================"
echo ""

VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=isDefault,Values=true" \
    --query "Vpcs[0].VpcId" --output text 2>&1)

if echo "$VPC_ID" | grep -iq "error"; then
    echo "ERROR finding default VPC: $VPC_ID"
    exit 1
fi

if [ "$VPC_ID" = "None" ] || [ -z "$VPC_ID" ]; then
    echo "ERROR: No default VPC found. Create one with 'aws ec2 create-default-
vpc'."
    exit 1
fi

echo "Default VPC: $VPC_ID"

# Get subnets in at least 2 different AZs (space-separated)
SUBNET_INFO=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=${VPC_ID}" "Name=default-for-az,Values=true" \
    --query "Subnets[*].[SubnetId,AvailabilityZone]" --output text 2>&1)

if echo "$SUBNET_INFO" | grep -iq "error"; then
    echo "ERROR finding subnets: $SUBNET_INFO"
    exit 1
fi

# Collect unique AZs and their subnet IDs
declare -A AZ_SUBNETS
```

```
while IFS=$'\t' read -r sid az; do
    if [ -z "${AZ_SUBNETS[$az]+x}" ]; then
        AZ_SUBNETS[$az]="$sid"
    fi
done <<< "$SUBNET_INFO"

AZ_COUNT=${#AZ_SUBNETS[@]}
if [ "$AZ_COUNT" -lt 2 ]; then
    echo "ERROR: DocumentDB requires subnets in at least 2 AZs. Found $AZ_COUNT."
    exit 1
fi

# Build space-separated subnet ID list
SUBNET_IDS=""
for az in "${!AZ_SUBNETS[@]}"; do
    if [ -n "$SUBNET_IDS" ]; then
        SUBNET_IDS="${SUBNET_IDS} ${AZ_SUBNETS[$az]}"
    else
        SUBNET_IDS="${AZ_SUBNETS[$az]}"
    fi
done

echo "Subnets (${AZ_COUNT} AZs): $SUBNET_IDS"
echo ""

################################################################################
# Step 3: Create subnet group
################################################################################
echo "==========================================="
echo "Step 3: Create DocumentDB subnet group"
echo "==========================================="
echo ""

SUBNET_GROUP_OUTPUT=$(aws docdb create-db-subnet-group \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --db-subnet-group-description "Subnet group for DocumentDB getting started" \
    --subnet-ids $SUBNET_IDS \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBSubnetGroup.DBSubnetGroupName" --output text 2>&1)

if echo "$SUBNET_GROUP_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating subnet group: $SUBNET_GROUP_OUTPUT"
    exit 1
fi
```

```
CREATED_RESOURCES+=("subnet-group:${SUBNET_GROUP_NAME}")
echo "Subnet group created: $SUBNET_GROUP_NAME"
echo ""

################################################################################
# Step 4: Create DocumentDB cluster
################################################################################
echo "==========================================="
echo "Step 4: Create DocumentDB cluster"
echo "==========================================="
echo ""

# Read password securely from file for cluster creation
MASTER_PASSWORD=$(cat "$TEMP_PASS_FILE")

CLUSTER_OUTPUT=$(aws docdb create-db-cluster \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --engine-version "$ENGINE_VERSION" \
    --master-username "$MASTER_USER" \
    --master-user-password "$MASTER_PASSWORD" \
    --db-subnet-group-name "$SUBNET_GROUP_NAME" \
    --storage-encrypted \
    --kms-key-id "alias/aws/docdb" \
    --no-deletion-protection \
    --enable-cloudwatch-logs-exports '["audit","error","general","slowquery"]' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBCluster.DBClusterIdentifier" --output text 2>&1)

# Clear password immediately after use
MASTER_PASSWORD=""

if echo "$CLUSTER_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating cluster: $CLUSTER_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("cluster:${CLUSTER_ID}")
echo "Cluster created: $CLUSTER_ID"
echo ""

wait_for_status "cluster" "$CLUSTER_ID" "available"
echo ""
```

```
################################################################################
# Step 5: Create DocumentDB instance
################################################################################
echo "============================================="
echo "Step 5: Create DocumentDB instance"
echo "============================================="
echo ""

INSTANCE_OUTPUT=$(aws docdb create-db-instance \
    --db-instance-identifier "$INSTANCE_ID" \
    --db-instance-class "$INSTANCE_CLASS" \
    --db-cluster-identifier "$CLUSTER_ID" \
    --engine docdb \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=documentdb-gs \
    --query "DBInstance.DBInstanceIdentifier" --output text 2>&1)

if echo "$INSTANCE_OUTPUT" | grep -iq "error"; then
    echo "ERROR creating instance: $INSTANCE_OUTPUT"
    exit 1
fi

CREATED_RESOURCES+=("instance:${INSTANCE_ID}")
echo "Instance created: $INSTANCE_ID"
echo ""

wait_for_status "instance" "$INSTANCE_ID" "available"
echo ""

################################################################################
# Step 6: Get cluster endpoint and security group
################################################################################
echo "============================================="
echo "Step 6: Get cluster endpoint and security group"
echo "============================================="
echo ""

CLUSTER_DETAILS=$(aws docdb describe-db-clusters \
    --db-cluster-identifier "$CLUSTER_ID" \
    --query "DBClusters[0].[Endpoint,VpcSecurityGroups[0].VpcSecurityGroupId]" \
    --output text 2>&1)

if echo "$CLUSTER_DETAILS" | grep -iq "error"; then
    echo "ERROR getting cluster details: $CLUSTER_DETAILS"
```

```
    exit 1
fi

CLUSTER_ENDPOINT=$(echo "$CLUSTER_DETAILS" | awk '{print $1}')
SG_ID=$(echo "$CLUSTER_DETAILS" | awk '{print $2}')

echo "Cluster endpoint: $CLUSTER_ENDPOINT"
echo "Security group: $SG_ID"
echo ""

################################################################################
# Step 7: Add security group ingress for port 27017 from user's IP
################################################################################
echo "============================================"
echo "Step 7: Add security group ingress rule"
echo "============================================"
echo ""

# Get the user's public IP with timeout and error handling
MY_IP=$(timeout 5 curl -s --max-time 5 https://checkip.amazonaws.com 2>/dev/null ||
 true)

if [ -z "$MY_IP" ] || echo "$MY_IP" | grep -iq "error\|could not\|failed"; then
    echo "WARNING: Could not determine public IP address. Skipping security group
 rule."
    echo "You must manually add an ingress rule for your IP to security group
 $SG_ID"
    MY_IP=""
else
    # Trim whitespace
    MY_IP=$(echo "$MY_IP" | tr -d '[:space:]')

    # Validate IP format (basic check)
    if ! echo "$MY_IP" | grep -qE '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}$'; then
        echo "WARNING: Invalid IP address format: $MY_IP. Skipping security group
 rule."
        MY_IP=""
    else
        echo "Your public IP: $MY_IP"

        SG_RULE_OUTPUT=$(aws ec2 authorize-security-group-ingress \
            --group-id "$SG_ID" \
            --protocol tcp \
```

```
            --port "$DOCDB_PORT" \
            --cidr "${MY_IP}/32" 2>&1)

        if echo "$SG_RULE_OUTPUT" | grep -iq "error"; then
            # Ignore if rule already exists
            if echo "$SG_RULE_OUTPUT" | grep -iq "Duplicate"; then
                echo "Ingress rule already exists."
            else
                echo "ERROR adding ingress rule: $SG_RULE_OUTPUT"
                exit 1
            fi
        else
            echo "Ingress rule added: TCP ${DOCDB_PORT} from ${MY_IP}/32"
            CREATED_RESOURCES+=("sg-rule:${SG_ID}:${MY_IP}")
        fi
    fi
fi

echo ""

################################################################################
# Step 8: Download CA certificate
################################################################################
echo "============================================"
echo "Step 8: Download Amazon DocumentDB CA certificate"
echo "============================================"
echo ""

CA_CERT_PATH="${TEMP_DIR}/global-bundle.pem"
CA_CERT_URL="https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem"

if timeout 10 curl -s --max-time 10 -o "$CA_CERT_PATH" "$CA_CERT_URL" 2>&1; then
    if [ -s "$CA_CERT_PATH" ]; then
        # Verify it's a valid PEM file and check file permissions
        if grep -q "BEGIN CERTIFICATE" "$CA_CERT_PATH"; then
            chmod 644 "$CA_CERT_PATH"
            echo "CA certificate downloaded to: $CA_CERT_PATH"
        else
            echo "WARNING: Downloaded file is not a valid PEM certificate."
            CA_CERT_PATH=""
        fi
    else
        echo "WARNING: Failed to download CA certificate (empty file)."
        CA_CERT_PATH=""
```

```
    fi
else
    echo "WARNING: Failed to download CA certificate (timeout or network error)."
    CA_CERT_PATH=""
fi

echo ""

##############################################################################
# Step 9: Display connection information
##############################################################################
echo "============================================"
echo "CONNECTION INFORMATION"
echo "============================================"
echo ""
echo "Cluster endpoint : $CLUSTER_ENDPOINT"
echo "Port             : $DOCDB_PORT"
echo "Master username  : $MASTER_USER"
echo "Secret name      : $SECRET_NAME (contains password)"
echo "Security group   : $SG_ID"
if [ -n "$CA_CERT_PATH" ]; then
    echo "CA certificate   : $CA_CERT_PATH"
fi
echo ""
echo "To connect with mongosh:"
if [ -n "$CA_CERT_PATH" ]; then
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} --tlsCAFile ${CA_CERT_PATH} \\"
else
    echo "  mongosh --tls --host ${CLUSTER_ENDPOINT} \\"
fi
echo "    --retryWrites false --username ${MASTER_USER} --password \$(aws
 secretsmanager get-secret-value --secret-id ${SECRET_NAME} --query SecretString --
output text)"
echo ""

##############################################################################
# Step 10: Cleanup
##############################################################################
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo ""
echo "Resources created:"
```

```
for r in "${CREATED_RESOURCES[@]}"; do
    echo "  - $r"
done
echo ""
echo "Automatically cleaning up all created resources..."
echo ""

cleanup_resources

echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AuthorizeSecurityGroupIngress
  - CreateDbCluster
  - CreateDbInstance
  - CreateDbSubnetGroup
  - CreateDefaultVpc
  - CreateSecret
  - DeleteDbCluster
  - DeleteDbInstance
  - DeleteDbSubnetGroup
  - DeleteSecret
  - DescribeDbClusters
  - DescribeDbInstances
  - DescribeSubnets
  - DescribeVpcs
  - GetSecretValue
  - RevokeSecurityGroupIngress
  - Wait

Fest codierte Geheimnisse in einen sicheren Geheimspeicher verschieben

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen

- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisiere das Geheimnis
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
```

```
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
```

```
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."
```

```
# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
```

```
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
```

```
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
```

```
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
```

```
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole

- DeleteSecret
- DetachRolePolicy
- GetCallerIdentity
- GetSecretValue
- PutResourcePolicy
- UpdateSecret

Amazon SES-Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

E-Mail-Versanddienste einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Bestätigen Sie eine E-Mail-Adresse
- Verifizieren Sie eine Domain (optional)
- Überprüfe deine Sendelimits
- Senden Sie eine Test-E-Mail
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SES Setup Script (v2)
# This script helps you set up Amazon SES for sending emails

# Initialize log file
LOG_FILE="ses-setup.log"
echo "Starting Amazon SES setup at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_cmd() {
    echo "Running: $1" | tee -a "$LOG_FILE"
    eval "$1" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local ignore_error="${4:-false}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        if [[ "$ignore_error" != "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}

# Function to clean up resources
cleanup_resources() {
```

```
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"

    # No physical resources to clean up for SES setup
    # Email identities can be deleted if needed
    if [[ -n "$EMAIL_ADDRESS" ]]; then
        echo "Deleting email identity: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$EMAIL_ADDRESS\""
    fi

    if [[ -n "$RECIPIENT_EMAIL" && "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
        echo "Deleting recipient email identity: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$RECIPIENT_EMAIL\""
    fi

    if [[ -n "$DOMAIN_NAME" ]]; then
        echo "Deleting domain identity: $DOMAIN_NAME" | tee -a "$LOG_FILE"
        log_cmd "aws ses delete-identity --identity \"$DOMAIN_NAME\""
    fi
}

# Track created resources
CREATED_RESOURCES=()

# Welcome message
echo "==============================================="
echo "Amazon SES Setup Script"
echo "==============================================="
echo "This script will help you set up Amazon SES for sending emails."
echo "You'll need to verify at least one email address that you own."
echo ""
echo "NOTE: New SES accounts are placed in the sandbox environment."
echo "In the sandbox, both sender AND recipient email addresses must be verified."
echo ""

# Get email address to verify
echo "Please enter an email address that you own and can access:"
EMAIL_ADDRESS="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"
echo "$EMAIL_ADDRESS"

# Verify email identity
echo "Verifying email address: $EMAIL_ADDRESS" | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address \"$EMAIL_ADDRESS\"")
```

```
check_error "$OUTPUT" $? "Failed to verify email address"

CREATED_RESOURCES+=("Email identity: $EMAIL_ADDRESS")
echo "A verification email has been sent to $EMAIL_ADDRESS."
echo "Please check your inbox and click the verification link before continuing."
echo ""
echo "Press Enter after you've verified your email address..."
sleep 1

# Check verification status
echo "Checking verification status..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses list-identities --identity-type EmailAddress")
check_error "$OUTPUT" $? "Failed to list identities"

OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --identities
 \"$EMAIL_ADDRESS\"")
check_error "$OUTPUT" $? "Failed to get verification attributes"

# Check if the email is verified
VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o '"VerificationStatus": "[^"]*' | cut
 -d'"' -f4)
if [[ "$VERIFICATION_STATUS" != "Success" ]]; then
    echo "Email address $EMAIL_ADDRESS is not verified yet. Please check your inbox
 and verify before continuing."
    echo "Exiting script..."
    exit 1
fi

# Ask if user wants to verify a domain
echo ""
echo "Do you want to verify a domain for sending emails? (y/n):"
VERIFY_DOMAIN="y"
echo "$VERIFY_DOMAIN"

if [[ "$VERIFY_DOMAIN" =~ ^[Yy] ]]; then
    echo "Please enter the domain name you want to verify:"
    DOMAIN_NAME="test-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1).example.com"
    echo "$DOMAIN_NAME"

    # Verify domain identity
    echo "Verifying domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses verify-domain-identity --domain \"$DOMAIN_NAME\"")
    check_error "$OUTPUT" $? "Failed to verify domain identity"
```

```
   # Extract verification token
   VERIFICATION_TOKEN=$(echo "$OUTPUT" | grep -o '"VerificationToken": "[^"]*' |
cut -d'"' -f4)

   CREATED_RESOURCES+=("Domain identity: $DOMAIN_NAME")

   echo ""
   echo "==============================================="
   echo "Domain Verification Instructions"
   echo "==============================================="
   echo "To verify your domain ownership, you need to add a TXT record"
   echo "to your domain's DNS settings with the following values:"
   echo ""
   echo "Record Type: TXT"
   echo "Record Name: _amazonses.$DOMAIN_NAME"
   echo "Record Value: $VERIFICATION_TOKEN"
   echo ""
   echo "After adding this DNS record, verification may take up to 72 hours."
   echo ""

   # Set up DKIM for the domain
   echo "Setting up DKIM for domain: $DOMAIN_NAME" | tee -a "$LOG_FILE"
   OUTPUT=$(log_cmd "aws ses verify-domain-dkim --domain \"$DOMAIN_NAME\"")
   check_error "$OUTPUT" $? "Failed to set up DKIM"

   # Extract DKIM tokens
   DKIM_TOKENS=$(echo "$OUTPUT" | grep -o '"DkimTokens": \[[^]]*\]' | sed
's/"DkimTokens": \[\|\]//g' | sed 's/,//g' | sed 's/"//g')

   echo "==============================================="
   echo "DKIM Configuration Instructions"
   echo "==============================================="
   echo "To configure DKIM for your domain, add the following CNAME records"
   echo "to your domain's DNS settings:"
   echo ""

   for token in $DKIM_TOKENS; do
       echo "Record Type: CNAME"
       echo "Record Name: ${token}._domainkey.$DOMAIN_NAME"
       echo "Record Value: ${token}.dkim.amazonses.com"
       echo ""
   done
```

```
    echo "After adding these DNS records, DKIM verification may take up to 72
 hours."
    echo ""
fi

# Check sending limits
echo "Checking your SES sending limits..." | tee -a "$LOG_FILE"
OUTPUT=$(log_cmd "aws ses get-send-quota")
check_error "$OUTPUT" $? "Failed to get sending quota"

# Extract quota information
MAX_SEND_RATE=$(echo "$OUTPUT" | grep -o '"MaxSendRate": [0-9.]*' | cut -d' ' -f2)
MAX_24_HOUR_SEND=$(echo "$OUTPUT" | grep -o '"Max24HourSend": [0-9.]*' | cut -d' ' -
f2)
SENT_LAST_24_HOURS=$(echo "$OUTPUT" | grep -o '"SentLast24Hours": [0-9.]*' | cut -d'
 ' -f2)

echo ""
echo "==============================================="
echo "Your SES Sending Limits"
echo "==============================================="
echo "Maximum send rate: $MAX_SEND_RATE emails/second"
echo "Maximum 24-hour send: $MAX_24_HOUR_SEND emails"
echo "Sent in the last 24 hours: $SENT_LAST_24_HOURS emails"
echo ""

# Ask if user wants to send a test email
echo "Do you want to send a test email? (y/n):"
SEND_TEST="y"
echo "$SEND_TEST"

if [[ "$SEND_TEST" =~ ^[Yy] ]]; then
    echo ""
    echo "==============================================="
    echo "SANDBOX ENVIRONMENT NOTICE"
    echo "==============================================="
    echo "Your account is likely in the SES sandbox environment."
    echo "In the sandbox, you can only send emails to verified email addresses."
    echo ""
    echo "Do you want to:"
    echo "1. Send a test email to the same verified address ($EMAIL_ADDRESS)"
    echo "2. Verify another email address to use as recipient"
    echo ""
    echo "Enter your choice (1 or 2):"
```

```
    RECIPIENT_CHOICE="1"
    echo "$RECIPIENT_CHOICE"

    if [[ "$RECIPIENT_CHOICE" == "1" ]]; then
        RECIPIENT_EMAIL="$EMAIL_ADDRESS"
    else
        echo "Please enter the recipient email address you want to verify:"
        RECIPIENT_EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 |
 head -n 1)@example.com"
        echo "$RECIPIENT_EMAIL"

        # Verify recipient email identity if different from sender
        if [[ "$RECIPIENT_EMAIL" != "$EMAIL_ADDRESS" ]]; then
            echo "Verifying recipient email address: $RECIPIENT_EMAIL" | tee -a
 "$LOG_FILE"
            OUTPUT=$(log_cmd "aws ses verify-email-identity --email-address
 \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to verify recipient email address"

            CREATED_RESOURCES+=("Email identity: $RECIPIENT_EMAIL")
            echo "A verification email has been sent to $RECIPIENT_EMAIL."
            echo "Please check the inbox and click the verification link before
 continuing."
            echo ""
            echo "Press Enter after you've verified the recipient email address..."
            sleep 1

            # Check recipient verification status
            OUTPUT=$(log_cmd "aws ses get-identity-verification-attributes --
identities \"$RECIPIENT_EMAIL\"")
            check_error "$OUTPUT" $? "Failed to get recipient verification
 attributes"

            # Check if the recipient email is verified
            RECIPIENT_VERIFICATION_STATUS=$(echo "$OUTPUT" | grep -o
 '"VerificationStatus": "[^"]*' | cut -d'"' -f4)
            if [[ "$RECIPIENT_VERIFICATION_STATUS" != "Success" ]]; then
                echo "Recipient email address $RECIPIENT_EMAIL is not verified yet."
                echo "You can try sending the email anyway, but it may fail."
            fi
        fi
    fi
```

```
    echo "Sending test email from $EMAIL_ADDRESS to $RECIPIENT_EMAIL..." | tee -a
 "$LOG_FILE"
    OUTPUT=$(log_cmd "aws ses send-email \
        --from \"$EMAIL_ADDRESS\" \
        --destination \"ToAddresses=$RECIPIENT_EMAIL\" \
        --message \"Subject={Data=SES Test
 Email,Charset=UTF-8},Body={Text={Data=This is a test email sent from Amazon SES
 using the AWS CLI,Charset=UTF-8}}\"")

    # Don't exit on send email error, just report it
    check_error "$OUTPUT" $? "Failed to send test email" "true"

    # Check if the email was sent successfully
    if [[ "$OUTPUT" =~ "MessageId" ]]; then
        # Extract message ID
        MESSAGE_ID=$(echo "$OUTPUT" | grep -o '"MessageId": "[^"]*' | cut -d'"' -f4)
        echo "Test email sent successfully! Message ID: $MESSAGE_ID"
    else
        echo "Failed to send test email. This is likely because your account is in
 the sandbox environment."
        echo "In the sandbox, both sender AND recipient email addresses must be
 verified."
    fi
    echo ""
fi

# Summary of created resources
echo ""
echo "==============================================="
echo "Setup Complete - Resources Created"
echo "==============================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo ""

# Ask if user wants to clean up resources
echo "==============================================="
echo "CLEANUP CONFIRMATION"
echo "==============================================="
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"
echo "$CLEANUP_CHOICE"
```

```
if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources have been preserved."
fi

echo ""
echo "==============================================="
echo "Amazon SES Setup Complete"
echo "==============================================="
echo "For production use, you may need to request to be moved out of the SES
 sandbox."
echo "Visit the SES console and navigate to 'Account dashboard' to request
 production access."
echo ""
echo "Log file: $LOG_FILE"
echo "==============================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - DeleteIdentity
  - GetIdentityVerificationAttributes
  - GetSendQuota
  - ListIdentities
  - SendEmail
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Amazon SNS-Beispiele unter Verwendung AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie ein Nachrichtenthema und veröffentlichen Sie Nachrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon SNS-Thema.
- Abonnieren Sie einen E-Mail-Endpunkt für das Thema
- Bestätigen Sie Ihr Abonnement
- Veröffentlichen Sie eine Nachricht zum Thema
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SNS Getting Started Script
# This script demonstrates how to create an SNS topic, subscribe to it, publish a
 message,
# and clean up resources.

set -euo pipefail

# Set up logging with secure file permissions
LOG_FILE="sns-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon SNS Getting Started Tutorial..."
echo "$(date)"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    if [ -n "${SUBSCRIPTION_ARN:-}" ] && [ "$SUBSCRIPTION_ARN" != "pending
 confirmation" ] && [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ]; then
        echo "Deleting subscription: $SUBSCRIPTION_ARN"
        if ! aws sns unsubscribe --subscription-arn "$SUBSCRIPTION_ARN" --region
 "$AWS_REGION" 2>/dev/null; then
            echo "Warning: Failed to delete subscription" >&2
        fi
    fi

    if [ -n "${TOPIC_ARN:-}" ]; then
        echo "Deleting topic: $TOPIC_ARN"
        if ! aws sns delete-topic --topic-arn "$TOPIC_ARN" --region "$AWS_REGION"
 2>/dev/null; then
            echo "Warning: Failed to delete topic" >&2
        fi
    fi

    return $exit_code
}

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi
```

```
# Set trap to cleanup on exit
trap cleanup_resources EXIT

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random topic name suffix using secure method
RANDOM_SUFFIX=$(openssl rand -hex 4)
TOPIC_NAME="my-topic-${RANDOM_SUFFIX}"

# Validate topic name length (max 256 characters)
if [ ${#TOPIC_NAME} -gt 256 ]; then
    handle_error "Topic name exceeds maximum length of 256 characters"
fi

# Step 1: Create an SNS topic
echo "Creating SNS topic: $TOPIC_NAME"
TOPIC_RESULT=$(aws sns create-topic \
    --name "$TOPIC_NAME" \
    --region "$AWS_REGION" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
    --output json) || handle_error "Failed to create SNS topic"

# Extract the topic ARN using jq for reliable parsing
TOPIC_ARN=$(echo "$TOPIC_RESULT" | jq -r '.TopicArn // empty') || handle_error
 "Failed to parse topic result"

if [ -z "$TOPIC_ARN" ]; then
    handle_error "Failed to extract topic ARN from result: $TOPIC_RESULT"
fi

# Validate ARN format
if [[ ! "$TOPIC_ARN" =~ ^arn:aws:sns:[a-z0-9-]+:[0-9]{12}:[a-zA-Z0-9_-]+$ ]]; then
```

```
    handle_error "Invalid SNS topic ARN format: $TOPIC_ARN"
fi

echo "Successfully created topic with ARN: $TOPIC_ARN"

# Step 2: Subscribe to the topic using Email-JSON protocol to reduce costs
echo ""
echo "==============================================="
echo "EMAIL SUBSCRIPTION"
echo "==============================================="
EMAIL_ADDRESS="test-${RANDOM_SUFFIX}@example.com"

# Validate email format (basic validation)
if [[ ! "$EMAIL_ADDRESS" =~ ^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]]; then
    handle_error "Invalid email format: $EMAIL_ADDRESS"
fi

echo "Subscribing email: $EMAIL_ADDRESS to topic using Email-JSON protocol"
SUBSCRIPTION_RESULT=$(aws sns subscribe \
    --topic-arn "$TOPIC_ARN" \
    --protocol email-json \
    --notification-endpoint "$EMAIL_ADDRESS" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to create subscription"

# Extract the subscription ARN using jq
SUBSCRIPTION_ARN=$(echo "$SUBSCRIPTION_RESULT" | jq -r '.SubscriptionArn // empty')
 || handle_error "Failed to parse subscription result"

echo "Subscription created: $SUBSCRIPTION_ARN"
echo "A confirmation email has been sent to $EMAIL_ADDRESS"
echo ""

# Tag the subscription
if [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ] && [ "$SUBSCRIPTION_ARN" !=
 "pending confirmation" ]; then
    aws sns tag-resource \
        --resource-arn "$SUBSCRIPTION_ARN" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
        --region "$AWS_REGION" 2>/dev/null || echo "Warning: Failed to tag
 subscription"
fi
```

```
# Step 3: List subscriptions to verify
echo "Listing subscriptions for topic: $TOPIC_ARN"
SUBSCRIPTIONS=$(aws sns list-subscriptions-by-topic --topic-arn "$TOPIC_ARN" --
region "$AWS_REGION" --output json) || handle_error "Failed to list subscriptions"

echo "Current subscriptions:"
echo "$SUBSCRIPTIONS" | jq '.'

# Get the confirmed subscription ARN with optimized jq query and improved error
 handling
CONFIRMED_SUBSCRIPTION=$(echo "$SUBSCRIPTIONS" | jq -r '.Subscriptions[]? |
 select(.SubscriptionArn != "PendingConfirmation") | .SubscriptionArn' 2>/dev/null |
 head -n 1)

if [ -n "$CONFIRMED_SUBSCRIPTION" ]; then
    SUBSCRIPTION_ARN="$CONFIRMED_SUBSCRIPTION"
else
    echo "Warning: No confirmed subscription found. You may not have confirmed the
 subscription yet."
    echo "The script will continue, but you may not receive the test message."
fi

# Step 4: Publish a message to the topic
echo ""
echo "Publishing a test message to the topic"
MESSAGE="Hello from Amazon SNS! This is a test message sent at $(date)."

# Validate message length (max 256 KB for SNS)
if [ ${#MESSAGE} -gt 262144 ]; then
    handle_error "Message exceeds maximum size of 256 KB"
fi

PUBLISH_RESULT=$(aws sns publish \
    --topic-arn "$TOPIC_ARN" \
    --message "$MESSAGE" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to publish message"

MESSAGE_ID=$(echo "$PUBLISH_RESULT" | jq -r '.MessageId // empty') || handle_error
 "Failed to parse publish result"

if [ -z "$MESSAGE_ID" ]; then
    handle_error "No message ID returned from publish operation"
fi
```

```
# Validate message ID format (UUID v4)
if [[ ! "$MESSAGE_ID" =~ ^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]
{12}$ ]]; then
    handle_error "Unexpected message ID format: $MESSAGE_ID"
fi

echo "Message published successfully with ID: $MESSAGE_ID"
echo "Check your email for the message."

# Pause to allow the user to check their email
echo ""
echo "Pausing for 3 seconds to allow message delivery..."
sleep 3

# Step 5: Clean up resources
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Resources created:"
echo "- SNS Topic: $TOPIC_ARN"
echo "- Subscription: ${SUBSCRIPTION_ARN:-N/A}"
echo ""
echo "Cleaning up resources to avoid unnecessary charges..."

echo ""
echo "Tutorial completed successfully!"
echo "$(date)"
echo "================================================"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateTopic
  - DeleteTopic
  - ListSubscriptionsByTopic
  - Veröffentlichen
  - Abonnieren
  - Unsubscribe

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config-Konfigurationsrekorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}
```

```
# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
```

```
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
```

```
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "==========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"
```

```
# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
```

```
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
```

```
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
```

```
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy

- Rm
- StartConfigurationRecorder
- StopConfigurationRecorder

Erste Schritte mit dem Geräteschutz im Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erforderliche IAM-Rollen erstellen
- Aktivieren Sie die Audit-Checks von IoT Device Defender
- Führen Sie ein On-Demand Audit durch
- Erstellen Sie eine Schadensbegrenzungsmaßnahme
- Wenden Sie Abhilfemaßnahmen auf Ergebnisse an
- Richten Sie SNS-Benachrichtigungen ein (optional)
- Aktivieren Sie die IoT-Protokollierung

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```
echo "====================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "====================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /dev/
null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
```

```
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/
null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
```

```
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
```

```
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
```

```
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
```

```
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "====================================================="
echo "Step 1: Creating required IAM roles"
echo "====================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
```

```
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "====================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "====================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1) ||
 true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
```

```
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "==================================================="
echo "Step 3: Running an on-demand audit"
echo "==================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
```

```
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null || echo
 "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
```

```
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "====================================================="
echo "Step 4: Creating a mitigation action"
echo "====================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting it
 first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1
 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)
```

```
if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 ||
 true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "====================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "====================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
```

```
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1) ||
 true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "====================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "====================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
```

```
echo "====================================================="
echo "Step 6: Setting up SNS notifications"
echo "====================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --output
 json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
```

```
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "====================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "====================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)
```

```
    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2 methods,
 continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved but
 could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "====================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "====================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "============================================"
echo "CLEANUP"
```

```
echo "============================================"
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true
```

```
# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --output
 json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null ||
 echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
```

```
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies

- ListTopics
- PutRolePolicy
- SetLoggingOptions
- SetV2LoggingOptions
- StartAuditMitigationActionsTask
- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

Beispiele für Step Functions mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie das with Bash-Skript AWS Command Line Interface mit Step Functions verwenden.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit der Workflow-Orchestrierung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Step Functions
- Erstellen Sie Ihre erste State-Machine
- Starten Sie die Ausführung Ihrer Zustandsmaschine
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS Step
 Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done
```

```
# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be less
 reliable."
    echo "Consider installing jq for better error handling: brew install jq (macOS)
 or apt-get install jq (Ubuntu)"
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"
```

```
# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
```

```
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] || [ "$wait_time"
 -gt 300 ]; then
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
```

```
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."

    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
```

```
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
$ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed to
delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
```

```
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
```

```
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
```

```
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
```

```
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")
```

```
if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)
```

```
check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
```

```
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
```

```
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)
```

```
check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
```

```
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
```

```
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
```

```
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
```

```
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3

    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
```

```
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "============================================"

# Cleanup
echo ""
echo "============================================"
echo "CLEANUP"
echo "============================================"
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

AWS STS Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS STS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
```

```
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
##############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
```

```
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API-Details finden Sie AssumeRole in der AWS CLI Befehlsreferenz.

Szenarien

Konfigurieren Sie die Container-Dienstkonnektivität

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie die VPC-Infrastruktur
- Einrichten der Protokollierung
- Erstellen Sie den ECS-Cluster
- Konfigurieren Sie IAM-Rollen
- Erstellen Sie den Dienst mit Service Connect
- Überprüfen der Bereitstellung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
```

```
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
```

```
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters "$resource_id"
 --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
```

```
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -d'"'
 -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true
```

```
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
```

```
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
```

```
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json >/
dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
```

```
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF
```

```
    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///tmp/
task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn": "[^"]*"'
 | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
```

```
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
```

```
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
```

```
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/null
 || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================"
    echo "CREATED RESOURCES"
    echo "============================================"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================"
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
```

```
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed to
 scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn" >/
dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-arn
 "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/dev/null
 || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id" 2>/
dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
```

```
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/null
 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"
```

```
    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
cleanup function."
        echo ""
```

```
        echo "To clean up resources later, you can use the AWS CLI commands or the
 AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid dependency
 issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - DetachRolePolicy

  - GetCallerIdentity

- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Container-Aufgabe für den serverlosen Starttyp

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Den Cluster erstellen
- Einer Aufgabendefinition erstellen
- Erstellen Sie den Dienst
- Bereinigen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error
```

```
# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
```

```
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|AccessDenied
\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
```

```
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security group
 $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
```

```
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "CLEANUP PROCESS"
    echo "============================================"
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."
```

```
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]]; then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME'" "Delete ECS service"
            else
                echo "WARNING: Failed to scale service. Attempting to delete
 anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME' --
service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]]; then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
 "Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete security
 group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."
```

```
            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix "$TASK_FAMILY"
 --query "taskDefinitionArns" --output text 2>/dev/null || echo "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
```

```
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME --
service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY " ]];
 then
            echo "6. Deregister task definitions: aws ecs list-task-definitions --
family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --task-
definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "============================================"
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "============================================"

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
```

```
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --assume-
role-policy-document file://trust-policy.json" "Create ECS task execution role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name ecsTaskExecutionRole --
policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
 "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================"
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================"
```

```
CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-ecs-
fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================"
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================"

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "logConfiguration": {
```

```
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-json
 file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================"
echo "STEP 4: SET UP NETWORKING"
echo "============================================"

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
```

```
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --group-
name '$SECURITY_GROUP_NAME' --description 'Security group for ECS Fargate tutorial
 - HTTP access' --vpc-id '$VPC_ID' --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-fargate-
linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"' |
 head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP inbound
 rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
```

```
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" --
query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================"
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================"

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-type
 FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets\":[\"$(echo
 "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups\":[\"$SECURITY_GROUP_ID
\"],\"assignPublicIp\":\"ENABLED\"}}' --tags key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================"
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================"
```

```
echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN" --
query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --output
 text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID" --
query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
```

```
    else
        echo ""
        echo "============================================"
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "============================================"
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "============================================"
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

Erstellen Sie eine Rest-API mit Funktions-Proxy-Integration

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für die Lambda-Ausführung
- Erstellen Sie eine Lambda-Funktion und stellen Sie sie bereit
- Erstellen Sie eine REST-API
- Konfigurieren Sie die Lambda-Proxy-Integration
- Bereitstellen und Testen der API
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
```

```
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
```

```
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
```

```
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
```

```
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query "items[?
pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
```

```
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
```

```
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed to
 delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning: Failed
 to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
 || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json
```

```
echo "Cleanup completed!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Erstellen und überwachen Sie einen Dienst
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
```

```
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
```

```
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
```

```
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
```

```
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
```

```
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
```

```
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."
```

```
    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
```

```
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
```

```
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
```

```
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
```

```
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "==========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "==========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair

- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

Einen verwalteten Überwachungsbereich erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle für Ihren Workspace

- Erstellen Sie einen Grafana-Workspace
- Konfigurieren Sie die Authentifizierung
- Konfigurieren Sie optionale Einstellungen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
```

```
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
```

```
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-grafana-
gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"
```

```
echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
```

```
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -d'"' -
f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================"
echo "WORKSPACE INFORMATION"
echo "============================================"
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================"

# Step 6: Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"
```

```
    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 | grep
 -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
```

```
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

Erstellen eines Monitoring-Dashboards mit einem Funktionsnamen als Variable

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein CloudWatch Dashboard
- Fügen Sie Lambda-Metrik-Widgets mit einer Funktionsnamenvariablen hinzu
- Überprüfen Sie das Dashboard
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}
```

```
# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with 'aws
 configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
```

```
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-dynamicdash
 \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
```

```
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/null
 --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"
```

```
# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}", { "stat":
 "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
```

```
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi
```

```
# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
```

```
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of the
 dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "============================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --region
 "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to delete
 it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach it
 manually."
        fi
```

```
        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetDashboard
  - Invoke
  - ListDashboards
  - ListFunctions
  - PutDashboard

Erste Schritte mit dem Konfigurationsmanagement

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config-Konfigurationsrekorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR
```

```
# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-name
 "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi
```

```
        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null || true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=========================================="
    echo "CREATED RESOURCES"
    echo "=========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
```

```
    fi
    echo "============================================"
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
```

```
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --create-
bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/null
 || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
```

```
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:${AWS_REGION}:
${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null || echo
 "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"' |
 head -1 | cut -d'"' -f4)
```

```
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"
```

```
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/dev/
null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --configuration-
recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
```

```
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm

- StartConfigurationRecorder
- StopConfigurationRecorder

Erste Schritte mit Container-Registern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Docker-Image
- Erstellen Sie ein Amazon ECR-Repository
- Löschen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "==================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "==================================================="
```

```
# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
 for installation instructions."
    exit 1
fi

# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure' to
 set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running this
 script."
    echo "Visit https://docs.docker.com/get-docker/ for installation instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"
```

```
    echo "====================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "====================================================="

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "====================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force || echo
 "Failed to delete repository, it may not exist or may have already been deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local image,
 it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove tagged
 image, it may not exist or may have already been deleted."
    fi
```

```
    echo "Cleanup completed."
    echo "==================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build Docker
 image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list Docker
 images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"
```

```
# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials are
 configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"

# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --tags
 key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-registry-
gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username AWS --
password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" || handle_error
 "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"
```

```
# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image to
 ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "====================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "====================================================="

# Ask user if they want to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
```

```
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "==================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "==================================================="
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- BatchDeleteImage
- CreateRepository
- DeleteRepository
- GetCallerIdentity
- GetLoginPassword

Erste Schritte mit Nachrichten im Internet der Dinge

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IoT-Ressourcen
- Konfigurieren Ihres Geräts
- Ausführen der Beispielanwendungen
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
```

```
}

# Function to check for errors
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --
new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
```

```
}

# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "==================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your region."
 | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE
```

```
# Create IoT policy
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-document file://
iot-policy.json --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-core-
gs"
check_error $?
```

```
# Create IoT thing
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn' --
output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active --
certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile $CERTS_DIR/
public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
```

```
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." | tee
 -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-python-
v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
```

```
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
```

```
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
```

```
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "==================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
```

```
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-status
 INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint

- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

Erste Schritte mit Feature Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
```

```
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-arn
 "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
```

```
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)
```

```
    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
```

```
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
```

```
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=tr
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
```

```
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
```

```
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3:////'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
```

```
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi
```

```
echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
```

```
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS" |
 grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
```

```
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- AttachRolePolicy
- CreateBucket
- CreateFeatureGroup
- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

Erste Schritte mit verwalteten Kubernetes-Clustern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine VPC für Ihren EKS-Cluster
- Erstellen Sie IAM-Rollen für Ihren EKS-Cluster
- Erstellen Ihres EKS-Clusters
- Konfigurieren Sie kubectl für die Kommunikation mit Ihrem Cluster
- Erstellen einer verwalteten Knotengruppe
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the AWS
 CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your operating
 system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
```

```
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/
kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"
```

```
# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --nodegroup-
name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
```

```
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources; exit
 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
```

```
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."

# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
```

```
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
```

```
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
```

```
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-kubeconfig
 --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"
```

```
NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn" --
output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl get
 nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or the
 nodes are still registering."
```

```
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl get
 all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
```

```
done
echo "============================================"

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --cluster-
name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --stack-
name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole

- DeleteStack
- DescribeCluster
- DescribeNodegroup
- DescribeStacks
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNodegroups
- UpdateKubeconfig
- Wait

Erste Schritte mit verwaltetem Streaming

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen MSK-Cluster
- Erstellen Sie IAM-Berechtigungen für den MSK-Zugriff
- Erstellen Sie einen Client-Computer
- Holen Sie sich Bootstrap-Broker
- Richten Sie den Client-Computer ein
- Erstellen Sie ein Thema und produce/consume Daten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
```

```
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group: $CLIENT_SG_ID";
 fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
```

```
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id" &>/
dev/null
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated" &>/
dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
```

```
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o "\"GroupId
\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing $sg_id,
 removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing $sg_id,
 removing them..."
```

```
                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No outbound
 all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed to
 terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group" "$CLIENT_SG_ID";
 then
        remove_security_group_references "$CLIENT_SG_ID"
```

```
        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are removed
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete key
pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
```

```
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to delete
 policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
```

```
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az (subnet:
 $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "==============================================="

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
```

```
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default VPC
 in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-az,Values=true"
 \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least 3
 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=$DEFAULT_VPC_ID"
 \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"
```

```
# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\", \"ClientSubnets
\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\", \"${SUBNET_ARRAY[2]}\"],
 \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-gs
 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --query
 "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
```

```
        echo "Cluster is now active!"
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "================================================"
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
```

```
                \"$CLUSTER_ARN\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-streaming-
for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
```

```
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "==============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@]; then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi
```

```
echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -d'"'
 -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"

# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)
```

```
if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi

echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
```

```
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
```

```
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-${RANDOM_SUFFIX}},
{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-streaming-for-
apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response: $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)

if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)
```

```
    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "==============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))

    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
```

```
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "==============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "==============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi
```

```
echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-iam-
auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/$LATEST_VERSION/
aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
```

```
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "==============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF

# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
```

```
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "==============================================="

# Display summary of created resources
echo ""
echo "==============================================="
echo "RESOURCE SUMMARY"
echo "==============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "==============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
```

```
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
```

```
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile

- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole
- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances

- TerminateInstances

- Wait

Erste Schritte mit Push-Benachrichtigungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Anwendung
- Aktivieren Sie Kanäle für Push-Benachrichtigungen
- Push-Benachrichtigung senden
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
```

```
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
```

```
        echo "ERROR: Command failed: $cmd" >&2
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
```

```
    echo ""

    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
```

```
        echo "  - mobiletargeting:CreateApp" >&2
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters" >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi
```

```
check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "==========================================="
echo "ENABLING FCM (GCM) CHANNEL"
```

```
echo "============================================"
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-demo-
only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real certificates)
echo ""
echo "============================================"
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================"
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --apns-
channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
```

```
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-channel
 command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "============================================"
echo "CREATING MESSAGE FILES"
echo "============================================"

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
```

```
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
echo "To send real messages, you would need to replace the sample device tokens with
 actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder tokens)
```

```
echo ""
echo "============================================"
echo "DEMONSTRATING MESSAGE SENDING"
echo "============================================"
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."
```

```
echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
```

```
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

Erste Schritte mit Such- und Analysesystemen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Service-Domain

- Laden Sie Daten auf Ihre Domain hoch
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
        echo "Checking if domain $DOMAIN_NAME exists before attempting to delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/null
 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
```

```
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
```

```
# This policy restricts access to specific actions and should be further restricted
 to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack creation
 failed"

DOMAIN_CREATED=true
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo "No";
 fi)"
```

```
# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel", "Malahide,
 Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas", "Stellate, Nick",
 "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese", "Spielbauer, Bruce",
 "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett, Charlie"], "title": "U.S.
 Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955, "actor":
 ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal", "Backus, Jim",
 "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen, Corey", "Birch, Paul",
 "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh, Nelson", "Williams,
```

```
 Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira", "McMahon, David",
 "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"

    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }
```

```
    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d:
-f2)

        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
$HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
$HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
then
                echo "Error: Request is being treated as anonymous (authentication
not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential backoff..."

            # Retry authentication test with exponential backoff
```

```
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT+1))/
$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)

                echo "Retry result (HTTP $HTTP_CODE):"
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
 $HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" |
 cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
 (HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
```

```
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding with
data operations."

        # Upload single document (matches tutorial exactly)
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
'"result"'; then
            echo "# Single document uploaded successfully! (HTTP $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
$UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
```

```
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
'"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
$BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
```

```
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "==========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "==========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-name "$DOMAIN_NAME"
 --query 'DomainStatus.{AdvancedSecurityOptions: AdvancedSecurityOptions,
 AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
```

```
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================"
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================"
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
```

```
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true
```

```
echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - GetCallerIdentity

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleichen von Zeichenfolgen hinzu
- Verwaltete Regeln hinzufügen
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
```

```
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "====================================================="
    echo "CLEANING UP RESOURCES"
    echo "====================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi
```

```
    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
```

```
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"
```

```
# Step 2: Add a String Match Rule
echo ""
echo "======================================================"
echo "STEP 2: Adding String Match Rule"
echo "======================================================"

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
 || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
```

```
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
"SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
\
        --region us-east-1 2>&1) || true
```

```
    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
```

```
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
"SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
\
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
attempts"
```

```
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "======================================================"
echo "STEP 4: Listing CloudFront Distributions"
echo "======================================================"

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "======================================================"
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "======================================================"

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
```

```
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
```

```
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

Fest codierte Geheimnisse in einen sicheren Geheimspeicher verschieben

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen
- Erstellen eines Secrets in Secrets Manager
- Aktualisieren Sie Ihren Anwendungscode
- Aktualisiere das Geheimnis
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "============================================"
    echo "RESOURCES CREATED"
    echo "============================================"

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
```

```
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-delete-
without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name "$RUNTIME_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME" --
query 'PolicyNames[]' --output text 2>/dev/null || true); do
```

```
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-name
 "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
```

```
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_urls
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
```

```
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/null
 || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').token_

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
```

```
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "========================================"
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
```

```
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue
  - PutResourcePolicy
  - UpdateSecret

Führen Sie CPU-Stresstests auf VM-Instanzen mithilfe von Fault Injection durch

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Rollen

- Erstellen Sie einen CloudWatch Alarm
- Erstellen Sie eine Versuchsvorlage
- Führen Sie das Experiment aus
- Überprüfe die Ergebnisse
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
```

```
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi
```

```
    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
```

```
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
```

```
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
```

```
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60
```

```
# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk
 -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
```

```
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF
```

```
# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail
```

```
    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk -
F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status later
 using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/null
 || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")
```

```
# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================"

# Prompt for cleanup
echo ""
```

```
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
```

```
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.

- AddRoleToInstanceProfile
- AttachRolePolicy
- CreateExperimentTemplate
- CreateInstanceProfile
- CreateRole
- DeleteAlarms
- DeleteExperimentTemplate
- DeleteInstanceProfile
- DeleteRole
- DeleteRolePolicy
- DescribeAlarms
- DescribeImages
- DetachRolePolicy
- GetCallerIdentity
- GetExperiment
- GetMetricData
- GetRole
- MonitorInstances
- PutMetricAlarm
- PutRolePolicy
- RemoveRoleFromInstanceProfile
- RunInstances
- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

Systemverwaltung einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Berechtigungen für Systems Manager

- Erstellen einer IAM-Rolle für Systems Manager
- Konfigurieren Sie Systems Manager
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
```

```
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
```

```
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
```

```
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
```

```
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
```

```
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
```

```
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
```

```
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
```

```
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
```

```
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"
```

```
# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
```

```
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
```

```
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
```

```
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole

- DetachRolePolicy
- GetCallerIdentity
- GetConfigurationManager

Support Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. Support

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit technischem Support

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Prüfen Sie die verfügbaren Dienste und den Schweregrad
- Einen Support-Fall erstellen
- Mitteilungen zu einem Fall hinzufügen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash
```

```
# AWS Support CLI Tutorial Script
# This script demonstrates how to use AWS Support API through AWS CLI

set -euo pipefail

# Security: Validate script location and permissions
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
readonly SCRIPT_DIR

# Security: Use secure temporary directory
readonly TEMP_DIR="$(mktemp -d)" || { echo "Failed to create temp directory"; exit
 1; }
trap "rm -rf '$TEMP_DIR'" EXIT

# Set up logging with secure permissions
LOG_FILE="${TEMP_DIR}/aws-support-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Validate AWS CLI is available
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed or not in PATH" >&2
    exit 1
fi

# Security: Check AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not properly configured" >&2
    exit 1
fi

{
    echo "Starting AWS Support Tutorial at $(date)"
    echo "Script: $0"
    echo "User: $(whoami)"
    echo "---"
} >> "$LOG_FILE"

# Function to log commands and their outputs securely
log_cmd() {
    local cmd="$1"
    echo "$(date): Running command" >> "$LOG_FILE"
    # Don't echo the actual command to prevent credential leakage
```

```
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
    local is_fatal="${4:-true}"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command returned status: $cmd_status" >> "$LOG_FILE"

        # Check for subscription error
        if [[ "$cmd_output" =~ "SubscriptionRequiredException" ]]; then
            {
                echo ""
                echo "====================================================="
                echo "IMPORTANT: This account does not have the required AWS Support
 plan."
                echo "You need a Business, Enterprise On-Ramp, or Enterprise Support
 plan"
                echo "to use the AWS Support API."
                echo ""
                echo "This script will now demonstrate the commands that would be
 run"
                echo "if you had the appropriate support plan, but will not execute
 them."
                echo "====================================================="
            } | tee -a "$LOG_FILE"

            DEMO_MODE=true
            return 0
        fi

        if [[ "$is_fatal" == "true" ]]; then
            cleanup_resources
            exit 1
        fi
    fi
}
```

```
# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    echo "No persistent resources were created that need cleanup." | tee -a
 "$LOG_FILE"
}

# Function to run a command in demo mode
demo_cmd() {
    local cmd="$1"
    local description="$2"

    {
        echo ""
        echo "DEMO: $description"
        echo "Command that would be executed:"
        echo "  [Command hidden for security]"
        echo ""
    } | tee -a "$LOG_FILE"
}

# Function to safely extract JSON values
extract_json_value() {
    local json_output="$1"
    local key="$2"

    echo "$json_output" | grep -o "\"$key\": \"[^\"]*\"" | head -1 | cut -d'"' -f4
 || echo ""
}

# Array to track created resources
declare -a CREATED_RESOURCES

# Initialize demo mode flag
DEMO_MODE=false

# Security: Validate input parameters
if [[ $# -gt 0 ]]; then
    echo "ERROR: This script does not accept parameters" >&2
    exit 1
fi

{
    echo "====================================================="
```

```
    echo "AWS Support CLI Tutorial"
    echo "====================================================="
    echo "This script demonstrates how to use AWS Support API"
    echo "Note: You must have a Business, Enterprise On-Ramp,"
    echo "or Enterprise Support plan to use the AWS Support API."
    echo "====================================================="
    echo ""
} | tee -a "$LOG_FILE"

# Step 1: Check available services
echo "Step 1: Checking available AWS Support services..." | tee -a "$LOG_FILE"
SERVICES_OUTPUT=$(log_cmd "aws support describe-services --language en" 2>&1) ||
 SERVICES_OUTPUT=""
check_error "$SERVICES_OUTPUT" $? "Failed to retrieve AWS Support services"

# If we're in demo mode, set default values
if [[ "$DEMO_MODE" == "true" ]]; then
    SERVICE_CODE="general-info"
    echo "Using demo service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
else
    # Extract a service code for demonstration using safer method
    SERVICE_CODE=$(extract_json_value "$SERVICES_OUTPUT" "code") || SERVICE_CODE=""
    if [[ -z "$SERVICE_CODE" ]]; then
        SERVICE_CODE="general-info"
        echo "Using default service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found service code: $SERVICE_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 2: Check available severity levels
echo "Step 2: Checking available severity levels..." | tee -a "$LOG_FILE"
if [[ "$DEMO_MODE" == "true" ]]; then
    demo_cmd "aws support describe-severity-levels --language en" "Check available
 severity levels"
    SEVERITY_CODE="low"
    echo "Using demo severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
else
    SEVERITY_OUTPUT=$(log_cmd "aws support describe-severity-levels --language en"
 2>&1) || SEVERITY_OUTPUT=""
    check_error "$SEVERITY_OUTPUT" $? "Failed to retrieve severity levels"

    SEVERITY_CODE=$(extract_json_value "$SEVERITY_OUTPUT" "code") ||
 SEVERITY_CODE=""
```

```
    if [[ -z "$SEVERITY_CODE" ]]; then
        SEVERITY_CODE="low"
        echo "Using default severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    else
        echo "Found severity code: $SEVERITY_CODE" | tee -a "$LOG_FILE"
    fi
fi

# Step 3: Create a test support case
{
    echo ""
    echo "====================================================="
    echo "SUPPORT CASE CREATION"
    echo "====================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "DEMO MODE: The following steps would create and manage a support case"
        echo "if you had a Business, Enterprise On-Ramp, or Enterprise Support
 plan."
        echo ""
    } | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"

    demo_cmd "aws support create-case" "Create a support case"

    CASE_ID="case-12345678910-2013-c4c1d2bf33c5cf47"
    echo "Demo case ID: $CASE_ID" | tee -a "$LOG_FILE"

    demo_cmd "aws support describe-cases" "List support cases"
    demo_cmd "aws support add-communication-to-case" "Add communication to case"
    demo_cmd "aws support describe-communications" "View case communications"
    demo_cmd "aws support resolve-case" "Resolve the support case"

else
    echo "Creating a test support case..." | tee -a "$LOG_FILE"

    USER_EMAIL="example@example.com"
    CC_EMAIL_PARAM="--cc-email-addresses $USER_EMAIL"

    # Create the case
```

```
    CASE_OUTPUT=$(log_cmd "aws support create-case --subject 'AWS CLI Tutorial Test
 Case' --service-code '$SERVICE_CODE' --category-code 'using-aws' --communication-
body 'This is a test case created as part of an AWS CLI tutorial.' --severity-code
 '$SEVERITY_CODE' --language 'en' $CC_EMAIL_PARAM" 2>&1) || CASE_OUTPUT=""

    check_error "$CASE_OUTPUT" $? "Failed to create support case"

    # Extract the case ID safely
    CASE_ID=$(extract_json_value "$CASE_OUTPUT" "caseId") || CASE_ID=""

    if [[ -n "$CASE_ID" ]]; then
        echo "Successfully created support case with ID: $CASE_ID" | tee -a
 "$LOG_FILE"
        CREATED_RESOURCES+=("Support Case: $CASE_ID")

        # Step 4: List the case we just created
        echo "" | tee -a "$LOG_FILE"
        echo "Step 4: Listing the support case we just created..." | tee -a
 "$LOG_FILE"
        CASES_OUTPUT=$(log_cmd "aws support describe-cases --case-id-list '$CASE_ID'
 --include-resolved-cases false --language 'en'" 2>&1) || CASES_OUTPUT=""

        check_error "$CASES_OUTPUT" $? "Failed to retrieve case details"

        # Step 5: Add a communication to the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 5: Adding a communication to the support case..." | tee -a
 "$LOG_FILE"
        COMM_OUTPUT=$(log_cmd "aws support add-communication-to-case --case-id
 '$CASE_ID' --communication-body 'This is an additional communication for the test
 case.' $CC_EMAIL_PARAM" 2>&1) || COMM_OUTPUT=""

        check_error "$COMM_OUTPUT" $? "Failed to add communication to case"

        # Step 6: View communications for the case
        echo "" | tee -a "$LOG_FILE"
        echo "Step 6: Viewing communications for the support case..." | tee -a
 "$LOG_FILE"
        COMMS_OUTPUT=$(log_cmd "aws support describe-communications --case-id
 '$CASE_ID' --language 'en'" 2>&1) || COMMS_OUTPUT=""

        check_error "$COMMS_OUTPUT" $? "Failed to retrieve case communications"

        # Step 7: Resolve the case
```

```
        {
            echo ""
            echo "==================================================="
            echo "CASE RESOLUTION"
            echo "==================================================="
            echo "Resolving the support case..."
        } | tee -a "$LOG_FILE"

        RESOLVE_OUTPUT=$(log_cmd "aws support resolve-case --case-id '$CASE_ID'"
 2>&1) || RESOLVE_OUTPUT=""

        check_error "$RESOLVE_OUTPUT" $? "Failed to resolve case"
        echo "Successfully resolved support case: $CASE_ID" | tee -a "$LOG_FILE"
    else
        echo "Could not extract case ID from the response." | tee -a "$LOG_FILE"
    fi
fi

# Display summary of created resources
{
    echo ""
    echo "==================================================="
    echo "TUTORIAL SUMMARY"
    echo "==================================================="
} | tee -a "$LOG_FILE"

if [[ "$DEMO_MODE" == "true" ]]; then
    {
        echo "This was a demonstration in DEMO MODE."
        echo "No actual AWS Support cases were created."
        echo "To use the AWS Support API, you need a Business, Enterprise On-Ramp,"
        echo "or Enterprise Support plan."
    } | tee -a "$LOG_FILE"
else
    {
        echo "Resources created during this tutorial:"
        if [[ ${#CREATED_RESOURCES[@]} -eq 0 ]]; then
            echo "No resources were created."
        else
            for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
                echo "- $resource"
            done
        fi
    } | tee -a "$LOG_FILE"
```

```
fi

{
    echo ""
    echo "Tutorial completed successfully!"
    echo "Log file: $LOG_FILE"
    echo "====================================================="
} | tee -a "$LOG_FILE"

# Display log file path to user
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddCommunicationToCase
  - CreateCase
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Beispiele für Systems Manager mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Skripts AWS Command Line Interface with Bash in Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Einen Container-Service für Instanzen virtueller Maschinen erstellen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen ECS-Cluster
- Erstellen und überwachen Sie einen Dienst
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container instance,
# registering a task definition, and creating a service using the EC2 launch type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"
```

```
# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "============================================"
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
```

```
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
"$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --reason
"Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
"$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi
```

```
    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds before
retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after $max_retries
attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
```

```
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
 then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
 remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log file
 for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi
```

```
    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --
query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}
```

```
# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING: Failed
 to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-group,Tags=[{Key=project,Value=
$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]" \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
```

```
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}
```

```
# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
```

```
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]]; then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
```

```
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
```

```
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"
```

```
    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCount,T

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME" --
output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
```

```
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME" --container-
instances "$CONTAINER_INSTANCE_ARN" --output table --query 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendingT
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "==========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "==========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try accessing
 http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
```

```
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================"
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================"
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
```

```
main "$@"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeContainerInstances
  - DescribeInstances
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs
  - GetCallerIdentity
  - GetParameters
  - GetRole
  - ListContainerInstances
  - ListServices
  - ListTasks
  - RegisterTaskDefinition
  - RunInstances

- StopTask
- TerminateInstances
- UpdateService
- Wait

Erste Schritte mit virtuellen Maschinen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Schlüsselpaares
- Eine Sicherheitsgruppe erstellen
- Anhalten und Starten Ihrer Instance
- Testen Sie die Elastic IP-Persistenz
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# EC2 Basics Tutorial Script - Revised
# This script demonstrates the basics of working with EC2 instances using AWS CLI
# Updated to use Amazon Linux 2023 and enhanced security settings

# Set up logging
LOG_FILE="ec2_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to log messages
log() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1"
}
```

```
# Function to handle errors
handle_error() {
  log "ERROR: $1"
  log "Cleaning up resources..."
  cleanup
  exit 1
}

# Function to clean up resources
cleanup() {
  log "Resources created:"

  if [ -n "$ASSOCIATION_ID" ]; then
    log "- Elastic IP Association: $ASSOCIATION_ID"
  fi

  if [ -n "$ALLOCATION_ID" ]; then
    log "- Elastic IP Allocation: $ALLOCATION_ID (IP: $ELASTIC_IP)"
  fi

  if [ -n "$INSTANCE_ID" ]; then
    log "- EC2 Instance: $INSTANCE_ID"
  fi

  if [ -n "$SECURITY_GROUP_ID" ]; then
    log "- Security Group: $SECURITY_GROUP_ID"
  fi

  if [ -n "$KEY_NAME" ]; then
    log "- Key Pair: $KEY_NAME (File: $KEY_FILE)"
  fi

  REPLY=y

  if [[ $REPLY =~ ^[Yy]$ ]]; then
    log "Starting cleanup..."

    # Track cleanup failures
    CLEANUP_FAILURES=0

    # Disassociate Elastic IP if it exists
    if [ -n "$ASSOCIATION_ID" ]; then
      log "Disassociating Elastic IP..."
      if ! aws ec2 disassociate-address --association-id "$ASSOCIATION_ID"; then
```

```
        log "Failed to disassociate Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Release Elastic IP if it exists
    if [ -n "$ALLOCATION_ID" ]; then
      log "Releasing Elastic IP..."
      if ! aws ec2 release-address --allocation-id "$ALLOCATION_ID"; then
        log "Failed to release Elastic IP"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Terminate instance if it exists
    if [ -n "$INSTANCE_ID" ]; then
      log "Terminating instance $INSTANCE_ID..."
      if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" > /dev/null;
then
        log "Failed to terminate instance"
        ((CLEANUP_FAILURES++))
      else
        log "Waiting for instance to terminate..."
        if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"; then
          log "Failed while waiting for instance to terminate"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Delete security group if it exists
    if [ -n "$SECURITY_GROUP_ID" ]; then
      log "Deleting security group..."
      if ! aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"; then
        log "Failed to delete security group"
        ((CLEANUP_FAILURES++))
      fi
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ]; then
      log "Deleting key pair..."
      if ! aws ec2 delete-key-pair --key-name "$KEY_NAME"; then
        log "Failed to delete key pair"
```

```
        ((CLEANUP_FAILURES++))
      fi

      # Remove key file
      if [ -f "$KEY_FILE" ]; then
        log "Removing key file..."
        if ! rm -f "$KEY_FILE"; then
          log "Failed to remove key file"
          ((CLEANUP_FAILURES++))
        fi
      fi
    fi

    # Report cleanup status
    if [ $CLEANUP_FAILURES -eq 0 ]; then
      log "Cleanup completed successfully."
    else
      log "WARNING: Cleanup completed with $CLEANUP_FAILURES failures. Some
 resources may not have been deleted properly."
    fi
  else
    log "Resources were not deleted."
  fi
}

# Generate random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
KEY_NAME="ec2-tutorial-key-$RANDOM_ID"
SG_NAME="ec2-tutorial-sg-$RANDOM_ID"

# Create a directory for the key file
KEY_DIR=$(mktemp -d)
KEY_FILE="$KEY_DIR/$KEY_NAME.pem"

log "Starting EC2 basics tutorial script"
log "Random identifier: $RANDOM_ID"
log "Key name: $KEY_NAME"
log "Security group name: $SG_NAME"

# Step 1: Create a key pair
log "Creating key pair..."
KEY_RESULT=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial' --
output text)
```

```
if [ $? -ne 0 ] || [ -z "$KEY_RESULT" ]; then
  handle_error "Failed to create key pair"
fi

echo "$KEY_RESULT" > "$KEY_FILE"
chmod 400 "$KEY_FILE"
log "Created key pair and saved to $KEY_FILE"

# Step 2: Create a security group
log "Creating security group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group \
  --group-name "$SG_NAME" \
  --description "Security group for EC2 tutorial" \
  --tag-specifications 'ResourceType=security-group,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --query "GroupId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$SECURITY_GROUP_ID" ]; then
  handle_error "Failed to create security group"
fi

log "Created security group: $SECURITY_GROUP_ID"

# Get current public IP address for SSH access
MY_IP=$(curl -s http://checkip.amazonaws.com)
if [ $? -ne 0 ] || [ -z "$MY_IP" ]; then
  handle_error "Failed to get current IP address"
fi

log "Adding SSH ingress rule for IP $MY_IP..."
aws ec2 authorize-security-group-ingress \
  --group-id "$SECURITY_GROUP_ID" \
  --protocol tcp \
  --port 22 \
  --cidr "$MY_IP/32" > /dev/null

if [ $? -ne 0 ]; then
  handle_error "Failed to add security group ingress rule"
fi

log "Added SSH ingress rule for IP $MY_IP"

# Step 3: Find an Amazon Linux 2023 AMI (updated from AL2)
```

```
log "Finding latest Amazon Linux 2023 AMI..."
AMI_ID=$(aws ssm get-parameters-by-path \
  --path "/aws/service/ami-amazon-linux-latest" \
  --query "Parameters[?contains(Name, 'al2023-ami-kernel-default-x86_64')].Value" \
  --output text | head -1)

if [ $? -ne 0 ] || [ -z "$AMI_ID" ]; then
  handle_error "Failed to find Amazon Linux 2023 AMI"
fi

log "Selected AMI: $AMI_ID"

# Get the architecture of the AMI
log "Getting AMI architecture..."
AMI_ARCH=$(aws ec2 describe-images \
  --image-ids "$AMI_ID" \
  --query "Images[0].Architecture" \
  --output text)

if [ $? -ne 0 ] || [ -z "$AMI_ARCH" ]; then
  handle_error "Failed to get AMI architecture"
fi

log "AMI architecture: $AMI_ARCH"

# Find a compatible instance type
log "Finding compatible instance type..."
# Directly use t2.micro for simplicity
INSTANCE_TYPE="t2.micro"
log "Using instance type: $INSTANCE_TYPE"

# Step 4: Launch an EC2 instance with enhanced security
log "Launching EC2 instance with IMDSv2 and encryption enabled..."
INSTANCE_ID=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type "$INSTANCE_TYPE" \
  --key-name "$KEY_NAME" \
  --security-group-ids "$SECURITY_GROUP_ID" \
  --metadata-options "HttpTokens=required,HttpEndpoint=enabled" \
  --block-device-mappings "DeviceName=/dev/xvda,Ebs={Encrypted=true}" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' 'ResourceType=volume,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=ec2-basics}]' \
  --count 1 \
```

```
  --query 'Instances[0].InstanceId' \
  --output text)

if [ $? -ne 0 ] || [ -z "$INSTANCE_ID" ]; then
  handle_error "Failed to launch EC2 instance"
fi

log "Launched instance $INSTANCE_ID. Waiting for it to start..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get instance details
INSTANCE_DETAILS=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].
{ID:InstanceId,Type:InstanceType,State:State.Name,PublicIP:PublicIpAddress}' \
  --output json)

if [ $? -ne 0 ]; then
  handle_error "Failed to get instance details"
fi

log "Instance details: $INSTANCE_DETAILS"

# Get the public IP address
PUBLIC_IP=$(echo "$INSTANCE_DETAILS" | grep -oP '"PublicIP": "\K[^"]+')
if [ -z "$PUBLIC_IP" ]; then
  handle_error "Failed to get instance public IP"
fi

log "Instance public IP: $PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$PUBLIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 6: Stop and Start the Instance
log "Stopping instance $INSTANCE_ID..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
```

```
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi

log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Get the new public IP address
NEW_PUBLIC_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$NEW_PUBLIC_IP" ]; then
  handle_error "Failed to get new public IP"
fi

log "Instance restarted with new public IP: $NEW_PUBLIC_IP"
log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$NEW_PUBLIC_IP"

# Step 7: Allocate and Associate an Elastic IP Address
log "Allocating Elastic IP address..."
ALLOCATION_RESULT=$(aws ec2 allocate-address \
  --domain vpc \
  --tag-specifications 'ResourceType=elastic-ip,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=ec2-basics}]' \
  --query '[PublicIp,AllocationId]' \
  --output text)

if [ $? -ne 0 ] || [ -z "$ALLOCATION_RESULT" ]; then
```

```
  handle_error "Failed to allocate Elastic IP"
fi

ELASTIC_IP=$(echo "$ALLOCATION_RESULT" | awk '{print $1}')
ALLOCATION_ID=$(echo "$ALLOCATION_RESULT" | awk '{print $2}')

log "Allocated Elastic IP: $ELASTIC_IP with ID: $ALLOCATION_ID"

log "Associating Elastic IP with instance..."
ASSOCIATION_ID=$(aws ec2 associate-address \
  --instance-id "$INSTANCE_ID" \
  --allocation-id "$ALLOCATION_ID" \
  --query "AssociationId" \
  --output text)

if [ $? -ne 0 ] || [ -z "$ASSOCIATION_ID" ]; then
  handle_error "Failed to associate Elastic IP"
fi

log "Associated Elastic IP with instance. Association ID: $ASSOCIATION_ID"
log "To connect to your instance using the Elastic IP, run: ssh -i $KEY_FILE ec2-
user@$ELASTIC_IP"

# Pause to allow user to connect if desired
sleep 2

# Step 8: Test the Elastic IP by Stopping and Starting the Instance
log "Stopping instance $INSTANCE_ID to test Elastic IP persistence..."
aws ec2 stop-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to stop instance"
fi

log "Waiting for instance to stop..."
aws ec2 wait instance-stopped --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to stop"
fi

log "Instance stopped. Starting instance again..."
aws ec2 start-instances --instance-ids "$INSTANCE_ID" > /dev/null
if [ $? -ne 0 ]; then
  handle_error "Failed to start instance"
fi
```

```
log "Waiting for instance to start..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
if [ $? -ne 0 ]; then
  handle_error "Failed while waiting for instance to start"
fi

# Verify the Elastic IP is still associated
CURRENT_IP=$(aws ec2 describe-instances \
  --instance-ids "$INSTANCE_ID" \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text)

if [ $? -ne 0 ] || [ -z "$CURRENT_IP" ]; then
  handle_error "Failed to get current public IP"
fi

log "Current public IP address: $CURRENT_IP"
log "Elastic IP address: $ELASTIC_IP"

if [ "$CURRENT_IP" = "$ELASTIC_IP" ]; then
  log "Success! The Elastic IP is still associated with your instance."
else
  log "Something went wrong. The Elastic IP is not associated with your instance."
fi

log "To connect to your instance, run: ssh -i $KEY_FILE ec2-user@$ELASTIC_IP"

# Step 9: Clean up resources
log "Tutorial completed successfully!"
cleanup

exit 0
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair

- DeleteSecurityGroup
- DescribeImages
- DescribeInstances
- DisassociateAddress
- GetParametersByPath
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- Wait

Systemverwaltung einrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Berechtigungen für Systems Manager
- Erstellen einer IAM-Rolle für Systems Manager
- Konfigurieren Sie Systems Manager
- Überprüfen Sie das Setup
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
```

```
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}
```

```
# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "==========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id --
policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
```

```
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --manager-
arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null
 || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and region."
echo "It will create IAM policies and roles, then enable Systems Manager features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
```

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmReadOnly",
       "Effect": "Allow",
       "Action": [
         "ssm:DescribeAutomationExecutions",
         "ssm:GetAutomationExecution",
         "ssm:ListAssociations",
         "ssm:DescribeAssociation",
         "ssm:ListDocuments",
         "ssm:ListResourceDataSync",
         "ssm:DescribePatchBaselines",
         "ssm:GetPatchBaseline",
         "ssm:DescribeMaintenanceWindows",
         "ssm:DescribeMaintenanceWindowTasks"
       ],
       "Resource": "*"
     },
     {
       "Sid": "SsmDocument",
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument",
         "ssm:DescribeDocument"
       ],
       "Resource": [
         "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
         "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
       ]
     },
     {
       "Sid": "SsmEnableExplorer",
       "Effect": "Allow",
       "Action": "ssm:StartAutomationExecution",
```

```
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
      {
        "Sid": "SsmExplorerRds",
        "Effect": "Allow",
        "Action": [
          "ssm:GetOpsSummary",
          "ssm:CreateResourceDataSync",
          "ssm:UpdateResourceDataSync"
        ],
        "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
      },
      {
        "Sid": "OrgsReadOnly",
        "Effect": "Allow",
        "Action": [
          "organizations:DescribeAccount",
          "organizations:DescribeOrganization",
          "organizations:ListDelegatedAdministrators",
          "organizations:ListRoots",
          "organizations:ListParents",
          "organizations:ListOrganizationalUnitsForParent",
          "organizations:DescribeOrganizationalUnit",
          "organizations:ListAWSServiceAccessForOrganization"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgsAdministration",
        "Effect": "Allow",
        "Action": [
          "organizations:EnableAWSServiceAccess",
          "organizations:RegisterDelegatedAdministrator",
          "organizations:DeregisterDelegatedAdministrator"
        ],
        "Resource": "*",
        "Condition": {
          "StringEquals": {
            "organizations:ServicePrincipal": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "member.org.stacksets.cloudformation.amazonaws.com",
              "resource-explorer-2.amazonaws.com"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnReadOnly",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ListStacks",
          "cloudformation:DescribeStacks",
          "cloudformation:ListStackSets",
          "cloudformation:DescribeOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "OrgCfnAccess",
        "Effect": "Allow",
        "Action": [
          "cloudformation:ActivateOrganizationsAccess"
        ],
        "Resource": "*"
      },
      {
        "Sid": "CfnStackActions",
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStack",
          "cloudformation:DeleteStack",
          "cloudformation:DescribeStackResources",
          "cloudformation:DescribeStackEvents",
          "cloudformation:GetTemplate",
          "cloudformation:RollbackStack",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStack"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*"
        ]
      },
      {
        "Sid": "CfnStackSetActions",
```

```
        "Effect": "Allow",
        "Action": [
          "cloudformation:CreateStackInstances",
          "cloudformation:CreateStackSet",
          "cloudformation:DeleteStackInstances",
          "cloudformation:DeleteStackSet",
          "cloudformation:DescribeStackInstance",
          "cloudformation:DetectStackSetDrift",
          "cloudformation:ListStackInstanceResourceDrifts",
          "cloudformation:DescribeStackSet",
          "cloudformation:DescribeStackSetOperation",
          "cloudformation:ListStackInstances",
          "cloudformation:ListStackSetOperations",
          "cloudformation:ListStackSetOperationResults",
          "cloudformation:TagResource",
          "cloudformation:UntagResource",
          "cloudformation:UpdateStackSet"
        ],
        "Resource": [
          "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
          "arn:aws:cloudformation:*:*:type/resource/*",
          "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
        ]
      },
      {
        "Sid": "ValidationReadonlyActions",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles",
          "iam:GetRole"
        ],
        "Resource": "*"
      },
      {
        "Sid": "IamRolesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:CreateRole",
          "iam:DeleteRole",
          "iam:GetRole",
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy",
          "iam:GetRolePolicy",
          "iam:ListRolePolicies"
```

```
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ]
      },
      {
        "Sid": "IamPassRole",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": [
              "ssm.amazonaws.com",
              "ssm-quicksetup.amazonaws.com",
              "cloudformation.amazonaws.com"
            ]
          }
        }
      },
      {
        "Sid": "IamRolesPoliciesMgmt",
        "Effect": "Allow",
        "Action": [
          "iam:AttachRolePolicy",
          "iam:DetachRolePolicy"
        ],
        "Resource": [
          "arn:aws:iam::*:role/AWS-QuickSetup-*",
          "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
        ],
        "Condition": {
          "ArnEquals": {
            "iam:PolicyARN": [
              "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
              "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
            ]
```

```
          }
        }
      },
      {
        "Sid": "CfnStackSetsSLR",
        "Effect": "Allow",
        "Action": [
          "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
          "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
          "arn:aws:iam::*:role/aws-service-role/accountdiscovery.ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM_AccountDiscovery",
          "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
          "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
        ]
      }
    ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --policy-
arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"
```

```
# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
```

```
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-arn
 $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================"
echo "CREATED RESOURCES"
echo "============================================"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================"
echo "CLEANUP CONFIRMATION"
echo "============================================"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up later." |
 tee -a "$LOG_FILE"
```

```
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

Beispiele für Amazon Textract mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit Amazon Textract Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit der Extraktion von Dokumententexten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines S3-Buckets
- Laden Sie ein Dokument auf S3 hoch
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "==================================================="
echo "Amazon Textract Getting Started Tutorial"
echo "==================================================="
echo "This script will guide you through using Amazon Textract to analyze document
 text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
```

```
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*/\1=***REDACTED***/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to delete
 document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ]; then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
```

```
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\).*/
\1=***REDACTED***/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
 default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output text
 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
```

```
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
 2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
 {"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text 50,50
 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
```

```
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "./$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256 2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ./$DOCUMENT_NAME s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
```

```
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====================================================="
echo "Analysis Summary"
echo "====================================================="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk '{print
 $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l || echo
 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l || echo
 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l || echo
 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
 echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l || echo
 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"' |
 wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' | wc -
l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
```

```
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "==================================================="
echo "RESOURCES CREATED"
echo "==================================================="
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-results.json)
 has been kept."

echo ""
```

```
echo "======================================================"
echo "Tutorial complete!"
echo "======================================================"
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AnalyzeDocument
  - Cp
  - Hilfe
  - Mb
  - Rb
  - Rm

Beispiele für VPC-Gitter mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des Skripts with Bash AWS Command Line Interface mit VPC Lattice Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen und Verwalten eines Service-Netzwerk-Mesh

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ein Servicenetzwerk erstellen

- Einen Service erstellen
- Verfügbare VPCs auflisten
- Listet Sicherheitsgruppen für die ausgewählte VPC auf
- Listet die Dienstzuordnungen auf
- VPC-Zuordnungen auflisten
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# VPC Lattice Service Network Tutorial Script
# This script demonstrates how to create and manage a VPC Lattice service network

set -euo pipefail

# Set up logging with secure permissions
LOG_FILE="vpc-lattice-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting VPC Lattice tutorial script at $(date)" > "$LOG_FILE"

# Function to log commands and their output
log_command() {
    local cmd="$1"
    echo "$(date): Running command: $cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return "${PIPESTATUS[0]}"
}

# Function to check for errors
check_error() {
    if [ "$1" -ne 0 ]; then
```

```
        echo "ERROR: Command failed with exit code $1" | tee -a "$LOG_FILE"
        echo "See $LOG_FILE for details"
        exit "$1"
    fi
}

# Function to validate AWS CLI is available
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH" | tee -a "$LOG_FILE"
        exit 1
    fi
}

# Function to validate input parameters
validate_input() {
    local input="$1"
    local param_name="$2"

    if [[ -z "$input" ]]; then
        echo "ERROR: $param_name is empty" | tee -a "$LOG_FILE"
        return 1
    fi

    # Validate against common injection patterns
    if [[ "$input" =~ [\;\$\`\|\&\<\>\(\)\{\}] ]]; then
        echo "ERROR: $param_name contains invalid characters" | tee -a "$LOG_FILE"
        return 1
    fi

    return 0
}

# Function to wait for a resource to be in the desired state
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"
    local desired_status="$3"
    local command="$4"
    local max_attempts=30
    local attempt=1
    local status=""

    validate_input "$resource_type" "resource_type" || return 1
```

```
   validate_input "$resource_id" "resource_id" || return 1
   validate_input "$desired_status" "desired_status" || return 1

   echo "Waiting for $resource_type $resource_id to be in state $desired_status..."
| tee -a "$LOG_FILE"

   while [ "$attempt" -le "$max_attempts" ]; do
       echo "Attempt $attempt of $max_attempts..." >> "$LOG_FILE"

       # Run the command to get the status and capture the output
       status_output=$(eval "$command" 2>&1) || true
       echo "$status_output" >> "$LOG_FILE"

       # For service networks, they do not have a status field in the output
       # We'll consider them active if we can retrieve them
       if [[ "$resource_type" == "Service Network" ]]; then
           if [[ "$status_output" == *"$resource_id"* ]]; then
               echo "$resource_type $resource_id is now active" | tee -a
"$LOG_FILE"
               return 0
           fi
       else
           # For other resources, extract the status field
           status=$(echo "$status_output" | grep -i "status" | awk -F'"' '{print
$4}' || true)
           echo "Current status: $status" >> "$LOG_FILE"

           if [[ "$status" == "$desired_status" ]]; then
               echo "$resource_type $resource_id is now in state $desired_status" |
tee -a "$LOG_FILE"
               return 0
           elif [[ "$status" == *"FAIL"* ]]; then
               echo "ERROR: $resource_type $resource_id failed to reach desired
state. Current status: $status" | tee -a "$LOG_FILE"
               return 1
           fi
       fi

       echo "Waiting for status change... (attempt $attempt/$max_attempts)" >>
"$LOG_FILE"
       sleep 10
       ((attempt++))
   done
```

```
    echo "ERROR: Timed out waiting for $resource_type $resource_id to reach state
 $desired_status" | tee -a "$LOG_FILE"
    return 1
}

# Cleanup function for trap
cleanup() {
    local exit_code=$?
    echo "Script interrupted or failed. Cleaning up..." | tee -a "$LOG_FILE"
    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Check prerequisites
check_aws_cli

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
SERVICE_NETWORK_NAME="lattice-network-${RANDOM_ID}"
SERVICE_NAME="lattice-service-${RANDOM_ID}"

# Store created resources for cleanup
declare -a CREATED_RESOURCES

echo "=== VPC Lattice Service Network Tutorial ===" | tee -a "$LOG_FILE"
echo "Random ID for this session: ${RANDOM_ID}" | tee -a "$LOG_FILE"

# Step 1: Create a VPC Lattice service network
echo -e "\n=== Step 1: Creating a VPC Lattice service network ===" | tee -a
 "$LOG_FILE"
echo "Creating service network: $SERVICE_NETWORK_NAME" | tee -a "$LOG_FILE"

SERVICE_NETWORK_OUTPUT=$(log_command "aws vpc-lattice create-service-
network --name '$SERVICE_NETWORK_NAME' --tags Key=project,Value=doc-smith
 Key=tutorial,Value=amazon-vpc-lattice-gs --output json")
check_error $?

# Extract the service network ID using jq for safety
SERVICE_NETWORK_ID=$(echo "$SERVICE_NETWORK_OUTPUT" | jq -r '.id // empty' 2>/dev/
null || true)
if [ -z "$SERVICE_NETWORK_ID" ]; then
    echo "ERROR: Failed to extract service network ID" | tee -a "$LOG_FILE"
    exit 1
```

```
fi

validate_input "$SERVICE_NETWORK_ID" "SERVICE_NETWORK_ID" || exit 1

echo "Service network created with ID: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Network: $SERVICE_NETWORK_ID")

# Wait for the service network to be active
wait_for_resource "Service Network" "$SERVICE_NETWORK_ID" "ACTIVE" "aws vpc-lattice
 get-service-network --service-network-identifier '$SERVICE_NETWORK_ID' --output
 json"
check_error $?

# Step 2: Create a VPC Lattice service
echo -e "\n=== Step 2: Creating a VPC Lattice service ===" | tee -a "$LOG_FILE"
echo "Creating service: $SERVICE_NAME" | tee -a "$LOG_FILE"

SERVICE_OUTPUT=$(log_command "aws vpc-lattice create-service --name '$SERVICE_NAME'
 --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-vpc-lattice-gs --
output json")
check_error $?

# Extract the service ID using jq for safety
SERVICE_ID=$(echo "$SERVICE_OUTPUT" | jq -r '.id // empty' 2>/dev/null || true)
if [ -z "$SERVICE_ID" ]; then
    echo "ERROR: Failed to extract service ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ID" "SERVICE_ID" || exit 1

echo "Service created with ID: $SERVICE_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service: $SERVICE_ID")

# Wait for the service to be active
wait_for_resource "Service" "$SERVICE_ID" "ACTIVE" "aws vpc-lattice get-service --
service-identifier '$SERVICE_ID' --output json"
check_error $?

# Step 3: Associate the service with the service network
echo -e "\n=== Step 3: Associating service with service network ===" | tee -a
 "$LOG_FILE"
```

```
SERVICE_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-service-
association --service-identifier '$SERVICE_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json")
check_error $?

# Extract the service association ID using jq for safety
SERVICE_ASSOC_ID=$(echo "$SERVICE_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
if [ -z "$SERVICE_ASSOC_ID" ]; then
    echo "ERROR: Failed to extract service association ID" | tee -a "$LOG_FILE"
    exit 1
fi

validate_input "$SERVICE_ASSOC_ID" "SERVICE_ASSOC_ID" || exit 1

echo "Service association created with ID: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
CREATED_RESOURCES+=("Service Association: $SERVICE_ASSOC_ID")

# Wait for the service association to be active
wait_for_resource "Service Association" "$SERVICE_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-service-association --service-network-service-
association-identifier '$SERVICE_ASSOC_ID' --output json"
check_error $?

# Step 4: List available VPCs to associate with the service network
echo -e "\n=== Step 4: Listing available VPCs ===" | tee -a "$LOG_FILE"

VPC_LIST=$(log_command "aws ec2 describe-vpcs --query 'Vpcs[*].[VpcId,Tags[?Key==
\`Name\`].Value|[0]]' --output text")
check_error $?

echo "Available VPCs:" | tee -a "$LOG_FILE"
echo "$VPC_LIST" | tee -a "$LOG_FILE"

# Step 5: Auto-select first available VPC
echo -e "\n=== Step 5: Associate a VPC with the service network ===" | tee -a
 "$LOG_FILE"

VPC_ID=$(echo "$VPC_LIST" | head -n 1 | awk '{print $1}')

if [ -z "$VPC_ID" ]; then
    echo "WARNING: No VPC ID found" | tee -a "$LOG_FILE"
    echo "Skipping VPC association step" | tee -a "$LOG_FILE"
else
```

```
    validate_input "$VPC_ID" "VPC_ID" || {
        echo "ERROR: VPC_ID validation failed"
        exit 1
    }

    echo "Auto-selected VPC: $VPC_ID" | tee -a "$LOG_FILE"

    # Step 6: List security groups for the selected VPC
    echo -e "\n=== Step 6: Listing security groups for VPC $VPC_ID ===" | tee -a
 "$LOG_FILE"

    SG_LIST=$(log_command "aws ec2 describe-security-groups --filters Name=vpc-
id,Values='$VPC_ID' --query 'SecurityGroups[*].[GroupId,GroupName]' --output text")
    check_error $?

    echo "Available Security Groups for VPC $VPC_ID:" | tee -a "$LOG_FILE"
    echo "$SG_LIST" | tee -a "$LOG_FILE"

    # Step 7: Auto-select first available security group
    echo -e "\n=== Step 7: Select a security group for the VPC association ===" |
 tee -a "$LOG_FILE"

    SG_ID=$(echo "$SG_LIST" | head -n 1 | awk '{print $1}')

    if [ -z "$SG_ID" ]; then
        echo "WARNING: No Security Group ID found" | tee -a "$LOG_FILE"
        echo "Skipping VPC association step" | tee -a "$LOG_FILE"
    else
        validate_input "$SG_ID" "SG_ID" || {
            echo "ERROR: SG_ID validation failed"
            exit 1
        }

        echo "Auto-selected Security Group: $SG_ID" | tee -a "$LOG_FILE"

        # Step 8: Associate the VPC with the service network
        echo -e "\n=== Step 8: Associating VPC with service network ===" | tee -a
 "$LOG_FILE"

        VPC_ASSOC_OUTPUT=$(log_command "aws vpc-lattice create-service-network-
vpc-association --vpc-identifier '$VPC_ID' --service-network-identifier
 '$SERVICE_NETWORK_ID' --security-group-ids '$SG_ID' --output json")
        check_error $?
```

```
        # Extract the VPC association ID using jq for safety
        VPC_ASSOC_ID=$(echo "$VPC_ASSOC_OUTPUT" | jq -r '.id // empty' 2>/dev/null
 || true)
        if [ -z "$VPC_ASSOC_ID" ]; then
            echo "ERROR: Failed to extract VPC association ID" | tee -a "$LOG_FILE"
        else
            validate_input "$VPC_ASSOC_ID" "VPC_ASSOC_ID" || exit 1

            echo "VPC association created with ID: $VPC_ASSOC_ID" | tee -a
 "$LOG_FILE"
            CREATED_RESOURCES+=("VPC Association: $VPC_ASSOC_ID")

            # Wait for the VPC association to be active
            wait_for_resource "VPC Association" "$VPC_ASSOC_ID" "ACTIVE" "aws vpc-
lattice get-service-network-vpc-association --service-network-vpc-association-
identifier '$VPC_ASSOC_ID' --output json"
            check_error $?
        fi
    fi
fi

# Step 9: Display information about the created resources
echo -e "\n=== Step 9: Displaying information about created resources ===" | tee -a
 "$LOG_FILE"

echo "Service Network Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID' --output json"

echo "Service Details:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice get-service --service-identifier '$SERVICE_ID' --output
 json"

echo "Service Network Service Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-service-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

echo "Service Network VPC Associations:" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice list-service-network-vpc-associations --service-
network-identifier '$SERVICE_NETWORK_ID' --output json"

# Step 10: Cleanup - Auto-confirm
echo -e "\n=== Step 10: Resource Cleanup ===" | tee -a "$LOG_FILE"
echo "Resources created in this tutorial:" | tee -a "$LOG_FILE"
```

```
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource" | tee -a "$LOG_FILE"
done

echo "Starting cleanup process..." | tee -a "$LOG_FILE"

# Delete resources in reverse order

# Delete VPC association if it was created
if [[ -n "${VPC_ASSOC_ID:-}" ]]; then
    echo "Deleting VPC association: $VPC_ASSOC_ID" | tee -a "$LOG_FILE"
    log_command "aws vpc-lattice delete-service-network-vpc-association --service-
network-vpc-association-identifier '$VPC_ASSOC_ID'" || true

    # Wait for the VPC association to be deleted
    echo "Waiting for VPC association to be deleted..." | tee -a "$LOG_FILE"
    sleep 30
fi

# Delete service association
echo "Deleting service association: $SERVICE_ASSOC_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network-service-association --service-
network-service-association-identifier '$SERVICE_ASSOC_ID'" || true

# Wait for the service association to be deleted
echo "Waiting for service association to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service
echo "Deleting service: $SERVICE_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service --service-identifier '$SERVICE_ID'" ||
 true

# Wait for the service to be deleted
echo "Waiting for service to be deleted..." | tee -a "$LOG_FILE"
sleep 30

# Delete service network
echo "Deleting service network: $SERVICE_NETWORK_ID" | tee -a "$LOG_FILE"
log_command "aws vpc-lattice delete-service-network --service-network-identifier
 '$SERVICE_NETWORK_ID'" || true

echo "Cleanup completed successfully!" | tee -a "$LOG_FILE"
```

```
echo -e "\n=== Tutorial completed! ===" | tee -a "$LOG_FILE"
echo "Log file: $LOG_FILE" | tee -a "$LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateService
  - CreateServiceNetwork
  - CreateServiceNetworkServiceAssociation
  - CreateServiceNetworkVpcAssociation
  - DeleteService
  - DeleteServiceNetwork
  - DeleteServiceNetworkServiceAssociation
  - DeleteServiceNetworkVpcAssociation
  - DescribeSecurityGroups
  - DescribeVpcs
  - GetService
  - GetServiceNetwork
  - GetServiceNetworkServiceAssociation
  - GetServiceNetworkVpcAssociation
  - ListServiceNetworkServiceAssociations
  - ListServiceNetworkVpcAssociations

AWS WAFV2 Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. AWS WAFV2

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erste Schritte mit Firewalls für Webanwendungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Web-ACL
- Fügen Sie eine Regel zum Abgleichen von Zeichenfolgen hinzu
- Verwaltete Regeln hinzufügen
- Konfigurieren der -Protokollierung
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====================================================="
echo "AWS WAF Getting Started Tutorial"
echo "====================================================="
```

```
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi

    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "==================================================="
    echo "CLEANING UP RESOURCES"
    echo "==================================================="
```

```
    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true

        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/
dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
```

```
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the AWS
 Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"

echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "==================================================="
echo "STEP 1: Creating Web ACL"
echo "==================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
```

```
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "====================================================="
echo "STEP 2: Adding String Match Rule"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
```

```
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
|| true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
```

```
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "====================================================="
```

```
echo "STEP 3: Adding AWS Managed Rules"
echo "====================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/null)
 || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
```

```
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=$METRIC_NAME"
 \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "====================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "====================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""
```

```
if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "====================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "====================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo "$first_dist" |
 grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text 2>/
dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution successfully."
```

```
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "==================================================="
echo "RESOURCE SUMMARY"
echo "==================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "==================================================="
echo "CLEANUP CONFIRMATION"
echo "==================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "==================================================="
echo "Tutorial completed!"
echo "==================================================="
echo "Log file: $LOG_FILE"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl

- ListDistributions
- UpdateWebAcl

WorkSpaces Beispiele mit AWS CLI mit Bash-Skript

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des with Bash-Skripts AWS Command Line Interface mit Aktionen ausführen und gängige Szenarien implementieren. WorkSpaces

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen und Verwalten persönlicher virtueller Desktops

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie WorkSpaces die Verfügbarkeit in Ihrer Region
- Listet verfügbare WorkSpaces Pakete auf
- Erstellen Sie eine WorkSpace
- Überprüfe den Status deiner WorkSpace
- Bereinigen von Ressourcen

AWS CLI mit Bash-Skript

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im Tutorials und Beispiele-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Script to create a WorkSpace in WorkSpaces Personal
# This script follows the workflow described in the AWS documentation
# https://docs.aws.amazon.com/workspaces/latest/adminguide/create-workspaces-
personal.html

# Set up logging
LOG_FILE="workspaces_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting WorkSpaces creation script"
echo "================================================"

# Initialize resource tracking array
declare -a CREATED_RESOURCES

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    exit 1
}

# Function to check if a command succeeded
check_command() {
    # Check for ResourceNotFound.User error specifically
    if echo "$1" | grep -q "ResourceNotFound.User"; then
        echo ""
        echo "ERROR: User not found in the directory."
        echo ""
        echo "This error occurs when the specified username doesn't exist in the
 directory."
        echo ""
        echo "To resolve this issue:"
        echo "1. Ensure the user exists in the directory before creating a
 WorkSpace."
        echo "2. For Simple AD and AWS Managed Microsoft AD:"
        echo "   - Connect to a directory-joined instance"
        echo "   - Use Active Directory tools to create the user"
```

```
        echo "   - See: https://docs.aws.amazon.com/workspaces/latest/adminguide/
manage-users.html"
        echo ""
        echo "3. For AD Connector:"
        echo "   - Create the user in your on-premises Active Directory"
        echo "   - Ensure proper synchronization with the AD Connector"
        echo ""
        echo "4. Alternatively, you can use the AWS Console to create a WorkSpace,"
        echo "   which can create the user automatically in some directory types."
        echo ""
        handle_error "User '$USERNAME' not found in directory '$DIRECTORY_ID'"
    # Check for other errors
    elif echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Step 0: Select AWS region
echo ""
echo "==============================================="
echo "AWS REGION SELECTION"
echo "==============================================="
echo "Enter the AWS region to use (e.g., us-east-1, us-west-2):"
AWS_REGION="${AWS_REGION:-us-east-1}"

if [ -z "$AWS_REGION" ]; then
    handle_error "Region cannot be empty"
fi

export AWS_DEFAULT_REGION="$AWS_REGION"
echo "Using AWS region: $AWS_REGION"

# Step 1: Prompt for directory ID
echo ""
echo "==============================================="
echo "DIRECTORY SELECTION"
echo "==============================================="
echo "Listing available directories..."

DIRECTORIES_OUTPUT=$(aws workspaces describe-workspace-directories --output json)
check_command "$DIRECTORIES_OUTPUT"
echo "$DIRECTORIES_OUTPUT"

# Extract directory IDs and display them
```

```
DIRECTORY_IDS=$(echo "$DIRECTORIES_OUTPUT" | grep -o '"DirectoryId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$DIRECTORY_IDS" ]; then
    echo "No directories found. Please create a directory first using AWS Directory
 Service."
    echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/register-deregister-directory.html"
    exit 1
fi

echo ""
echo "Available directory IDs:"
echo "$DIRECTORY_IDS"
echo ""
echo "Enter the directory ID you want to use:"
DIRECTORY_ID=$(echo "$DIRECTORY_IDS" | head -1)

# Validate directory ID
if ! echo "$DIRECTORY_IDS" | grep -q "$DIRECTORY_ID"; then
    echo "Directory ID $DIRECTORY_ID not found in the list of available
 directories."
    echo "Please check the ID and try again."
    exit 1
fi

echo "Selected directory ID: $DIRECTORY_ID"

# Step 2: Check if directory is registered with WorkSpaces
echo ""
echo "==============================================="
echo "CHECKING DIRECTORY REGISTRATION"
echo "==============================================="

REGISTERED=$(echo "$DIRECTORIES_OUTPUT" | grep -A 5 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -c "\"State\": \"REGISTERED\"")

if [ "$REGISTERED" -eq 0 ]; then
    echo "Directory $DIRECTORY_ID is not registered with WorkSpaces. Registering
 now..."
    REGISTER_OUTPUT=$(aws workspaces register-workspace-directory --directory-id
 "$DIRECTORY_ID")
    check_command "$REGISTER_OUTPUT"
    echo "Directory registration initiated. This may take a few minutes."
```

```
    # Add to resource tracking
    CREATED_RESOURCES+=("Directory registration: $DIRECTORY_ID")

    # Wait for directory to be registered
    echo "Waiting for directory registration to complete..."
    sleep 30

    # Check registration status
    REGISTRATION_CHECK=$(aws workspaces describe-workspace-directories --directory-
ids "$DIRECTORY_ID")
    check_command "$REGISTRATION_CHECK"

    REGISTRATION_STATE=$(echo "$REGISTRATION_CHECK" | grep -o '"State": "[^"]*' |
 cut -d'"' -f4)
    if [ "$REGISTRATION_STATE" != "REGISTERED" ]; then
        echo "Directory registration is still in progress. Current state:
 $REGISTRATION_STATE"
        echo "Please check the AWS console for the final status."
        echo "You may need to wait a few minutes before proceeding."
    else
        echo "Directory successfully registered with WorkSpaces."
    fi
else
    echo "Directory $DIRECTORY_ID is already registered with WorkSpaces."
fi

# Get directory type to provide appropriate user guidance
DIRECTORY_TYPE=$(echo "$DIRECTORIES_OUTPUT" | grep -A 10 "\"DirectoryId\":
 \"$DIRECTORY_ID\"" | grep -o '"DirectoryType": "[^"]*' | cut -d'"' -f4)
echo "Directory type: $DIRECTORY_TYPE"

# Display user creation guidance based on directory type
echo ""
echo "==============================================="
echo "USER CREATION GUIDANCE"
echo "==============================================="
case "$DIRECTORY_TYPE" in
    "SimpleAD" | "MicrosoftAD")
        echo "For $DIRECTORY_TYPE, users must be created using Active Directory
 tools."
        echo "1. Connect to a directory-joined EC2 instance"
        echo "2. Use Active Directory Users and Computers to create users"
```

```
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
    "ADConnector")
        echo "For AD Connector, users must exist in your on-premises Active
 Directory."
        echo "1. Create the user in your on-premises Active Directory"
        echo "2. Ensure the user is in an OU that is within the scope of your AD
 Connector"
        echo "3. For detailed instructions, see: https://docs.aws.amazon.com/
directoryservice/latest/admin-guide/ad_connector_management.html"
        ;;
    *)
        echo "For this directory type, ensure users exist before creating
 WorkSpaces."
        echo "For detailed instructions, see: https://docs.aws.amazon.com/
workspaces/latest/adminguide/manage-users.html"
        ;;
esac
echo ""

# Step 3: List available bundles
echo ""
echo "==============================================="
echo "BUNDLE SELECTION"
echo "==============================================="
echo "Listing available WorkSpace bundles..."

# Get bundles with a format that's easier to parse
BUNDLES_OUTPUT=$(aws workspaces describe-workspace-bundles --owner AMAZON --output
 text --query "Bundles[*].
[BundleId,Name,ComputeType.Name,RootStorage.Capacity,UserStorage.Capacity]")
check_command "$BUNDLES_OUTPUT"

# Extract bundle information and display in a numbered list
echo "Available bundles:"
echo "-----------------"
echo "NUM | BUNDLE ID | NAME | COMPUTE TYPE | ROOT STORAGE | USER STORAGE"
echo "-------------------------------------------------------------------"

# Create arrays to store bundle information
declare -a BUNDLE_IDS
declare -a BUNDLE_NAMES
```

```
# Process the output to extract bundle information
COUNT=1
while IFS=$'\t' read -r BUNDLE_ID BUNDLE_NAME COMPUTE_TYPE ROOT_STORAGE USER_STORAGE
 || [[ -n "$BUNDLE_ID" ]]; do
    # Store in arrays
    BUNDLE_IDS[$COUNT]="$BUNDLE_ID"
    BUNDLE_NAMES[$COUNT]="$BUNDLE_NAME"

    # Display with number
    echo "$COUNT | $BUNDLE_ID | $BUNDLE_NAME | $COMPUTE_TYPE | $ROOT_STORAGE GB |
 $USER_STORAGE GB"

    ((COUNT++))
done <<< "$BUNDLES_OUTPUT"

# Prompt for selection
echo ""
echo "Enter the number of the bundle you want to use (1-$((COUNT-1))):"
BUNDLE_SELECTION="1"

# Validate selection
if ! [[ "$BUNDLE_SELECTION" =~ ^[0-9]+$ ]] || [ "$BUNDLE_SELECTION" -lt 1 ] ||
 [ "$BUNDLE_SELECTION" -ge "$COUNT" ]; then
    handle_error "Invalid bundle selection. Please enter a number between 1 and
 $((COUNT-1))."
fi

# Get the selected bundle ID
BUNDLE_ID="${BUNDLE_IDS[$BUNDLE_SELECTION]}"
BUNDLE_NAME="${BUNDLE_NAMES[$BUNDLE_SELECTION]}"

echo "Selected bundle: $BUNDLE_NAME (ID: $BUNDLE_ID)"

# Step 4: Prompt for username
echo ""
echo "==============================================="
echo "USER INFORMATION"
echo "==============================================="
echo "Enter the username for the WorkSpace:"
USERNAME="testuser-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n 1)"

echo "NOTE: The user must already exist in the directory for the WorkSpace creation
 to succeed."
```

```
echo "If you're using Simple AD or AWS Managed Microsoft AD, the user must be
 created using Active Directory tools."
echo "If you're using AD Connector, the user must exist in your on-premises Active
 Directory."
echo ""

echo "Enter the user's first name:"
FIRST_NAME="Test"

echo "Enter the user's last name:"
LAST_NAME="User"

echo "Enter the user's email address:"
EMAIL="user-$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 8 | head -n
 1)@example.com"

# Step 5: Choose running mode
echo ""
echo "================================================"
echo "RUNNING MODE SELECTION"
echo "================================================"
echo "Select running mode:"
echo "1. AlwaysOn (billed monthly)"
echo "2. AutoStop (billed hourly)"
RUNNING_MODE_CHOICE="1"

if [ "$RUNNING_MODE_CHOICE" = "1" ]; then
    RUNNING_MODE="ALWAYS_ON"
    AUTO_STOP_TIMEOUT=""
else
    RUNNING_MODE="AUTO_STOP"
    AUTO_STOP_TIMEOUT=60
fi

echo "Selected running mode: $RUNNING_MODE"

# Step 6: Add tags (optional)
echo ""
echo "================================================"
echo "TAGS (OPTIONAL)"
echo "================================================"
echo "Would you like to add tags to your WorkSpace? (y/n):"
ADD_TAGS="n"
```

```
TAGS_JSON=""
if [ "$ADD_TAGS" = "y" ] || [ "$ADD_TAGS" = "Y" ]; then
    echo "Enter tag key (e.g., Department):"
    read -r TAG_KEY

    echo "Enter tag value (e.g., IT):"
    read -r TAG_VALUE

    TAGS_JSON="[{\"Key\":\"$TAG_KEY\",\"Value\":\"$TAG_VALUE\"}]"
fi

# Step 7: Create the WorkSpace
echo ""
echo "==============================================="
echo "CREATING WORKSPACE"
echo "==============================================="
echo "Creating WorkSpace with the following parameters:"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Bundle ID: $BUNDLE_ID"
echo "Running Mode: $RUNNING_MODE"
if [ -n "$TAGS_JSON" ]; then
    echo "Tags: $TAG_KEY=$TAG_VALUE"
fi

# Create JSON for workspace properties
if [ "$RUNNING_MODE" = "AUTO_STOP" ]; then
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\",
\"RunningModeAutoStopTimeoutInMinutes\":$AUTO_STOP_TIMEOUT}"
else
    PROPERTIES_JSON="{\"RunningMode\":\"$RUNNING_MODE\"}"
fi

# Create JSON for workspaces parameter
WORKSPACE_JSON="{\"DirectoryId\":\"$DIRECTORY_ID\",\"UserName\":\"$USERNAME\",
\"BundleId\":\"$BUNDLE_ID\",\"WorkspaceProperties\":$PROPERTIES_JSON"

# Add tags if specified
if [ -n "$TAGS_JSON" ]; then
    USER_TAGS=$(echo "$TAGS_JSON" | sed 's/^\[//;s/\]$//')
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[$USER_TAGS,{\"Key\":\"project\",
\"Value\":\"doc-smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
else
```

```
    WORKSPACE_JSON="$WORKSPACE_JSON,\"Tags\":[{\"Key\":\"project\",\"Value\":\"doc-
smith\"},{\"Key\":\"tutorial\",\"Value\":\"workspaces-personal\"}]"
fi

# Close the JSON object
WORKSPACE_JSON="$WORKSPACE_JSON}"

# Construct the create-workspaces command
CREATE_COMMAND="aws workspaces create-workspaces --workspaces '$WORKSPACE_JSON'"

echo "Executing: $CREATE_COMMAND"
CREATE_OUTPUT=$(eval "$CREATE_COMMAND")
check_command "$CREATE_OUTPUT"
echo "$CREATE_OUTPUT"

# Extract WorkSpace ID
WORKSPACE_ID=$(echo "$CREATE_OUTPUT" | grep -o '"WorkspaceId": "[^"]*' | head -1 |
 cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    handle_error "Failed to extract WorkSpace ID from creation output."
fi

echo "WorkSpace creation initiated. WorkSpace ID: $WORKSPACE_ID"
CREATED_RESOURCES+=("WorkSpace: $WORKSPACE_ID")

# Step 8: Check WorkSpace status
echo ""
echo "==============================================="
echo "CHECKING WORKSPACE STATUS"
echo "==============================================="
echo "Checking status of WorkSpace $WORKSPACE_ID..."

# Initial status check
STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids "$WORKSPACE_ID")
check_command "$STATUS_OUTPUT"
echo "$STATUS_OUTPUT"

WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 | cut
 -d'"' -f4)
echo "Current WorkSpace state: $WORKSPACE_STATE"

# Wait for WorkSpace to be available (this can take 20+ minutes)
echo ""
```

```
echo "WorkSpace creation is in progress. This can take 20+ minutes."
echo "The script will check the status every 60 seconds."
echo "Press Ctrl+C to exit the script at any time. The WorkSpace will continue to be
 created."

while [ "$WORKSPACE_STATE" = "PENDING" ]; do
    echo "$(date): WorkSpace state is still PENDING. Waiting 60 seconds before
 checking again..."
    sleep 60

    STATUS_OUTPUT=$(aws workspaces describe-workspaces --workspace-ids
 "$WORKSPACE_ID")
    check_command "$STATUS_OUTPUT"

    WORKSPACE_STATE=$(echo "$STATUS_OUTPUT" | grep -o '"State": "[^"]*' | head -1 |
 cut -d'"' -f4)
    echo "$(date): Current WorkSpace state: $WORKSPACE_STATE"

    # If state is ERROR or UNHEALTHY, exit
    if [ "$WORKSPACE_STATE" = "ERROR" ] || [ "$WORKSPACE_STATE" = "UNHEALTHY" ];
 then
        handle_error "WorkSpace creation failed. Final state: $WORKSPACE_STATE"
    fi

    # If state is AVAILABLE, break the loop
    if [ "$WORKSPACE_STATE" = "AVAILABLE" ]; then
        break
    fi
done

# Step 9: Display WorkSpace information
echo ""
echo "================================================"
echo "WORKSPACE CREATION COMPLETE"
echo "================================================"
echo "WorkSpace has been successfully created!"
echo "WorkSpace ID: $WORKSPACE_ID"
echo "Directory ID: $DIRECTORY_ID"
echo "Username: $USERNAME"
echo "Running Mode: $RUNNING_MODE"

# Step 10: Remind about invitation emails
echo ""
echo "================================================"
```

```
echo "INVITATION EMAILS"
echo "================================================"
echo "IMPORTANT: If you're using AD Connector or a trust relationship, or if the
 user already exists in Active Directory,"
echo "invitation emails are not sent automatically. You'll need to manually send an
 invitation email."
echo "For more information, see: https://docs.aws.amazon.com/workspaces/latest/
adminguide/manage-workspaces-users.html#send-invitation"

# Step 11: Cleanup confirmation
echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

echo ""
echo "Do you want to clean up all created resources? (y/n):"
CLEANUP_CHOICE="y"

if [ "$CLEANUP_CHOICE" = "y" ] || [ "$CLEANUP_CHOICE" = "Y" ]; then
    echo ""
    echo "================================================"
    echo "CLEANING UP RESOURCES"
    echo "================================================"

    # Terminate WorkSpace
    if [ -n "$WORKSPACE_ID" ]; then
        echo "Terminating WorkSpace $WORKSPACE_ID..."
        TERMINATE_OUTPUT=$(aws workspaces terminate-workspaces --terminate-
workspace-requests WorkspaceId="$WORKSPACE_ID")
        check_command "$TERMINATE_OUTPUT"
        echo "$TERMINATE_OUTPUT"
        echo "WorkSpace termination initiated. This may take a few minutes."
    fi

    # Deregister directory (only if we registered it in this script)
    if [[ " ${CREATED_RESOURCES[*]} " == *"Directory registration:
 $DIRECTORY_ID"* ]]; then
        echo "Deregistering directory $DIRECTORY_ID from WorkSpaces..."
```

```
        DEREGISTER_OUTPUT=$(aws workspaces deregister-workspace-directory --
directory-id "$DIRECTORY_ID")
        check_command "$DEREGISTER_OUTPUT"
        echo "$DEREGISTER_OUTPUT"
        echo "Directory deregistration initiated. This may take a few minutes."
    fi

    echo "Cleanup completed."
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo ""
echo "================================================"
echo "SCRIPT COMPLETED"
echo "================================================"
echo "Log file: $LOG_FILE"
echo "Thank you for using the WorkSpaces creation script!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI -Befehlsreferenz.
  - CreateWorkspaces
  - DeregisterWorkspaceDirectory
  - DescribeWorkspaceBundles
  - DescribeWorkspaceDirectories
  - DescribeWorkspaces
  - RegisterWorkspaceDirectory
  - TerminateWorkspaces

Codebeispiele für SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für C++-Entwicklerhandbuch — Weitere Informationen zur Verwendung von C++ mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele unter Verwendung von SDK für C++
- API-Gateway-Beispiele unter Verwendung von SDK für C++
- Aurora-Beispiele unter Verwendung von SDK für C++
- Auto-Scaling-Beispiele unter Verwendung von SDK für C++
- CloudTrail Beispiele zur Verwendung des SDK für C++
- CloudWatch Beispiele zur Verwendung des SDK für C++
- CloudWatch Protokolliert Beispiele mithilfe des SDK für C++
- CodeBuild Beispiele zur Verwendung des SDK für C++
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für C++
- DynamoDB-Beispiele unter Verwendung von SDK für C++
- Amazon-EC2-Beispiele unter Verwendung von SDK für C++
- EventBridge Beispiele zur Verwendung des SDK für C++
- AWS Glue Beispiele zur Verwendung des SDK für C++
- HealthImaging Beispiele zur Verwendung des SDK für C++
- IAM-Beispiele unter Verwendung von SDK für C++
- AWS IoT Beispiele zur Verwendung des SDK für C++
- AWS IoT data Beispiele zur Verwendung des SDK für C++
- Lambda-Beispiele unter Verwendung von SDK für C++
- MediaConvert Beispiele zur Verwendung des SDK für C++

- Amazon-RDS-Beispiele unter Verwendung von SDK für C++
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für C++
- Amazon-Rekognition-Beispiele unter Verwendung von SDK für C++
- Amazon-S3-Beispiele unter Verwendung von SDK für C++
- Secrets-Manager-Beispiele unter Verwendung von SDK für C++
- Amazon-SES-Beispiele unter Verwendung von SDK für C++
- Amazon-SNS-Beispiele unter Verwendung von SDK für C++
- Amazon-SQS-Beispiele unter Verwendung von SDK für C++
- AWS STS Beispiele mit SDK für C++
- Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für C++

ACM-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von ACM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AddTagsToCertificate`

Das folgende Codebeispiel zeigt die Verwendung`AddTagsToCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add tags to an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::addTagsToCertificate(const Aws::String &certificateArn,
                                       const Aws::String &tagKey,
                                       const Aws::String &tagValue,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::AddTagsToCertificateRequest request;
    Aws::Vector<Aws::ACM::Model::Tag> tags;
    Aws::ACM::Model::Tag tag;

    tag.WithKey(tagKey).WithValue(tagValue);
    tags.push_back(tag);

    request.WithCertificateArn(certificateArn).WithTags(tags);

    Aws::ACM::Model::AddTagsToCertificateOutcome outcome =
            acmClient.AddTagsToCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: addTagsToCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey <<
                  "' and value '" << tagValue <<
```

```
                    "' added to certificate with ARN '" <<
                    certificateArn << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie AddTagsToCertificate in der AWS SDK für C++ API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::deleteCertificate(const Aws::String &certificateArn,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::DeleteCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DeleteCertificateOutcome outcome =
            acmClient.DeleteCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteCertificate: " <<
                    outcome.GetError().GetMessage() << std::endl;
```

```
    }
    else {
        std::cout << "Success: The certificate with the ARN '" <<
                  certificateArn << "' is deleted." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteCertificate in der AWS SDK für C++ API-Referenz.

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::describeCertificate(const Aws::String &certificateArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::DescribeCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DescribeCertificateOutcome outcome =
            acm_client.DescribeCertificate(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error: DescribeCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::ACM::Model::CertificateDetail certificate =
                outcome.GetResult().GetCertificate();

        std::cout << "Success: Information about certificate "
                     "with ARN '" << certificateArn << "':" << std::endl <<
std::endl;

        std::cout << "ARN:                 " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << "Authority ARN:       " <<
                  certificate.GetCertificateAuthorityArn() << std::endl;
        std::cout << "Created at (GMT):    " <<
                  certificate.GetCreatedAt().ToGmtString(
                          Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
        std::cout << "Domain name:         " << certificate.GetDomainName()
                  << std::endl;

        Aws::Vector<Aws::ACM::Model::DomainValidation> options =
                certificate.GetDomainValidationOptions();

        if (!options.empty()) {
            std::cout << std::endl << "Domain validation information: "
                      << std::endl << std::endl;

            for (auto &validation: options) {
                std::cout << "  Domain name:              " <<
                          validation.GetDomainName() << std::endl;

                const Aws::ACM::Model::ResourceRecord &record =
                        validation.GetResourceRecord();

                std::cout << "  Resource record name:     " <<
                          record.GetName() << std::endl;

                Aws::ACM::Model::RecordType recordType = record.GetType();
                Aws::String type;

                switch (recordType) {
                    case Aws::ACM::Model::RecordType::CNAME:
```

```
                type = "CNAME";
                break;
            case Aws::ACM::Model::RecordType::NOT_SET:
                type = "Not set";
                break;
            default:
                type = "Cannot determine.";
                break;
        }

        std::cout << "  Resource record type:     " << type <<
                  std::endl;

        std::cout << "  Resource record value:    " <<
                  record.GetValue() << std::endl;

        std::cout << "  Validation domain:        " <<
                  validation.GetValidationDomain() << std::endl;

        Aws::Vector<Aws::String> emails =
                validation.GetValidationEmails();

        if (!emails.empty()) {
            std::cout << "  Validation emails:" << std::endl <<
                      std::endl;

            for (auto &email: emails) {
                std::cout << "    " << email << std::endl;
            }

            std::cout << std::endl;
        }

        Aws::ACM::Model::ValidationMethod validationMethod =
                validation.GetValidationMethod();
        Aws::String method;

        switch (validationMethod) {
            case Aws::ACM::Model::ValidationMethod::DNS:
                method = "DNS";
                break;
            case Aws::ACM::Model::ValidationMethod::EMAIL:
                method = "Email";
                break;
```

```
                    case Aws::ACM::Model::ValidationMethod::NOT_SET:
                        method = "Not set";
                        break;
                    default:
                        method = "Cannot determine";
                }

                std::cout << "  Validation method:        " <<
                          method << std::endl;

                Aws::ACM::Model::DomainStatus domainStatus =
                        validation.GetValidationStatus();
                Aws::String status;

                switch (domainStatus) {
                    case Aws::ACM::Model::DomainStatus::FAILED:
                        status = "Failed";
                        break;
                    case Aws::ACM::Model::DomainStatus::NOT_SET:
                        status = "Not set";
                        break;
                    case Aws::ACM::Model::DomainStatus::PENDING_VALIDATION:
                        status = "Pending validation";
                        break;
                    case Aws::ACM::Model::DomainStatus::SUCCESS:
                        status = "Success";
                        break;
                    default:
                        status = "Cannot determine";
                }

                std::cout << "  Domain validation status: " << status <<
                          std::endl << std::endl;

            }
        }

        Aws::Vector<Aws::ACM::Model::ExtendedKeyUsage> usages =
                certificate.GetExtendedKeyUsages();

        if (!usages.empty()) {
            std::cout << std::endl << "Extended key usages:" <<
                      std::endl << std::endl;
```

```
            for (auto &usage: usages) {
                Aws::ACM::Model::ExtendedKeyUsageName usageName =
                        usage.GetName();
                Aws::String name;

                switch (usageName) {
                    case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
                        name = "Any";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
                        name = "Code signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
                        name = "Custom";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
                        name = "Email protection";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
                        name = "IPSEC end system";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
                        name = "IPSEC tunnel";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
                        name = "IPSEC user";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
                        name = "None";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
                        name = "Not set";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
                        name = "OCSP signing";
                        break;
                    case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
                        name = "Time stamping";
                        break;
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
                        name = "TLS web client authentication";
                        break;
```

```
                    case
Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
                        name = "TLS web server authentication";
                        break;
                    default:
                        name = "Cannot determine";
                }

                std::cout << "  Name: " << name << std::endl;
                std::cout << "  OID:  " << usage.GetOID() <<
                          std::endl << std::endl;
            }

            std::cout << std::endl;
        }

        Aws::ACM::Model::CertificateStatus certificateStatus =
                certificate.GetStatus();
        Aws::String status;

        switch (certificateStatus) {
            case Aws::ACM::Model::CertificateStatus::EXPIRED:
                status = "Expired";
                break;
            case Aws::ACM::Model::CertificateStatus::FAILED:
                status = "Failed";
                break;
            case Aws::ACM::Model::CertificateStatus::INACTIVE:
                status = "Inactive";
                break;
            case Aws::ACM::Model::CertificateStatus::ISSUED:
                status = "Issued";
                break;
            case Aws::ACM::Model::CertificateStatus::NOT_SET:
                status = "Not set";
                break;
            case Aws::ACM::Model::CertificateStatus::PENDING_VALIDATION:
                status = "Pending validation";
                break;
            case Aws::ACM::Model::CertificateStatus::REVOKED:
                status = "Revoked";
                break;
            case Aws::ACM::Model::CertificateStatus::VALIDATION_TIMED_OUT:
                status = "Validation timed out";
```

```
                break;
            default:
                status = "Cannot determine";
        }

        std::cout << "Status:              " << status << std::endl;

        if (certificate.GetStatus() ==
            Aws::ACM::Model::CertificateStatus::FAILED) {
            Aws::ACM::Model::FailureReason failureReason =
                    certificate.GetFailureReason();
            Aws::String reason;

            switch (failureReason) {
                case
Aws::ACM::Model::FailureReason::ADDITIONAL_VERIFICATION_REQUIRED:
                    reason = "Additional verification required";
                    break;
                case Aws::ACM::Model::FailureReason::CAA_ERROR:
                    reason = "CAA error";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_NOT_ALLOWED:
                    reason = "Domain not allowed";
                    break;
                case Aws::ACM::Model::FailureReason::DOMAIN_VALIDATION_DENIED:
                    reason = "Domain validation denied";
                    break;
                case Aws::ACM::Model::FailureReason::INVALID_PUBLIC_DOMAIN:
                    reason = "Invalid public domain";
                    break;
                case Aws::ACM::Model::FailureReason::NOT_SET:
                    reason = "Not set";
                    break;
                case Aws::ACM::Model::FailureReason::NO_AVAILABLE_CONTACTS:
                    reason = "No available contacts";
                    break;
                case Aws::ACM::Model::FailureReason::OTHER:
                    reason = "Other";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_ACCESS_DENIED:
                    reason = "PCA access denied";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARGS:
                    reason = "PCA invalid args";
```

```
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_ARN:
                    reason = "PCA invalid ARN";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_DURATION:
                    reason = "PCA invalid duration";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_INVALID_STATE:
                    reason = "PCA invalid state";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_LIMIT_EXCEEDED:
                    reason = "PCA limit exceeded";
                    break;
                case
Aws::ACM::Model::FailureReason::PCA_NAME_CONSTRAINTS_VALIDATION:
                    reason = "PCA name constraints validation";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_REQUEST_FAILED:
                    reason = "PCA request failed";
                    break;
                case Aws::ACM::Model::FailureReason::PCA_RESOURCE_NOT_FOUND:
                    reason = "PCA resource not found";
                    break;
                default:
                    reason = "Cannot determine";
            }

            std::cout << "Failure reason:      " << reason << std::endl;
        }

        if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::REVOKED)
{
            std::cout << "Revoked at (GMT):    " <<
                      certificate.GetRevokedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;

            Aws::ACM::Model::RevocationReason revocationReason =
                    certificate.GetRevocationReason();
            Aws::String reason;

            switch (revocationReason) {
                case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
                    reason = "Affiliation changed";
```

```
                    break;
                case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
                    reason = "AA compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::CA_COMPROMISE:
                    reason = "CA compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::CERTIFICATE_HOLD:
                    reason = "Certificate hold";
                    break;
                case Aws::ACM::Model::RevocationReason::CESSATION_OF_OPERATION:
                    reason = "Cessation of operation";
                    break;
                case Aws::ACM::Model::RevocationReason::KEY_COMPROMISE:
                    reason = "Key compromise";
                    break;
                case Aws::ACM::Model::RevocationReason::NOT_SET:
                    reason = "Not set";
                    break;
                case Aws::ACM::Model::RevocationReason::PRIVILEGE_WITHDRAWN:
                    reason = "Privilege withdrawn";
                    break;
                case Aws::ACM::Model::RevocationReason::REMOVE_FROM_CRL:
                    reason = "Revoke from CRL";
                    break;
                case Aws::ACM::Model::RevocationReason::SUPERCEDED:
                    reason = "Superceded";
                    break;
                case Aws::ACM::Model::RevocationReason::UNSPECIFIED:
                    reason = "Unspecified";
                    break;
                default:
                    reason = "Cannot determine";
            }

            std::cout << "Revocation reason:   " << reason << std::endl;
        }

        if (certificate.GetType() == Aws::ACM::Model::CertificateType::IMPORTED) {
            std::cout << "Imported at (GMT):   " <<
                      certificate.GetImportedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }
```

```
        Aws::Vector<Aws::String> inUseBys = certificate.GetInUseBy();

        if (!inUseBys.empty()) {
            std::cout << std::endl << "In use by:" << std::endl << std::endl;

            for (auto &in_use_by: inUseBys) {
                std::cout << "  " << in_use_by << std::endl;
            }

            std::cout << std::endl;
        }

        if (certificate.GetType() == Aws::ACM::Model::CertificateType::AMAZON_ISSUED
&&
            certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Issued at (GMT):     " <<
                      certificate.GetIssuedAt().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        std::cout << "Issuer:              " << certificate.GetIssuer() <<
                  std::endl;

        Aws::ACM::Model::KeyAlgorithm keyAlgorithm =
                certificate.GetKeyAlgorithm();
        Aws::String algorithm;

        switch (keyAlgorithm) {
            case Aws::ACM::Model::KeyAlgorithm::EC_prime256v1:
                algorithm = "P-256 (secp256r1, prime256v1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp384r1:
                algorithm = "P-384 (secp384r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::EC_secp521r1:
                algorithm = "P-521 (secp521r1)";
                break;
            case Aws::ACM::Model::KeyAlgorithm::NOT_SET:
                algorithm = "Not set";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_1024:
                algorithm = "RSA 1024";
```

```
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_2048:
                algorithm = "RSA 2048";
                break;
            case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
                algorithm = "RSA 4096";
                break;
            default:
                algorithm = "Cannot determine";
        }

        std::cout << "Key algorithm:       " << algorithm << std::endl;

        if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
            std::cout << "Not valid after (GMT): " <<
                      certificate.GetNotAfter().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
            std::cout << "Not valid before (GMT): " <<
                      certificate.GetNotBefore().ToGmtString(
                              Aws::Utils::DateFormat::ISO_8601)
                      << std::endl;
        }

        Aws::ACM::Model::CertificateTransparencyLoggingPreference loggingPreference
=

certificate.GetOptions().GetCertificateTransparencyLoggingPreference();
        Aws::String preference;

        switch (loggingPreference) {
            case
Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED:
                preference = "Disabled";
                break;
            case Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED:
                preference = "Enabled";
                break;
            case Aws::ACM::Model::CertificateTransparencyLoggingPreference::NOT_SET:
                preference = "Not set";
                break;
            default:
                preference = "Cannot determine";
        }
```

```
        std::cout << "Logging preference:  " << preference << std::endl;

        std::cout << "Serial:              " << certificate.GetSerial() <<
                  std::endl;
        std::cout << "Signature algorithm: "
                  << certificate.GetSignatureAlgorithm() << std::endl;
        std::cout << "Subject:             " << certificate.GetSubject() <<
                  std::endl;

        Aws::ACM::Model::CertificateType certificateType = certificate.GetType();
        Aws::String type;

        switch (certificateType) {
            case Aws::ACM::Model::CertificateType::AMAZON_ISSUED:
                type = "Amazon issued";
                break;
            case Aws::ACM::Model::CertificateType::IMPORTED:
                type = "Imported";
                break;
            case Aws::ACM::Model::CertificateType::NOT_SET:
                type = "Not set";
                break;
            case Aws::ACM::Model::CertificateType::PRIVATE_:
                type = "Private";
                break;
            default:
                type = "Cannot determine";
        }

        std::cout << "Type:                " << type << std::endl;

        Aws::Vector<Aws::String> altNames =
                certificate.GetSubjectAlternativeNames();

        if (!altNames.empty()) {
            std::cout << std::endl << "Alternative names:" <<
                      std::endl << std::endl;

            for (auto &alt_name: altNames) {
                std::cout << "  " << alt_name << std::endl;
            }

            std::cout << std::endl;
```

```
        }
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeCertificate in der AWS SDK für C++ API-Referenz.

`ExportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ExportCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Export an AWS Certificate Manager (ACM)  certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param passphrase: A passphrase to decrypt the exported certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::exportCertificate(const Aws::String &certificateArn,
                                    const Aws::String &passphrase,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ExportCertificateRequest request;
    Aws::Utils::CryptoBuffer cryptoBuffer(
            reinterpret_cast<const unsigned char *>(passphrase.c_str()),
            passphrase.length());
    request.WithCertificateArn(certificateArn).WithPassphrase(cryptoBuffer);

    Aws::ACM::Model::ExportCertificateOutcome outcome =
```

```
            acm_client.ExportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ExportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate:       " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl << std::endl;
        std::cout << "Private key:       " << std::endl << std::endl <<
                  result.GetPrivateKey() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie ExportCertificate in der AWS SDK für C++ API-Referenz.

`GetCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::getCertificate(const Aws::String &certificateArn,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::GetCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::GetCertificateOutcome outcome =
            acmClient.GetCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: GetCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate: " << std::endl << std::endl <<
                  result.GetCertificate() << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
                  result.GetCertificateChain() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetCertificate in der AWS SDK für C++ API-Referenz.

`ImportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ImportCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Import an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateFile: Path to certificate to import.
  \param privateKeyFile: Path to file containing a private key.
  \param certificateChainFile: Path to file containing a PEM encoded certificate
 chain.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                    const Aws::String &privateKeyFile,
                                    const Aws::String &certificateChainFile,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream privateKeyInstream(privateKeyFile.c_str());
    if (!privateKeyInstream) {
        std::cerr << "Error: The private key file '" << privateKeyFile <<
                  "' does not exist." << std::endl;

        return false;
    }

    std::ifstream certificateChainInStream(certificateChainFile.c_str());
    if (!certificateChainInStream) {
        std::cerr << "Error: The certificate chain file '"
                  << certificateChainFile << "' does not exist." << std::endl;
```

```
        return false;
    }

    Aws::String certificate;
    certificate.assign(std::istreambuf_iterator<char>(certificateInStream),
                       std::istreambuf_iterator<char>());

    Aws::String privateKey;
    privateKey.assign(std::istreambuf_iterator<char>(privateKeyInstream),
                      std::istreambuf_iterator<char>());

    Aws::String certificateChain;

certificateChain.assign(std::istreambuf_iterator<char>(certificateChainInStream),
                            std::istreambuf_iterator<char>());

    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ImportCertificateRequest request;

    request.WithCertificate(Aws::Utils::ByteBuffer((unsigned char *)
                                                           certificate.c_str(),
                                                   certificate.size()))
            .WithPrivateKey(Aws::Utils::ByteBuffer((unsigned char *)
                                                           privateKey.c_str(),
                                                   privateKey.size()))
            .WithCertificateChain(Aws::Utils::ByteBuffer((unsigned char *)

certificateChain.c_str(),
                                                         certificateChain.size()));

    Aws::ACM::Model::ImportCertificateOutcome outcome =
            acmClient.ImportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ImportCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Certificate associated with ARN '" <<
                  outcome.GetResult().GetCertificateArn() << "' imported."
```

```
                  << std::endl;

        return true;
    }
}
```

- API-Details finden Sie ImportCertificate in der AWS SDK für C++ API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the AWS Certificate Manager (ACM) certificates in an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ListCertificatesRequest request;
    Aws::Vector<Aws::ACM::Model::CertificateSummary> allCertificates;
    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::ACM::Model::ListCertificatesOutcome outcome =
                acmClient.ListCertificates(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListCertificates: " <<
                      outcome.GetError().GetMessage() << std::endl;

            return false;
        }
        else {
            const Aws::ACM::Model::ListCertificatesResult &result =
 outcome.GetResult();

            const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates =
                    result.GetCertificateSummaryList();
            allCertificates.insert(allCertificates.end(), certificates.begin(),
                                   certificates.end());

            nextToken = result.GetNextToken();
        }
    } while (!nextToken.empty());

    if (!allCertificates.empty()) {
        for (const Aws::ACM::Model::CertificateSummary &certificate:
 allCertificates) {
            std::cout << "Certificate ARN: " <<
                      certificate.GetCertificateArn() << std::endl;
            std::cout << "Domain name:     " <<
                      certificate.GetDomainName() << std::endl << std::endl;
        }
    }
    else {
        std::cout << "No available certificates found in account."
                  << std::endl;
    }

    return true;
}
```

- API-Details finden Sie ListCertificates in der AWS SDK für C++ API-Referenz.

`ListTagsForCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the tags for an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listTagsForCertificate(const Aws::String &certificateArn,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ListTagsForCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::ListTagsForCertificateOutcome outcome =
            acm_client.ListTagsForCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cout << "Error: ListTagsForCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Information about tags for "
                     "certificate with ARN '"
                  << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        Aws::Vector<Aws::ACM::Model::Tag> tags =
                result.GetTags();

        if (tags.size() > 0) {
```

```
            for (const Aws::ACM::Model::Tag &tag: tags) {
                std::cout << "Key:   " << tag.GetKey() << std::endl;
                std::cout << "Value: " << tag.GetValue()
                          << std::endl << std::endl;
            }
        }
        else {
            std::cout << "No tags found." << std::endl;
        }

        return true;
    }
}
```

- API-Details finden Sie ListTagsForCertificate in der AWS SDK für C++ API-Referenz.

`RemoveTagsFromCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTagsFromCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Remove a tag from an ACM certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param tagKey: The key for the tag.
  \param tagValue: The value for the tag.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::removeTagsFromCertificate(const Aws::String &certificateArn,
                                            const Aws::String &tagKey,
                                            const Aws::String &tagValue,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::Vector<Aws::ACM::Model::Tag> tags;

    Aws::ACM::Model::Tag tag;
    tag.SetKey(tagKey);

    tags.push_back(tag);

    Aws::ACM::Model::RemoveTagsFromCertificateRequest request;
    request.WithCertificateArn(certificateArn)
            .WithTags(tags);

    Aws::ACM::Model::RemoveTagsFromCertificateOutcome outcome =
            acmClient.RemoveTagsFromCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RemoveTagFromCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey << "' removed from "
                  << "certificate with ARN '" << certificateArn << "'." <<
 std::endl;

        return true;
    }
}
```

- API-Details finden Sie RemoveTagsFromCertificate in der AWS SDK für C++ API-Referenz.

`RenewCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RenewCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Renew an AWS Certificate Manager (ACM) certificate.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::renewCertificate(const Aws::String &certificateArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RenewCertificateRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::RenewCertificateOutcome outcome =
            acmClient.RenewCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RenewCertificate: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Renewed certificate with ARN '"
                  << certificateArn << "'." << std::endl;

        return true;
    }
}
```

- API-Details finden Sie RenewCertificate in der AWS SDK für C++ API-Referenz.

`RequestCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RequestCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Request an AWS Certificate Manager (ACM) certificate.
/*!
  \param domainName: A fully qualified domain name.
  \param idempotencyToken: Customer chosen string for idempotency.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::requestCertificate(const Aws::String &domainName,
                                     const Aws::String &idempotencyToken,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RequestCertificateRequest request;
    request.WithDomainName(domainName)
            .WithIdempotencyToken(idempotencyToken);

    Aws::ACM::Model::RequestCertificateOutcome outcome =
            acmClient.RequestCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "RequestCertificate error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The newly requested certificate's "
                     "ARN is '" <<
                  outcome.GetResult().GetCertificateArn() <<
                  "'." << std::endl;
```

```
        return true;
    }
}
```

- API-Details finden Sie RequestCertificate in der AWS SDK für C++ API-Referenz.

`ResendValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung`ResendValidationEmail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Resend the email that requests domain ownership validation.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param domainName: A fully qualified domain name.
  \param validationDomain: The base validation domain that will act as the suffix
                           of the email addresses.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::resendValidationEmail(const Aws::String &certificateArn,
                                        const Aws::String &domainName,
                                        const Aws::String &validationDomain,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ResendValidationEmailRequest request;
    request.WithCertificateArn(certificateArn)
            .WithDomain(domainName)
            .WithValidationDomain(validationDomain);

    Aws::ACM::Model::ResendValidationEmailOutcome outcome =
```

```
            acmClient.ResendValidationEmail(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "ResendValidationEmail error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The validation email has been resent."
                  << std::endl;

        return true;
    }
}
```

- API-Details finden Sie ResendValidationEmail in der AWS SDK für C++ API-Referenz.

`UpdateCertificateOptions`

Das folgende Codebeispiel zeigt die Verwendung`UpdateCertificateOptions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an AWS Certificate Manager (ACM) certificate option.
/*!
  \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
  \param loggingEnabled: Boolean specifying logging enabled.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::ACM::updateCertificateOption(const Aws::String &certificateArn,
                                          bool loggingEnabled,
                                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::UpdateCertificateOptionsRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::CertificateOptions options;

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(
                Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED);
    }
    else {
        options.SetCertificateTransparencyLoggingPreference(

 Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED);
    }

    request.SetOptions(options);

    Aws::ACM::Model::UpdateCertificateOptionsOutcome outcome =
            acmClient.UpdateCertificateOptions(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "UpdateCertificateOption error: " <<
                  outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The option '"
                  << (loggingEnabled ? "enabled" : "disabled") << "' has been set
 for "
                                                                  "the certificate
 with the ARN '"
                  << certificateArn << "'."
                  << std::endl;

        return true;
    }
}
```

- API-Details finden Sie UpdateCertificateOptions in der AWS SDK für C++ API-Referenz.

API-Gateway-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von API Gateway Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Aurora-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_aurora.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
```

```
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für C++ -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für C++

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
            dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " << DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family << std::endl;
            }
        }
```

```
        int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX,
                                NO_SOURCE,
                                autoIncrementParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
```

```
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value between ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                          << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
```

```
                    << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a source
of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
                                client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
```

```
        std::cout << "Let's create a DB cluster." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters): ");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
{
            std::cout << "  " << index << ": " << engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                                1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " << engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
```

```
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }
```

```
        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
```

```
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
            return false;
```

```
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql' shell
 to the database.
    displayConnection(dbCluster);

    printAsterisksLine();

    if (askYesNoQuestion(
            "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
        Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                               Aws::String(Aws::Utils::UUID::RandomUUID()));
        {
            std::cout << "Creating a snapshot named " << snapshotID << "." <<
 std::endl;
            std::cout << "This typically takes a few minutes." << std::endl;

            // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
        }

        std::cout << "Waiting for the snapshot to become available." << std::endl;

        Aws::RDS::Model::DBClusterSnapshot snapshot;
        counter = 0;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

```
            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
            "Do you want to delete the DB cluster, DB instance, and parameter group
 (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
```

```
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,
                                            Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
```

```
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
```

```
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
```

```
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
```

```
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
```

```
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
```

```
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " << counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." << std::endl;
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetEngine(engineName);
    request.SetEngineVersion(engineVersionName);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBClusterOutcome outcome =
            client.CreateDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBCluster. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK für C++ -API-Referenz.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example cluster parameter group.");

    Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
            client.CreateDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK für C++ API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
```

```
            return false;
        }
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK für C++ API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetEngine(engineName);
    request.SetDBInstanceClass(dbInstanceClass);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
            client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
```

```
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für C++ -API-Referenz.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
```

```
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK für C++ -API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK für C++ API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für C++ -API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
```

```
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
            dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK für C++ API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
```

```
/*!
 \sa getDBClusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,
                                            Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
```

```
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK für C++ API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
```

```
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS SDK für C++ API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
```

```
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für C++ -API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();
```

```
            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für C++ API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
```

```
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für C++ API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
            client.ModifyDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully modified."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
```

```
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_autoscaling.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2 Auto
 Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());
```

```
        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." << std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

```
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
```

```
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following Availability
Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
            "Do you want to collect metrics about the A"
            "Auto Scaling group during this demo (y/n)?  ");
```

```
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
             "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
```

```
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName << "."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will start
"
              << "to replace the terminated EC2 instance."
              << std::endl;
    std::cout << "The currently running EC2 instances are:" << std::endl;

    if (autoScalingGroups.empty()) {
        std::cerr << "Error describing groups. No groups returned." << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    int instanceNumber = 1;
    Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
```

```
    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to stop?
",
                                            1,
                                            static_cast<int>(instanceIDs.size()));

    // 10. Terminate an EC2 instance in the Auto Scaling group.
    {
        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group '"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
```

```
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
```

```
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const Aws::AutoScaling::AutoScalingClient
 &client) {
    bool ready = false;
```

```
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." << std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
 &instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails &details:
 instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
```

```
                                                const Aws::AutoScaling::AutoScalingClient
&autoScalingClient,
                                                const Aws::EC2::EC2Client &ec2Client) {
   bool result = true;

   // 14. Delete the Auto Scaling group.
   if (!groupName.empty() &&
       (askYesNoQuestion(
               Aws::String("Delete the Auto Scaling group '") + groupName +
               "'  (y/n)?"))) {
       {
           Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
           request.SetAutoScalingGroupName(groupName);
           request.SetMinSize(0);
           request.SetDesiredCapacity(0);

           Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                   autoScalingClient.UpdateAutoScalingGroup(request);

           if (outcome.IsSuccess()) {
               std::cout
                       << "The minimum size and desired capacity of the Auto
Scaling group "
                       << "was set to zero before terminating the instances."
                       << std::endl;
           }
           else {
               std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                         << outcome.GetError().GetMessage() << std::endl;
               result = false;
           }
       }

       Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
       if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                              autoScalingClient)) {
           if (!autoScalingGroups.empty()) {
               Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                       autoScalingGroups[0].GetInstances());
               for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                   request.SetInstanceId(instanceID);
                   request.SetShouldDecrementDesiredCapacity(true);
```

```
Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
                            autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                    request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before deleting
the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

```
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                         const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
```

```
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);

        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;

    }
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK für C++ API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                autoScalingClient.DeleteAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling group '" << groupName << "' was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
            }
        }
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK für C++ API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für C++ API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK für C++ API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
```

```
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK für C++ API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
```

```
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK für C++ API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK für C++ API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
        }
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK für C++ API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
    }
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK für C++ API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetMaxSize(3);

    Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
            autoScalingClient.UpdateAutoScalingGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK für C++ API-Referenz.

CloudTrail Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren CloudTrail.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateTrail`

Das folgende Codebeispiel zeigt die Verwendung`CreateTrail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which creates an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param bucketName: The Amazon S3 bucket designate for publishing logs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
```

```
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateTrail in der AWS SDK für C++ API-Referenz.

`DeleteTrail`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTrail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which deletes an AWS CloudTrail trail.
/*!
  \param trailName: The name of the CloudTrail trail.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
```

```
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteTrail in der AWS SDK für C++ API-Referenz.

`DescribeTrail`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTrail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which describes the AWS CloudTrail trails in an account.
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/

bool AwsDoc::CloudTrail::describeTrails(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;
```

```
    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
 outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." << outcome.GetError().GetMessage()
                  << std::endl;
    }
    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeTrail in der AWS SDK für C++ API-Referenz.

`LookupEvents`

Das folgende Codebeispiel zeigt die Verwendung`LookupEvents`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Routine which looks up events captured by AWS CloudTrail.
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::lookupEvents(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);
```

```
    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;

    size_t count = 0;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
 cloudtrail.LookupEvents(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
 outcome.GetResult().GetEvents();
            count += events.size();
            allEvents.insert(allEvents.end(), events.begin(), events.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty() && count <= 50); // Limit to 50 events.

    std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

    for (auto &event: allEvents) {
        std::cout << "Event name: " << event.GetEventName() << std::endl;
        std::cout << "Event source: " << event.GetEventSource() << std::endl;
        std::cout << "Event id: " << event.GetEventId() << std::endl;
        std::cout << "Resources: " << std::endl;
        for (auto &resource: event.GetResources()) {
            std::cout << "  " << resource.GetResourceName() << std::endl;
        }
    }

    return true;
}
```

- API-Details finden Sie LookupEvents in der AWS SDK für C++ API-Referenz.

CloudWatch Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Löschen Sie den Alarm.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- API-Details finden Sie DeleteAlarms in der AWS SDK für C++ API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Beschreiben Sie die Alarme.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
```

```
            done = next_token.empty();
        }
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK für C++ API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Deaktivieren Sie die Alarmaktionen.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);

        auto disableAlarmActionsOutcome =
 cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
```

```
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK für C++ API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Aktivieren Sie die Alarmaktionen.

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
    request.SetEvaluationPeriods(1);
```

```
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);

    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- API-Details finden Sie EnableAlarmActions in der AWS SDK für C++ API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

Listen Sie die Metriken auf.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
```

```
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API-Details finden Sie ListMetrics in der AWS SDK für C++ API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Erstellen Sie den Alarm, um die Metrik zu beobachten.

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
        if (!outcome.IsSuccess())
        {
```

```
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK für C++ API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Fügen Sie Daten in eine Metrik ein.

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");
```

```
        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- API-Details finden Sie PutMetricData in der AWS SDK für C++ API-Referenz.

CloudWatch Protokolliert Beispiele mithilfe des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Löschen Sie den Abonnementfilter.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetLogGroupName(log_group);

        auto outcome = cwl.DeleteSubscriptionFilter(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        } else {
            std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API-Details finden Sie DeleteSubscriptionFilter in der AWS SDK für C++ API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

Listen Sie die Abonnementfilter auf.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
        request.SetLogGroupName(log_group);
        request.SetLimit(1);

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = cwl.DescribeSubscriptionFilters(
                    request);
            if (!outcome.IsSuccess()) {
                std::cout << "Failed to describe CloudWatch subscription filters "
                    << "for log group " << log_group << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
```

```
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(32) << "Name" <<
                    std::setw(64) << "FilterPattern" << std::setw(64) <<
                    "DestinationArn" << std::endl;
                header = true;
            }

            const auto &filters = outcome.GetResult().GetSubscriptionFilters();
            for (const auto &filter : filters) {
                std::cout << std::left << std::setw(32) <<
                    filter.GetFilterName() << std::setw(64) <<
                    filter.GetFilterPattern() << std::setw(64) <<
                    filter.GetDestinationArn() << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- API-Details finden Sie DescribeSubscriptionFilters in der AWS SDK für C++ API-Referenz.

`PutSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`PutSubscriptionFilter`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
```

```
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie den Abonnementfilter.

```
        Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
        Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
        request.SetFilterName(filter_name);
        request.SetFilterPattern(filter_pattern);
        request.SetLogGroupName(log_group);
        request.SetDestinationArn(dest_arn);
        auto outcome = cwl.PutSubscriptionFilter(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch logs subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
        }
```

- API-Details finden Sie PutSubscriptionFilter in der AWS SDK für C++ API-Referenz.

CodeBuild Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren CodeBuild.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListBuilds`

Das folgende Codebeispiel zeigt die Verwendung`ListBuilds`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the CodeBuild builds.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
 codeBuildClient.ListBuilds(
                listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
```

```
            const Aws::Vector<Aws::String> &ids =
listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {

                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
                Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
                        getBuildsRequest);

                if (getBuildsOutcome.IsSuccess()) {
                    const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
                    std::cout << builds.size() << " build(s) found." << std::endl;
                    for (auto val: builds) {
                        std::cout << val.GetId() << std::endl;
                    }
                } else {
                    std::cerr << "Error getting builds"
                              << getBuildsOutcome.GetError().GetMessage() <<
std::endl;
                    return false;
                }
            } else {
                std::cout << "No builds found." << std::endl;
            }

            // Get the next token for pagination.

            nextToken = listBuildsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing builds"
                      << listBuildsOutcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!nextToken.

            empty()

            );
```

```
    return true;
}
```

- API-Details finden Sie ListBuilds in der AWS SDK für C++ API-Referenz.

`ListProjects`

Das folgende Codebeispiel zeigt die Verwendung`ListProjects`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
 codeBuildClient.ListProjects(
```

```
                listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
 outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(), projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }

        else {
            std::cerr << "Error listing projects" << outcome.GetError().GetMessage()
                      << std::endl;
        }

    } while (!nextToken.empty());

    std::cout << allProjects.size() << " project(s) found." << std::endl;
    for (auto project: allProjects) {
        std::cout << project << std::endl;
    }

    return true;
}
```

- API-Details finden Sie ListProjects in der AWS SDK für C++ API-Referenz.

StartBuild

Das folgende Codebeispiel zeigt die Verwendung`StartBuild`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Start an AWS CodeBuild project build.
/*!
```

```
  \param projectName: A CodeBuild project name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome = codeBuildClient.StartBuild(
            startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
                  << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie StartBuild in der AWS SDK für C++ API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_cognito.cpp.

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito client
 and lists the Amazon Cognito
```

```
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
 listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
```

```
            } else {
                std::cerr << "ListUserPools error: " <<
 listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListUserPools in der AWS SDK für C++ API-Referenz.

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);
```

```
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API-Details finden Sie AdminGetUser in der AWS SDK für C++ API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK für C++ API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK für C++ API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for example
 Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            return false;
        }
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK für C++ API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
```

```
                    << outcome.GetError().GetMessage()
                    << std::endl;
            return false;
        }
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK für C++ API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API-Details finden Sie DeleteUser in der AWS SDK für C++ API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK für C++ API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
```

```
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API-Details finden Sie SignUp in der AWS SDK für C++ API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
```

```
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK für C++ API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
```

```
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);
    bool userExists = false;
    do {
        // 1. Add a user with a username, password, and email address.
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was successful."
                      << std::endl;
        }
```

```
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (userExists);

    printAsterisksLine();
    std::cout << "Retrieving status of " << userName << " in the user pool."
              << std::endl;
    // 2. Confirm that the user was added to the user pool.
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
```

```
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();
```

```
    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
 session, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-factor
 authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for example
 Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
```

```
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
(VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
```

```
   if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
       return false;
   }

   Aws::String accessToken;
   {
       Aws::String mfaCode = askQuestion(
               "Re-enter the 6 digit code displayed in the authenticator app: ");

       // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

```
        std::cout << "You have successfully added a user to Amazon Cognito."
                  << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added? (y/
n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);
```

```
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

DynamoDB-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS dynamodb)

# Set this project's name.
project("hello_dynamodb")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_dynamodb.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
 *  A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 (DynamoDB) client and lists the
 *  DynamoDB tables.
 *
 *  main function
 *
 *  Usage: 'hello_dynamodb'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do {
            const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamodbClient.ListTables(
                    listTablesRequest);
            if (!outcome.IsSuccess()) {
                std::cout << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
                result = 1;
                break;
            }

            for (const auto &tableName: outcome.GetResult().GetTableNames()) {
```

```
                std::cout << tableName << std::endl;
            }

            listTablesRequest.SetExclusiveStartTableName(
                    outcome.GetResult().GetLastEvaluatedTableName());

        } while (!listTablesRequest.GetExclusiveStartTableName().empty());
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListTables in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    {
```

```
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table with partition: year (N) and sort: title (S).
 (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

            // 9. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }
    }

//! Scenario to modify and query a DynamoDB table.
/*!
  \sa dynamodbGettingStartedScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
 std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
 ",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
```

```
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
                putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    // 3. Update the rating and plot of the movie by using an update expression.
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);
        plot = askQuestion(Aws::String("You summarized the plot as '") + plot +
```

```
                            "'.\nWhat would you say now? ");

        Aws::DynamoDB::Model::UpdateItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
Aws::DynamoDB::Model::AttributeValue().SetS(title));
        request.AddKey(YEAR_KEY, Aws::DynamoDB::Model::AttributeValue().SetN(year));
        std::stringstream expressionStream;
        expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
                         << INFO_KEY << "." << PLOT_KEY << " =:p";
        request.SetUpdateExpression(expressionStream.str());
        request.SetExpressionAttributeValues({
                                                    {":r",
Aws::DynamoDB::Model::AttributeValue().SetN(
                                                            rating)},
                                                    {":p",
Aws::DynamoDB::Model::AttributeValue().SetS(
                                                            plot)}
                                                });

        request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

        const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
                request);
        if (!result.IsSuccess()) {
            std::cerr << "Error updating movie " + result.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    // 4. Put 250 movies in the table from moviedata.json.
    {
        std::cout << "Adding movies from a json file to the database." << std::endl;
        const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
        const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
        Aws::String jsonString = getMovieJSON();
        if (!jsonString.empty()) {
            Aws::Utils::Json::JsonValue json(jsonString);
            Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
json.View().AsArray();
```

```
            Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

            // To add movies with a cross-section of years, use an appropriate
increment
            // value for iterating through the database.
            size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
            for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
                writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> putItems
= movieJsonViewToAttributeMap(
                        movieJsons[i]);
                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(putItems);
                writeRequests.back().SetPutRequest(putRequest);
                if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                    Aws::DynamoDB::Model::BatchWriteItemRequest request;
                    request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
                    const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                            request);
                    if (!outcome.IsSuccess()) {
                        std::cerr << "Unable to batch write movie data: "
                                  << outcome.GetError().GetMessage()
                                  << std::endl;
                        writeRequests.clear();
                        break;
                    }
                    else {
                        std::cout << "Added batch of " << writeRequests.size()
                                  << " movies to the database."
                                  << std::endl;
                    }
                    writeRequests.clear();
                }
            }
        }
    }

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;

    // 5. Get a movie by Key (partition + sort).
    {
        Aws::String titleToGet("King Kong");
```

```
        Aws::String answer = askQuestion(Aws::String(
                "Let's move on...Would you like to get info about '" + titleToGet +
                "'? (y/n) "));
        if (answer == "y") {
            Aws::DynamoDB::Model::GetItemRequest request;
            request.SetTableName(MOVIE_TABLE_NAME);
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(1933));

            const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
                    request);
            if (!result.IsSuccess()) {
                std::cerr << "Error " << result.GetError().GetMessage();
            }
            else {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
                if (!item.empty()) {
                    std::cout << "\nHere's what I found:" << std::endl;
                    printMovieInfo(item);
                }
                else {
                    std::cout << "\nThe movie was not found in the database."
                              << std::endl;
                }
            }
        }
    }

    // 6. Use Query with a key condition expression to return all movies
    //    released in a given year.
    Aws::String doAgain = "n";
    do {
        Aws::DynamoDB::Model::QueryRequest req;

        req.SetTableName(MOVIE_TABLE_NAME);

        // "year" is a DynamoDB reserved keyword and must be replaced with an
        // expression attribute name.
        req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
        req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});
```

```
        int yearToMatch = askQuestionForIntRange(
                "\nLet's get a list of movies released in"
                " a given year. Enter a year between 1972 and 2018 ",
                1972, 2018);
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
        attributeValues.emplace(":valueToMatch",
                                Aws::DynamoDB::Model::AttributeValue().SetN(
                                        yearToMatch));
        req.SetExpressionAttributeValues(attributeValues);

        const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
        if (result.IsSuccess()) {
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "\nThere were " << items.size()
                          << " movies in the database from "
                          << yearToMatch << "." << std::endl;
                for (const auto &item: items) {
                    printMovieInfo(item);
                }
                doAgain = "n";
            }
            else {
                std::cout << "\nNo movies from " << yearToMatch
                          << " were found in the database"
                          << std::endl;
                doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
            }
        }
        else {
            std::cerr << "Failed to Query items: " << result.GetError().GetMessage()
                      << std::endl;
        }

    } while (doAgain == "y");

    //  7. Use Scan to return movies released within a range of years.
    //     Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
    {
        int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
```

```
                                                "for movies in the database. Enter a
start year: ",
                                                1972, 2018);
       int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                            startYear, 2018);
       Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
       do {
           Aws::DynamoDB::Model::ScanRequest scanRequest;
           scanRequest.SetTableName(MOVIE_TABLE_NAME);
           scanRequest.SetFilterExpression(
                   "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
           scanRequest.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

           Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
           attributeValues.emplace(":startYear",
                                   Aws::DynamoDB::Model::AttributeValue().SetN(
                                           startYear));
           attributeValues.emplace(":endYear",
                                   Aws::DynamoDB::Model::AttributeValue().SetN(
                                           endYear));
           scanRequest.SetExpressionAttributeValues(attributeValues);

           if (!exclusiveStartKey.empty()) {
               scanRequest.SetExclusiveStartKey(exclusiveStartKey);
           }

           const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
                   scanRequest);
           if (result.IsSuccess()) {
               const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
               if (!items.empty()) {
                   std::stringstream stringStream;
                   stringStream << "\nFound " << items.size() << " movies in one
scan."
                                << " How many would you like to see? ";
                   size_t count = askQuestionForInt(stringStream.str());
                   for (size_t i = 0; i < count && i < items.size(); ++i) {
                       printMovieInfo(items[i]);
                   }
               }
               else {
```

```
                    std::cout << "\nNo movies in the database between " << startYear
<<
                              " and " << endYear << "." << std::endl;
                }

                exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
                if (!exclusiveStartKey.empty()) {
                    std::cout << "Not all movies were retrieved. Scanning for more."
                              << std::endl;
                }
                else {
                    std::cout << "All movies were retrieved with this scan."
                              << std::endl;
                }
            }
            else {
                std::cerr << "Failed to Scan movies: "
                          << result.GetError().GetMessage() << std::endl;
            }
        } while (!exclusiveStartKey.empty());
    }

    // 8. Delete a movie. (DeleteItem)
    {
        std::stringstream stringStream;
        stringStream << "\nWould you like to delete the movie " << title
                     << " from the database? (y/n) ";
        Aws::String answer = askQuestion(stringStream.str());
        if (answer == "y") {
            Aws::DynamoDB::Model::DeleteItemRequest request;
            request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
            request.AddKey(TITLE_KEY,
                           Aws::DynamoDB::Model::AttributeValue().SetS(title));
            request.SetTableName(MOVIE_TABLE_NAME);

            const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(
                    request);
            if (result.IsSuccess()) {
                std::cout << "\nRemoved \"" << title << "\" from the database."
                          << std::endl;
            }
            else {
```

```
                std::cerr << "Failed to delete the movie: "
                          << result.GetError().GetMessage()
                          << std::endl;
            }
        }
    }

    return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
  \sa movieJsonViewToAttributeMap()
  \param jsonView: Json view object.
  \return map: Map that can be used in a DynamoDB request.
 */
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
        const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
 std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
 std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
 std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }
```

```
        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
```

```
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
 dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
```

```
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
```

```
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
```

```
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
```

```
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
```

```
                << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
    // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";
```

```
        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für C++ API-Referenz.

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Batch get items from different Amazon DynamoDB tables.
/*!
  \sa batchGetItem()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::batchGetItem(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages, #v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");
```

```
    // Table2: Set key name, type, and value to search.
    std::vector<Aws::String> idValues = {"102", "103", "201"};
    for (const Aws::String &id: idValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetN(id);
        keys.emplace("Id", key);
        table2KeysAndAttributes.AddKeys(keys);
    }

    requestItems.emplace(table2Name, table2KeysAndAttributes);

    bool result = true;
    do {  // Use a do loop to handle pagination.
        request.SetRequestItems(requestItems);
        const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
                request);

        if (outcome.IsSuccess()) {
            for (const auto &responsesMapEntry: outcome.GetResult().GetResponses())
{
                Aws::String tableName = responsesMapEntry.first;
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
                std::cout << "Retrieved " << tableResults.size()
                          << " responses for table '" << tableName << "'.\n"
                          << std::endl;
                if (tableName == "Forum") {

                    std::cout << "Name | Category | Message | Views" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Name").GetS() << " | ";
                        std::cout << item.at("Category").GetS() << " | ";
                        std::cout << (item.count("Message") == 0 ? "" : item.at(
                                "Messages").GetN()) << " | ";
                        std::cout << (item.count("Views") == 0 ? "" : item.at(
                                "Views").GetN()) << std::endl;
                    }
                }
                else {
                    std::cout << "Title | Price | Color" << std::endl;
```

```
                    for (const Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Title").GetS() << " | ";
                        std::cout << (item.count("Price") == 0 ? "" : item.at(
                                "Price").GetN());
                        if (item.count("Color")) {
                            std::cout << " | ";
                            for (const
 std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
                                    "Color").GetL())
                                std::cout << listItem->GetS() << " ";
                        }
                        std::cout << std::endl;
                    }
                }
                std::cout << std::endl;
            }

            // If necessary, repeat request for remaining items.
            requestItems = outcome.GetResult().GetUnprocessedKeys();
        }
        else {
            std::cerr << "Batch get item failed: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;
        }
    } while (!requestItems.empty());

    return result;
}
```

- API-Details finden Sie BatchGetItem in der AWS SDK für C++ API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Batch write items from a JSON file.
/*!
  \sa batchWriteItem()
  \param jsonFilePath: JSON file path.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

/*
 * The input for this routine is a JSON file that you can download from the
 following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 generate
 * JSON strings when constructing the BatchWriteItem request, essentially outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
```

```
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
        //  key - table name
        //  value - list of request objects
        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                      << tableName << "' is not a list." << std::endl;
            continue;
        }

        Aws::Utils::Array<Aws::Utils::Json::JsonView> entries =
entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
                                               writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
dynamoClient.BatchWriteItem(
            batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return outcome.IsSuccess();
```

```
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
  \sa addWriteRequests()
  \param tableName: Name of the table for the write operations.
  \param requestsJson: Request data in JSON format.
  \param writeRequests: Vector to receive the WriteRequest objects.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                        const
 Aws::Utils::Array<Aws::Utils::Json::JsonView> &requestsJson,

 Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonView &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                      << requestsEntry.WriteReadable() << std::endl;
            return false;
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonView> requestsMap =
 requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonView &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                                            attributes)) {
                    std::cerr << "Error getting attributes "
                              << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
```

```
                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(attributes);
                writeRequests.push_back(
                        Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                                putRequest));
            }
            else {
                std::cerr << "Error: unimplemented request type '" << requestType
                          << "'." << std::endl;
            }
        }
    }

    return true;
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
  \sa getAttributeObjectsMap()
  \param jsonView: JSONView of attribute records.
  \param writeRequests: Map to receive the AttributeValue objects.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView &jsonView,
                                         Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
 jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                      << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,
                           Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }
```

```
    return true;
}
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für C++ API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon DynamoDB table.
/*!
  \sa createTable()
  \param tableName: Name for the DynamoDB table.
  \param primaryKey: Primary key for the DynamoDB table.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
              " with a simple primary key: \"" << primaryKey << "\"." << std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);
```

```
    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
 dynamoClient.CreateTable(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
                  << outcome.GetResult().GetTableDescription().GetTableName() <<
                  " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
```

```
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie CreateTable in der AWS SDK für C++ API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an item from an Amazon DynamoDB table.
/*!
  \sa deleteItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
 dynamoClient.DeleteItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
```

```
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie DeleteItem in der AWS SDK für C++ API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon DynamoDB table.
/*!
  \sa deleteTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

- API-Details finden Sie DeleteTable in der AWS SDK für C++ API-Referenz.

DescribeTable

Das folgende Codebeispiel zeigt die VerwendungDescribeTable.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an Amazon DynamoDB table.
/*!
  \sa describeTable()
  \param tableName: The DynamoDB table name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
 dynamoClient.DescribeTable(
            request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
 outcome.GetResult().GetTable();
        std::cout << "Table name  : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN   : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
```

```
                  << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                          td.GetTableStatus()) << std::endl;
        std::cout << "Item count  : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
 td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
 std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
 std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
 td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

 Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
                              a.GetAttributeType()) <<
                      ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
 outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeTable in der AWS SDK für C++ API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
```

```
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
    //  3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
```

```
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());
```

```
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }
```

Verwenden Sie eine DELETE-Anweisung, um ein Element zu löschen.

```
    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
```

```
        }
    }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für C++ API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get an item from an Amazon DynamoDB table.
/*!
  \sa getItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
```

```
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
 dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
 outcome.GetResult().GetItem();
        if (!item.empty()) {
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << "Values: " << i.first << ": " << i.second.GetS()
                          << std::endl;
        }
        else {
            std::cout << "No item found with the key " << partitionKey << std::endl;
        }
    }
    else {
        std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetItem in der AWS SDK für C++ API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the Amazon DynamoDB tables for the current AWS account.
/*!
  \sa listTables()
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
 dynamoClient.ListTables(
                listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
                outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}
```

- API-Details finden Sie ListTables in der AWS SDK für C++ API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Put an item in an Amazon DynamoDB table.
/*!
  \sa putItem()
  \param tableName: The table name.
  \param artistKey: The artist key. This is the partition key for the table.
  \param artistValue: The artist value.
  \param albumTitleKey: The album title key.
  \param albumTitleValue: The album title value.
  \param awardsKey: The awards key.
  \param awardsValue: The awards value.
  \param songTitleKey: The song title key.
  \param songTitleValue: The song title value.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                               const Aws::String &artistKey,
                               const Aws::String &artistValue,
                               const Aws::String &albumTitleKey,
                               const Aws::String &albumTitleValue,
                               const Aws::String &awardsKey,
                               const Aws::String &awardsValue,
                               const Aws::String &songTitleKey,
                               const Aws::String &songTitleValue,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
            artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
 Aws::DynamoDB::Model::AttributeValue().SetS(
            albumTitleValue));
    putItemRequest.AddItem(awardsKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

 Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));
```

```
    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
```

```
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie PutItem in der AWS SDK für C++ API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Perform a query on an Amazon DynamoDB Table and retrieve items.
/*!
  \sa queryItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param projectionExpression: The projections expression, which is ignored if
 empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */
```

```
/*
 * The partition key attribute is searched with the specified value. By default, all
 fields and values
 * contained in the item are returned. If an optional projection expression is
 * specified on the command line, only the specified fields and values are
 * returned.
 */

bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &projectionExpression,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);

    if (!projectionExpression.empty()) {
        request.SetProjectionExpression(projectionExpression);
    }

    // Set query key condition expression.
    request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

    // Set Expression AttributeValues.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch", partitionValue);

    request.SetExpressionAttributeValues(attributeValues);

    bool result = true;

    // "exclusiveStartKey" is used for pagination.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> exclusiveStartKey;
    do {
        if (!exclusiveStartKey.empty()) {
            request.SetExclusiveStartKey(exclusiveStartKey);
            exclusiveStartKey.clear();
        }
        // Perform Query operation.
```

```
        const Aws::DynamoDB::Model::QueryOutcome &outcome =
 dynamoClient.Query(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            if (!items.empty()) {
                std::cout << "Number of items retrieved from Query: " <<
 items.size()
                          << std::endl;
                // Iterate each item and print.
                for (const auto &item: items) {
                    std::cout
                            <<
 "*****************************************************"
                            << std::endl;
                    // Output each retrieved field and its value.
                    for (const auto &i: item)
                        std::cout << i.first << ": " << i.second.GetS() <<
 std::endl;
                }
            }
            else {
                std::cout << "No item found in table: " << tableName << std::endl;
            }

            exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Query items: " <<
 outcome.GetError().GetMessage();
            result = false;
            break;
        }
    } while (!exclusiveStartKey.empty());

    return result;
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für C++ -API-Referenz.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Scan an Amazon DynamoDB table.
/*!
  \sa scanTable()
  \param tableName: Name for the DynamoDB table.
  \param projectionExpression: An optional projection expression, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                 const Aws::String &projectionExpression,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
 all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
 dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
```

```
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
 Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            all_items.insert(all_items.end(), items.begin(), items.end());

            last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
        }
        else {
            std::cerr << "Failed to Scan items: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!last_evaluated_key.empty());

    if (!all_items.empty()) {
        std::cout << "Number of items retrieved from scan: " << all_items.size()
                  << std::endl;
        // Iterate each item and print.
        for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 &itemMap: all_items) {
            std::cout << "*****************************************************"
                      << std::endl;
            // Output each retrieved field and its value.
            for (const auto &itemEntry: itemMap)
                std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                          << std::endl;
        }
    }

    else {
        std::cout << "No items found in table: " << tableName << std::endl;
    }

    return true;
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für C++ -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an Amazon DynamoDB table item.
/*!
  \sa updateItem()
  \param tableName: The table name.
  \param partitionKey: The partition key.
  \param partitionValue: The value for the partition key.
  \param attributeKey: The key for the attribute to be updated.
  \param attributeValue: The value for the attribute to be updated.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */

/*
 *  The example code only sets/updates an attribute value. It processes
 *  the attribute value as a string, even if the value could be interpreted
 *  as a number. Also, the example code does not remove an existing attribute
 *  from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                  const Aws::String &partitionKey,
                                  const Aws::String &partitionValue,
                                  const Aws::String &attributeKey,
                                  const Aws::String &attributeValue,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
```

```
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
 expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);

    // Update the item.
    const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
 dynamoClient.UpdateItem(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Item was updated" << std::endl;
    } else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
```

```
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für C++ API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update a DynamoDB table.
/*!
  \sa updateTable()
  \param tableName: Name for the DynamoDB table.
  \param readCapacity: Provisioned read capacity.
  \param writeCapacity: Provisioned write capacity.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long writeCapacity,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput
 values"
              << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;

 provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
            writeCapacity);

 request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome =
 dynamoClient.UpdateTable(
```

```
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    } else {
        const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
        if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
            error.GetMessage().find("The provisioned throughput for the table will
 not change") != std::string::npos) {
            std::cout << "The provisioned throughput for the table will not change."
 << std::endl;
        } else {
            std::cerr << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return waitTableActive(tableName, dynamoClient);
}
```

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
```

```
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie UpdateTable in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
```

```
/*!
  \sa partiqlBatchExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
                "On a scale of 1 - 10, how do you rate it? ",
                1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
 "));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
              << (titles.size() == 1 ? " movie " : " movies ")
              << "to the table using a batch \"INSERT\" statement." << std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
```

```
                  << INFO_KEY << "': ?}";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

            // Create attribute for the info map.
            Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            ratingAttribute->SetN(ratings[i]);
            infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                    ALLOCATION_TAG.c_str());
            plotAttribute->SetS(plots[i]);
            infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
            attributes.push_back(infoMapAttribute);
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

```
    }

    std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
              << std::endl;

    // 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();
```

```
                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    // 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

    for (size_t i = 0; i < titles.size(); ++i) {
        ratings[i] = askQuestionForFloatRange(
                Aws::String("\nLet's update your the movie, \"") + titles[i] +
                ".\nYou rated it  " + std::to_string(ratings[i])
                + ", what new rating would you give it? ", 1, 10);
    }

    std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";


        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
```

```
attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
              << std::endl;

    // 5. Get the updated data for multiple movies using "Select" statements.
(BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
```

```
        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
                request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movies information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
              << std::endl;

    // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
```

```
            attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

 attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
 dynamoClient.BatchExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::N);
```

```
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }

    // Wait for table to become active.
```

```
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
```

```
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für C++ API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        //  1. Create a table. (CreateTable)
        if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

            AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

            // 7. Delete the table. (DeleteTable)
            AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
        }

//! Scenario to modify and query a DynamoDB table using single PartiQL statements.
/*!
  \sa partiqlExecuteScenario()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
```

```
    int year;
    Aws::String plot;
    {
        title = askQuestion(
                "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?
",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
              << std::endl;

    //  3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to retrieve movie information: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            // Print the retrieved movie information.
            const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

            if (items.size() == 1) {
                printMovieInfo(items[0]);
```

```
            }
            else {
                std::cerr << "Error: " << items.size() << " movies were retrieved. "
                          << " There should be only one movie." << std::endl;
            }
        }
    }

    //  4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
                Aws::String("\nLet's update your movie.\nYou rated it  ") +
                std::to_string(rating)
                + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
                  << INFO_KEY << "." << RATING_KEY << "=? WHERE "
                  << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
                request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                      << outcome.GetError().GetMessage();
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;
```

```
   //  5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
   {
       Aws::DynamoDB::Model::ExecuteStatementRequest request;
       std::stringstream sqlStream;
       sqlStream << "SELECT * FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                 << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

       request.SetStatement(sqlStream.str());

       Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
       attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
       request.SetParameters(attributes);

       Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
               request);
       if (!outcome.IsSuccess()) {
           std::cerr << "Failed to retrieve the movie information: "
                     << outcome.GetError().GetMessage() << std::endl;
           return false;
       }
       else {
           const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

           const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

           if (items.size() == 1) {
               printMovieInfo(items[0]);
           }
           else {
               std::cerr << "Error: " << items.size() << " movies were retrieved. "
                         << " There should be only one movie." << std::endl;
           }
       }
   }

   std::cout << "Deleting the movie" << std::endl;

   // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
   {
```

```
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM  \"" << MOVIE_TABLE_NAME << "\" WHERE "
                  << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
 dynamoClient.ExecuteStatement(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                      << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Movie successfully deleted." << std::endl;
    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
  \sa createMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
```

```
                Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
                Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
                Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
                PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
                request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorType() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "
                          << result.GetError().GetMessage();
                return false;
            }
        }
    }
```

```
    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
                  << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
 clientConfiguration)) {
            return false;
        }
        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
                  << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
  \sa deleteMoviesDynamoDBTable()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
 dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                  << result.GetResult().GetTableDescription().GetTableName()
                  << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

```
//! Query a newly created DynamoDB table until it is active.
/*!
  \sa waitTableActive()
  \param waitTableActive: The DynamoDB table's name.
  \param dynamoClient: A DynamoDB client.
  \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
 &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
 dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
 result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                      << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für C++ API-Referenz.

Amazon-EC2-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon EC2 beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
project("hello_ec2")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
```

```
#include <iomanip>
#include <iostream>

/*
 *  A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 Cloud (Amazon EC2) client and describes
 *  the Amazon EC2 instances.
 *
 *  main function
 *
 *  Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)argv;

    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                              std::setw(48) << "Name" <<
                              std::setw(20) << "ID" <<
                              std::setw(25) << "Ami" <<
                              std::setw(15) << "Type" <<
                              std::setw(15) << "State" <<
                              std::setw(15) << "Monitoring" << std::endl;
```

```
                    header = true;
                }

                const std::vector<Aws::EC2::Model::Reservation> &reservations =
                        outcome.GetResult().GetReservations();

                for (const auto &reservation: reservations) {
                    const std::vector<Aws::EC2::Model::Instance> &instances =
                            reservation.GetInstances();
                    for (const auto &instance: instances) {
                        Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                        instance.GetState().GetName());

                        Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                        instance.GetInstanceType());

                        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                        instance.GetMonitoring().GetState());
                        Aws::String name = "Unknown";

                        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                     [](const Aws::EC2::Model::Tag
&tag) {
                                                         return tag.GetKey() ==
"Name";
                                                     });
                        if (nameIter != tags.cend()) {
                            name = nameIter->GetValue();
                        }
                        std::cout <<
                                  std::setw(48) << name <<
                                  std::setw(20) << instance.GetInstanceId() <<
                                  std::setw(25) << instance.GetImageId() <<
                                  std::setw(15) << typeString <<
                                  std::setw(15) << instanceStateString <<
                                  std::setw(15) << monitorString << std::endl;
```

```
                    }
                }

                if (!outcome.GetResult().GetNextToken().empty()) {
                    request.SetNextToken(outcome.GetResult().GetNextToken());
                } else {
                    done = true;
                }
            } else {
                std::cerr << "Failed to describe EC2 instances:" <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für C++ API-Referenz.

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Allocate an Elastic IP address and associate it with an Amazon Elastic Compute
 Cloud
```

```
//! (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param[out] publicIPAddress: String to return the public IP address.
  \param[out] allocationID: String to return the allocation ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
 Aws::String &publicIPAddress,
                                               Aws::String &allocationID,
                                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
            ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response = outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();


    return true;
}
```

- API-Details finden Sie AllocateAddress in der AWS SDK für C++ API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

//! Associate an Elastic IP address with an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param allocationId: An Elastic IP allocation ID.
  \param[out] associationID: String to receive the association ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: True if the address was associated with the instance; otherwise,
 false.
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const Aws::String
 &allocationId,
                                    Aws::String &associationID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest request;
    request.SetInstanceId(instanceId);
    request.SetAllocationId(allocationId);

    Aws::EC2::Model::AssociateAddressOutcome outcome =
 ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
                  " with instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
                  " with instance " << instanceId << std::endl;
        associationID = outcome.GetResult().GetAssociationId();
    }
```

```
    return outcome.IsSuccess();
}
```

- API-Details finden Sie AssociateAddress in der AWS SDK für C++ API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
/*!
  \param groupID: The EC2 group ID.
  \param clientConfiguration: The ClientConfiguration object.
  \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
 authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
 authorizeSecurityGroupIngressOutcome =

 ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." << std::endl;
```

```
    } else {
        std::cerr << "Error authorizing security group ingress: "
                  << authorizeSecurityGroupIngressOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}
```

Hilfsfunktion zum Erstellen einer Eingangsregel.

```
//! Build a sample ingress rule.
/*!
  \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
  \return void:
 */
void buildSampleIngressRule(
        Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request) {
    Aws::String ingressIPRange = "203.0.113.0/24";  // Configure this for your
 allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK für C++ API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param keyFilePath: File path where the credentials are stored. Ignored if it is
 an empty string;
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
 &keyFilePath,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
 ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - "  << keyPairName << ". " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
                  keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                      keyFilePath << std::endl;
        }
```

```
    }

    return outcome.IsSuccess();

}
```

- API-Details finden Sie CreateKeyPair in der AWS SDK für C++ API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create a security group.
/*!
  \param groupName: A security group name.
  \param description: A description.
  \param vpcID: A virtual private cloud (VPC) ID.
  \param[out] groupIDResult: A string to receive the group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                      const Aws::String &description,
                                      const Aws::String &vpcID,
                                      Aws::String &groupIDResult,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
```

```
    request.SetDescription(description);
    request.SetVpcId(vpcID);

    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
            ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully created security group named " << groupName <<
              std::endl;


    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK für C++ API-Referenz.

`CreateTags`

Das folgende Codebeispiel zeigt die Verwendung`CreateTags`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add or overwrite only the specified tags for the specified Amazon Elastic
 Compute Cloud (Amazon EC2) resource or resources.
/*!
  \param resources: The resources for the tags.
  \param tags: Vector of tags.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
 ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateTags in der AWS SDK für C++ API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
```

```
  \param keyPairName: A name for a key pair.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK für C++ API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a security group.
```

```
/*!
  \param securityGroupID: A security group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
 ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK für C++ API-Referenz.

`DescribeAddresses`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAddresses`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Elastic IP addresses.
```

```
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeAddresses(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
 ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
                  std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
                  std::setw(30) << "Allocation ID" << std::setw(25) <<
                  "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
 outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                    Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                            address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                      address.GetInstanceId() << std::setw(15) <<
                      address.GetPublicIp() << std::setw(10) << domainString <<
                      std::setw(30) << address.GetAllocationId() << std::setw(25)
                      << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeAddresses in der AWS SDK für C++ API-Referenz.

`DescribeAvailabilityZones`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAvailabilityZones`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! DescribeAvailabilityZones
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
 ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "ZoneName" <<
                  std::setw(20) << "State" <<
                  std::setw(32) << "Region" << std::endl;

        const auto &zones =
                outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

 Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                            zone.GetState());
            std::cout << std::left <<
                      std::setw(32) << zone.GetZoneName() <<
                      std::setw(20) << stateString <<
                      std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
```

```
                    outcome.GetError().GetMessage() << std::endl;

    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeAvailabilityZones in der AWS SDK für C++ API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated with
 an account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
 ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                          std::setw(48) << "Name" <<
                          std::setw(20) << "ID" <<
```

```
                        std::setw(25) << "Ami" <<
                        std::setw(15) << "Type" <<
                        std::setw(15) << "State" <<
                        std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                    outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
                        reservation.GetInstances();
                for (const auto &instance: instances) {
                    Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                                    instance.GetState().GetName());

                    Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                                    instance.GetInstanceType());

                    Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                                    instance.GetMonitoring().GetState());
                    Aws::String name = "Unknown";

                    const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                                                 [](const Aws::EC2::Model::Tag &tag)
{
                                                     return tag.GetKey() == "Name";
                                                 });
                    if (nameIter != tags.cend()) {
                        name = nameIter->GetValue();
                    }
                    std::cout <<
                              std::setw(48) << name <<
                              std::setw(20) << instance.GetInstanceId() <<
                              std::setw(25) << instance.GetImageId() <<
```

```
                                std::setw(15) << typeString <<
                                std::setw(15) << instanceStateString <<
                                std::setw(15) << monitorString << std::endl;
                            }
                        }

                        if (!outcome.GetResult().GetNextToken().empty()) {
                            request.SetNextToken(outcome.GetResult().GetNextToken());
                        } else {
                            done = true;
                        }
                    } else {
                        std::cerr << "Failed to describe EC2 instances:" <<
                                  outcome.GetError().GetMessage() << std::endl;
                        return false;
                    }
                }

                return true;
            }
```

- API-Details finden Sie DescribeInstances in der AWS SDK für C++ API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeKeyPairs(
```

```
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
 ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "Name" <<
                  std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
                outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                      std::setw(32) << key_pair.GetKeyName() <<
                      std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK für C++ API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
```

```
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeRegions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
 ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "RegionName" <<
                  std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                      std::setw(32) << region.GetRegionName() <<
                      std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();

}
```

- API-Details finden Sie DescribeRegions in der AWS SDK für C++ API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
 specific group.
/*!
  \param groupID: A group ID, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
 ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
                      std::setw(32) << "Name" <<
                      std::setw(30) << "GroupId" <<
                      std::setw(30) << "VpcId" <<
                      std::setw(64) << "Description" << std::endl;

            const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
                    outcome.GetResult().GetSecurityGroups();
```

```
            for (const auto &securityGroup: securityGroups) {
                std::cout << std::left <<
                          std::setw(32) << securityGroup.GetGroupName() <<
                          std::setw(30) << securityGroup.GetGroupId() <<
                          std::setw(30) << securityGroup.GetVpcId() <<
                          std::setw(64) << securityGroup.GetDescription() <<
                          std::endl;
            }
        } else {
            std::cerr << "Failed to describe security groups:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return true;
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für C++ API-Referenz.

`MonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`MonitorInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
 ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to enable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
 ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
                  instanceId << ": " <<
                  monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
                  instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}
```

- API-Details finden Sie MonitorInstances in der AWS SDK für C++ API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
 ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to reboot on instance. A dry run should trigger
 an error."
                <<
                std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }
```

```
    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
 ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie RebootInstances in der AWS SDK für C++ API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);
```

```
    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                  allocationID << ":" << outcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                  allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie ReleaseAddress in der AWS SDK für C++ API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceName: A name for the EC2 instance.
  \param amiId: An Amazon Machine Image (AMI) identifier.
  \param[out] instanceID: String to return the instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
            runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
 runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
                  " based on ami " << amiId << ":" <<
                  runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}
```

- API-Details finden Sie RunInstances in der AWS SDK für C++ API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
 ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to start instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
 ec2Client.StartInstances(startRequest);
```

```
    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}
```

- API-Details finden Sie StartInstances in der AWS SDK für C++ API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Stop an EC2 instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
 ec2Client.StopInstances(request);
```

```
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to stop instance. A dry run should trigger an
 error."
                << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
                  << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
 ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
                  std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
              std::endl;
}
```

- API-Details finden Sie StopInstances in der AWS SDK für C++ API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
            ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
                  "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
                  ", " <<
                  outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie TerminateInstances in der AWS SDK für C++ API-Referenz.

`UnmonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`UnmonitorInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Disable monitoring for an EC2 instance.
/*!
  \param instanceId: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to disable monitoring on instance. A dry run
 should trigger an error."
                <<
                std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }
```

```
    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
 ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
                  << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
                  std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
                  instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}
```

- API-Details finden Sie UnmonitorInstances in der AWS SDK für C++ API-Referenz.

EventBridge Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Senden Sie das Ereignis.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
        event_entry.SetDetail(MakeDetails(event_key, event_value));
        event_entry.SetDetailType("sampleSubmitted");
        event_entry.AddResources(resource_arn);
        event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

        Aws::CloudWatchEvents::Model::PutEventsRequest request;
        request.AddEntries(event_entry);

        auto outcome = cwe.PutEvents(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to post CloudWatch event: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully posted CloudWatch event" << std::endl;
        }
```

- API-Details finden Sie PutEvents in der AWS SDK für C++ API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie die Regel.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;
        Aws::CloudWatchEvents::Model::PutRuleRequest request;
        request.SetName(rule_name);
        request.SetRoleArn(role_arn);
        request.SetScheduleExpression("rate(5 minutes)");
        request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

        auto outcome = cwe.PutRule(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events rule " <<
                rule_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
        else
```

```
        {
            std::cout << "Successfully created CloudWatch events rule " <<
                rule_name << " with resulting Arn " <<
                outcome.GetResult().GetRuleArn() << std::endl;
        }
```

- API-Details finden Sie PutRule in der AWS SDK für C++ API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Fügen Sie das Ziel hinzu.

```
        Aws::CloudWatchEvents::EventBridgeClient cwe;

        Aws::CloudWatchEvents::Model::Target target;
        target.SetArn(lambda_arn);
        target.SetId(target_id);

        Aws::CloudWatchEvents::Model::PutTargetsRequest request;
```

```
        request.SetRule(rule_name);
        request.AddTargets(target);

        auto putTargetsOutcome = cwe.PutTargets(request);
        if (!putTargetsOutcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout <<
                "Successfully created CloudWatch events target for rule "
                << rule_name << std::endl;
        }
```

- API-Details finden Sie PutTargets in der AWS SDK für C++ API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für C++

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
```

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListJobs in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String &bucketName,
```

```
                                                        const Aws::String &roleName,
                                                        const
Aws::Client::ClientConfiguration &clientConfig) {
   Aws::Glue::GlueClient client(clientConfig);

   Aws::String roleArn;
   if (!getRoleArn(roleName, roleArn, clientConfig)) {
       std::cerr << "Error getting role ARN for role." << std::endl;
       return false;
   }

   // 1. Upload the job script to the S3 bucket.
   {
       std::cout << "Uploading the job script '"
                 << AwsDoc::Glue::PYTHON_SCRIPT
                 << "'." << std::endl;

       if (!AwsDoc::Glue::uploadFile(bucketName,
                                     AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                     AwsDoc::Glue::PYTHON_SCRIPT,
                                     clientConfig)) {
           std::cerr << "Error uploading the job file." << std::endl;
           return false;
       }
   }

   // 2. Create a crawler.
   {
       Aws::Glue::Model::S3Target s3Target;
       s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
       Aws::Glue::Model::CrawlerTargets crawlerTargets;
       crawlerTargets.AddS3Targets(s3Target);

       Aws::Glue::Model::CreateCrawlerRequest request;
       request.SetTargets(crawlerTargets);
       request.SetName(CRAWLER_NAME);
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
       request.SetRole(roleArn);

       Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

       if (outcome.IsSuccess()) {
```

```
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<
                      Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);
```

```
        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                        crawlerState)
                                << ". After " << iterations
                                << " seconds elapsed."
                                << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                                << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }
```

```
            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 5. Get a database.
    {
        Aws::Glue::Model::GetDatabaseRequest request;
        request.SetName(CRAWLER_DATABASE_NAME);

        Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

        if (outcome.IsSuccess()) {
            const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

            std::cout << "Successfully retrieve the database\n" <<
                      database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 6. Get tables.
    Aws::String tableName;
```

```
    {
        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(), tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " << all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }
```

```
    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;
```

```
            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                          jobRunState) <<
                                  ". " << iterator <<
```

```
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(), objects.end());
                continuationToken = outcome.GetResult().GetNextContinuationToken();
            }
```

```
            else {
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }

        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
first ") +
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; ++i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
    Aws::String jobName;
```

```
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
```

```
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
    }

    // 12. Get a single job run.
```

```
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
 jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;
```

```
    // 13. Delete a job.
    if (!job.empty()) {
        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    // 14. Delete a database.
    if (!database.empty()) {
        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    // 15. Delete a crawler.
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);
```

```
        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);
```

```
    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }
```

```
        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
                                    object.GetKey()));
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." << std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
                          << std::endl;
            }

            continuationToken =
listObjectsOutcome.GetResult().GetNextContinuationToken();
```

```
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." << std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
            return false;
        }
```

- API-Details finden Sie CreateCrawler in der AWS SDK für C++ API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
```

```
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- API-Details finden Sie CreateJob in der AWS SDK für C++ API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

- API-Details finden Sie DeleteCrawler in der AWS SDK für C++ API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- API-Details finden Sie DeleteDatabase in der AWS SDK für C++ API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- API-Details finden Sie DeleteJob in der AWS SDK für C++ API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<
                      Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- API-Details finden Sie GetCrawler in der AWS SDK für C++ API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetDatabaseRequest request;
       request.SetName(CRAWLER_DATABASE_NAME);

       Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

       if (outcome.IsSuccess()) {
           const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

           std::cout << "Successfully retrieve the database\n" <<
                     database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
       }
       else {
           std::cerr << "Error getting the database.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
```

- API-Details finden Sie GetDatabase in der AWS SDK für C++ API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API-Details finden Sie GetJobRun in der AWS SDK für C++ API-Referenz.

GetJobRuns

Das folgende Codebeispiel zeigt die VerwendungGetJobRuns.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
```

```
            }
        } while (!nextToken.empty());
```

- API-Details finden Sie GetJobRuns in der AWS SDK für C++ API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetTablesRequest request;
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        std::vector<Aws::Glue::Model::Table> all_tables;
        Aws::String nextToken; // Used for pagination.
        do {
            Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
                all_tables.insert(all_tables.end(), tables.begin(), tables.end());
                nextToken = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " << all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- API-Details finden Sie GetTables in der AWS SDK für C++ API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
```

```
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
```

- API-Details finden Sie ListJobs in der AWS SDK für C++ API-Referenz.

StartCrawler

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
               std::cout << "Successfully started crawler." << std::endl;
           }

           std::cout << "This may take a while to run." << std::endl;

           Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
           int iterations = 0;
           while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
               std::this_thread::sleep_for(std::chrono::seconds(1));
               ++iterations;
               if ((iterations % 10) == 0) { // Log status every 10 seconds.
                   std::cout << "Crawler status " <<
```

```
Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                    crawlerState)
                            << ". After " << iterations
                            << " seconds elapsed."
                            << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- API-Details finden Sie StartCrawler in der AWS SDK für C++ API-Referenz.

StartJobRun

Das folgende Codebeispiel zeigt die VerwendungStartJobRun.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);
```

```
                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                          jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
```

```
            }
        }
        else {
            std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                         clientConfig);
            return false;
        }
```

- API-Details finden Sie StartJobRun in der AWS SDK für C++ API-Referenz.

HealthImaging Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo HealthImaging

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von HealthImaging beginnen.

SDK für C++

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
 need to uncomment this
    # and set the proper subdirectory to the executable location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_health_imaging.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_health_imaging.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 *  A "Hello HealthImaging" starter application which initializes an AWS
 HealthImaging (HealthImaging) client
 *  and lists the HealthImaging data stores in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_health-imaging'
 *
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    //   Optional: change the log level for debugging.
    //   options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
 medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;
```

```
        Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 allDataStoreSummaries;
        Aws::String nextToken; // Used for paginated results.
        do {
            if (!nextToken.empty()) {
                listDatastoresRequest.SetNextToken(nextToken);
            }
            Aws::MedicalImaging::Model::ListDatastoresOutcome listDatastoresOutcome
 =
                    medicalImagingClient.ListDatastores(listDatastoresRequest);
            if (listDatastoresOutcome.IsSuccess()) {
                const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
 &dataStoreSummaries =
                        listDatastoresOutcome.GetResult().GetDatastoreSummaries();
                allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                             dataStoreSummaries.cbegin(),
                                             dataStoreSummaries.cend());
                nextToken = listDatastoresOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "ListDatastores error: "
                          << listDatastoresOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allDataStoreSummaries.size() << " HealthImaging data "
                  << ((allDataStoreSummaries.size() == 1) ?
                      "store was retrieved." : "stores were retrieved.") <<
 std::endl;

        for (auto const &dataStoreSummary: allDataStoreSummaries) {
            std::cout << "  Datastore: " << dataStoreSummary.GetDatastoreName()
                      << std::endl;
            std::cout << "  Datastore ID: " << dataStoreSummary.GetDatastoreId()
                      << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- API-Details finden Sie ListDatastores in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktionen

`DeleteImageSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteImageSet`.

SDK für C++

```
//! Routine which deletes an AWS HealthImaging image set.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::deleteImageSet(
        const Aws::String &dataStoreID, const Aws::String &imageSetID,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
 client.DeleteImageSet(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
                  << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data store
 "
```

```
                    << dataStoreID << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteImageSet in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`GetDICOMImportJob`.

SDK für C++

```
//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
```

```
                    << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}
```

- API-Details finden Sie GetDICOMImportJob in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt die Verwendung`GetImageFrame`.

SDK für C++

```
//! Routine which downloads an AWS HealthImaging image frame.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The image set ID.
  \param frameID: The image frame ID.
  \param jphFile: File to store the downloaded frame.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                            const Aws::String &imageSetID,
                                            const Aws::String &frameID,
                                            const Aws::String &jphFile,
                                            const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
```

```
    imageFrameInformation.SetImageFrameId(frameID);
    request.SetImageFrameInformation(imageFrameInformation);

    Aws::MedicalImaging::Model::GetImageFrameOutcome outcome = client.GetImageFrame(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved image frame." << std::endl;
        auto &buffer = outcome.GetResult().GetImageFrameBlob();

        std::ofstream outfile(jphFile, std::ios::binary);
        outfile << buffer.rdbuf();
    }
    else {
        std::cout << "Error retrieving image frame." <<
 outcome.GetError().GetMessage()
                  << std::endl;

    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetImageFrame in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSetMetadata`.

SDK für C++

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
```

```
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 "", outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." << std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
```

```
        }
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
        if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
 versionID, outputFilePath, clientConfig))
        {
            std::cout << "Successfully retrieved image set metadata." << std::endl;
            std::cout << "Metadata stored in: " << outputFilePath << std::endl;
        }
```

- API-Details finden Sie GetImageSetMetadata in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt die Verwendung`SearchImageSets`.

SDK für C++

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param searchCriteria: A search criteria instance.
  \param imageSetResults: Vector to receive the image set IDs.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
 Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
 &imageSetResults,
```

```
                                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
 client.SearchImageSets(
                request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
 outcome.GetResult().GetImageSetsMetadataSummaries()) {
                imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}
```

Anwendungsfall 1: EQUAL-Operator.

```
        Aws::Vector<Aws::String> imageIDsForPatientID;
        Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
        Aws::Vector<Aws::MedicalImaging::Model::SearchFilter> patientIDSearchFilters
 = {

 Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Operato
```

```
 .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patien
        };

        searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
        bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

 searchCriteriaEqualsPatientID,
                                                                imageIDsForPatientID,
                                                                clientConfig);
        if (result) {
            std::cout << imageIDsForPatientID.size() << " image sets found for the
 patient with ID '"
            <<  patientID << "'." << std::endl;
            for (auto &imageSetResult : imageIDsForPatientID) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudyDate und DICOMStudyTime.

```
         Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

 useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTim
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

 useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(

 .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeStrin
%m%d"))
        .WithDICOMStudyTime("000000.000"));

        Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
        useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

 useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
```

```
        useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

        Aws::Vector<Aws::String> usesCase2Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase2SearchCriteria,
                                                          usesCase2Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase2Results.size() << " image sets found for between
1999/01/01 and present."
                      <<  std::endl;
            for (auto &imageSetResult : usesCase2Results) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;

useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::DateF

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;

useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
        useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
        useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

        Aws::Vector<Aws::String> usesCase3Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                          useCase3SearchCriteria,
                                                          usesCase3Results,
                                                          clientConfig);
        if (result) {
            std::cout << usesCase3Results.size() << " image sets found for created
between 2023/11/30 and present."
```

```
                    <<  std::endl;
            for (auto &imageSetResult : usesCase3Results) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

Anwendungsfall #4: EQUAL-Operator für DICOMSeriesInstanceUID und BETWEEN für updatedAt und Sortierung der Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z",Aws::Utils::DateF

        Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

        Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
        useCase4SearchFilterBetween.SetValues({useCase4StartDate, useCase4EndDate});

useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

        Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
        seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

        Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
        useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

        Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
        useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
useCase4SearchFilterEqual});

        Aws::MedicalImaging::Model::Sort useCase4Sort;
        useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
        useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

        useCase4SearchCriteria.SetSort(useCase4Sort);

        Aws::Vector<Aws::String> usesCase4Results;
        result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
```

```
                                                                    useCase4SearchCriteria,
                                                                    usesCase4Results,
                                                                    clientConfig);
        if (result) {
            std::cout << usesCase4Results.size() << " image sets found for EQUAL
 operator "
            << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response
\n"
            <<  "in ASC order on updatedAt field." <<  std::endl;
            for (auto &imageSetResult : usesCase4Results) {
                std::cout << "  Image set with ID '" << imageSetResult << std::endl;
            }
        }
```

- API-Details finden Sie in der API-Referenz. SearchImageSets AWS SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

StartDICOMImportJob

Das folgende Codebeispiel zeigt die Verwendung`StartDICOMImportJob`.

SDK für C++

```
//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
```

```
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}
```

- API-Details finden Sie StartDICOMImportJob in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien

Erste Schritte mit Bildsätzen und Bild-Frames

Das folgende Codebeispiel zeigt, wie DICOM-Dateien importiert und Bildrahmen heruntergeladen werden. HealthImaging

Die Implementierung ist als Befehlszeilenanwendung strukturiert.

- Richten Sie Ressourcen für einen DICOM-Import ein.
- Importieren Sie DICOM-Dateien in einen Datenspeicher.
- Rufen Sie die Bildsatz-IDs für den Importauftrag ab.
- Rufen Sie die Bild-Frame-IDs für den Importauftrag ab.
- Laden Sie die Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.
- Bereinigen Sie die Ressourcen.

SDK für C++

Erstellen Sie einen CloudFormation Stapel mit den erforderlichen Ressourcen.

```
    Aws::String inputBucketName;
    Aws::String outputBucketName;
    Aws::String dataStoreId;
    Aws::String roleArn;
    Aws::String stackName;

    if (askYesNoQuestion(
            "Would you like to let this workflow create the resources for you? (y/n)
")) {
        stackName = askQuestion(
                "Enter a name for the AWS CloudFormation stack to create. ");
        Aws::String dataStoreName = askQuestion(
                "Enter a name for the HealthImaging datastore to create. ");

        Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
                stackName,
                dataStoreName,
                clientConfiguration);
```

```
        if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
outputBucketName,
                             roleArn)) {
            return false;
        }

        std::cout << "The following resources have been created." << std::endl;
        std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
                  << std::endl;
        std::cout << "An Amazon S3 input bucket named: " << inputBucketName << "."
                  << std::endl;
        std::cout << "An Amazon S3 output bucket named: " << outputBucketName << "."
                  << std::endl;
        std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
        askQuestion("Enter return to continue.", alwaysTrueTest);
    }
    else {
        std::cout << "You have chosen to use preexisting resources:" << std::endl;
        dataStoreId = askQuestion(
                "Enter the data store ID of the HealthImaging datastore you wish to
use: ");
        inputBucketName = askQuestion(
                "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
                "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
                "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
```

Kopieren Sie DICOM-Dateien in den Amazon-S3-Import-Bucket.

```
    std::cout
            << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
            << "Commons (IDC) Collections." << std::endl;
    std::cout << "Here is the link to their website." << std::endl;
    std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
    std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
              << std::endl;
```

```
    std::cout
            << "First one of the DICOM folders in the IDC collection must be copied
 to your\n"
               "input S3 bucket."
            << std::endl;
    std::cout << "You have the choice of one of the following "
              << IDC_ImageChoices.size() << " folders to copy." << std::endl;

    int index = 1;
    for (auto &idcChoice: IDC_ImageChoices) {
        std::cout << index << " - " << idcChoice.mDescription << std::endl;
        index++;
    }
    int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

    Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
    Aws::String inputDirectory = "input";

    std::cout << "The files in the directory '" << fromDirectory << "' in the bucket
 '"
              << IDC_S3_BucketName << "' will be copied " << std::endl;
    std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
              << "' in the bucket '" << inputBucketName << "'." << std::endl;
    askQuestion("Enter return to start the copy.", alwaysTrueTest);

    if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
            IDC_S3_BucketName,
            fromDirectory,
            inputBucketName,
            inputDirectory, clientConfiguration)) {
        std::cerr << "This workflow will exit because of an error." << std::endl;
        cleanup(stackName, dataStoreId, clientConfiguration);
        return false;
    }
```

Importieren Sie die DICOM-Dateien in den Amazon-S3-Datenspeicher.

```
bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                               const Aws::String &inputBucketName,
                                               const Aws::String &inputDirectory,
                                               const Aws::String &outputBucketName,
                                               const Aws::String &outputDirectory,
```

```
                                                const Aws::String &roleArn,
                                                Aws::String &importJobId,
                                                const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = false;
    if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                            outputBucketName, outputDirectory, roleArn, importJobId,
                            clientConfiguration)) {
        std::cout << "DICOM import job started with job ID " << importJobId << "."
                  << std::endl;
        result = waitImportJobCompleted(dataStoreID, importJobId,
 clientConfiguration);
        if (result) {
            std::cout << "DICOM import job completed." << std::endl;

        }
    }

    return result;
}

//! Routine which starts a HealthImaging import job.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
 files.
  \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
  \param outputBucketName: The name of the S3 bucket for the output.
  \param outputDirectory: The directory in the S3 bucket to store the output.
  \param roleArn: The ARN of the IAM role with permissions for the import.
  \param importJobId: A string to receive the import job ID.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
 "/";
```

```
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
 startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
 startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
            startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
                  << startDICOMImportJobOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}


//! Routine which waits for a DICOM import job to complete.
/*!
 * @param dataStoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String &datastoreID,
                                                     const Aws::String &importJobId,
                                                     const
 Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
 Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
 getDicomImportJobOutcome = getDICOMImportJob(
                datastoreID, importJobId,
```

```
                clientConfiguration);

        if (getDicomImportJobOutcome.IsSuccess()) {
            jobStatus =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

            std::cout << "DICOM import job status: " <<

 Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
                              jobStatus) << std::endl;
        }
        else {
            std::cerr << "Failed to get import job status because "
                      << getDicomImportJobOutcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    }

    return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param importJobID: The DICOM import job ID
  \param clientConfig: Aws client configuration.
  \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
 client.GetDICOMImportJob(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
                  << outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome;
}
```

Rufen Sie Bildsätze ab, die durch den DICOM-Importauftrag erstellt wurden.

```
bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
 &datastoreID,
                                                        const Aws::String
 &importJobId,
                                                        Aws::Vector<Aws::String>
 &imageSets,
                                                        const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome =
 getDICOMImportJob(
            datastoreID, importJobId, clientConfiguration);
    bool result = false;
    if (getDicomImportJobOutcome.IsSuccess()) {
        auto outputURI =
 getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
        Aws::Http::URI uri(outputURI);
        const Aws::String &bucket = uri.GetAuthority();
        Aws::String key = uri.GetPath();

        Aws::S3::S3Client s3Client(clientConfiguration);
        Aws::S3::Model::GetObjectRequest objectRequest;
        objectRequest.SetBucket(bucket);
        objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

        auto getObjectOutcome = s3Client.GetObject(objectRequest);
        if (getObjectOutcome.IsSuccess()) {
            auto &data = getObjectOutcome.GetResult().GetBody();

            std::stringstream stringStream;
            stringStream << data.rdbuf();

            try {
                // Use JMESPath to extract the image set IDs.
                // https://jmespath.org/specification.html
```

```
                std::string jmesPathExpression =
 "jobSummary.imageSetsSummary[].imageSetId";
                jsoncons::json doc = jsoncons::json::parse(stringStream.str());

                jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,

 jmesPathExpression);\
                for (auto &imageSet: imageSetsJson.array_range()) {
                    imageSets.push_back(imageSet.as_string());
                }

                result = true;
            }
            catch (const std::exception &e) {
                std::cerr << e.what() << '\n';
            }

        }
        else {
            std::cerr << "Failed to get object because "
                      << getObjectOutcome.GetError().GetMessage() << std::endl;
        }

    }
    else {
        std::cerr << "Failed to get import job status because "
                  << getDicomImportJobOutcome.GetError().GetMessage() << std::endl;
    }

    return result;
}
```

Rufen Sie Bild-Frame-Informationen für Bildsätze ab.

```
bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
 &dataStoreID,
                                                         const Aws::String
 &imageSetID,
                                                         const Aws::String
 &outDirectory,
                                                         Aws::Vector<ImageFrameInfo>
 &imageFrames,
```

```
                                                        const
Aws::Client::ClientConfiguration &clientConfiguration) {
   Aws::String fileName = outDirectory + "/" + imageSetID + "_metadata.json.gzip";
   bool result = false;
   if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for version
ID.
                           fileName, clientConfiguration)) {
       try {
           std::string metadataGZip;
           {
               std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
               if (!inFileStream) {
                   throw std::runtime_error("Failed to open file " + fileName);
               }

               std::stringstream stringStream;
               stringStream << inFileStream.rdbuf();
               metadataGZip = stringStream.str();
           }
           std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                        metadataGZip.size());
           // Use JMESPath to extract the image set IDs.
           // https://jmespath.org/specification.html
           jsoncons::json doc = jsoncons::json::parse(metadataJson);
           std::string jmesPathExpression = "Study.Series.*.Instances[].*[]";
           jsoncons::json instances = jsoncons::jmespath::search(doc,

jmesPathExpression);
           for (auto &instance: instances.array_range()) {
               jmesPathExpression = "DICOM.RescaleSlope";
               std::string rescaleSlope = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();
               jmesPathExpression = "DICOM.RescaleIntercept";
               std::string rescaleIntercept = jsoncons::jmespath::search(instance,

jmesPathExpression).to_string();

               jmesPathExpression = "ImageFrames[][]";
               jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,

jmesPathExpression);
```

```
                for (auto &imageFrame: imageFramesJson.array_range()) {
                    ImageFrameInfo imageFrameIDs;
                    imageFrameIDs.mImageSetId = imageSetID;
                    imageFrameIDs.mImageFrameId = imageFrame.find(
                            "ID")->value().as_string();
                    imageFrameIDs.mRescaleIntercept = rescaleIntercept;
                    imageFrameIDs.mRescaleSlope = rescaleSlope;
                    imageFrameIDs.MinPixelValue = imageFrame.find(
                            "MinPixelValue")->value().as_string();
                    imageFrameIDs.MaxPixelValue = imageFrame.find(
                            "MaxPixelValue")->value().as_string();

                    jmesPathExpression =
 "max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
                    jsoncons::json checksumJson =
 jsoncons::jmespath::search(imageFrame,

 jmesPathExpression);
                    imageFrameIDs.mFullResolutionChecksum =
 checksumJson.as_integer<uint32_t>();

                    imageFrames.emplace_back(imageFrameIDs);
                }
            }

            result = true;
        }
        catch (const std::exception &e) {
            std::cerr << "getImageFramesForImageSet failed because " << e.what()
                      << std::endl;
        }
    }

    return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
  \param dataStoreID: The HealthImaging data store ID.
  \param imageSetID: The HealthImaging image set ID.
  \param versionID: The HealthImaging image set version ID, ignored if empty.
  \param outputFilePath: The path where the metadata will be stored as gzipped json.
  \param clientConfig: Aws client configuration.
  \\return bool: Function succeeded.
```

```
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
 client.GetImageSetMetadata(
            request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Laden Sie Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.

```
bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
        const Aws::String &dataStoreID,
        const Aws::Vector<ImageFrameInfo> &imageFrames,
        const Aws::String &outDirectory,
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
```

```
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
            clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
outDirectory](
                const Aws::MedicalImaging::MedicalImagingClient *client,
                const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
                Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext>
&context) {

                if (!handleGetImageFrameResult(outcome, outDirectory, imageFrame)) {
                    std::cerr << "Failed to download and convert image frame: "
                              << imageFrame.mImageFrameId << " from image set: "
                              << imageFrame.mImageSetId << std::endl;
                    result = false;
                }

                count--;
                if (count <= 0) {

                    semaphore.ReleaseAll();
                }
        }; // End of 'getImageFrameAsyncLambda' lambda.

        medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                                getImageFrameAsyncLambda);
    }

    if (count > 0) {
        semaphore.WaitOne();
```

```
    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
                  << std::endl;
    }

    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
        const Aws::String &jphFile,
        uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected value."
                  << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
 std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
```

```
        decodeParameters->cod_format = 2; // BMP image format.

        std::strncpy(decodeParameters->infile, jphFile.c_str(),
                     OPJ_PATH_LEN);

        inFileStream = opj_stream_create_default_file_stream(
                decodeParameters->infile, true);
        if (!inFileStream) {
            throw std::runtime_error(
                    "Unable to create input file stream for file '" + jphFile +
"'.");
        }

        decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
        if (!decompressorCodec) {
            throw std::runtime_error("Failed to create decompression codec.");
        }

        int decodeMessageLevel = 1;
        if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
            std::cerr << "Failed to setup codec logging." << std::endl;
        }

        if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
            throw std::runtime_error("Failed to setup decompression codec.");
        }
        if (!opj_codec_set_threads(decompressorCodec, 4)) {
            throw std::runtime_error("Failed to set decompression codec threads.");
        }

        if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
            throw std::runtime_error("Failed to read header.");
        }

        if (!opj_decode(decompressorCodec, inFileStream,
                        outputImage)) {
            throw std::runtime_error("Failed to decode.");
        }

        if (DEBUGGING) {
            std::cout << "image width : " << outputImage->x1 - outputImage->x0
                      << std::endl;
            std::cout << "image height : " << outputImage->y1 - outputImage->y0
                      << std::endl;
```

```
            std::cout << "number of channels: " << outputImage->numcomps
                      << std::endl;
            std::cout << "colorspace : " << outputImage->color_space << std::endl;
        }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved image
 bitmap and
//! then verifies the checksum of the bitmap.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;
```

```
            uint32_t toIndex = (row * width) * numOfChannels + channel;

            for (uint32_t col = 0; col < width; col++) {
                uint32_t fromIndex = fromRowStart + col / image->comps[channel].dx;

                buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

                toIndex += numOfChannels;
            }
        }
    }

    // Verify checksum.
    boost::crc_32_type crc32;
    crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                        buffer.size() * sizeof(myType));

    bool result = crc32.checksum() == crc32Checksum;
    if (!result) {
        std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                  << crc32Checksum << ", actual - " << crc32.checksum()
                  << std::endl;
    }

    return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return  bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                     uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;
```

```
        if (signedData) {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,
                                                                   crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,
                                                                    crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,
                                                                    crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
signed bytes - "
                            << bytes << std::endl;
                    break;
            }
        }
        else {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<uint8_t>(image,
                                                                    crc32Checksum);
                    break;
                case 2 :
                    result = verifyChecksumForImageForType<uint16_t>(image,
                                                                     crc32Checksum);
                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,
                                                                     crc32Checksum);
                    break;
                default:
                    std::cerr
                            << "verifyChecksumForImage, unsupported data type,
unsigned bytes - "
                            << bytes << std::endl;
                    break;
            }
        }
```

```
        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                      << " signed "
                      << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
                  << std::endl;
    }
    return result;
}
```

Bereinigen von Ressourcen.

```
bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                      const Aws::String &dataStoreId,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
            "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {
```

```
        result = true;
        for (auto &imageSetID: imageSetIDs) {
            result &= deleteImageSet(datastoreID, imageSetID, clientConfiguration);
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die iam.cpp-Quellendatei.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and Access
 Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) << policy.GetArn()
 <<
                          std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
```

```
                            std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
```

```
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName << std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
```

```
                        << outcome.GetResult().GetUser().GetUserName()
                        << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.
        request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
```

```
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome = client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
```

```
        }
        else {
            std::cout << "Successfully created a policy with name, " << policyName
<<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
```

```
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() << std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
```

```
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." << std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() << std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
```

```
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }
```

```
        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                                return policy.GetPolicyArn() == policyArn;
                        })) {
            std::cout << "Policy " << policyArn <<
                      " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
```

```
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
 iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                  roleName << ": " << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role " <<
                  roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für C++ API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
```

```
    Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- API-Details finden Sie CreateAccessKey in der AWS SDK für C++ API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
```

```
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateAccountAlias in der AWS SDK für C++ API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                    std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "       ],"
                 << "       \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";

    return stringStream.str();
}
```

- API-Details finden Sie CreatePolicy in der AWS SDK für C++ API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateRole in der AWS SDK für C++ API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
              create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

return create_outcome.IsSuccess();
```

- API-Details finden Sie CreateUser in der AWS SDK für C++ API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK für C++ API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteAccountAlias in der AWS SDK für C++ API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeletePolicy in der AWS SDK für C++ API-Referenz.

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
                  << std::endl;
    }

    return result;
}
```

- API-Details finden Sie DeleteServerCertificate in der AWS SDK für C++ API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ": " <<
              outcome.GetError().GetMessage() << std::endl;;
}
else {
    std::cout << "Successfully deleted IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
```

- API-Details finden Sie DeleteUser in der AWS SDK für C++ API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role "
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK für C++ API-Referenz.

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung`GetAccessKeyLastUsed`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);
```

```
    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome = iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetAccessKeyLastUsed in der AWS SDK für C++ API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                            const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);
```

```
    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetPolicy in der AWS SDK für C++ API-Referenz.

`GetServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetServerCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);
```

```
    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                 << std::endl << "Body: " << certificate.GetCertificateBody() <<
                 std::endl << "Chain: " << certificate.GetCertificateChain() <<
                 std::endl;
    }

    return result;
}
```

- API-Details finden Sie GetServerCertificate in der AWS SDK für C++ API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
```

```
        }
    }

    return true;
}
```

- API-Details finden Sie ListAccessKeys in der AWS SDK für C++ API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
```

```
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API-Details finden Sie ListAccountAliases in der AWS SDK für C++ API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;
```

```
    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12) <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für C++ API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
```

```
                        certificate.GetServerCertificateId() << std::setw(80) <<
                        certificate.GetArn() << std::setw(14) <<
                        certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                        std::setw(14) <<
                        certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str())
 <<
                        std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API-Details finden Sie ListServerCertificates in der AWS SDK für C++ API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;
```

```
    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API-Details finden Sie ListUsers in der AWS SDK für C++ API-Referenz.

`PutRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutRolePolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie PutRolePolicy in der AWS SDK für C++ API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateAccessKey in der AWS SDK für C++ API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateServerCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String &currentCertificateName,
                                          const Aws::String &newCertificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName << ":"
 <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }

    return result;
}
```

- API-Details finden Sie UpdateServerCertificate in der AWS SDK für C++ API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateUser in der AWS SDK für C++ API-Referenz.

AWS IoT Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren AWS IoT.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für C++

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
```

```
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_iot.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
```

```
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() << std::endl;
                break;
```

```
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK für C++ -API-Referenz.

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Shadow.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.

- Lösche eine AWS IoT Sache.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erschaffe AWS IoT irgendwas.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
```

```
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Generieren Sie ein Gerätezertifikat und fügen Sie es an.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing? (y/
n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n) "))
 {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN, certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName, clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
            return false;
```

```
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string is
 empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
 ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
```

```
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" << publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
```

```
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Führe verschiedene Operationen an dem AWS IoT Ding aus.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
 std::endl;
```

```
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
 Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
 std::endl;

    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String endpoint;
    if (!describeEndpoint(endpoint, clientConfiguration)) {
        std::cerr << "Exiting because getEndpoint failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
    printAsterisksLine();

    std::cout << "Now the certificates in your account will be listed." <<
 std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!listCertificates(clientConfiguration)) {
        std::cerr << "Exiting because listCertificates failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
    std::cout << "A thing shadow refers to a feature that enables you to create a
 virtual representation, or \"shadow,\"\n"
    << "of a physical device or thing. The thing shadow allows you to synchronize
 and control the state of a device between\n"
    << "the cloud and the device itself. and the AWS IoT service. For example, you
 can write and retrieve JSON data from a thing shadow." << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
```

```
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n" <<
std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to access
data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation template."
<< std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
```

```
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."  <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://docs.aws.amazon.com/
iot/latest/developerguide/iot-sql-reference.html" << std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
```

```
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/latest/
developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account? (y/
n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGISTR

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnectiv
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
```

```
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
```

```
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome = iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
```

```
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Bereinigen von Ressourcen.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                              askYesNoQuestion(
                                      "Delete the CloudFormation stack '" +
 stackName +
                                      "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
```

```
                                                        certificateARN + "'? (y/n) ")))
 {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing
 "
                  << thingName << std::endl;
    }
```

```
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Attach a principal to an AWS IoT thing.
```

```
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie AttachThingPrincipal in der AWS SDK für C++ API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string is
 empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
 ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
```

```
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" << publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS SDK für C++ API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateThing in der AWS SDK für C++ API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateTopicRule in der AWS SDK für C++ API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);
```

```
    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteCertificate in der AWS SDK für C++ API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
```

```
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteThing in der AWS SDK für C++ API-Referenz.

`DeleteTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopicRule`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);
```

```
    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteTopicRule in der AWS SDK für C++ API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
```

```
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK für C++ API-Referenz.

`DescribeThing`

Das folgende Codebeispiel zeigt die Verwendung`DescribeThing`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DescribeThing in der AWS SDK für C++ API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
            detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing
 "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DetachThingPrincipal in der AWS SDK für C++ API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());
```

```
    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- API-Details finden Sie ListCertificates in der AWS SDK für C++ API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
```

```
    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- API-Details finden Sie SearchIndex in der AWS SDK für C++ API-Referenz.

`UpdateIndexingConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateIndexingConfiguration`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration object
 which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {
        request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
```

```
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateIndexingConfiguration in der AWS SDK für C++ API-Referenz.

`UpdateThing`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThing`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
```

```
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateThing in der AWS SDK für C++ API-Referenz.

AWS IoT data Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetThingShadow in der AWS SDK für C++ API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateThingShadow in der AWS SDK für C++ API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

# Set this project's name.
project("hello_lambda")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
```

```
                                        # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_lambda.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>

/*
 *  A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
 client and lists the Lambda functions.
 *
 *  main function
 *
 *  Usage: 'hello_lambda'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);
```

```
        std::vector<Aws::String> functions;
        Aws::String marker; // Used for pagination.

        do {
            Aws::Lambda::Model::ListFunctionsRequest request;
            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::Lambda::Model::ListFunctionsOutcome outcome =
lambdaClient.ListFunctions(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Lambda::Model::ListFunctionsResult &listFunctionsResult =
outcome.GetResult();
                std::cout << listFunctionsResult.GetFunctions().size()
                          << " lambda functions were retrieved." << std::endl;

                for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: listFunctionsResult.GetFunctions()) {
                    functions.push_back(functionConfiguration.GetFunctionName());
                    std::cout << functions.size() << "  "
                              << functionConfiguration.GetDescription() <<
std::endl;
                    std::cout << "   "
                              <<
Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                      functionConfiguration.GetRuntime()) << ": "
                              << functionConfiguration.GetHandler()
                              << std::endl;
                }
                marker = listFunctionsResult.GetNextMarker();
            } else {
                std::cerr << "Error with Lambda::ListFunctions. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!marker.empty());
    }
```

```
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListFunctions in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get started with functions scenario.
/*!
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
```

```
    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
 function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
```

```
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                                buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " << seconds
                      << " seconds elapsed." << std::endl;
            break;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&
                 outcome.GetError().GetMessage().find("role") >= 0) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout
                        << "Waiting for the IAM role to become available as a
 CreateFunction parameter. "
                        << seconds
                        << " seconds elapsed." << std::endl;

                std::cout << outcome.GetError().GetMessage() << std::endl;
            }
        }
        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

```
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (60 > seconds);

    std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

    // 3.  Invoke the Lambda function.
    {
        int increment = askQuestionForInt("Enter an increment integer: ");

        Aws::Lambda::Model::InvokeResult invokeResult;
        Aws::Utils::Json::JsonValue jsonPayload;
        jsonPayload.WithString("action", "increment");
        jsonPayload.WithInteger("number", increment);
        if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
                                 invokeResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
                {
                    std::cout << INCREMENT_RESUlT_PREFIX
                              << iter->second.AsInteger() << std::endl;
                }
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                        invokeResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
            }
        }
    }

    std::cout
            << "The Lambda function will now be updated with new code. Press return
to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);
```

```
    // 4.  Update the Lambda function code.
    {
        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
```

```
            << "This function uses an environment variable to control the logging
level."
            << std::endl;
    std::cout
            << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
            << std::endl;
    seconds = 0;

    // 5.  Update the Lambda function configuration.
    do {
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

            // RESOURCE_IN_USE: function code update not completed.
        else if (outcome.GetError().GetErrorType() !=
                 Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
            if ((seconds % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Lambda function update in progress . After " <<
seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

```
    } while (0 < seconds);

    if (0 > seconds) {
        std::cerr << "Function failed to become active." << std::endl;
    }
    else {
        std::cout << "Updated function active after " << seconds << " seconds."
                  << std::endl;
    }

    std::cout
            << "\nThe new code applies an arithmetic operator to two variables, x an
y."
            << std::endl;
    std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-by"};
    for (size_t i = 0; i < operators.size(); ++i) {
        std::cout << "   " << i + 1 << " " << operators[i] << std::endl;
    }

    // 6.  Invoke the updated Lambda function.
    do {
        int operatorIndex = askQuestionForIntRange("Select an operator index 1 - 4
", 1,
                                                   4);
        int x = askQuestionForInt("Enter an integer for the x value ");
        int y = askQuestionForInt("Enter an integer for the y value ");

        Aws::Utils::Json::JsonValue calculateJsonPayload;
        calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
        calculateJsonPayload.WithInteger("x", x);
        calculateJsonPayload.WithInteger("y", y);
        Aws::Lambda::Model::InvokeResult calculatedResult;
        if (invokeLambdaFunction(calculateJsonPayload,
                                 Aws::Lambda::Model::LogType::Tail,
                                 calculatedResult, client)) {
            Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
            Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
                    jsonValue.View().GetAllObjects();
            auto iter = values.find("result");
            if (iter != values.end() && iter->second.IsIntegerType()) {
                std::cout << ARITHMETIC_RESUlT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsInteger() << std::endl;
            }
```

```
            else if (iter != values.end() && iter->second.IsFloatingPointType()) {
                std::cout << ARITHMETIC_RESULT_PREFIX << x << " "
                          << operators[operatorIndex - 1] << " "
                          << y << " is " << iter->second.AsDouble() << std::endl;
            }
            else {
                std::cout << "There was an error in execution. Here is the log."
                          << std::endl;
                Aws::Utils::ByteBuffer buffer =
Aws::Utils::HashingUtils::Base64Decode(
                        calculatedResult.GetLogResult());
                std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
            }
        }

        answer = askQuestion("Would you like to try another operation? (y/n) ");
    } while (answer == "y");

    std::cout
            << "A list of the lambda functions will be retrieved. Press return to
continue, ";
    std::getline(std::cin, answer);

    // 7.  List the Lambda functions.

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;
```

```
            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    // 8.  Get a Lambda function.
    if (!functions.empty()) {
        std::stringstream question;
        question << "Choose a function to retrieve between 1 and " <<
functions.size()
                 << " ";
        int functionIndex = askQuestionForIntRange(question.str(), 1,

static_cast<int>(functions.size()));

        Aws::String functionName = functions[functionIndex - 1];

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
        }
```

```
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout << "The resources will be deleted. Press return to continue, ";
    std::getline(std::cin, answer);

    // 9.  Delete the Lambda function.
    bool result = deleteLambdaFunction(client);

    // 10. Delete the IAM role.
    return result && deleteIamRole(clientConfig);
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue &jsonPayload,
                                     Aws::Lambda::Model::LogType logType,
                                     Aws::Lambda::Model::InvokeResult &invokeResult,
                                     const Aws::Lambda::LambdaClient &client) {
    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
 resources are
     * available.  The Invoke function is called repeatedly until the resources are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
```

```
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

            // ACCESS_DENIED: because the role is not available yet.
            // RESOURCE_CONFLICT: because the Lambda function is being created or
 updated.
        else if ((outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
                 (outcome.GetError().GetErrorType() ==
                  Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout << "Waiting for the invoke api to be available, status "
 <<
                          ((outcome.GetError().GetErrorType() ==
                            Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                            "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
 seconds
                          << " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
    } while (seconds < 60);

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#if USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);
```

```
        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
#if  defined(__x86_64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
        request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif
        request.SetRole(roleArn);
        request.SetHandler(LAMBDA_HANDLER_NAME);
        request.SetPublish(true);
        Aws::Lambda::Model::FunctionCode code;
        std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();

        code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
 buffer.str().c_str(),
                                               buffer.str().length()));
        request.SetCode(code);

        Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
                request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The lambda function was successfully created. " << seconds
                      << " seconds elapsed." << std::endl;
            break;
        }

        else {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

- API-Details finden Sie CreateFunction in der AWS SDK für C++ API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::DeleteFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);

    Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
            request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API-Details finden Sie DeleteFunction in der AWS SDK für C++ API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::GetFunctionRequest request;
        request.SetFunctionName(functionName);

        Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

        if (outcome.IsSuccess()) {
            std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with Lambda::GetFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API-Details finden Sie GetFunction in der AWS SDK für C++ API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
                "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
```

```
            break;
        }

        else {
            std::cerr << "Error with Lambda::InvokeRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            break;
        }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für C++ -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    std::vector<Aws::String> functions;
    Aws::String marker;

    do {
        Aws::Lambda::Model::ListFunctionsRequest request;
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
```

```
                request);

        if (outcome.IsSuccess()) {
            const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
            std::cout << result.GetFunctions().size()
                      << " lambda functions were retrieved." << std::endl;

            for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
                functions.push_back(functionConfiguration.GetFunctionName());
                std::cout << functions.size() << "  "
                          << functionConfiguration.GetDescription() << std::endl;
                std::cout << "   "
                          << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                                  functionConfiguration.GetRuntime()) << ": "
                          << functionConfiguration.GetHandler()
                          << std::endl;
            }
            marker = result.GetNextMarker();
        }
        else {
            std::cerr << "Error with Lambda::ListFunctions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());
```

- API-Details finden Sie ListFunctions in der AWS SDK für C++ API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
 (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionCodeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                               std::ios_base::in | std::ios_base::binary);
        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
 std::endl;

#if USE_CPP_LAMBDA_FUNCTION
            std::cerr
                    << "The cpp Lambda function must be built following the
 instructions in the cpp_lambda/README.md file. "
                    << std::endl;
#endif
            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
                Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                       buffer.str().length()));
        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
 client.UpdateFunctionCode(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                      << outcome.GetError().GetMessage()
```

```
                        << std::endl;
        }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für C++ API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                      << std::endl;
            break;
        }

        else {
```

```
            std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert Beispiele zur Verwendung des SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren MediaConvert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an AWS Elemental MediaConvert job.
/*!
  \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                           Access Management (IAM) role for the job.
  \param fileInput: A URI to an input file that is stored in Amazon Simple Storage
 Service
                    (Amazon S3) or on an HTTP(S) server.
  \param fileOutput: A URI for an Amazon S3 output location and the output file name
 base.
  \param jobSettingsFile: An optional JSON settings file.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
```

```
                                        const Aws::String &jobSettingsFile,
                                        const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

   createJobRequest.SetRole(mediaConvertRole);
   Aws::Http::HeaderValueCollection hvc;
   hvc.emplace("Customer", "Amazon");
   createJobRequest.SetUserMetadata(hvc);

   if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
   {
       std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
       if (!jobSettingsStream) {
           std::cerr << "Unable to open the job template file." << std::endl;
           return false;
       }
       std::vector<char> buffer(jobSettingsStream.tellg());
       jobSettingsStream.seekg(0);
       jobSettingsStream.read(buffer.data(), buffer.size());
       std::string jobSettingsJSON(buffer.data(), buffer.size());
       size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
       if (pos != std::string::npos) {
           jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER), fileInput);
       }

       pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
       if (pos != std::string::npos) {
           jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
       }
       Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
       Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);

       createJobRequest.SetSettings(jobSettings);
   }
   else { // Configure the job settings programmatically.
       Aws::MediaConvert::Model::JobSettings jobSettings;
       jobSettings.SetAdAvailOffset(0);
       Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
       jobSettings.SetTimecodeConfig(timecodeConfig);
```

```
        // Configure the output group.
        Aws::MediaConvert::Model::OutputGroup outputGroup;
        outputGroup.SetName("File Group");
        Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
        outputGroupSettings.SetType(
                Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
        Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
        fileGroupSettings.SetDestination(fileOutput);
        outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
        outputGroup.SetOutputGroupSettings(outputGroupSettings);

        Aws::MediaConvert::Model::Output output;
        output.SetNameModifier("_1");

        Aws::MediaConvert::Model::VideoDescription videoDescription;
        videoDescription.SetScalingBehavior(
                Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);
        videoDescription.SetTimecodeInsertion(
                Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);
        videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
        videoDescription.SetSharpness(50);

videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
        videoDescription.SetDropFrameTimecode(
                Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
        videoDescription.SetColorMetadata(
                Aws::MediaConvert::Model::ColorMetadata::INSERT);

        Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
        videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
        Aws::MediaConvert::Model::H264Settings h264Settings;
        h264Settings.SetNumberReferenceFrames(3);
        h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
        h264Settings.SetSoftness(0);
        h264Settings.SetGopClosedCadence(1);
        h264Settings.SetGopSize(90);
        h264Settings.SetSlices(1);
        h264Settings.SetGopBReference(
                Aws::MediaConvert::Model::H264GopBReference::DISABLED);
        h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
        h264Settings.SetSpatialAdaptiveQuantization(
                Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
```

```
        h264Settings.SetTemporalAdaptiveQuantization(

Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
        h264Settings.SetFlickerAdaptiveQuantization(

Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
        h264Settings.SetEntropyEncoding(
                Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
        h264Settings.SetBitrate(5000000);
        h264Settings.SetFramerateControl(
                Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
        h264Settings.SetRateControlMode(
                Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
        h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
        h264Settings.SetMinIInterval(0);
        h264Settings.SetAdaptiveQuantization(
                Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);
        h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
        h264Settings.SetFieldEncoding(
                Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
        h264Settings.SetSceneChangeDetect(
                Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
        h264Settings.SetQualityTuningLevel(
                Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
        h264Settings.SetFramerateConversionAlgorithm(

Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
        h264Settings.SetUnregisteredSeiTimecode(
                Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
        h264Settings.SetGopSizeUnits(
                Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
        h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
        h264Settings.SetFramerateNumerator(30);
        h264Settings.SetFramerateDenominator(1);
        h264Settings.SetParNumerator(1);
        h264Settings.SetParDenominator(1);
        videoCodecSettings.SetH264Settings(h264Settings);
        videoDescription.SetCodecSettings(videoCodecSettings);
```

```
        output.SetVideoDescription(videoDescription);

        Aws::MediaConvert::Model::AudioDescription audioDescription;
        audioDescription.SetLanguageCodeControl(
                Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
        audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
        Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
        audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
        Aws::MediaConvert::Model::AacSettings aacSettings;
        aacSettings.SetAudioDescriptionBroadcasterMix(

Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
        aacSettings.SetRateControlMode(
                Aws::MediaConvert::Model::AacRateControlMode::CBR);
        aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
        aacSettings.SetCodingMode(
                Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
        aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
        aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
        aacSettings.SetBitrate(64000);
        audioCodecSettings.SetAacSettings(aacSettings);
        audioDescription.SetCodecSettings(audioCodecSettings);
        Aws::Vector<Aws::MediaConvert::Model::AudioDescription> audioDescriptions;
        audioDescriptions.emplace_back(audioDescription);
        output.SetAudioDescriptions(audioDescriptions);

        Aws::MediaConvert::Model::ContainerSettings mp4container;
        mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
        Aws::MediaConvert::Model::Mp4Settings mp4Settings;
        mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
        mp4Settings.SetMoovPlacement(
                Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
        mp4container.SetMp4Settings(mp4Settings);
        output.SetContainerSettings(mp4container);

        outputGroup.AddOutputs(output);
        jobSettings.AddOutputGroups(outputGroup);

        // Configure inputs.
        Aws::MediaConvert::Model::Input input;
```

```
        input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
        input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
        input.SetFilterStrength(0);

 input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

 input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
        input.SetTimecodeSource(
                Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
        input.SetFileInput(fileInput);

        Aws::MediaConvert::Model::AudioSelector audioSelector;
        audioSelector.SetOffset(0);
        audioSelector.SetDefaultSelection(
                Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
        audioSelector.SetProgramSelection(1);
        audioSelector.SetSelectorType(
                Aws::MediaConvert::Model::AudioSelectorType::TRACK);
        audioSelector.AddTracks(1);
        input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

        Aws::MediaConvert::Model::VideoSelector videoSelector;
        videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
        input.SetVideoSelector(videoSelector);

        jobSettings.AddInputs(input);

        createJobRequest.SetSettings(jobSettings);
    }

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
    Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(
            createJobRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Job successfully created with ID - "
                  << outcome.GetResult().GetJob().GetId() << std::endl;
    }
    else {
        std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateJob in der AWS SDK für C++ API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the information for a specific completed transcoding job.
/*!
  \param jobID: A job ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
    const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
            request);
    if (outcome.IsSuccess()) {
        std::cout << outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
                  << std::endl;
    }
    else {
        std::cerr << "DescribeEndpoints error - " << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetJob in der AWS SDK für C++ API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of created jobs.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::MediaConvert::listJobs(
        const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;
    Aws::String nextToken; // Used to handle paginated results.
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome = client.ListJobs(
                request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                    outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
```

```
                std::cout << "  " << job.Jsonize().View().WriteReadable() <<
 std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
            break;

        }
    } while (!nextToken.empty());


    return result;
}
```

- API-Details finden Sie ListJobs in der AWS SDK für C++ API-Referenz.

Amazon-RDS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_rds")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_rds.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>
#include <iostream>

/*
 *  A "Hello Rds" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client and
 *  describes the Amazon RDS instances.
 *
 *  main function
 *
 *  Usage: 'hello_rds'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                    rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with RDS::DescribeDBInstances. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << instanceDBIDs.size() << " RDS instances found." << std::endl;
        for (auto &instanceDBID: instanceDBIDs) {
            std::cout << "   Instance: " << instanceDBID << std::endl;
        }
```

```
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für C++

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon RDS instance and demonstrates several operations
//! on that instance.
/*!
 \sa gettingStartedWithDBInstances()
 \param clientConfiguration: AWS client configuration.
```

```
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
              << std::endl;
    std::cout << "get started with DB instances demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB parameter group named '" <<
              PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB parameter group already exists.
        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
```

```
    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available engine versions for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available database engine versions for " << DB_ENGINE
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family << std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB parameter group.
        Aws::RDS::Model::CreateDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example parameter group.");

        Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
                client.CreateDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your parameter group."
              << std::endl;

    Aws::String marker;
    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4.  Get the parameters in the DB parameter group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                         autoIncrementParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&
            (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                      << " is described as: " <<
                      autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                          << autoIncParameter.GetParameterValue()
                          << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                    autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                        Aws::String("Enter a new value in the range ") +
                        autoIncParameter.GetAllowedValues() + ": ",
                        splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);

            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
```

```
                        << std::endl;
            }
        }
    }

    {
        // 5.  Modify the auto increment parameters in the group.
        Aws::RDS::Model::ModifyDBParameterGroupRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
                client.ModifyDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    std::cout
            << "You can get a list of parameters you've set by specifying a source
of 'user'."
            << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6.  Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                         client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << "  " << userParameter.GetParameterName() << ", " <<
                  userParameter.GetDescription() << ", parameter value - "
                  << userParameter.GetParameterValue() << std::endl;
    }
```

```
    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

    Aws::RDS::Model::DBInstance dbInstance;
    // 7.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters): ");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
                                 client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
{
            std::cout << "  " << index << ": " << engineVersion.GetEngineVersion()
                      << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
```

```
                                                                                1];

        Aws::String dbInstanceClass;
        // 9.  Get a list of micro instance classes.
        if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                        engineVersion.GetEngineVersion(),
                                        dbInstanceClass,
                                        client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB instance is configured to use your custom parameter
group '"
                  << PARAMETER_GROUP_NAME << "',\n"
                  << "selected engine version " << engineVersion.GetEngineVersion()
                  << ",\n"
                  << "selected DB instance class '" << dbInstanceClass << "',"
                  << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
                  << " storage.\nThis typically takes several minutes." <<
std::endl;

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
```

```
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    printAsterisksLine();
```

```
   // 12. Display the connection string that can be used to connect a 'mysql' shell
to the database.
   displayConnection(dbInstance);

   printAsterisksLine();

   if (askYesNoQuestion(
           "Do you want to create a snapshot of your DB instance (y/n)? ")) {
       Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
                              Aws::String(Aws::Utils::UUID::RandomUUID()));
       {
           std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
           std::cout << "This typically takes a few minutes." << std::endl;

           // 13. Create a snapshot of the DB instance.
           Aws::RDS::Model::CreateDBSnapshotRequest request;
           request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
           request.SetDBSnapshotIdentifier(snapshotID);

           Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                   client.CreateDBSnapshot(request);

           if (outcome.IsSuccess()) {
               std::cout << "Snapshot creation has started."
                         << std::endl;
           }
           else {
               std::cerr << "Error with RDS::CreateDBSnapshot. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
               return false;
           }
       }

       std::cout << "Waiting for snapshot to become available." << std::endl;

       Aws::RDS::Model::DBSnapshot snapshot;
       counter = 0;
       do {
           std::this_thread::sleep_for(std::chrono::seconds(1));
           ++counter;
```

```
            if (counter > 600) {
                std::cerr << "Wait for snapshot to be available timed out ofter "
                          << counter
                          << " seconds." << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 14. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
```

```
            "Do you want to delete the DB instance and parameter group (y/n)? ")) {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
 client);
    }

    return result;
}


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
```

```
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,
                                      Aws::Vector<Aws::RDS::Model::DBEngineVersion>
 &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }
```

```
    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
```

```
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
```

```
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
{
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database engine
are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
```

```
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        std::cout
                << "Waiting for DB instance to delete before deleting the parameter
 group."
                << std::endl;
```

```
        std::cout << "This may take a while." << std::endl;

        int counter = 0;
        Aws::RDS::Model::DBInstance dbInstance;
        do {
            std::this_thread::sleep_for(std::chrono::seconds(1));
            ++counter;
            if (counter > 800) {
                std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                          << " seconds." << std::endl;
                return false;
            }

            dbInstance = Aws::RDS::Model::DBInstance();
            // 16. Wait for the DB instance to be deleted.
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
            }

            if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) == 0)
{
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (dbInstance.DBInstanceIdentifierHasBeenSet());
    }

    if (!parameterGroupName.empty()) {
        // 17. Delete the parameter group.
        Aws::RDS::Model::DeleteDBParameterGroupRequest request;
        request.SetDBParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
                client.DeleteDBParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                      << outcome.GetError().GetMessage()
```

```
                        << std::endl;
            result = false;
        }
    }

    return result;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBName(DB_NAME);
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        request.SetEngine(engineVersion.GetEngine());
        request.SetEngineVersion(engineVersion.GetEngineVersion());
        request.SetDBInstanceClass(dbInstanceClass);
        request.SetStorageType(DB_STORAGE_TYPE);
        request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für C++ -API-Referenz.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
            client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK für C++ API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBSnapshotRequest request;
            request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
                    client.CreateDBSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with RDS::CreateDBSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für C++ -API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBInstanceRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);
        request.SetSkipFinalSnapshot(true);
        request.SetDeleteAutomatedBackups(true);

        Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB instance deletion has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::DeleteDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für C++ -API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBParameterGroupRequest request;
    request.SetDBParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
            client.DeleteDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK für C++ API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                      const Aws::String &parameterGroupFamily,
                                      Aws::Vector<Aws::RDS::Model::DBEngineVersion>
 &engineVersionsResult,
                                      const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
```

```
            request.SetMarker(marker);
        }


        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
 engineVersions.begin(),
                                        engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

    } while (!marker.empty());


    return true;
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für C++ API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für C++ - API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB parameter group named '" <<
                      PARAMETER_GROUP_NAME << "' already exists." << std::endl;
            dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
```

```
        }
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK für C++ API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                  const Aws::String &namePrefix,
                                  const Aws::String &source,
                                  Aws::Vector<Aws::RDS::Model::Parameter>
 &parametersResult,
                                  const Aws::RDS::RDSClient &client) {
    Aws::String marker;
```

```
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
                client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für C++ - API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBSnapshotsRequest request;
            request.SetDBSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
                    client.DescribeDBSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBSnapshots()[0];
            }
            else {
                std::cerr << "Error with RDS::DescribeDBSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für C++ - API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
```

```
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
 {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                  instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available micro DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which micro DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für C++ API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::ModifyDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
            client.ModifyDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully modified."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für C++ Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon-Rekognition-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Rekognition

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon Rekognition beginnen.

SDK für C++

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
```

```
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_rekognition.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 *  A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
 *  lists the Amazon Rekognition collections in the current account and region.
 *
 *  main function
 *
 *  Usage: 'hello_rekognition'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
                rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
 outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                      << std::endl;
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- API-Details finden Sie ListCollections in der AWS SDK für C++ API-Referenz.

Aktionen

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);
```

```
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DetectLabels in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Amazon-S3-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.
```

```
    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_s3.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        // You don't normally have to test that you are authenticated. But the S3
 service permits anonymous requests, thus the s3Client will return "success" and 0
 buckets even if you are unauthenticated, which can be confusing to a new user.
        auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.

- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
```

```
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param bucketNamePrefix: A prefix for a bucket name.
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
        const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(
```

```
Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                            clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath << "'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
```

```
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath << "'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
```

```
                        << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
```

```
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(), objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" << bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`AbortMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`AbortMultipartUpload`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie AbortMultipartUpload in der AWS SDK für C++ API-Referenz.

`CompleteMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CompleteMultipartUpload`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

 const Aws::String &key,

 const Aws::String &uploadID,

 const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

 const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- API-Details finden Sie CompleteMultipartUpload in der AWS SDK für C++ API-Referenz.

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                            const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " << fromBucket
 <<
                  " to " << toBucket << "." << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CopyObject in der AWS SDK für C++ API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                        clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
```

```
                    " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateBucket in der AWS SDK für C++ API-Referenz.

`CreateMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultipartUpload`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
```

```
    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}
```

- API-Details finden Sie CreateMultipartUpload in der AWS SDK für C++ API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
```

```
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteBucket in der AWS SDK für C++ API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK für C++ API-Referenz.

`DeleteBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketWebsite`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteBucketWebsite in der AWS SDK für C++ API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteObject in der AWS SDK für C++ API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteObjects in der AWS SDK für C++ API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
```

```
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/
```

```
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- API-Details finden Sie GetBucketAcl in der AWS SDK für C++ API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
                  policy_stream.str() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK für C++ API-Referenz.

`GetBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketWebsite`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
```

```
                    << std::endl
                    << "Index page : "
                    << websiteResult.GetIndexDocument().GetSuffix()
                    << std::endl
                    << "Error page: "
                    << websiteResult.GetErrorDocument().GetKey()
                    << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetBucketWebsite in der AWS SDK für C++ API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
```

```
                    err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetObject in der AWS SDK für C++ API-Referenz.

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAcl`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
```

```
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                      << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

```
//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- API-Details finden Sie GetObjectAcl in der AWS SDK für C++ API-Referenz.

`GetObjectAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of the
 object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);
```

```
        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
```

```
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
```

```
                std::cerr << "Error retrieving object attributes for object parts."
 <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

- API-Details finden Sie GetObjectAttributes in der AWS SDK für C++ API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }
```

```
    return result;
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für C++ API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
```

```
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für C++ API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);
```

```
    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
```

```
                    << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- API-Details finden Sie PutBucketAcl in der AWS SDK für C++ API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}


//! Build a policy JSON string.
```

```
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- API-Details finden Sie PutBucketPolicy in der AWS SDK für C++ API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie PutBucketWebsite in der AWS SDK für C++ API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
```

```
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie PutObject in der AWS SDK für C++ API-Referenz.

`PutObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectAcl`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                              const Aws::String &granteePermission, const
 Aws::String &granteeType,
                              const Aws::String &granteeID, const Aws::String
 &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
```

```
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
            s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- API-Details finden Sie PutObjectAcl in der AWS SDK für C++ API-Referenz.

`UploadPart`

Das folgende Codebeispiel zeigt die Verwendung`UploadPart`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
 algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
 &calculatedHash,
                                                         const
 std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
```

```
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- API-Details finden Sie UploadPart in der AWS SDK für C++ API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, um ein Objekt herunterzuladen.

```
//! Routine which demonstrates creating a pre-signed URL to download an object from
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

Führen Sie einen Download unter Verwendung von libcurl durch.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
```

```
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();
```

```
    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

Generieren Sie eine vorsignierte URL, um ein Objekt hochzuladen.

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

Führen Sie einen Upload unter Verwendung von libcurl durch.

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}
```

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }
```

```
    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
```

```
                    << std::endl;
        return false;
    }
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Arbeiten mit Amazon-S3-Objektintegrität

Das folgende Codebeispiel zeigt, wie Sie mit Features der S3-Objektintegrität arbeiten.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Objektintegrität zeigt.

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
 std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
 std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
```

```
   std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

   printAsterisksLine();
   std::cout << "Choose from one of the following checksum algorithms."
             << std::endl;

   for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
       std::cout << "  " << hashMethod << " - " << stringForHashMethod(hashMethod)
                 << std::endl;
   }

   HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                         SHA256);


   gUseCalculatedChecksum = !askYesNoQuestion(
           "Let the SDK calculate the checksum for you? (y/n) ");

   printAsterisksLine();

   std::cout << "The workflow will now upload a file using PutObject."
             << std::endl;
   std::cout << "Object integrity will be verified using the "
             << stringForHashMethod(chosenHashMethod) << " algorithm."
             << std::endl;
   if (gUseCalculatedChecksum) {
       std::cout
               << "A checksum computed by this workflow will be used for object
integrity verification,"
               << std::endl;
       std::cout << "except for the TransferManager upload." << std::endl;
   } else {
       std::cout
               << "A checksum computed by the SDK will be used for object integrity
verification."
               << std::endl;
   }

   pressEnterToContinue();
   printAsterisksLine();

   std::shared_ptr<Aws::IOStream> inputData =
```

```
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
```

```
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                 client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;
```

```
   // Retrieve all the hashes for the TransferManager upload.
   if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
       std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   AwsDoc::S3::Hasher locallyCalculatedFinalHash;
   std::vector<Aws::String> locallyCalculatedPartHashes;

   // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
   if (!calculatePartHashesForFile(transferManagerHashMethod, MULTI_PART_TEST_FILE,
                                   UPLOAD_BUFFER_SIZE,
                                   locallyCalculatedFinalHash,
                                   locallyCalculatedPartHashes)) {
       std::cerr << "Exiting because of an error in calculatePartHashesForFile." <<
std::endl;
       cleanUp(bucketName, clientConfiguration);
       return false;
   }

   verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

   printAsterisksLine();

   key = "mp_";
   key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

   multiPartUploadExplanations(key, chosenHashMethod);

   pressEnterToContinue();

   std::shared_ptr<Aws::IOStream> largeFileInputData =
           Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                         MULTI_PART_TEST_FILE,
```

```
                                        std::ios_base::in |
                                        std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5;  // The default hash method for multipart
uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
              stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." << std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
```

```
        std::cerr << "Exiting because of an error in retrieveObjectHash for
 multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
 workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object integrity
 hashing methods.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded object.
   \param hashMethod: The hashing algorithm to use when calculating the hash value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
```

```
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

 request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
```

```
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of the
 object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
```

```
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
 result.GetObjectParts().GetParts();
```

```
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
 NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object parts."
 <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual parts
 retrieved from the remote source.
```

```
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                       const Hasher &localHash,
                                       const Aws::String &uploadtype,
                                       HASH_METHOD hashMethod,
                                       const std::vector<Aws::String>
 &retrievedPartHashes,
                                       const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash <<
 std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
 retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
 std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do not
 match" << std::endl;
```

```
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
 "'"
                      << std::endl;
            std::cerr << "Remote number of hashes - '"
                      << retrievedPartHashes.size()
                      << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not match
 for part " << i + 1
                          << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                          << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                          << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
                                         const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" << std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                          const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
```

```
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's Transfer
 Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    bool useDefaultHashMethod,
                                    const std::shared_ptr<Aws::S3::S3Client>
 &client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm = getChecksumAlgorithmForHashMethod(
                    hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback = transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
            transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle = transfer_manager-
>UploadFile(MULTI_PART_TEST_FILE,
```

```
            bucket, key,

            "text/plain",

            Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
            uploadHandle->GetStatus() == Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle->GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being uploaded
 to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
   \param[out] partHashes: The vector that will store the calculated hash values for
 each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes) {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
```

```
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize - uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
```

```
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
 algorithm,
                            ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
```

```
                                                                const Aws::String
 &calculatedHash,
                                                                const
 std::shared_ptr<Aws::IOStream> &body,
                                                                const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
```

```
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

 const Aws::String &key,

 const Aws::String &uploadID,

 const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,
```

```
 const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {
```

```
   // Get object size.
   ioStream->seekg(0, ioStream->end);
   size_t objectSize = ioStream->tellg();
   ioStream->seekg(0, ioStream->beg);

   Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
   if (!useDefaultHashMethod) {
       if (hashMethod != MD5) {
           checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
       }
   }
   Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
   if (uploadID.empty()) {
       return false;
   }

   std::vector<unsigned char> totalHashBuffer;
   bool uploadSucceeded = true;
   std::streamsize uploadedBytes = 0;
   int partNumber = 1;
   Aws::Vector<Aws::S3::Model::CompletedPart> parts;
   while (uploadedBytes < objectSize) {
       std::cout << "Uploading part " << partNumber << "." << std::endl;

       std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
       std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
       ioStream->read((char *) buffer.data(), bytesToRead);
       Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
       std::shared_ptr<Aws::IOStream> body =
               Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                              &preallocatedStreamBuf);

       Hasher hasher;
       if (!hasher.calculateObjectHash(*body, hashMethod)) {
           std::cerr << "Error calculating hash." << std::endl;
           uploadSucceeded = false;
```

```
            break;
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        Aws::String calculatedHash;
        if (gUseCalculatedChecksum) {
            calculatedHash = base64HashString;
        }
        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,
                                                                        client);
        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:
```

```
completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }

            parts.push_back(completedPart);
        } else {
            std::cerr << "Error uploading part. " <<
                      uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                 key,

                                 uploadID,

                                 parts,

                                 client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod)) {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
```

```
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
```

```
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not valid."
 << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration &clientConfiguration)
 {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
        }
    } else {
```

```
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
            << std::endl;
    printAsterisksLine();
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an object.
\n"
            << "The default checksum algorithm for PutObject and MultiPart upload is
 an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting the
 SDK calculate the value."
              << std::endl;
    std::cout
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;

    std::cout
```

```
            << "This workflow will locally compute checksums for files uploaded to
 an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums are
 calculated."
            << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using the
 GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
            << std::endl;
    std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
```

```
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when uploading
 large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally and
 then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part. The
 final checksum is a concatenation of"
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
```

```
            << " in the section \"Using part-level checksums for multipart uploads
\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
 file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                   | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();
```

```
    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

Secrets-Manager-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Secrets Manager Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
  \param secretID: The ID for the secret.
  \return bool: Function succeeded.
 */
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
 secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
 secretsManagerClient.GetSecretValue(
            request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                  << getSecretValueOutcome.GetResult().GetSecretString() <<
 std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                  << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
```

```
}
```

- API-Details finden Sie GetSecretValue in der AWS SDK für C++ API-Referenz.

Amazon-SES-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptFilter`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..
```

```
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
 (CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
                                      Aws::SES::Model::ReceiptFilterPolicy policy,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
 sesClient.CreateReceiptFilter(
            createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
                  createReceiptFilterOutcome.GetError().GetMessage() << std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}
```

- API-Details finden Sie CreateReceiptFilter in der AWS SDK für C++ API-Referenz.

`CreateReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRule`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param s3BucketName: The name of the S3 bucket for incoming mail.
  \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
  \param ruleSetName: The name of the rule set where the receipt rule is added.
  \param recipients: Aws::Vector of recipients.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);

    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
```

```
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateReceiptRule in der AWS SDK für C++ API-Referenz.

`CreateReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRuleSet`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
 sesClient.CreateReceiptRuleSet(
            createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateReceiptRuleSet in der AWS SDK für C++ API-Referenz.

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateTemplate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Email Service (Amazon SES) template.
```

```
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;

    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
            createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
    else {
        std::cerr << "Error creating template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateTemplate in der AWS SDK für C++ API-Referenz.

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentity`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete the specified identity (an email address or a domain).
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
            deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteIdentity in der AWS SDK für C++ API-Referenz.

`DeleteReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptFilter`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
 sesClient.DeleteReceiptFilter(
            deleteReceiptFilterRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error deleting receipt filter. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteReceiptFilter in der AWS SDK für C++ API-Referenz.

`DeleteReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRule`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &receiptRuleSetName,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome = sesClient.DeleteReceiptRule(
            deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
```

```
    else {
        std::cout << "Error deleting receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteReceiptRule in der AWS SDK für C++ API-Referenz.

`DeleteReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRuleSet`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
 sesClient.DeleteReceiptRuleSet(
```

```
            deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteReceiptRuleSet in der AWS SDK für C++ API-Referenz.

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTemplate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;
```

```
    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
            deleteTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteTemplate in der AWS SDK für C++ API-Referenz.

`GetTemplate`

Das folgende Codebeispiel zeigt die Verwendung`GetTemplate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Get a template's attributes.
/*!
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
            getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetTemplate in der AWS SDK für C++ API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the identities associated with this account.
/*!
  \param identityType: The identity type enum. "NOT_SET" is a valid option.
  \param identities; A vector to receive the retrieved identities.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                 Aws::Vector<Aws::String> &identities,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome = sesClient.ListIdentities(
                listIdentitiesRequest);

        if (outcome.IsSuccess()) {
            const auto &retrievedIdentities = outcome.GetResult().GetIdentities();
            if (!retrievedIdentities.empty()) {
                identities.insert(identities.cend(), retrievedIdentities.cbegin(),
                                  retrievedIdentities.cend());
            }
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error listing identities. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    return true;
}
```

- API-Details finden Sie ListIdentities in der AWS SDK für C++ API-Referenz.

`ListReceiptFilters`

Das folgende Codebeispiel zeigt die Verwendung`ListReceiptFilters`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! List the receipt filters associated with this account.
/*!
  \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
 &filters,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
 sesClient.ListReceiptFilters(
            listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie ListReceiptFilters in der AWS SDK für C++ API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send an email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param subject: Email subject.
  \param htmlBody: Email body as HTML. At least one body data is required.
  \param textBody: Email body as plain text. At least one body data is required.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                            const Aws::String &subject,
                            const Aws::String &htmlBody,
                            const Aws::String &textBody,
                            const Aws::String &senderEmailAddress,
                            const Aws::Vector<Aws::String> &ccAddresses,
                            const Aws::String &replyToAddress,
                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
```

```
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::Body message_body;
    if (!htmlBody.empty()) {
        message_body.SetHtml(
                Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
    }

    if (!textBody.empty()) {
        message_body.SetText(
                Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
    }

    Aws::SES::Model::Message message;
    message.SetBody(message_body);
    message.SetSubject(
            Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

    Aws::SES::Model::SendEmailRequest sendEmailRequest;
    sendEmailRequest.SetDestination(destination);
    sendEmailRequest.SetMessage(message);
    if (!senderEmailAddress.empty()) {
        sendEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendEmail(sendEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- API-Details finden Sie SendEmail in der AWS SDK für C++ API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send a templated email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
 &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
```

```
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "\":\"" << pair.second << "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- API-Details finden Sie SendTemplatedEmail in der AWS SDK für C++ API-Referenz.

`UpdateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTemplate`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
```

```
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
 sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
    } else {
        std::cerr << "Error updating template. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie UpdateTemplate in der AWS SDK für C++ API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Add an email address to the list of identities associated with this account and
//! initiate verification.
/*!
  \param emailAddress; The email address to add.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                         const Aws::Client::ClientConfiguration
 &clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);
```

```
    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
 sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }

    else
    {
        std::cerr << "Error initiating email verification. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie VerifyEmailIdentity in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon-SNS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SNS beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.
```

```
    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_sns.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
```

```
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << " topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
```

```
}
```

- API-Details finden Sie ListTopics in der AWS SDK für C++ API-Referenz.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
```

```
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateTopic in der AWS SDK für C++ API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);
```

```
    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteTopic in der AWS SDK für C++ API-Referenz.

`GetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetSMSAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the default settings for sending SMS messages from your AWS account by
 using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
```

```
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für C++ -API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK für C++ API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
```

```
                                newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- API-Details finden Sie ListSubscriptions in der AWS SDK für C++ API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " << outcome.GetError().GetMessage()
 <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- API-Details finden Sie ListTopics in der AWS SDK für C++ API-Referenz.

Veröffentlichen

Das folgende Codebeispiel zeigt die VerwendungPublish.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                        "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
```

```
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

`SetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSMSAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

So verwenden Sie Amazon SNS zum Festlegen des DefaultSMSType-Attributs.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
```

```
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für C++ -API-Referenz.

`Abonnieren`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
```

```
                            const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
```

```
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
```

```
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

Abonnieren Sie ein Thema mit einem Filter.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
```

```
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
                                << "will be added to this subscription." <<
 std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
```

```
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
```

```
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für C++ -API-Referenz.

Abonnement beenden

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für C++ -API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an SMS
 text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is
 in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction that
 you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
```

```
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
```

```
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication and
message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or automatically
generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic, any
message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
```

```
                << "within the five-minute deduplication interval, is accepted but
 not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a deduplication
 ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);
```

```
        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " << std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

  request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
```

```
                                        "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is "
```

```
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
```

```
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
```

```
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }
```

```
    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received in
the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
```

```
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
```

```
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
```

```
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }
```

```
    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
```

```
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}
```

```
//! Create an IAM policy that gives an SQS queue permission to receive messages from
 an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Amazon-SQS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für C++

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
 for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
 running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
 need to uncomment this
    # and set the proper subdirectory to the executables' location.
```

```
    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
        hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei „hello_sqs.cpp“.

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 *  A "Hello SQS" starter application that initializes an Amazon Simple Queue
 Service
 *  (Amazon SQS) client and lists the SQS queues in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sqs'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SQS::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
```

```
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SQS::Model::ListQueuesRequest request;

            Aws::SQS::Model::ListQueuesOutcome outcome =
 sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &pageOfQueueUrls =
 outcome.GetResult().GetQueueUrls();
                if (!pageOfQueueUrls.empty()) {
                    allQueueUrls.insert(allQueueUrls.cend(),
 pageOfQueueUrls.cbegin(),
                                        pageOfQueueUrls.cend());
                }
            }
            else {
                std::cerr << "Error with SQS::ListQueues. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());


        std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
 queue"
                  << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allQueueUrls.empty()) {
            std::cout << "Here are your queue URLs." << std::endl;
            for (const Aws::String &queueUrl: allQueueUrls) {
                std::cout << "  * " << queueUrl << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- API-Details finden Sie ListQueues in der AWS SDK für C++ API-Referenz.

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung`ChangeMessageVisibility`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Changes the visibility timeout of a message in an Amazon Simple Queue Service
//! (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageReceiptHandle: A message receipt handle.
  \param visibilityTimeoutSeconds: Visibility timeout in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ChangeMessageVisibilityRequest request;
    request.SetQueueUrl(queue_url);
    request.SetReceiptHandle(messageReceiptHandle);
    request.SetVisibilityTimeout(visibilityTimeoutSeconds);
```

```
    auto outcome = sqsClient.ChangeMessageVisibility(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully changed visibility of message " <<
                  messageReceiptHandle << " from queue " << queue_url << std::endl;
    }
    else {
        std::cout << "Error changing visibility of message from queue "
                  << queue_url << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie ChangeMessageVisibility in der AWS SDK für C++ API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Create an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::CreateQueueRequest request;
    request.SetQueueName(queueName);

    const Aws::SQS::Model::CreateQueueOutcome outcome =
 sqsClient.CreateQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created queue " << queueName << " with a queue
 URL "
                  << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie CreateQueue in der AWS SDK für C++ API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
```

```
  \param messageReceiptHandle: A message receipt handle.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                const Aws::String &messageReceiptHandle,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
                  << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteMessage in der AWS SDK für C++ API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK für C++ API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueURL: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::SQS::Model::DeleteQueueRequest request;
    request.SetQueueUrl(queueURL);

    const Aws::SQS::Model::DeleteQueueOutcome outcome =
 sqsClient.DeleteQueue(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted queue with url " << queueURL <<
                  std::endl;
    }
    else {
        std::cerr << "Error deleting queue " << queueURL << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- API-Details finden Sie DeleteQueue in der AWS SDK für C++ API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }

            }
```

```
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;


            }
```

- API-Details finden Sie GetQueueAttributes in der AWS SDK für C++ API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueName: An Amazon SQS queue name.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueUrlRequest request;
    request.SetQueueName(queueName);
```

```
    const Aws::SQS::Model::GetQueueUrlOutcome outcome =
 sqsClient.GetQueueUrl(request);
    if (outcome.IsSuccess()) {
        std::cout << "Queue " << queueName << " has url " <<
                  outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie GetQueueUrl in der AWS SDK für C++ API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS account.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
    Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
                listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
 outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}
```

- API-Details finden Sie ListQueues in der AWS SDK für C++ API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::ReceiveMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMaxNumberOfMessages(1);

    const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
            request);
    if (outcome.IsSuccess()) {

        const Aws::Vector<Aws::SQS::Model::Message> &messages =
                outcome.GetResult().GetMessages();
        if (!messages.empty()) {
            const Aws::SQS::Model::Message &message = messages[0];
            std::cout << "Received message:" << std::endl;
            std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
            std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
 std::endl;
            std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
        }
        else {
```

```
            std::cout << "No messages received from queue " << queueUrl <<
                      std::endl;

        }
    }
    else {
        std::cerr << "Error receiving message from queue " << queueUrl << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für C++ API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param messageBody: A message body.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                              const Aws::String &messageBody,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
 sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
                  std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie SendMessage in der AWS SDK für C++ API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
 queue.
```

```
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param attributeName: An attribute name enum.
  \param attribute: The attribute value as a string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
 attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            attributeName,
            attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute  " <<

 Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
                         attributeName)
                  << " with value " << attribute << " in queue " <<
                  queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for  queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

Konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
//! dead-letter queue.
/*!
  \param srcQueueUrl: An Amazon SQS queue URL.
  \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS dead-
letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                     const Aws::String &deadLetterQueueARN,
                                     int maxReceiveCount,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
 maxReceiveCount);

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(srcQueueUrl);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set dead letter queue for queue  " <<
                  srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
    }
    else {
        std::cerr << "Error setting dead letter queue for queue " <<
                  srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
  \param queueArn: An Amazon SQS ARN for the dead-letter queue.
  \param maxReceiveCount: The max receive count of a message before it is sent to
 the dead-letter queue.
  \return Aws::String: Policy as JSON string.
 */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}
```

Konfigurieren Sie eine Amazon-SQS-Warteschlange für die Verwendung einer Langabfrage.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
/*!
  \param queueUrl: An Amazon SQS queue URL.
  \param pollTimeSeconds: The receive message wait time in seconds.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                               const Aws::String &pollTimeSeconds,
                                               const
 Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```
    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
            pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
 sqsClient.SetQueueAttributes(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
                  queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
                  queueURL << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK für C++ API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the queues."
              << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication and
 message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
```

```
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or automatically
 generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic, any
 message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted but
 not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a deduplication
 ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;
```

```
        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " << std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
```

```
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." << std::endl;
            }
            else {
```

```
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is "
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
```

```
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
```

```
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then only
 the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
```

```
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN << "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
```

```
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received in
the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
```

```
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

```
        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);
```

```
            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);
```

```
        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages from
 an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                           const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
```

```
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

AWS STS Beispiele mit SDK für C++

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
```

```
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- API-Details finden Sie AssumeRole in der AWS SDK für C++ API-Referenz.

Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Streaming Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`StartStreamTranscription`

Das folgende Codebeispiel zeigt die Verwendung. `StartStreamTranscription`

SDK für C++

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
 SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                []( const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
```

```
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
```

```
                    //the number of characters extracted by the last read()
 operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
 std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" << std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };
```

```
        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK für C++ API-Referenz.

Szenarien

Transkribieren einer Audiodatei

Das folgende Code-Beispiel zeigt, wie mit Amazon Transcribe-Streaming eine Transkription einer Quell-Audiodatei generiert wird.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;
```

```
        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
 managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
 SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
                []( const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
 &error) {
                        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
                });
        //SetTranscriptEventCallback called for every 'chunk' of file transcripted.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
                for (auto &&r: ev.GetTranscript().GetResults()) {
                    if (r.GetIsPartial()) {
                        std::cout << "[partial] ";
                    }
                    else {
                        std::cout << "[Final] ";
                    }
                    for (auto &&alt: r.GetAlternatives()) {
                        std::cout << alt.GetTranscript() << std::endl;
                    }
                }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
                MediaEncoding::pcm); // wav and aiff files are PCM formats.
```

```
        request.SetEventStreamHandler(handler);

        auto OnStreamReady = [](AudioStream &stream) {
                Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
                if (!file.is_open()) {
                    std::cerr << "Failed to open " << FILE_NAME << '\n';
                }
                std::array<char, BUFFER_SIZE> buf;
                int i = 0;
                while (file) {
                    file.read(&buf[0], buf.size());

                    if (!file)
                        std::cout << "File: only " << file.gcount() << " could be
read"
                                  << std::endl;

                    Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
                    AudioEvent event(std::move(bits));
                    if (!stream) {
                        std::cerr << "Failed to create a stream" << std::endl;
                        break;
                    }
                    //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
                    //the number of characters extracted by the last read()
operation.
                    if (file.gcount() > 0) {
                        if (!stream.WriteAudioEvent(event)) {
                            std::cerr << "Failed to write an audio event" <<
std::endl;
                            break;
                        }
                    }
                    else {
                        break;
                    }
                    std::this_thread::sleep_for(std::chrono::milliseconds(
                            25)); // Slow down because we are streaming from a file.
                }
                if (!stream.WriteAudioEvent(
                        AudioEvent())) {
```

```
                    // Per the spec, we have to send an empty event (an event
 without a payload) at the end.
                    std::cerr << "Failed to send an empty frame" << std::endl;
                }
                else {
                    std::cout << "Successfully sent the empty frame" << std::endl;
                }
                stream.flush();
                stream.Close();
        };

        Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
        auto OnResponseCallback = [&signaling](
                const TranscribeStreamingServiceClient * /*unused*/,
                const Model::StartStreamTranscriptionRequest & /*unused*/,
                const Model::StartStreamTranscriptionOutcome &outcome,
                const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

                if (!outcome.IsSuccess()) {
                    std::cerr << "Transcribe streaming error "
                              << outcome.GetError().GetMessage() << std::endl;
                }

                signaling.Release();
        };

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
 OnResponseCallback,
                                             nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
 done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK für C++ API-Referenz.

Codebeispiele für SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Go V2 mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for Go V2 Entwicklerhandbuch — Mehr über die Verwendung von Go with AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- API-Gateway-Beispiele unter Verwendung von SDK für Go V2
- Aurora-Beispiele unter Verwendung von SDK für Go V2
- Codebeispiele für Amazon Bedrock unter Verwendung von SDK für Go V2
- Codebeispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Go V2
- CloudFormation Beispiele mit SDK für Go V2
- CloudWatch Protokolliert Beispiele mithilfe des SDK für Go V2
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Go V2
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Go V2
- DynamoDB-Beispiele unter Verwendung von SDK für Go V2
- IAM-Beispiele unter Verwendung von SDK für Go V2
- Kinesis-Beispiele unter Verwendung von SDK für Go V2

- Lambda-Beispiele unter Verwendung von SDK für Go V2
- Beispiele für Amazon MSK unter Verwendung von SDK für Go V2
- Beispiele für Partner Central unter Verwendung von SDK für Go V2
- Beispiele für Amazon RDS unter Verwendung von SDK für Go V2
- Beispiele für Amazon Redshift unter Verwendung von SDK für Go V2
- Beispiele für Amazon S3 unter Verwendung von SDK für Go V2
- Beispiele für Amazon SNS unter Verwendung von SDK für Go V2
- Beispiele für Amazon SQS unter Verwendung von SDK für Go V2

API-Gateway-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit API Gateway verwenden.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden AWS. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- AWS Beiträge der Gemeinschaft

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Go V2

> Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Aurora-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Go V2 mit Aurora Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up to
 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
```

```
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für Go -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service (Amazon
 RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
```

```
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB cluster parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
```

```
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
 families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName, "")
```

```
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName, updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source of
 'user'.")
 userParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a database
// of a specified type. The database is also configured to use a custom DB cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {
```

```
 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?\n",
 engineChoices)
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName, dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
```

```
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
 if err != nil {
  panic(err)
 }
 if dbInstance == nil {
  log.Println("Let's create a database instance in your DB cluster.")
  log.Println("First, choose a DB instance type:")
  instOpts, err := scenario.dbClusters.GetOrderableInstances(
   ctx, *cluster.Engine, *cluster.EngineVersion)
  if err != nil {
   panic(err)
  }
  var instChoices []string
  for _, opt := range instOpts {
   instChoices = append(instChoices, *opt.DBInstanceClass)
  }
  slices.Sort(instChoices)
  instChoices = slices.Compact(instChoices)
  instIndex := scenario.questioner.AskChoice(
   "Which DB instance class do you want to use?\n", instChoices)
  log.Println("Creating a database instance. This typically takes several minutes.")
  dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
   ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier, *cluster.Engine,
```

```
    instChoices[instIndex])
   if err != nil {
    panic(err)
   }
   for *dbInstance.DBInstanceStatus != "available" {
    scenario.helper.Pause(30)
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
  *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Println("Instance data:")
  log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
  log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
  log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
  log.Printf("\tEngine: %v\n", *dbInstance.Engine)
  log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
  log.Println(strings.Repeat("-", 88))
  return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster and
 tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
  log.Println(
   "You can now connect to your database using your favorite MySql client.\n" +
    "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
    "that is running in the same VPC as your database cluster. Pass the endpoint,\n"
  +
    "port, and administrator user name to 'mysql' and enter your password\n" +
    "when prompted:")
  log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
   *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
  log.Println("For more information, see the User Guide for Aurora:\n" +
   "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Co
  log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
```

```
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context, clusterName
 string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
 snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster parameter
 group.
// Before the DB cluster parameter group can be deleted, all associated DB instances
 and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
```

```
  log.Printf("Deleting database instance %v.\n", *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx, *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
```

```
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
```

```
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
```

```
    DBClusterParameterGroupName: aws.String(parameterGroupName),
   })
  if err != nil {
   log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
   return err
  } else {
   return nil
  }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
 string, params []types.Parameter) error {
```

```
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified engine
 and
// engine version.
```

```
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
 parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
```

```
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
 string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
 string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
```

```
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
```

```
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified engine
 and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
 parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {
```

```
 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK für Go -API-Referenz.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK für Go API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
 string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK für Go API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
 string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
```

```
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für Go -API-Referenz.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
```

```
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK für Go -API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK für Go API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
 error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für Go -API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
```

```
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK für Go API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
```

```
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK für Go API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS SDK für Go API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
```

```
    err = nil
   } else {
    log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
   }
   return nil, err
  } else {
   return &output.DBClusters[0], err
  }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für Go -API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für Go API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Go -API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
```

```
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für Go API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
```

```
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
 string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK für Go API-Referenz.

Codebeispiele für Amazon Bedrock unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon Bedrock verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock beginnen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrock"
)

const region = "us-east-1"

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock client and
// list the available foundation models in your account and the chosen region.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(region))
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 bedrockClient := bedrock.NewFromConfig(sdkConfig)
 result, err := bedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})
```

```
 if err != nil {
  fmt.Printf("Couldn't list foundation models. Here's why: %v\n", err)
  return
 }
 if len(result.ModelSummaries) == 0 {
  fmt.Println("There are no foundation models.")
 }
 for _, modelSummary := range result.ModelSummaries {
  fmt.Println(*modelSummary.ModelId)
 }
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für Go API-Referenz.

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Bedrock-Basismodelle.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/bedrock"
 "github.com/aws/aws-sdk-go-v2/service/bedrock/types"
)

// FoundationModelWrapper encapsulates Amazon Bedrock actions used in the examples.
```

```
// It contains a Bedrock service client that is used to perform foundation model
 actions.
type FoundationModelWrapper struct {
 BedrockClient *bedrock.Client
}



// ListPolicies lists Bedrock foundation models that you can use.
func (wrapper FoundationModelWrapper) ListFoundationModels(ctx context.Context)
 ([]types.FoundationModelSummary, error) {

 var models []types.FoundationModelSummary

 result, err := wrapper.BedrockClient.ListFoundationModels(ctx,
 &bedrock.ListFoundationModelsInput{})

 if err != nil {
  log.Printf("Couldn't list foundation models. Here's why: %v\n", err)
 } else {
  models = result.ModelSummaries
 }
 return models, err
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für Go API-Referenz.

Codebeispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon Bedrock Runtime verwenden.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Amazon Nova

- Amazon Titan Image Generator
- Anthropic Claude

Erste Schritte

Hallo Amazon Bedrock Runtime

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock Runtime beginnen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "flag"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock Runtime client
// and invokes Anthropic Claude using the Converse API.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {

 region := flag.String("region", "us-east-1", "The AWS region")
 flag.Parse()
```

```
fmt.Printf("Using AWS region: %s\n", *region)

ctx := context.Background()
sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(*region))
if err != nil {
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}

client := bedrockruntime.NewFromConfig(sdkConfig)

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
modelId := "global.anthropic.claude-haiku-4-5-20251001-v1:0"

// Start a conversation with the user message.
prompt := "Hello. In a short paragraph, explain what you can do."

message := types.Message{
 Content: []types.ContentBlock{
  &types.ContentBlockMemberText{Value: prompt},
 },
 Role: types.ConversationRoleUser,
}

fmt.Printf("Model: %s\n", modelId)
fmt.Printf("Prompt: %s\n\n", prompt)

result, err := client.Converse(ctx, &bedrockruntime.ConverseInput{
 ModelId:  aws.String(modelId),
 Messages: []types.Message{message},
})

if err != nil {
 log.Fatalf("ERROR: Can't invoke '%s'. Reason: %v\n", modelId, err)
}

// Extract and print the response text.
responseMessage, ok := result.Output.(*types.ConverseOutputMemberMessage)
if !ok {
 log.Fatal("ERROR: Unexpected output type from Converse API")
```

```
 }
 responseContentBlock := responseMessage.Value.Content[0]
 text, ok := responseContentBlock.(*types.ContentBlockMemberText)
 if !ok {
  log.Fatal("ERROR: Unexpected content block type from Converse API")
 }
 fmt.Printf("Response: %s\n", text.Value)
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
func (wrapper ConverseWrapper) ConverseNova(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
 modelId := "amazon.nova-lite-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
```

```
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

Amazon Titan Image Generator

InvokeModel

Die folgenden Codebeispiele zeigen, wie Sie Amazon Titan Image in Amazon Bedrock aufrufen, um ein Bild zu generieren.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
import (
 "context"
 "encoding/json"
 "log"
 "strings"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



type TitanImageRequest struct {
 TaskType              string                `json:"taskType"`
 TextToImageParams     TextToImageParams     `json:"textToImageParams"`
 ImageGenerationConfig ImageGenerationConfig `json:"imageGenerationConfig"`
}
type TextToImageParams struct {
 Text string `json:"text"`
}
type ImageGenerationConfig struct {
 NumberOfImages int     `json:"numberOfImages"`
 Quality        string  `json:"quality"`
 CfgScale       float64 `json:"cfgScale"`
 Height         int     `json:"height"`
 Width          int     `json:"width"`
 Seed           int64   `json:"seed"`
}

type TitanImageResponse struct {
 Images []string `json:"images"`
}

// Invokes the Titan Image model to create an image using the input provided
// in the request body.
func (wrapper InvokeModelWrapper) InvokeTitanImage(ctx context.Context, prompt
 string, seed int64) (string, error) {
 modelId := "amazon.titan-image-generator-v2:0"

 body, err := json.Marshal(TitanImageRequest{
  TaskType: "TEXT_IMAGE",
  TextToImageParams: TextToImageParams{
   Text: prompt,
  },
```

```
  ImageGenerationConfig: ImageGenerationConfig{
   NumberOfImages: 1,
   Quality:        "standard",
   CfgScale:       8.0,
   Height:         512,
   Width:          512,
   Seed:           seed,
  },
 })

 if err != nil {
  log.Fatal("failed to marshal", err)
 }

 output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
 &bedrockruntime.InvokeModelInput{
  ModelId:     aws.String(modelId),
  ContentType: aws.String("application/json"),
  Body:        body,
 })

 if err != nil {
  ProcessError(err, modelId)
 }

 var response TitanImageResponse
 if err := json.Unmarshal(output.Body, &response); err != nil {
  log.Fatal("failed to unmarshal", err)
 }

 base64ImageData := response.Images[0]

 return base64ImageData, nil

}
```

- API-Details finden Sie InvokeModel in der AWS SDK für Go API-Referenz.

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
import (
 "context"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
 "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// ConverseWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke Bedrock.
type ConverseWrapper struct {
 BedrockRuntimeClient *bedrockruntime.Client
}



func (wrapper ConverseWrapper) ConverseClaude(ctx context.Context, prompt string)
 (string, error) {
 var content = types.ContentBlockMemberText{
  Value: prompt,
 }
 var message = types.Message{
  Content: []types.ContentBlock{&content},
  Role:    "user",
 }
```

```
 modelId := "anthropic.claude-3-haiku-20240307-v1:0"
 var converseInput = bedrockruntime.ConverseInput{
  ModelId:  aws.String(modelId),
  Messages: []types.Message{message},
 }
 response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
 if err != nil {
  ProcessError(err, modelId)
 }

 responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
 responseContentBlock := responseText.Value.Content[0]
 text, _ := responseContentBlock.(*types.ContentBlockMemberText)
 return text.Value, nil

}
```

- Details zur API finden Sie unter Converse in der AWS SDK für Go -API-Referenz.

CloudFormation Beispiele mit SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS SDK für Go V2 mit verwenden CloudFormation.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeStacks`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStacks`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

- API-Details finden Sie DescribeStacks in der AWS SDK für Go API-Referenz.

CloudWatch Protokolliert Beispiele mithilfe des SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Go V2 mit CloudWatch Protokollen Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für Go V2

Binden Sie die erforderlichen Dateien ein.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {
 eventsChan := stream.Events()
 for {
  event := <-eventsChan
  switch e := event.(type) {
  case *types.StartLiveTailResponseStreamMemberSessionStart:
   log.Println("Received SessionStart event")
  case *types.StartLiveTailResponseStreamMemberSessionUpdate:
   for _, logEvent := range e.Value.SessionResults {
    log.Println(*logEvent.Message)
   }
  default:
   // Handle on-stream exceptions
   if err := stream.Err(); err != nil {
    log.Fatalf("Error occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
    return
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Starten Sie die Live-Tail-Sitzung.

```
 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error, " + err.Error())
 }
 client := cloudwatchlogs.NewFromConfig(cfg)

 request := &cloudwatchlogs.StartLiveTailInput{
  LogGroupIdentifiers:   logGroupIdentifiers,
  LogStreamNames:        logStreamNames,
  LogEventFilterPattern: logEventFilterPattern,
 }

 response, err := client.StartLiveTail(context.TODO(), request)
```

```
// Handle pre-stream Exceptions
if err != nil {
 log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- API-Details finden Sie StartLiveTail in der AWS SDK für Go API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon Cognito Identity Provider verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
```

```
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API-Details finden Sie ListUserPools in der AWS SDK für Go API-Referenz.

Aktionen

`AdminCreateUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminCreateUser`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- API-Details finden Sie AdminCreateUser in der AWS SDK für Go API-Referenz.

`AdminSetUserPassword`

Das folgende Codebeispiel zeigt die Verwendung`AdminSetUserPassword`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- API-Details finden Sie AdminSetUserPassword in der AWS SDK für Go API-Referenz.

`ConfirmForgotPassword`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmForgotPassword`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- API-Details finden Sie ConfirmForgotPassword in der AWS SDK für Go API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- API-Details finden Sie DeleteUser in der AWS SDK für Go API-Referenz.

`ForgotPassword`

Das folgende Codebeispiel zeigt die Verwendung`ForgotPassword`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- API-Details finden Sie ForgotPassword in der AWS SDK für Go API-Referenz.

`InitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`InitiateAuth`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
```

```
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- API-Details finden Sie InitiateAuth in der AWS SDK für Go API-Referenz.

`ListUserPools`

Das folgende Codebeispiel zeigt die Verwendung`ListUserPools`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- API-Details finden Sie ListUserPools in der AWS SDK für Go API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
```

```
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- API-Details finden Sie SignUp in der AWS SDK für Go API-Referenz.

`UpdateUserPool`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUserPool`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
```

```
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- API-Details finden Sie UpdateUserPool in der AWS SDK für Go API-Referenz.

Szenarien

Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie bekannte Amazon-Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
```

```
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
```

```
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's email
 matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least eight
 characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
```

```
    signedUp = true
   }
  }
  log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

  log.Println(strings.Repeat("-", 88))

  return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
 string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
```

```
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
```

```
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
 from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
```

```
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
 UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
 user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
```

```
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
```

```
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
```

```
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
```

```
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
```

```
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

```
// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
```

```
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
```

```
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
```

```
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
```

```
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch migriert werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `MigrateUser`-Trigger aufzurufen.
- Melden Sie sich bei Amazon Cognito mit einem Benutzernamen und einer E-Mail-Adresse an, die sich nicht im Benutzerpool befinden.
- Die Lambda-Funktion durchsucht eine DynamoDB-Tabelle und migriert bekannte Benutzer automatisch in den Benutzerpool.
- Führen Sie den Ablauf „Passwort vergessen“ aus, um das Passwort für den migrierten Benutzer zurückzusetzen.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function take
 action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)
```

```
 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username and
 email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user is
 migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
 will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.", user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
```

```
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
 actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
 Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to Cognito,
 you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
```

```
    panic(err)
   }
  } else {
   confirmed = true
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
```

```
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `MigrateUser`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
 error) {
 log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n", user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
```

```
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
```

```
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
```

```
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
```

```
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
```

```
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
```

```
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}
```

```
// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
```

```
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
```

```
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
```

```
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

Das folgende Codebeispiel veranschaulicht, wie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon-Cognito-Benutzerauthentifizierung geschrieben werden.

- Verwenden Sie Administratorfunktionen, um einen Benutzer zu einem Benutzerpool hinzuzufügen.
- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PostAuthentication`-Trigger aufzurufen.
- Melden Sie den neuen Benutzer bei Amazon Cognito an.
- Die Lambda-Funktion schreibt benutzerdefinierte Informationen in CloudWatch Logs und in eine DynamoDB-Tabelle.
- Ermitteln Sie benutzerdefinierte Daten aus der DynamoDB-Tabelle, zeigen Sie diese an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
```

```
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
 from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function take
 action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
 Cognito trigger.\n",
  activityLogArn, userPoolId)
```

```
 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
 string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the DynamoDB
 table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
```

```
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PostAuthentication`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "fmt"
 "log"
 "os"
```

```
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
 logs and
```

```
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
```

```
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
```

```
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}
```

```
// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
```

```
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
```

```
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
```

```
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
```

```
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}
```

```
// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
```

```
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
```

```
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
```

```
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
```

```
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
```

```
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon DocumentDB verwenden.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
```

```
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

DynamoDB-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit DynamoDB verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams auf der ganzen Welt erstellt wurden und verwaltet werden. AWS Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunMovieScenario is an interactive example that shows you how to use the AWS SDK
 for Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
//  1. Create a table that can hold movie data.
//  2. Put, get, and update a single movie in the table.
//  3. Write movie data to the table from a sample JSON file.
//  4. Query for movies that were released in a given year.
//  5. Scan for movies that were released in a range of years.
//  6. Delete a movie from the table.
//  7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is loaded
// into the named table.
func RunMovieScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner,
 tableName string,
 movieSampler actions.IMovieSampler) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB getting started demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{TableName: tableName,
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
```

```
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}

var customMovie actions.Movie
customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
 demotools.NotEmpty{})
customMovie.Year = questioner.AskInt("What year was it released?",
 demotools.NotEmpty{}, demotools.InIntRange{Lower: 1900, Upper: 2030})
customMovie.Info = map[string]interface{}{}
customMovie.Info["rating"] = questioner.AskFloat64(
 "Enter a rating between 1 and 10:",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
 demotools.NotEmpty{})
err = tableBasics.AddMovie(ctx, customMovie)
if err == nil {
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
 "What new rating would you give it?",
 demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
 demotools.NotEmpty{})
attributes, err := tableBasics.UpdateMovie(ctx, customMovie)
if err == nil {
 log.Printf("Updated %v with new values.\n", customMovie.Title)
 for _, attVal := range attributes {
  for valKey, val := range attVal {
```

```
    log.Printf("\t%v: %v\n", valKey, val)
   }
  }
 }
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",
 movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(ctx, movies, 250)
if err != nil {
 panic(err)
} else {
 log.Printf("Added %v movies to the table.\n", written)
}

show := 10
if show > written {
 show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
 log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
 "Enter the number of a movie to get info about it: ",
 demotools.InIntRange{Lower: 1, Upper: show},
)
movie, err := tableBasics.GetMovie(ctx, movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
 log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018},
)
releases, err := tableBasics.Query(ctx, releaseYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
 } else {
```

```
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
endYear := questioner.AskInt("Enter another year: ",
 demotools.InIntRange{Lower: 1972, Upper: 2018})
releases, err = tableBasics.Scan(ctx, startYear, endYear)
if err == nil {
 if len(releases) == 0 {
  log.Printf("I couldn't find any movies released between %v and %v!\n", startYear,
endYear)
 } else {
  log.Printf("Found %v movies. In this list, the plot is <nil> because "+
   "we used a projection expression when scanning for items to return only "+
   "the title, year, and rating.\n", len(releases))
  for _, movie = range releases {
   log.Println(movie)
  }
 }
}
log.Println(strings.Repeat("-", 88))

var tables []string
if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
 tables, err = tableBasics.ListTables(ctx)
 if err == nil {
  log.Printf("Found %v tables:", len(tables))
  for _, table := range tables {
   log.Printf("\t%v", table)
  }
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
 err = tableBasics.DeleteMovie(ctx, customMovie)
}
```

```
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 if questioner.AskBool("Delete the table, too? (y/n)", "y") {
  err = tableBasics.DeleteTable(ctx)
 } else {
  log.Println("Don't forget to delete the table when you're done or you might " +
   "incur charges on your account.")
 }
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
```

```
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die DynamoDB-Aktionen aufrufen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
```

```
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
   AttributeName: aws.String("title"),
   KeyType:       types.KeyTypeRange,
  }},
  TableName:   aws.String(basics.TableName),
  BillingMode: types.BillingModePayPerRequest,
 })
 if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
 } else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
   TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
   log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
 }
 return tableDesc, err
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't list tables. Here's why: %v\n", err)
   break
  } else {
```

```
    tableNames = append(tableNames, output.TableNames...)
   }
  }
  return tableNames, err
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}



// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
   TableName:                 aws.String(basics.TableName),
   Key:                       movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
```

```
    ExpressionAttributeValues: expr.Values(),
    UpdateExpression:          expr.Update(),
    ReturnValues:              types.ReturnValueUpdatedNew,
   })
   if err != nil {
    log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
   } else {
    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
     log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
   }
  }
  return attributeMap, err
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
```

```
    }
   }
   _, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
   if err != nil {
    log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
  basics.TableName, err)
   } else {
    written += len(writeReqs)
   }
   start = end
   end += batchSize
  }

  return written, err
}



// GetMovie gets movie data from the DynamoDB table by using the primary composite
 key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// Query gets all movies in the DynamoDB table that were released in the specified
 year.
// The function uses the `expression` package to build the key condition expression
```

```
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}



// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
```

```
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:          expr.Filter(),
   ProjectionExpression:      expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

```
// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
 return err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query

- Scan
- UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Funktionsempfängerstruktur für das Beispiel.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
 the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
 {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
```

```
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    outMovies = append(outMovies, movie)
   }
  }
 }
 return outMovies, err
}
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
 of
// multiple movies that already exist in the DynamoDB table.
```

```
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
```

```
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
```

```
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für Go API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
```

```
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
 maxMovies int) (int, error) {
 var err error
 var item map[string]types.AttributeValue
 written := 0
 batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
 start := 0
 end := start + batchSize
 for start < maxMovies && start < len(movies) {
  var writeReqs []types.WriteRequest
  if end > len(movies) {
   end = len(movies)
  }
  for _, movie := range movies[start:end] {
   item, err = attributevalue.MarshalMap(movie)
   if err != nil {
    log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
 movie.Title, err)
   } else {
```

```
    writeReqs = append(
     writeReqs,
     types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
   }
  }
  _, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
   RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
  if err != nil {
   log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
 basics.TableName, err)
  } else {
   written += len(writeReqs)
  }
  start = end
  end += batchSize
 }

 return written, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
```

```
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für Go API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
 (*types.TableDescription, error) {
 var tableDesc *types.TableDescription
 table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
  AttributeDefinitions: []types.AttributeDefinition{{
   AttributeName: aws.String("year"),
   AttributeType: types.ScalarAttributeTypeN,
  }, {
   AttributeName: aws.String("title"),
   AttributeType: types.ScalarAttributeTypeS,
  }},
  KeySchema: []types.KeySchemaElement{{
   AttributeName: aws.String("year"),
   KeyType:       types.KeyTypeHash,
  }, {
```

```
    AttributeName: aws.String("title"),
    KeyType:       types.KeyTypeRange,
   }},
   TableName:   aws.String(basics.TableName),
   BillingMode: types.BillingModePayPerRequest,
  })
  if err != nil {
   log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
  } else {
   waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
   err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
    TableName: aws.String(basics.TableName)}, 5*time.Minute)
   if err != nil {
    log.Printf("Wait for table exists failed. Here's why: %v\n", err)
   }
   tableDesc = table.TableDescription
   log.Printf("Ccreating table test")
  }
  return tableDesc, err
}
```

- API-Details finden Sie CreateTable in der AWS SDK für Go API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
 _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
  TableName: aws.String(basics.TableName), Key: movie.GetKey(),
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
```

```
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie DeleteItem in der AWS SDK für Go API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
 _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
  TableName: aws.String(basics.TableName)})
 if err != nil {
  log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
 }
```

```
 return err
}
```

- API-Details finden Sie DeleteTable in der AWS SDK für Go API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
 exists := true
 _, err := basics.DynamoDbClient.DescribeTable(
  ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
 )
 if err != nil {
  var notFoundEx *types.ResourceNotFoundException
  if errors.As(err, &notFoundEx) {
   log.Printf("Table %v does not exist.\n", basics.TableName)
   err = nil
  } else {
   log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
 basics.TableName, err)
  }
  exists = false
 }
 return exists, err
}
```

- API-Details finden Sie DescribeTable in der AWS SDK für Go API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Funktionsempfängerstruktur für das Beispiel.

```
import (
 "context"
```

```
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
 by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Verwenden Sie eine SELECT-Anweisung, um eine Liste der Elemente abzurufen und die Ergebnisse zu projizieren.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
 table.
// pageSize is not typically required and is used to show how to paginate the
 results.
```

```
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
 }
 return output, err
}
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
```

```
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}
```

Verwenden Sie eine DELETE-Anweisung, um ein Element zu löschen.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
 table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}
```

```
// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für Go API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
```

```
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// GetMovie gets movie data from the DynamoDB table by using the primary composite
 key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 movie := Movie{Title: title, Year: year}
 response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key: movie.GetKey(), TableName: aws.String(basics.TableName),
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Item, &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)
```

```
// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie GetItem in der AWS SDK für Go API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
 var tableNames []string
 var output *dynamodb.ListTablesOutput
 var err error
 tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
 &dynamodb.ListTablesInput{})
 for tablePaginator.HasMorePages() {
  output, err = tablePaginator.NextPage(ctx)
  if err != nil {
```

```
    log.Printf("Couldn't list tables. Here's why: %v\n", err)
    break
   } else {
    tableNames = append(tableNames, output.TableNames...)
   }
  }
  return tableNames, err
}
```

- API-Details finden Sie ListTables in der AWS SDK für Go API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
```

```
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
 item, err := attributevalue.MarshalMap(movie)
 if err != nil {
  panic(err)
 }
 _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
  TableName: aws.String(basics.TableName), Item: item,
 })
 if err != nil {
  log.Printf("Couldn't add item to table. Here's why: %v\n", err)
 }
 return err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
```

```
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie PutItem in der AWS SDK für Go API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Query gets all movies in the DynamoDB table that were released in the specified
 year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
 error) {
 var err error
 var response *dynamodb.QueryOutput
 var movies []Movie
 keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
```

```
 expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
 if err != nil {
  log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
 } else {
  queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
 &dynamodb.QueryInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   KeyConditionExpression:    expr.KeyCondition(),
  })
  for queryPaginator.HasMorePages() {
   response, err = queryPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
 releaseYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
```

```
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für Go -API-Referenz.

Scan

Das folgende Codebeispiel zeigt die VerwendungScan.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
 ([]Movie, error) {
 var movies []Movie
```

```
 var err error
 var response *dynamodb.ScanOutput
 filtEx := expression.Name("year").Between(expression.Value(startYear),
 expression.Value(endYear))
 projEx := expression.NamesList(
  expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
 expr, err :=
 expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
 if err != nil {
  log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
 } else {
  scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
 &dynamodb.ScanInput{
   TableName:                 aws.String(basics.TableName),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   FilterExpression:          expr.Filter(),
   ProjectionExpression:      expr.Projection(),
  })
  for scanPaginator.HasMorePages() {
   response, err = scanPaginator.NextPage(ctx)
   if err != nil {
    log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
     startYear, endYear, err)
    break
   } else {
    var moviePage []Movie
    err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
    if err != nil {
     log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
     break
    } else {
     movies = append(movies, moviePage...)
    }
   }
  }
 }
 return movies, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
```

```
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für Go -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
 examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}
```

```
// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
 (map[string]map[string]interface{}, error) {
 var err error
 var response *dynamodb.UpdateItemOutput
 var attributeMap map[string]map[string]interface{}
 update := expression.Set(expression.Name("info.rating"),
 expression.Value(movie.Info["rating"]))
 update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
 expr, err := expression.NewBuilder().WithUpdate(update).Build()
 if err != nil {
  log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
 } else {
  response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
   TableName:                aws.String(basics.TableName),
   Key:                      movie.GetKey(),
   ExpressionAttributeNames:  expr.Names(),
   ExpressionAttributeValues: expr.Values(),
   UpdateExpression:         expr.Update(),
   ReturnValues:             types.ReturnValueUpdatedNew,
  })
  if err != nil {
   log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
  } else {
   err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
   if err != nil {
    log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
   }
  }
 }
 return attributeMap, err
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
```

```
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für Go API-Referenz.

Szenarien

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und Stapel von PartiQL-Abfragen ausführt.

```
import (
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
```

```
// to run batches of PartiQL statements to query a table that stores data about
 movies.
//
//   - Use batches of PartiQL statements to add, get, update, and delete data for
//     individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
 log.Println(strings.Repeat("-", 88))

 tableBasics := actions.TableBasics{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }
 runner := actions.PartiQLRunner{
  DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
  TableName:      tableName,
 }

 exists, err := tableBasics.TableExists(ctx)
 if err != nil {
  panic(err)
 }
 if !exists {
  log.Printf("Creating table %v...\n", tableName)
  _, err = tableBasics.CreateMovieTable(ctx)
  if err != nil {
   panic(err)
  } else {
   log.Printf("Created table %v.\n", tableName)
  }
```

```
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{{
 Title: "House PartiQL",
 Year:  currentYear - 5,
 Info: map[string]interface{}{
  "plot":   "Wacky high jinks result from querying a mysterious database.",
  "rating": 8.5}}, {
 Title: "House PartiQL 2",
 Year:  currentYear - 3,
 Info: map[string]interface{}{
  "plot":   "Moderate high jinks result from querying another mysterious
database.",
  "rating": 6.5}}, {
 Title: "House PartiQL 3",
 Year:  currentYear - 1,
 Info: map[string]interface{}{
  "plot":   "Tepid high jinks result from querying yet another mysterious
database.",
  "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
err = runner.AddMovieBatch(ctx, customMovies)
if err == nil {
 log.Printf("Added %v movies to the table.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(ctx, customMovies)
if err == nil {
 for _, movie := range movies {
  log.Println(movie)
 }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
```

```
 log.Println("Updating a batch of movies with new ratings.")
 err = runner.UpdateMovieBatch(ctx, customMovies, newRatings)
 if err == nil {
  log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Getting projected data from the table to verify our update.")
 log.Println("Using a page size of 2 to demonstrate paging.")
 projections, err := runner.GetAllMovies(ctx, 2)
 if err == nil {
  log.Println("All movies:")
  for _, projection := range projections {
   log.Println(projection)
  }
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Deleting a batch of movies.")
 err = runner.DeleteMovieBatch(ctx, customMovies)
 if err == nil {
  log.Printf("Deleted %v movies.\n", len(customMovies))
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
```

```
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                 `dynamodbav:"title"`
 Year  int                    `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die PartiQL-Anweisungen ausführen.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
 the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
 {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(fmt.Sprintf(
    "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
 runner.TableName)),
   Parameters: params,
  }
 }
```

```
 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
 }
 return err
}



// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
 from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
 ([]Movie, error) {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 var outMovies []Movie
 if err != nil {
  log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
 } else {
  for _, response := range output.Responses {
   var movie Movie
   err = attributevalue.UnmarshalMap(response.Item, &movie)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
```

```
    } else {
     outMovies = append(outMovies, movie)
    }
   }
  }
  return outMovies, err
}



// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
 table.
// pageSize is not typically required and is used to show how to paginate the
 results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
 ([]map[string]interface{}, error) {
 var output []map[string]interface{}
 var response *dynamodb.ExecuteStatementOutput
 var err error
 var nextToken *string
 for moreData := true; moreData; {
  response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
   Statement: aws.String(
    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
   Limit:     aws.Int32(pageSize),
   NextToken: nextToken,
  })
  if err != nil {
   log.Printf("Couldn't get movies. Here's why: %v\n", err)
   moreData = false
  } else {
   var pageOutput []map[string]interface{}
   err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
   if err != nil {
    log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
   } else {
    log.Printf("Got a page of length %v.\n", len(response.Items))
    output = append(output, pageOutput...)
   }
   nextToken = response.NextToken
   moreData = nextToken != nil
  }
```

```
 }
 return output, err
}



// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
 of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
 ratings []float64) error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{ratings[index],
 movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
 runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
 }
 return err
}



// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
 movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
 error {
 statementRequests := make([]types.BatchStatementRequest, len(movies))
```

```
 for index, movie := range movies {
  params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
  if err != nil {
   panic(err)
  }
  statementRequests[index] = types.BatchStatementRequest{
   Statement: aws.String(
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
   Parameters: params,
  }
 }

 _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
 &dynamodb.BatchExecuteStatementInput{
  Statements: statementRequests,
 })
 if err != nil {
  log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
 }
 return err
}
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für Go API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
import (
 "context"
 "fmt"
 "log"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual
 movies.
//
// This example creates an Amazon DynamoDB service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(ctx context.Context, sdkConfig aws.Config, tableName
 string) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
```

```
log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
 DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
 TableName:      tableName,
}
runner := actions.PartiQLRunner{
 DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
 TableName:      tableName,
}

exists, err := tableBasics.TableExists(ctx)
if err != nil {
 panic(err)
}
if !exists {
 log.Printf("Creating table %v...\n", tableName)
 _, err = tableBasics.CreateMovieTable(ctx)
 if err != nil {
  panic(err)
 } else {
  log.Printf("Created table %v.\n", tableName)
 }
} else {
 log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
 Title: "24 Hour PartiQL People",
 Year:  currentYear,
 Info: map[string]interface{}{
  "plot":   "A group of data developers discover a new query language they can't
stop using.",
  "rating": 9.9,
 },
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(ctx, customMovie)
if err == nil {
```

```
 log.Printf("Added %v to the movie table.\n", customMovie.Title)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
 customMovie.Year)
 movie, err := runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
 if err == nil {
  log.Println(movie)
 }
 log.Println(strings.Repeat("-", 88))

 newRating := 6.6
 log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
 newRating)
 err = runner.UpdateMovie(ctx, customMovie, newRating)
 if err == nil {
  log.Printf("Updated %v with a new rating.\n", customMovie.Title)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Getting data again to verify the update.")
 movie, err = runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
 if err == nil {
  log.Println(movie)
 }
 log.Println(strings.Repeat("-", 88))

 log.Printf("Deleting movie '%v'.\n", customMovie.Title)
 err = runner.DeleteMovie(ctx, customMovie)
 if err == nil {
  log.Printf("Deleted %v.\n", customMovie.Title)
 }

 err = tableBasics.DeleteTable(ctx)
 if err == nil {
  log.Printf("Deleted table %v.\n", tableBasics.TableName)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Movie-Struktur, die in diesem Beispiel verwendet wird.

```
import (
 "archive/zip"
 "bytes"
 "encoding/json"
 "fmt"
 "io"
 "log"
 "net/http"

 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
 key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
 key,
// and Info is additional data.
type Movie struct {
 Title string                `dynamodbav:"title"`
 Year  int                   `dynamodbav:"year"`
 Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
 title, err := attributevalue.Marshal(movie.Title)
 if err != nil {
  panic(err)
 }
 year, err := attributevalue.Marshal(movie.Year)
 if err != nil {
  panic(err)
 }
 return map[string]types.AttributeValue{"title": title, "year": year}
}
```

```
// String returns the title, year, rating, and plot of a movie, formatted for the
 example.
func (movie Movie) String() string {
 return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
  movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

Erstellen Sie eine Struktur und Methoden, die PartiQL-Anweisungen ausführen.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
 the
// specified table.
type PartiQLRunner struct {
 DynamoDbClient *dynamodb.Client
 TableName      string
}



// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
 movie.Info})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
```

```
  Statement: aws.String(
   fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
 }
 return err
}



// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
 by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
 (Movie, error) {
 var movie Movie
 params, err := attributevalue.MarshalList([]interface{}{title, year})
 if err != nil {
  panic(err)
 }
 response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
 } else {
  err = attributevalue.UnmarshalMap(response.Items[0], &movie)
  if err != nil {
   log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
  }
 }
 return movie, err
}



// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
```

```
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
 float64) error {
 params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
 movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
 }
 return err
}



// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
 table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
 params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
 if err != nil {
  panic(err)
 }
 _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
 &dynamodb.ExecuteStatementInput{
  Statement: aws.String(
   fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
    runner.TableName)),
  Parameters: params,
 })
 if err != nil {
  log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
 err)
 }
 return err
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für Go API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error)
 {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }
```

```
 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)
```

```
type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
 error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Go V2

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

IAM-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Go V2 mit IAM Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von IAM beginnen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access Management
 (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
```

```
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für Go API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
```

```
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration to
// another configuration. This is used to attach the middleware stubber to clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
```

```
    log.Printf("Something went wrong with the demo.\n")
    log.Println(r)
   }
  }()

  log.Println(strings.Repeat("-", 88))
  log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
  demo.")
  log.Println(strings.Repeat("-", 88))

  user := scenario.CreateUser(ctx)
  accessKey := scenario.CreateAccessKey(ctx, user)
  role := scenario.CreateRoleAndPolicies(ctx, user)
  noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
  scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
  scenario.Cleanup(ctx, user, role)

  log.Println(strings.Repeat("-", 88))
  log.Println("Thanks for watching!")
  log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
  log.Println("Let's create an example user with no permissions.")
  userName := scenario.questioner.Ask("Enter a name for the example user:",
  demotools.NotEmpty{})
  user, err := scenario.userWrapper.GetUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  if user == nil {
   user, err = scenario.userWrapper.CreateUser(ctx, userName)
   if err != nil {
    panic(err)
   }
   log.Printf("Created user %v.\n", *user.UserName)
  } else {
   log.Printf("User %v already exists.\n", *user.UserName)
  }
  log.Println(strings.Repeat("-", 88))
  return user
}
```

```
// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3 buckets
 for
// the current account and attaches the policy to a newly created role. It also adds
 an
// inline policy to the specified user that grants the user permission to assume the
 role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context, user
 *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
```

```
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's access
 key
// credentials and tries to list buckets for the account. Because the user does not
 have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return an
 AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
  ))
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

 s3Client := s3.NewFromConfig(noPermsConfig)
 _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
```

```
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n" +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
 } else {
  log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
   "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
 return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
//  1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//     the user's access key credentials.
//  2. Gets temporary credentials by assuming the role that grants permission to
 list the
//     buckets.
//  3. Creates an Amazon S3 client from the temporary credentials.
//  4. Lists buckets for the account. Because the temporary credentials are
 generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx context.Context,
 noPermsConfig *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and try
 again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
  RoleArn:         role.Arn,
  RoleSessionName: aws.String("AssumeRoleExampleSession"),
  DurationSeconds: aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
```

```
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n" +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
```

```
  err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
*policy.PolicyArn)
  if err != nil {
   panic(err)
  }
  err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
  if err != nil {
   panic(err)
  }
  log.Printf("Detached policy %v from role %v and deleted the policy.\n",
   *policy.PolicyName, *role.RoleName)
 }
 err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Deleted role %v.\n", *role.RoleName)

 userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 for _, userPol := range userPols {
  err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
 }
 keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 for _, key := range keys {
  err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName, *key.AccessKeyId)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
*user.UserName)
 }
 err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
 if err != nil {
  panic(err)
```

```
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
 Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

Definieren Sie eine Struktur, die Kontoaktionen umschließt.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

Definieren Sie eine Struktur, die Richtlinienaktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
```

```
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
```

```
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

Definieren Sie eine Struktur, die Rollenaktionen umschließt.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
```

```
    Effect:    "Allow",
    Principal: map[string]string{"AWS": trustedUserArn},
    Action:    []string{"sts:AssumeRole"},
   }},
  }
  policyBytes, err := json.Marshal(trustPolicy)
  if err != nil {
   log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
  trustedUserArn, err)
   return nil, err
  }
  result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(string(policyBytes)),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  } else {
   role = result.Role
  }
  return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
```

```
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
 string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
 roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
```

```
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

Definieren Sie eine Struktur, die Benutzeraktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
```

```
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
```

```
    }
   }
  } else {
   user = result.User
  }
  return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a policy
 that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
```

```
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
```

```
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy

- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
 roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für Go API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
```

```
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

- API-Details finden Sie CreateAccessKey in der AWS SDK für Go API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
```

```
    Action:   actions,
    Resource: aws.String(resourceArn),
   }},
  }
  policyBytes, err := json.Marshal(policyDoc)
  if err != nil {
   log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
  resourceArn, err)
   return nil, err
  }
  result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
   PolicyDocument: aws.String(string(policyBytes)),
   PolicyName:     aws.String(policyName),
  })
  if err != nil {
   log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
  } else {
   policy = result.Policy
  }
  return policy, err
 }
```

- API-Details finden Sie CreatePolicy in der AWS SDK für Go API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
```

```
 } else {
  role = result.Role
 }
 return role, err
}
```

- API-Details finden Sie CreateRole in der AWS SDK für Go API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
 string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK für Go API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- API-Details finden Sie CreateUser in der AWS SDK für Go API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK für Go API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- API-Details finden Sie DeletePolicy in der AWS SDK für Go API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- API-Details finden Sie DeleteRole in der AWS SDK für Go API-Referenz.

`DeleteServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServiceLinkedRole`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
```

```
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

- API-Details finden Sie DeleteServiceLinkedRole in der AWS SDK für Go API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- API-Details finden Sie DeleteUser in der AWS SDK für Go API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- API-Details finden Sie DeleteUserPolicy in der AWS SDK für Go API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK für Go API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
```

```
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK für Go API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}
```

```
// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- API-Details finden Sie GetPolicy in der AWS SDK für Go API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
```

```
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- API-Details finden Sie GetRole in der AWS SDK für Go API-Referenz.

`GetUser`

Das folgende Codebeispiel zeigt die Verwendung`GetUser`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}
```

- API-Details finden Sie GetUser in der AWS SDK für Go API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
```

```
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- API-Details finden Sie ListAccessKeys in der AWS SDK für Go API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
```

```
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK für Go API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
 ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(ctx, &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

- API-Details finden Sie ListGroups in der AWS SDK für Go API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
```

```
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für Go API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
```

```
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- API-Details finden Sie ListRolePolicies in der AWS SDK für Go API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
```

```
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- API-Details finden Sie ListRoles in der AWS SDK für Go API-Referenz.

`Listet SAML-Anbieter auf`

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für Go -API-Referenz.

`ListUserPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListUserPolicies`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
```

```
}
```

- API-Details finden Sie ListUserPolicies in der AWS SDK für Go API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
```

```
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- API-Details finden Sie ListUsers in der AWS SDK für Go API-Referenz.

`PutUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutUserPolicy`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)
```

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a policy
 that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- API-Details finden Sie PutUserPolicy in der AWS SDK für Go API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Go V2 mit Kinesis Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)
```

```
func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }

  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Lambda-Beispiele unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Go V2 mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren AWS Teams erstellt wurden und verwaltet werden. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
)

// main uses the AWS SDK for Go (v2) to create an AWS Lambda client and list up to
 10
// functions in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 lambdaClient := lambda.NewFromConfig(sdkConfig)

 maxItems := 10
 fmt.Printf("Let's list up to %v functions for your account.\n", maxItems)
 result, err := lambdaClient.ListFunctions(ctx, &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 if err != nil {
```

```
  fmt.Printf("Couldn't list functions for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Functions) == 0 {
  fmt.Println("You don't have any functions!")
 } else {
  for _, function := range result.Functions {
   fmt.Printf("\t%v\n", *function.FunctionName)
  }
 }
}
```

- API-Details finden Sie ListFunctions in der AWS SDK für Go API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein interaktives Szenario, das Ihnen zeigt, wie Sie mit Lambda-Funktionen loslegen können.

```
import (
 "archive/zip"
 "bytes"
 "context"
 "encoding/base64"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "os"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 iamtypes "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/lambda/actions"
)

// GetStartedFunctionsScenario shows you how to use AWS Lambda to perform the
 following
// actions:
//
//  1. Create an AWS Identity and Access Management (IAM) role and Lambda function,
 then upload handler code.
//  2. Invoke the function with a single parameter and get results.
//  3. Update the function code and configure with an environment variable.
//  4. Invoke the function with new parameters and get results. Display the returned
 execution log.
//  5. List the functions for your account, then clean up resources.
type GetStartedFunctionsScenario struct {
 sdkConfig       aws.Config
 functionWrapper actions.FunctionWrapper
 questioner      demotools.IQuestioner
 helper          IScenarioHelper
 isTestRun       bool
}
```

```
// NewGetStartedFunctionsScenario constructs a GetStartedFunctionsScenario instance
 from a configuration.
// It uses the specified config to get a Lambda client and create wrappers for the
 actions
// used in the scenario.
func NewGetStartedFunctionsScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedFunctionsScenario {
 lambdaClient := lambda.NewFromConfig(sdkConfig)
 return GetStartedFunctionsScenario{
  sdkConfig:       sdkConfig,
  functionWrapper: actions.FunctionWrapper{LambdaClient: lambdaClient},
  questioner:      questioner,
  helper:          helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedFunctionsScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Lambda get started with functions demo.")
 log.Println(strings.Repeat("-", 88))

 role := scenario.GetOrCreateRole(ctx)
 funcName := scenario.CreateFunction(ctx, role)
 scenario.InvokeIncrement(ctx, funcName)
 scenario.UpdateFunction(ctx, funcName)
 scenario.InvokeCalculator(ctx, funcName)
 scenario.ListFunctions(ctx)
 scenario.Cleanup(ctx, role, funcName)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

```
// GetOrCreateRole checks whether the specified role exists and returns it if it
 does.
// Otherwise, a role is created that specifies Lambda as a trusted principal.
// The AWSLambdaBasicExecutionRole managed policy is attached to the role and the
 role
// is returned.
func (scenario GetStartedFunctionsScenario) GetOrCreateRole(ctx context.Context)
 *iamtypes.Role {
 var role *iamtypes.Role
 iamClient := iam.NewFromConfig(scenario.sdkConfig)
 log.Println("First, we need an IAM role that Lambda can assume.")
 roleName := scenario.questioner.Ask("Enter a name for the role:",
 demotools.NotEmpty{})
 getOutput, err := iamClient.GetRole(ctx, &iam.GetRoleInput{
  RoleName: aws.String(roleName)})
 if err != nil {
  var noSuch *iamtypes.NoSuchEntityException
  if errors.As(err, &noSuch) {
   log.Printf("Role %v doesn't exist. Creating it....\n", roleName)
  } else {
   log.Panicf("Couldn't check whether role %v exists. Here's why: %v\n",
    roleName, err)
  }
 } else {
  role = getOutput.Role
  log.Printf("Found role %v.\n", *role.RoleName)
 }
 if role == nil {
  trustPolicy := PolicyDocument{
   Version: "2012-10-17",
   Statement: []PolicyStatement{{
    Effect:    "Allow",
    Principal: map[string]string{"Service": "lambda.amazonaws.com"},
    Action:    []string{"sts:AssumeRole"},
   }},
  }
  policyArn := "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
  createOutput, err := iamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(trustPolicy.String()),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  }
```

```
  role = createOutput.Role
  _, err = iamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
   PolicyArn: aws.String(policyArn),
   RoleName:  aws.String(roleName),
  })
  if err != nil {
   log.Panicf("Couldn't attach a policy to role %v. Here's why: %v\n", roleName,
 err)
  }
  log.Printf("Created role %v.\n", *role.RoleName)
  log.Println("Let's give AWS a few seconds to propagate resources...")
  scenario.helper.Pause(10)
 }
 log.Println(strings.Repeat("-", 88))
 return role
}

// CreateFunction creates a Lambda function and uploads a handler written in Python.
// The code for the Python handler is packaged as a []byte in .zip format.
func (scenario GetStartedFunctionsScenario) CreateFunction(ctx context.Context, role
 *iamtypes.Role) string {
 log.Println("Let's create a function that increments a number.\n" +
  "The function uses the 'lambda_handler_basic.py' script found in the \n" +
  "'handlers' directory of this project.")
 funcName := scenario.questioner.Ask("Enter a name for the Lambda function:",
 demotools.NotEmpty{})
 zipPackage := scenario.helper.CreateDeploymentPackage("lambda_handler_basic.py",
 fmt.Sprintf("%v.py", funcName))
 log.Printf("Creating function %v and waiting for it to be ready.", funcName)
 funcState := scenario.functionWrapper.CreateFunction(ctx, funcName,
 fmt.Sprintf("%v.lambda_handler", funcName),
  role.Arn, zipPackage)
 log.Printf("Your function is %v.", funcState)
 log.Println(strings.Repeat("-", 88))
 return funcName
}

// InvokeIncrement invokes a Lambda function that increments a number. The function
// parameters are contained in a Go struct that is used to serialize the parameters
 to
// a JSON payload that is passed to the function.
// The result payload is deserialized into a Go struct that contains an int value.
func (scenario GetStartedFunctionsScenario) InvokeIncrement(ctx context.Context,
 funcName string) {
```

```
 parameters := actions.IncrementParameters{Action: "increment"}
 log.Println("Let's invoke our function. This function increments a number.")
 parameters.Number = scenario.questioner.AskInt("Enter a number to increment:",
 demotools.NotEmpty{})
 log.Printf("Invoking %v with %v...\n", funcName, parameters.Number)
 invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, parameters, false)
 var payload actions.LambdaResultInt
 err := json.Unmarshal(invokeOutput.Payload, &payload)
 if err != nil {
  log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
   funcName, err)
 }
 log.Printf("Invoking %v with %v returned %v.\n", funcName, parameters.Number,
 payload)
 log.Println(strings.Repeat("-", 88))
}

// UpdateFunction updates the code for a Lambda function by uploading a simple
 arithmetic
// calculator written in Python. The code for the Python handler is packaged as a
// []byte in .zip format.
// After the code is updated, the configuration is also updated with a new log
// level that instructs the handler to log additional information.
func (scenario GetStartedFunctionsScenario) UpdateFunction(ctx context.Context,
 funcName string) {
 log.Println("Let's update the function to an arithmetic calculator.\n" +
  "The function uses the 'lambda_handler_calculator.py' script found in the \n" +
  "'handlers' directory of this project.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 log.Println("Creating deployment package...")
 zipPackage :=
 scenario.helper.CreateDeploymentPackage("lambda_handler_calculator.py",
  fmt.Sprintf("%v.py", funcName))
 log.Println("...and updating the Lambda function and waiting for it to be ready.")
 funcState := scenario.functionWrapper.UpdateFunctionCode(ctx, funcName, zipPackage)
 log.Printf("Updated function %v. Its current state is %v.", funcName, funcState)
 log.Println("This function uses an environment variable to control logging level.")
 log.Println("Let's set it to DEBUG to get the most logging.")
 scenario.functionWrapper.UpdateFunctionConfiguration(ctx, funcName,
  map[string]string{"LOG_LEVEL": "DEBUG"})
 log.Println(strings.Repeat("-", 88))
}
```

```
// InvokeCalculator invokes the Lambda calculator function. The parameters are
 stored in a
// Go struct that is used to serialize the parameters to a JSON payload. That
 payload is then passed
// to the function.
// The result payload is deserialized to a Go struct that stores the result as
 either an
// int or float32, depending on the kind of operation that was specified.
func (scenario GetStartedFunctionsScenario) InvokeCalculator(ctx context.Context,
 funcName string) {
 wantInvoke := true
 choices := []string{"plus", "minus", "times", "divided-by"}
 for wantInvoke {
  choice := scenario.questioner.AskChoice("Select an arithmetic operation:\n",
 choices)
  x := scenario.questioner.AskInt("Enter a value for x:", demotools.NotEmpty{})
  y := scenario.questioner.AskInt("Enter a value for y:", demotools.NotEmpty{})
  log.Printf("Invoking %v %v %v...", x, choices[choice], y)
  calcParameters := actions.CalculatorParameters{
   Action: choices[choice],
   X:      x,
   Y:      y,
  }
  invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, calcParameters,
 true)
  var payload any
  if choice == 3 { // divide-by results in a float.
   payload = actions.LambdaResultFloat{}
  } else {
   payload = actions.LambdaResultInt{}
  }
  err := json.Unmarshal(invokeOutput.Payload, &payload)
  if err != nil {
   log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
    funcName, err)
  }
  log.Printf("Invoking %v with %v %v %v returned %v.\n", funcName,
   calcParameters.X, calcParameters.Action, calcParameters.Y, payload)
  scenario.questioner.Ask("Press Enter to see the logs from the call.")
  logRes, err := base64.StdEncoding.DecodeString(*invokeOutput.LogResult)
  if err != nil {
   log.Panicf("Couldn't decode log result. Here's why: %v\n", err)
  }
  log.Println(string(logRes))
```

```
  wantInvoke = scenario.questioner.AskBool("Do you want to calculate again? (y/n)",
 "y")
 }
 log.Println(strings.Repeat("-", 88))
}

// ListFunctions lists up to the specified number of functions for your account.
func (scenario GetStartedFunctionsScenario) ListFunctions(ctx context.Context) {
 count := scenario.questioner.AskInt(
  "Let's list functions for your account. How many do you want to see?",
 demotools.NotEmpty{})
 functions := scenario.functionWrapper.ListFunctions(ctx, count)
 log.Printf("Found %v functions:", len(functions))
 for _, function := range functions {
  log.Printf("\t%v", *function.FunctionName)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup removes the IAM and Lambda resources created by the example.
func (scenario GetStartedFunctionsScenario) Cleanup(ctx context.Context, role
 *iamtypes.Role, funcName string) {
 if scenario.questioner.AskBool("Do you want to clean up resources created for this
 example? (y/n)",
  "y") {
  iamClient := iam.NewFromConfig(scenario.sdkConfig)
  policiesOutput, err := iamClient.ListAttachedRolePolicies(ctx,
   &iam.ListAttachedRolePoliciesInput{RoleName: role.RoleName})
  if err != nil {
   log.Panicf("Couldn't get policies attached to role %v. Here's why: %v\n",
    *role.RoleName, err)
  }
  for _, policy := range policiesOutput.AttachedPolicies {
   _, err = iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
    PolicyArn: policy.PolicyArn, RoleName: role.RoleName,
   })
   if err != nil {
    log.Panicf("Couldn't detach policy %v from role %v. Here's why: %v\n",
     *policy.PolicyArn, *role.RoleName, err)
   }
  }
  _, err = iamClient.DeleteRole(ctx, &iam.DeleteRoleInput{RoleName: role.RoleName})
  if err != nil {
   log.Panicf("Couldn't delete role %v. Here's why: %v\n", *role.RoleName, err)
```

```
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  scenario.functionWrapper.DeleteFunction(ctx, funcName)
  log.Printf("Deleted function %v.\n", funcName)
 } else {
  log.Println("Okay. Don't forget to delete the resources when you're done with
 them.")
 }
}

// IScenarioHelper abstracts I/O and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 CreateDeploymentPackage(sourceFile string, destinationFile string) *bytes.Buffer
}

// ScenarioHelper lets the caller specify the path to Lambda handler functions.
type ScenarioHelper struct {
 HandlerPath string
}

// Pause waits for the specified number of seconds.
func (helper *ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// CreateDeploymentPackage creates an AWS Lambda deployment package from a source
 file. The
// deployment package is stored in .zip format in a bytes.Buffer. The buffer can be
// used to pass a []byte to Lambda when creating the function.
// The specified destinationFile is the name to give the file when it's deployed to
 Lambda.
func (helper *ScenarioHelper) CreateDeploymentPackage(sourceFile string,
 destinationFile string) *bytes.Buffer {
 var err error
 buffer := &bytes.Buffer{}
 writer := zip.NewWriter(buffer)
 zFile, err := writer.Create(destinationFile)
 if err != nil {
  log.Panicf("Couldn't create destination archive %v. Here's why: %v\n",
 destinationFile, err)
 }
```

```
 sourceBody, err := os.ReadFile(fmt.Sprintf("%v/%v", helper.HandlerPath,
 sourceFile))
 if err != nil {
  log.Panicf("Couldn't read handler source file %v. Here's why: %v\n",
   sourceFile, err)
 } else {
  _, err = zFile.Write(sourceBody)
  if err != nil {
   log.Panicf("Couldn't write handler %v to zip archive. Here's why: %v\n",
    sourceFile, err)
  }
 }
 err = writer.Close()
 if err != nil {
  log.Panicf("Couldn't close zip writer. Here's why: %v\n", err)
 }
 return buffer
}
```

Erstellen einer Struktur, die die einzelnen Lambda-Aktionen umschließt.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}
```

```
// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
 types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
```

```
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
 string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
 err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
```

```
  }
 }
 return state
}



// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
 err)
 }
}



// ListFunctions lists up to maxItems functions for the account. This function uses
 a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

```
// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}
```

```
// IncrementParameters is used to serialize parameters to the increment Lambda
 handler.
type IncrementParameters struct {
 Action string `json:"action"`
 Number int    `json:"number"`
}

// CalculatorParameters is used to serialize parameters to the calculator Lambda
 handler.
type CalculatorParameters struct {
 Action string `json:"action"`
 X      int    `json:"x"`
 Y      int    `json:"y"`
}

// LambdaResultInt is used to deserialize an int result from a Lambda handler.
type LambdaResultInt struct {
 Result int `json:"result"`
}

// LambdaResultFloat is used to deserialize a float32 result from a Lambda handler.
type LambdaResultFloat struct {
 Result float32 `json:"result"`
}
```

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
```

```
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
```

```
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// CreateFunction creates a new Lambda function from code contained in the
 zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
```

```
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
 until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
 string, handlerName string,
 iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
  Code:         &types.FunctionCode{ZipFile: zipPackage.Bytes()},
  FunctionName: aws.String(functionName),
  Role:         iamRoleArn,
  Handler:      aws.String(handlerName),
  Publish:      true,
  Runtime:      types.RuntimePython39,
 })
 if err != nil {
  var resConflict *types.ResourceConflictException
  if errors.As(err, &resConflict) {
   log.Printf("Function %v already exists.\n", functionName)
   state = types.StateActive
  } else {
   log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
  }
 } else {
  waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- API-Details finden Sie CreateFunction in der AWS SDK für Go API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
 string) {
 _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
 }
```

```
}
```

- API-Details finden Sie DeleteFunction in der AWS SDK für Go API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// GetFunction gets data about the Lambda function specified by functionName.
```

```
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
 types.State {
 var state types.State
 funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
  FunctionName: aws.String(functionName),
 })
 if err != nil {
  log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
 } else {
  state = funcOutput.Configuration.State
 }
 return state
}
```

- API-Details finden Sie GetFunction in der AWS SDK für Go API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
```

```
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// Invoke invokes the Lambda function specified by functionName, passing the
 parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
 tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
 parameters any, getLog bool) *lambda.InvokeOutput {
 logType := types.LogTypeNone
 if getLog {
  logType = types.LogTypeTail
 }
 payload, err := json.Marshal(parameters)
 if err != nil {
  log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
 }
 invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
  FunctionName: aws.String(functionName),
  LogType:      logType,
  Payload:      payload,
 })
 if err != nil {
  log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
 }
 return invokeOutput
}
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für Go -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// ListFunctions lists up to maxItems functions for the account. This function uses
 a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
 []types.FunctionConfiguration {
 var functions []types.FunctionConfiguration
 paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
 &lambda.ListFunctionsInput{
  MaxItems: aws.Int32(int32(maxItems)),
 })
 for paginator.HasMorePages() && len(functions) < maxItems {
  pageOutput, err := paginator.NextPage(ctx)
```

```
  if err != nil {
   log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
  }
  functions = append(functions, pageOutput.Functions...)
 }
 return functions
}
```

- API-Details finden Sie ListFunctions in der AWS SDK für Go API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
```

```
 LambdaClient *lambda.Client
}



// UpdateFunctionCode updates the code for the Lambda function specified by
 functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
 lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
 string, zipPackage *bytes.Buffer) types.State {
 var state types.State
 _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
 &lambda.UpdateFunctionCodeInput{
  FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
 })
 if err != nil {
  log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
 err)
 } else {
  waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
  funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
   FunctionName: aws.String(functionName)}, 1*time.Minute)
  if err != nil {
   log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
 functionName, err)
  } else {
   state = funcOutput.Configuration.State
  }
 }
 return state
}
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für Go API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "encoding/json"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/lambda"
 "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
 LambdaClient *lambda.Client
}



// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
 functionName string, envVars map[string]string) {
 _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
 &lambda.UpdateFunctionConfigurationInput{
  FunctionName: aws.String(functionName),
  Environment:  &types.Environment{Variables: envVars},
 })
 if err != nil {
  log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
 err)
```

```
 }
}
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für Go API-Referenz.

Szenarien

Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie bekannte Amazon-Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's email
 matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least eight
 characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
```

```
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
 string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
```

```
 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
 from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
```

```
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
 UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
 user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
```

```
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
```

```
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
```

```
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
```

```
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

```
// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
```

```
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
```

```
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
```

```
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)
```

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
```

```
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
```

```
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Migrieren Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie bekannte Amazon Cognito-Benutzer mit einer Lambda-Funktion automatisch migriert werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `MigrateUser`-Trigger aufzurufen.
- Melden Sie sich bei Amazon Cognito mit einem Benutzernamen und einer E-Mail-Adresse an, die sich nicht im Benutzerpool befinden.
- Die Lambda-Funktion durchsucht eine DynamoDB-Tabelle und migriert bekannte Benutzer automatisch in den Benutzerpool.
- Führen Sie den Ablauf „Passwort vergessen“ aus, um das Passwort für den migrierten Benutzer zurückzusetzen.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
```

```
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function take
 action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}
```

```
// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username and
 email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user is
 migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
 will be used to confirm user migration\n" +
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.", user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
```

```
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
 actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
 Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
  log.Println("To complete this example and successfully migrate a user to Cognito,
 you must enter an email\n" +
   "you own that can receive a confirmation code.")
  return
 }
 codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
 user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)
 code := runner.questioner.Ask("Check your email and enter it here:")

 confirmed := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !confirmed {
  log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
 user.UserName, password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   confirmed = true
```

```
  }
 }
 log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
 log.Println("Signing in with your username and password...")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
 password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
```

```
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `MigrateUser`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}
```

```
// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
 error) {
 log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n", user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }
```

```
 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
```

```
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
```

```
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
```

```
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
```

```
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
```

```
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
```

```
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
  nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Schreiben von benutzerdefinierten Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon Cognito-Benutzerauthentifizierung

Das folgende Codebeispiel veranschaulicht, wie benutzerdefinierte Aktivitätsdaten mit einer Lambda-Funktion nach der Amazon-Cognito-Benutzerauthentifizierung geschrieben werden.

- Verwenden Sie Administratorfunktionen, um einen Benutzer zu einem Benutzerpool hinzuzufügen.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PostAuthentication`-Trigger aufzurufen.
- Melden Sie den neuen Benutzer bei Amazon Cognito an.
- Die Lambda-Funktion schreibt benutzerdefinierte Informationen in CloudWatch Logs und in eine DynamoDB-Tabelle.
- Ermitteln Sie benutzerdefinierte Daten aus der DynamoDB-Tabelle, zeigen Sie diese an und bereinigen Sie dann die Ressourcen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
```

```
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
 IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
```

```
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
 from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function take
 action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
 Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
 string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
```

```
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the DynamoDB
 table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")
```

```
 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Behandeln Sie den `PostAuthentication`-Trigger mit einer Lambda-Funktion.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
```

```
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
 logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
```

```
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

Erstellen Sie eine Struktur, die allgemeine Aufgaben ausführt.

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
```

```
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
 example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
 err)
```

```
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
 Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
 string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with your
 Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

Erstellen Sie eine Struktur, die Amazon Cognito-Aktionen umschließt.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
 specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
```

```
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
 err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
```

```
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
   }
  } else {
   confirmed = output.UserConfirmed
  }
  return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
```

```
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
 error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool. This
 method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
 userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a user
 without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
```

```
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

Erstellen Sie eine Struktur, die DynamoDB-Aktionen umhüllt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
```

```
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
 i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
```

```
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
 err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
 error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

Erstellen Sie eine Struktur, die CloudWatch Logs-Aktionen umschließt.

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
```

```
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

Erstellen Sie eine Struktur, die Aktionen umschließt. CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
 StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
```

```
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}
```

```
// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources \n"
 +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
 nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"
```

```
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
```

```
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)

 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
 if len(kinesisEvent.Records) == 0 {
  log.Printf("empty Kinesis event received")
  return nil
 }

 for _, record := range kinesisEvent.Records {
  log.Printf("processed Kinesis event with EventId: %v", record.EventID)
  recordDataBytes := record.Kinesis.Data
  recordDataText := string(recordDataBytes)
  log.Printf("record data: %v", recordDataText)
  // TODO: Do interesting work based on the new data
 }
 log.Printf("successfully processed %v records", len(kinesisEvent.Records))
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-lambda-go/events"
 "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error)
 {
 if len(event.Records) == 0 {
  return nil, fmt.Errorf("received empty event")
 }

 for _, record := range event.Records {
   LogDynamoDBRecord(record)
 }

 message := fmt.Sprintf("Records processed: %d", len(event.Records))
 return &message, nil
}

func main() {
 lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
 fmt.Println(record.EventID)
 fmt.Println(record.EventName)
 fmt.Printf("%+v\n", record.Change)
```

```
}
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"

 "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
 Events []Record `json:"events"`
}

type Record struct {
 Event struct {
  OperationType string `json:"operationType"`
  NS            struct {
   DB   string `json:"db"`
   Coll string `json:"coll"`
  } `json:"ns"`
```

```
  FullDocument interface{} `json:"fullDocument"`
 } `json:"event"`
}

func main() {
 lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
 fmt.Println("Loading function")
 for _, record := range event.Events {
  logDocumentDBEvent(record)
 }

 return "OK", nil
}

func logDocumentDBEvent(record Record) {
 fmt.Printf("Operation type: %s\n", record.Event.OperationType)
 fmt.Printf("db: %s\n", record.Event.NS.DB)
 fmt.Printf("collection: %s\n", record.Event.NS.Coll)
 docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")
 fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
```

```
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)
```

```
func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
```

```
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
 (map[string]interface{}, error) {
 batchItemFailures := []map[string]interface{}{}

 for _, record := range kinesisEvent.Records {
  curRecordSequenceNumber := ""

  // Process your record
  if /* Your record processing condition here */ {
   curRecordSequenceNumber = record.Kinesis.SequenceNumber
  }

  // Add a condition to check if the record processing failed
  if curRecordSequenceNumber != "" {
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": curRecordSequenceNumber})
  }
 }

 kinesisBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return kinesisBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
 ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
 BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
 error) {
 var batchItemFailures []BatchItemFailure
 curRecordSequenceNumber := ""

 for _, record := range event.Records {
  // Process your record
```

```
  curRecordSequenceNumber = record.Change.SequenceNumber
 }

 if curRecordSequenceNumber != "" {
  batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
 curRecordSequenceNumber})
 }

 batchResult := BatchResult{
  BatchItemFailures: batchItemFailures,
 }

 return &batchResult, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main
```

```
import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
 error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}

func main() {
 lambda.Start(handler)
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Go V2

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Go SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für Amazon MSK unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon MSK verwenden.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Go.

```
package main

import (
 "encoding/base64"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
 for key, records := range event.Records {
  fmt.Println("Key:", key)

  for _, record := range records {
   fmt.Println("Record:", record)

   decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
   message := string(decodedValue)
   fmt.Println("Message:", message)
  }
 }
}

func main() {
 lambda.Start(handler)
}
```

Beispiele für Partner Central unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Partner Central verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`GetOpportunity`.

SDK für Go V2

Rufen Sie eine Opportunity ab.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
```

```
 })

 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- API-Details finden Sie GetOpportunity in der AWS SDK für Go API-Referenz.

`ListOpportunities`

Das folgende Codebeispiel zeigt die Verwendung`ListOpportunities`.

SDK für Go V2

Listen Sie Opportunitys auf.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }
```

```
 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- API-Details finden Sie ListOpportunities in der AWS SDK für Go API-Referenz.

Beispiele für Amazon RDS unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon RDS verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen
- Serverless-Beispiele

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database Service
 (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
```

```
  return
 }
 rdsClient := rds.NewFromConfig(sdkConfig)
 const maxInstances = 20
 fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
 output, err := rdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
 if err != nil {
  fmt.Printf("Couldn't list DB instances: %v\n", err)
  return
 }
 if len(output.DBInstances) == 0 {
  fmt.Println("No DB instances found.")
 } else {
  for _, instance := range output.DBInstances {
   fmt.Printf("DB instance %v has database %v.\n", *instance.DBInstanceIdentifier,
    *instance.DBName)
  }
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Go -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "fmt"
 "log"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
 "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
//  1. Create a custom DB parameter group and set parameter values.
//  2. Create a DB instance that is configured to use the parameter group. The DB
 instance
//     also contains a database.
//  3. Take a snapshot of the DB instance.
//  4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
 sdkConfig  aws.Config
 instances  actions.DbInstances
 questioner demotools.IQuestioner
 helper     IScenarioHelper
```

```
 isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
 configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) GetStartedInstances {
 rdsClient := rds.NewFromConfig(sdkConfig)
 return GetStartedInstances{
  sdkConfig:  sdkConfig,
  instances:  actions.DbInstances{RdsClient: rdsClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 instanceName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
 Instance demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
 parameterGroup)
 scenario.DisplayConnection(instance)
 scenario.CreateSnapshot(ctx, instance)
 scenario.Cleanup(ctx, instance, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
```

```
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBParameterGroup {

 log.Printf("Checking for an existing DB parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
 families)
  log.Println("Creating a DB parameter group.")
  _, err = scenario.instances.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
```

```
  }
 }
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName, "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("To get a list of parameters that you set previously, specify a source
 of 'user'.")
 userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
 "user")
```

```
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
 group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context, instanceName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

 log.Println("Checking for an existing DB instance.")
 instance, err := scenario.instances.GetInstance(ctx, instanceName)
 if err != nil {
  panic(err)
 }
 if instance == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator username for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.AskPassword(
   "Enter a password for the administrator (at least 8 characters): ", 7)
  engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
   *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  engineIndex := scenario.questioner.AskChoice(
   "The available engines for your parameter group are:\n", engineChoices)
  engineSelection := engineVersions[engineIndex]
  instOpts, err := scenario.instances.GetOrderableInstances(ctx,
 *engineSelection.Engine,
   *engineSelection.EngineVersion)
  if err != nil {
   panic(err)
```

```
 }
 optSet := map[string]struct{}{}
 for _, opt := range instOpts {
  if strings.Contains(*opt.DBInstanceClass, "micro") {
   optSet[*opt.DBInstanceClass] = struct{}{}
  }
 }
 var optChoices []string
 for opt := range optSet {
  optChoices = append(optChoices, opt)
 }
 sort.Strings(optChoices)
 optIndex := scenario.questioner.AskChoice(
  "The available micro DB instance classes for your database engine are:\n",
optChoices)
 storageType := "standard"
 allocatedStorage := int32(5)
 log.Printf("Creating a DB instance named %v and database %v.\n"+
  "The DB instance is configured to use your custom parameter group %v,\n"+
  "selected engine %v,\n"+
  "selected DB instance class %v,"+
  "and %v GiB of %v storage.\n"+
  "This typically takes several minutes.",
  instanceName, dbName, *parameterGroup.DBParameterGroupName,
*engineSelection.EngineVersion,
  optChoices[optIndex], allocatedStorage, storageType)
 instance, err = scenario.instances.CreateInstance(
  ctx, instanceName, dbName, *engineSelection.Engine,
*engineSelection.EngineVersion,
  *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
  allocatedStorage, adminUsername, adminPassword)
 if err != nil {
  panic(err)
 }
 for *instance.DBInstanceStatus != "available" {
  scenario.helper.Pause(30)
  instance, err = scenario.instances.GetInstance(ctx, instanceName)
  if err != nil {
   panic(err)
  }
 }
 log.Println("Instance created and available.")
}
log.Println("Instance data:")
```

```
 log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
 log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
 log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
 log.Printf("\tEngine: %v\n", *instance.Engine)
 log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
 log.Println(strings.Repeat("-", 88))
 return instance
}

// DisplayConnection displays connection information about a DB instance and tips
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance) {
 log.Println(
  "You can now connect to your database by using your favorite MySQL client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
   "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
   "port, and administrator username to 'mysql'. Then, enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
 log.Println("For more information, see the User Guide for RDS:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
 available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
 *types.DBInstance) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
 scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
 snapshotId)
  snapshot, err := scenario.instances.CreateSnapshot(ctx,
 *instance.DBInstanceIdentifier, snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
```

```
    if err != nil {
     panic(err)
    }
   }
   log.Println("Snapshot data:")
   log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
   log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
   log.Printf("\tStatus: %v\n", *snapshot.Status)
   log.Printf("\tEngine: %v\n", *snapshot.Engine)
   log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
   log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
   log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
   log.Println(strings.Repeat("-", 88))
  }
 }

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
 first be deleted.
func (scenario GetStartedInstances) Cleanup(
 ctx context.Context, instance *types.DBInstance, parameterGroup
 *types.DBParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance and parameter group (y/n)? ", "y")
 {
  log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
  err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance to delete. This typically takes several minutes.")
  for instance != nil {
   scenario.helper.Pause(30)
   instance, err = scenario.instances.GetInstance(ctx,
 *instance.DBInstanceIdentifier)
   if err != nil {
    panic(err)
   }
  }
  log.Printf("Deleting parameter group %v.", *parameterGroup.DBParameterGroupName)
  err = scenario.instances.DeleteParameterGroup(ctx,
 *parameterGroup.DBParameterGroupName)
```

```
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Amazon-RDS-Aktionen zu verwalten.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}



// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
```

```
 }
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
```

```
 return params, err
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}



// GetSnapshot gets a DB instance snapshot.
```

```
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
 (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
 string,
 storageType string, allocatedStorage int32, adminName string, adminPassword string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

```
// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

```
// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
```

```
    instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
   }
  }
  return instanceOptions, err
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
 string, dbName string,
 dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
 string,
 storageType string, allocatedStorage int32, adminName string, adminPassword string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBName:               aws.String(dbName),
  DBParameterGroupName: aws.String(parameterGroupName),
  Engine:               aws.String(dbEngine),
  EngineVersion:        aws.String(dbEngineVersion),
  DBInstanceClass:      aws.String(dbInstanceClass),
  StorageType:          aws.String(storageType),
  AllocatedStorage:     aws.Int32(allocatedStorage),
  MasterUsername:       aws.String(adminName),
  MasterUserPassword:   aws.String(adminPassword),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
```

```
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für Go -API-Referenz.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
```

```
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBParameterGroup, error) {

 output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
  &rds.CreateDBParameterGroupInput{
   DBParameterGroupName:   aws.String(parameterGroupName),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
   Description:            aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBParameterGroup, err
 }
}
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK für Go API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
 string, snapshotName string) (
 *types.DBSnapshot, error) {
 output, err := instances.RdsClient.CreateDBSnapshot(ctx,
 &rds.CreateDBSnapshotInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBSnapshot, nil
 }
}
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für Go -API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für Go -API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
  &rds.DeleteDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK für Go API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
```

```
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für Go API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetInstance gets data about a DB instance.
```

```
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := instances.RdsClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Go -API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
```

```
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBParameterGroup, error) {
 output, err := instances.RdsClient.DescribeDBParameterGroups(
  ctx, &rds.DescribeDBParameterGroupsInput{
   DBParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBParameterGroups[0], err
 }
}
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK für Go API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
  &rds.DescribeDBParametersInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Source:               aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
```

```
    break
   } else {
    params = append(params, output.Parameters...)
   }
  }
  return params, err
}
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für Go - API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}
```

```
// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
 (*types.DBSnapshot, error) {
 output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
  &rds.DescribeDBSnapshotsInput{
   DBSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBSnapshots[0], nil
 }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für Go - API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
```

```
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instanceOptions []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instance options: %v\n", err)
   break
  } else {
   instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
  }
 }
 return instanceOptions, err
}
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK für Go API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
 RdsClient *rds.Client
}



// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
  &rds.ModifyDBParameterGroupInput{
   DBParameterGroupName: aws.String(parameterGroupName),
   Parameters:           params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
```

```
}
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK für Go API-Referenz.

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Go.

```
/*
Golang v2 code here.
*/

package main

import (
 "context"
 "database/sql"
 "encoding/json"
 "fmt"
 "os"

 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
 _ "github.com/go-sql-driver/mysql"
)
```

```
type MyEvent struct {
 Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

 var dbName string = os.Getenv("DatabaseName")
 var dbUser string = os.Getenv("DatabaseUser")
 var dbHost string = os.Getenv("DBHost") // Add hostname without https
 var dbPort int = os.Getenv("Port")      // Add port number
 var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
 var region string = os.Getenv("AWS_REGION")

 cfg, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  panic("configuration error: " + err.Error())
 }

 authenticationToken, err := auth.BuildAuthToken(
  context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
 if err != nil {
  panic("failed to create authentication token: " + err.Error())
 }

 dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
  dbUser, authenticationToken, dbEndpoint, dbName,
 )

 db, err := sql.Open("mysql", dsn)
 if err != nil {
  panic(err)
 }

 defer db.Close()

 var sum int
 err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
 if err != nil {
  panic(err)
 }
 s := fmt.Sprint(sum)
 message := fmt.Sprintf("The selected sum is: %s", s)
```

```
 messageBytes, err := json.Marshal(message)
 if err != nil {
  return nil, err
 }

 messageString := string(messageBytes)
 return map[string]interface{}{
  "statusCode": 200,
  "headers":    map[string]string{"Content-Type": "application/json"},
  "body":       messageString,
 }, nil
}

func main() {
 lambda.Start(HandleRequest)
}
```

Beispiele für Amazon Redshift unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon Redshift verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Redshift

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Redshift beginnen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 redshiftClient := redshift.NewFromConfig(sdkConfig)
 count := 20
 fmt.Printf("Let's list up to %v clusters for your account.\n", count)
```

```
 result, err := redshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  MaxRecords: aws.Int32(int32(count)),
 })
 if err != nil {
  fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Clusters) == 0 {
  fmt.Println("You don't have any clusters!")
  return
 }
 for _, cluster := range result.Clusters {
  fmt.Printf("\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK für Go API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Redshift-Clusters.
- Listen Sie die Datenbanken im Cluster auf.
- Erstellen einer Tabelle mit dem Namen „Movies“.
- Befüllen der Tabelle „Movies“.
- Abfragen der Tabelle „Movies“ nach Jahr.
- Ändern des Redshift-Clusters.
- Löschen Sie den Amazon-Redshift-Cluster.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package scenarios

import (
 "context"
 "encoding/json"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
 redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
 "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
```

```
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
 functions so that
// they are simpler to read and understand.
type RedshiftBasicsScenario struct {
 sdkConfig         aws.Config
 helper            IScenarioHelper
 questioner        demotools.IQuestioner
 pauser            demotools.IPausable
 filesystem        demotools.IFileSystem
 redshiftActor     *actions.RedshiftActions
 redshiftDataActor *actions.RedshiftDataActions
 secretsmanager    *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
 secure service.
type SecretsManager struct {
 SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner, pauser
 demotools.IPausable, filesystem demotools.IFileSystem, helper IScenarioHelper)
 RedshiftBasicsScenario {
 scenario := RedshiftBasicsScenario{
  sdkConfig:         sdkConfig,
  helper:            helper,
  questioner:        questioner,
  pauser:            pauser,
  filesystem:        filesystem,
  secretsmanager:    &SecretsManager{SecretsManagerClient:
 secretsmanager.NewFromConfig(sdkConfig)},
  redshiftActor:     &actions.RedshiftActions{RedshiftClient:
 redshift.NewFromConfig(sdkConfig)},
  redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
 redshiftdata.NewFromConfig(sdkConfig)},
```

```
 }
 return scenario
}


// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
 ID    int    `json:"id"`
 Title string `json:"title"`
 Year  int    `json:"year"`
}


// User makes it easier to get the User data back from SecretsManager and use it
 later.
type User struct {
 Username string `json:"userName"`
 Password string `json:"userPassword"`
}

// Run runs the RedshiftBasics interactive example that shows you how to use Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
 sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

 user := User{}
 secretId := "s3express/basics/secrets"
```

```
clusterId := "demo-cluster-1"
maintenanceWindow := "wed:07:30-wed:08:00"
databaseName := "dev"
tableName := "Movies"
fileName := "Movies.json"
nodeType := "ra3.xlplus"
clusterType := "single-node"

defer func() {
 if r := recover(); r != nil {
  log.Println("Something went wrong with the demo.")
  _, isMock := runner.questioner.(*demotools.MockQuestioner)
  if isMock || runner.questioner.AskBool("Do you want to see the full error message
(y/n)?", "y") {
   log.Println(r)
  }
  runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
runner.questioner)
 }
}()

// Retrieve the userName and userPassword from SecretsManager
output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
&secretsmanager.GetSecretValueInput{
 SecretId: aws.String(secretId),
})
if err != nil {
 log.Printf("There was a problem getting the secret value: %s", err)
 log.Printf("Please make sure to create a secret named 's3express/basics/secrets'
with keys of 'userName' and 'userPassword'.")
 panic(err)
}

err = json.Unmarshal([]byte(*output.SecretString), &user)
if err != nil {
 log.Printf("There was a problem parsing the secret value from JSON: %s", err)
 panic(err)
}

// Create the Redshift cluster
_, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username,
user.Password, nodeType, clusterType, true)
if err != nil {
 var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault
```

```
 if errors.As(err, &clusterAlreadyExistsFault) {
  log.Println("Cluster already exists. Continuing.")
 } else {
  log.Println("Error creating cluster.")
  panic(err)
 }
}

// Wait for the cluster to become available
waiter := redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
 ClusterIdentifier: aws.String(clusterId),
}, 5*time.Minute)
if err != nil {
 log.Println("An error occurred waiting for the cluster.")
 panic(err)
}

// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
 log.Println("Something went wrong trying to get information about the cluster.")
 panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s", *describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
 log.Printf("Failed to list databases: %v\n", err)
 panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil
```

```
 result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId, databaseName,
 tableName, user.Username, runner.pauser, []string{"title VARCHAR(256)", "year
 INT"})
 if err != nil {
  log.Printf("Failed to create table: %v\n", err)
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }
 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 // Populate the "Movies" table
 runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName, user.Username,
 fileName)

 // Query the "Movies" table by year
 log.Println("Query the Movies table by year.")
 year := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
  demotools.InIntRange{Lower: 2012, Upper: 2014})
 runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
 year)

 // Modify the cluster's maintenance window
 runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

 // Delete the Redshift cluster if confirmed
 runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
 runner.questioner)

 log.Println("Thanks for watching!")
}
```

```
// cleanUpResources asks the user if they would like to delete each resource created
 during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further deletion
 attempts are skipped.
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string,
 questioner demotools.IQuestioner) {
 deleted := false
 var err error = nil
 if questioner.AskBool("Do you want to delete the entire cluster? This will clean up
 all resources. (y/n)", "y") {
  deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
  if err != nil {
   log.Printf("Error deleting cluster: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete the dev table? This will
 clean up all inserted records but keep your cluster intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId, databaseName,
 tableName, userName)
  if err != nil {
   log.Printf("Error deleting movies table: %v", err)
  }
 }
 if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
 table? This will clean up all inserted records but keep your cluster and table
 intact. (y/n)", "y") {
  deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
 databaseName, tableName, userName, runner.pauser)
  if err != nil {
   log.Printf("Error deleting data rows: %v", err)
  }
 }
 if !deleted {
  log.Print("Please manually delete any unwanted resources.")
 }
}


// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
 objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string, filesystem
 demotools.IFileSystem) ([]Movie, error) {
```

```
 file, err := filesystem.OpenFile("../../resources/sample_files/" + fileName)
 if err != nil {
  return nil, err
 }
 defer filesystem.CloseFile(file)

 var movies []Movie
 err = json.NewDecoder(file).Decode(&movies)
 if err != nil {
  return nil, err
 }

 return movies, nil
}



// PopulateMoviesTable reads data from the <fileName> file and inserts records into
 the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, fileName
 string) {
 log.Println("Populate the " + tableName + " table using the " + fileName + "
 file.")
 numRecords := runner.questioner.AskInt(
  fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
  demotools.InIntRange{Lower: 10, Upper: 100})

 movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
 if err != nil {
  log.Printf("Failed to load movies from JSON: %v\n", err)
  panic(err)
 }

 var sqlStatements []string

 for i, movie := range movies {
  if i >= numRecords {
   break
  }

  sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
   tableName,
```

```
    strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
  escape them
    movie.Year)

  sqlStatements = append(sqlStatements, sqlStatement)
 }

 input := &redshiftdata.BatchExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sqls:              sqlStatements,
 }

 result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to execute batch statement: %v\n", err)
  panic(err)
 }

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Result:  result,
  Input:   describeInput,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute batch insert query: %v\n", err)
  return
 }
 log.Printf("Successfully executed batch statement\n")

 log.Printf("%d records were added to the Movies table.\n", numRecords)
}



// QueryMoviesByYear retrieves only movies from the "Movies" table which match the
 given year.
```

```
func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
 clusterId string, databaseName string, tableName string, userName string, year int)
 {

 sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
 year)

 input := &redshiftdata.ExecuteStatementInput{
  ClusterIdentifier: aws.String(clusterId),
  Database:          aws.String(databaseName),
  DbUser:            aws.String(userName),
  Sql:               aws.String(sqlStatement),
 }

 result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
 }

 log.Println("The identifier of the statement is ", *result.Id)

 describeInput := redshiftdata.DescribeStatementInput{
  Id: result.Id,
 }

 query := actions.RedshiftQuery{
  Context: ctx,
  Input:   describeInput,
  Result:  result,
 }
 err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
 if err != nil {
  log.Printf("Failed to execute query: %v\n", err)
  panic(err)
 }
 log.Printf("Successfully executed query\n")

 getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
 *result.Id)
 if err != nil {
  log.Printf("Failed to query movies: %v\n", err)
  panic(err)
 }
```

```
 for _, row := range getResultOutput.Records {
  for _, col := range row {
   title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
   if !ok {
    log.Println("Failed to parse the field")
   } else {
    log.Printf("The Movie title field is %s\n", title.Value)
   }
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// CreateCluster sends a request to create a cluster with the given clusterId using
 the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
 userName string, userPassword string, nodeType string, clusterType string,
 publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
 // Create a new Redshift cluster
 input := &redshift.CreateClusterInput{
  ClusterIdentifier:  aws.String(clusterId),
  MasterUserPassword: aws.String(userPassword),
  MasterUsername:     aws.String(userName),
  NodeType:           aws.String(nodeType),
  ClusterType:        aws.String(clusterType),
  PubliclyAccessible: aws.Bool(publiclyAccessible),
 }
 var opErr *types.ClusterAlreadyExistsFault
 output, err := actor.RedshiftClient.CreateCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster already exists")
  return nil, nil
 } else if err != nil {
  log.Printf("Failed to create Redshift cluster: %v\n", err)
  return nil, err
```

```
 }

 log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
 return output, nil
}
```

- API-Details finden Sie CreateCluster in der AWS SDK für Go API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}
```

```
// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
 (bool, error) {
 input := redshift.DeleteClusterInput{
  ClusterIdentifier:        aws.String(clusterId),
  SkipFinalClusterSnapshot: aws.Bool(true),
 }
 _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
 var opErr *types.ClusterNotFoundFault
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster was not found. Where could it be?")
  return false, err
 } else if err != nil {
  log.Printf("Failed to delete Redshift cluster: %v\n", err)
  return false, err
 }
 waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
 err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 }, 5*time.Minute)
 if err != nil {
  log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
 }
 log.Printf("The cluster %s was deleted\n", clusterId)
 return true, nil
}
```

- API-Details finden Sie DeleteCluster in der AWS SDK für Go API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// DescribeClusters returns information about the given cluster.
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId string)
 (*redshift.DescribeClustersOutput, error) {
 input, err := actor.RedshiftClient.DescribeClusters(ctx,
 &redshift.DescribeClustersInput{
  ClusterIdentifier: aws.String(clusterId),
 })
 var opErr *types.AccessToClusterDeniedFault
 if errors.As(err, &opErr) {
  println("Access to cluster denied.")
  panic(err)
 } else if err != nil {
  println("Failed to describe Redshift clusters.")
  return nil, err
 }
 return input, nil
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK für Go API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "errors"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/redshift"
 "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)



// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
 RedshiftClient *redshift.Client
}



// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
 maintenanceWindow string) *redshift.ModifyClusterOutput {
 // Modify the cluster's maintenance window
 input := &redshift.ModifyClusterInput{
  ClusterIdentifier:          aws.String(clusterId),
  PreferredMaintenanceWindow: aws.String(maintenanceWindow),
 }

 var opErr *types.InvalidClusterStateFault
```

```
 output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
 if err != nil && errors.As(err, &opErr) {
  log.Println("Cluster is in an invalid state.")
  panic(err)
 } else if err != nil {
  log.Printf("Failed to modify Redshift cluster: %v\n", err)
  panic(err)
 }

 log.Printf("The cluster was successfully modified and now has %s as the maintenance
 window\n", *output.Cluster.PreferredMaintenanceWindow)
 return output
}
```

- API-Details finden Sie ModifyCluster in der AWS SDK für Go API-Referenz.

Beispiele für Amazon S3 unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon S3 verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "errors"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
```

```
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  var ae smithy.APIError
  if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
   fmt.Println("You don't have permission to list buckets for this account.")
  } else {
   fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  }
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für Go API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.

- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie eine Struktur, die die vom Szenario verwendete Bucket- und Objektaktionen umschließt.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error)
 {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
```

```
    log.Printf("Either you don't have access to bucket %v or another error occurred.
 "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
 objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
```

```
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
```

```
    err = noKey
   } else {
    log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
  err)
   }
   return err
  }
  defer result.Body.Close()
  file, err := os.Create(fileName)
  if err != nil {
   log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
   return err
  }
  defer file.Close()
  body, err := io.ReadAll(result.Body)
  if err != nil {
   log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
  }
  _, err = file.Write(body)
  return err
}



// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
```

```
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
 objectKey string, folderName string) error {
 objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(objectDest),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, bucketName)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectDest)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
  }
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
```

```
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
 return objects, err
}
```

```
// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
 objectKeys []string) error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
 })
 if err != nil || len(output.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucketName)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
  } else if len(output.Errors) > 0 {
   for _, outErr := range output.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *output.Errors[0].Message)
  }
 } else {
  for _, delObjs := range output.Deleted {
   err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
 time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.Key)
   }
  }
 }
 return err
}
```

```
// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie mit S3-Buckets und Objekten arbeiten.

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
 log.Printf("Let's list up to %v buckets for your account:", count)
```

```
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
 panic(err)
}
if len(buckets) == 0 {
 log.Println("You don't have any buckets!")
} else {
 if count > len(buckets) {
  count = len(buckets)
 }
 for _, bucket := range buckets[:count] {
  log.Printf("\t%v\n", *bucket.Name)
 }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
```

```
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting objects.")
 err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
 if err != nil {
  panic(err)
 }
 log.Println("Deleting bucket.")
 err = bucketBasics.DeleteBucket(ctx, bucketName)
 if err != nil {
  panic(err)
 }
```

```
  log.Printf("Deleting downloaded file %v.\n", downloadFileName)
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
 destinationBucket string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
```

```
  Key:        aws.String(objectKey),
 })
 if err != nil {
  var notActive *types.ObjectNotInActiveTierError
  if errors.As(err, &notActive) {
   log.Printf("Couldn't copy object %s from %s because the object isn't in the
 active tier.\n",
    objectKey, sourceBucket)
   err = notActive
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }
 return err
}
```

- API-Details finden Sie CopyObject in der AWS SDK für Go API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Buckets mit Standardkonfiguration.

```
import (
 "bytes"
```

```
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
 string) error {
 _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", name)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", name)
   err = exists
  }
```

```
 } else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
 }
 return err
}
```

Erstellen eines Buckets mit Object Lock und warten, bis er existiert.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
```

```
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- API-Details finden Sie CreateBucket in der AWS SDK für Go API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
 error {
 _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
```

```
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucketName)
   err = noBucket
  } else {
   log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
  }
 } else {
  err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
  } else {
   log.Printf("Deleted %s.\n", bucketName)
  }
 }
 return err
}
```

- API-Details finden Sie DeleteBucket in der AWS SDK für Go API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
```

```
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,
 versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
```

```
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
 key, bucket)
  } else {
   deleted = true
  }
 }
 return deleted, err
}
```

- API-Details finden Sie DeleteObject in der AWS SDK für Go API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
```

```
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
```

```
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  } else {
   for _, delObjs := range delOut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
     ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
  time.Minute)
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
  *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
  return err
}
```

- API-Details finden Sie DeleteObjects in der AWS SDK für Go API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
```

```
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
 objectKey, bucketName)
   err = noKey
  } else {
   log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
 err)
  }
  return err
 }
```

```
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
 }
 _, err = file.Write(body)
 return err
}
```

- API-Details finden Sie GetObject in der AWS SDK für Go API-Referenz.

`GetObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLegalHold`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
```

```
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }
```

```
 return status, err
}
```

- API-Details finden Sie GetObjectLegalHold in der AWS SDK für Go API-Referenz.

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLockConfiguration`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS SDK für Go API-Referenz.

`GetObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectRetention`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
```

```
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- API-Details finden Sie GetObjectRetention in der AWS SDK für Go API-Referenz.

`HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung`HeadBucket`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
```

```
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
 (bool, error) {
 _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error occurred.
 "+
     "Here's what happened: %v\n", bucketName, err)
   }
```

```
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- API-Details finden Sie HeadBucket in der AWS SDK für Go API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error)
 {
 var err error
 var output *s3.ListBucketsOutput
 var buckets []types.Bucket
 bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
 &s3.ListBucketsInput{})
 for bucketPaginator.HasMorePages() {
  output, err = bucketPaginator.NextPage(ctx)
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
    err = apiErr
   } else {
    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
   }
   break
  } else {
   buckets = append(buckets, output.Buckets...)
  }
 }
 return buckets, err
}
```

- API-Details finden Sie ListBuckets in der AWS SDK für Go API-Referenz.

`ListObjectVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectVersions`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
```

```
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- API-Details finden Sie ListObjectVersions in der AWS SDK für Go API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
 ([]types.Object, error) {
 var err error
 var output *s3.ListObjectsV2Output
 input := &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 }
 var objects []types.Object
 objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
 for objectPaginator.HasMorePages() {
  output, err = objectPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucketName)
    err = noBucket
   }
   break
  } else {
   objects = append(objects, output.Contents...)
  }
 }
 return objects, err
}
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für Go API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Platzieren Sie ein Objekt in einem Bucket, indem Sie die Low-Level-API verwenden.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
 objectKey string, fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   var apiErr smithy.APIError
   if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
     "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
     "or the multipart upload API (5TB max).", bucketName)
   } else {
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
     fileName, bucketName, objectKey, err)
   }
  } else {
   err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
   }
  }
 }
 return err
}
```

Laden Sie ein Objekt mithilfe eines Transfer-Managers in einen Bucket hoch.

```
import (
```

```
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
 contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
```

```
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- API-Details finden Sie PutObject in der AWS SDK für Go API-Referenz.

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLegalHold`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)
```

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- API-Details finden Sie PutObjectLegalHold in der AWS SDK für Go API-Referenz.

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLockConfiguration`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)
 error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
```

```
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an existing
 bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS SDK für Go API-Referenz.

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectRetention`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- API-Details finden Sie PutObjectRetention in der AWS SDK für Go API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Vorsignierungsaktionen umschließen.

```
import (
 "context"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(ctx, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
```

```
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object from
 a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
 objectKey string) (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(ctx,
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
```

```
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why: %v
\n", objectKey, err)
 }
 return request, err
}



func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName string,
 objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error) {
 request, err := presigner.PresignClient.PresignPostObject(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(options *s3.PresignPostOptions) {
  options.Expires = time.Duration(lifetimeSecs) * time.Second
 })
 if err != nil {
  log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v\n",
 bucketName, objectKey, err)
 }
 return request, nil
}
```

Führen Sie ein interaktives Beispiel aus, das vorsignierte URLs generiert und verwendet, um ein S3-Objekt hochzuladen, herunterzuladen und zu löschen.

```
import (
 "bytes"
 "context"
 "io"
 "log"
 "mime/multipart"
 "net/http"
 "os"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)



// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local file
 to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the object
 to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()
```

```
log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
 "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
```

```
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
```

```
 presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName, uploadKey,
 60)
 if err != nil {
  panic(err)
 }
 log.Printf("Got a presigned post request to url %v with values %v\n",
 presignPostRequest.URL, presignPostRequest.Values)
 log.Println("Using net/http multipart to send the request...")
 uploadFile, err = os.Open(uploadFilename)
 if err != nil {
  panic(err)
 }
 defer uploadFile.Close()
 multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
 presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
 if err != nil {
  panic(err)
 }
 log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
 multiPartResponse.StatusCode)

 log.Println("Let's presign a request to delete the object.")
 questioner.Ask("Press Enter when you're ready.")
 presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Got a presigned %v request to URL:\n\t%v\n",
 presignedDelRequest.Method,
  presignedDelRequest.URL)
 log.Println("Using net/http to send the request...")
 delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie einen HTTP-Request-Wrapper, der im Beispiel verwendet wird, um HTTP-Anfragen zu stellen.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
 *http.Response, err error) {
 postRequest, err := http.NewRequest("POST", url, body)
 if err != nil {
  return nil, err
 }
 postRequest.Header.Set("Content-Type", contentType)
 return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
```

```
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error) {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Sperren von Amazon-S3-Objekten

Das folgende Codebeispiel zeigt, wie Sie mit S3-Object-Lock-Features arbeiten.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
import (
 "context"
 "fmt"
 "log"
 "strings"

 "s3_object_lock/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
```

```
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx, fmt.Sprintf("%s.
%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
```

```
    if err != nil {
     switch err.(type) {
     case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
      log.Printf("Couldn't create bucket %s.\n", bucketName)
     default:
      panic(err)
     }
     break
    }
    scenario.resources.demoBuckets[info.name] = &DemoBucket{
     name:       bucketName,
     objectKeys: []string{},
    }
    log.Printf("Created bucket %s.\n", bucketName)
   }

   if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
   } else {
    success = true
   }
  }

  log.Println("S3 buckets created.")
  log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
```

```
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention period
 in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
```

```
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx context.Context)
 {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold to
 %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
      default:
       panic(err)
      }
     } else {
      log.Printf("Legal hold set on %s.\n", objKey)
     }
    }
   case 1:
```

```
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to %s
 in %s?\n"+
     "Reminder: Only a user with the s3:BypassGovernanceRetention permission is able
 to delete this object\n"+
     "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
    if scenario.questioner.AskBool(q, "y") {
     err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
      default:
       panic(err)
      }
     } else {
      log.Printf("Retention period set to 1 for %s.", objKey)
      bucket.retentionEnabled = true
     }
    }
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
```

```
 "List all objects and buckets.",
 "Attempt to delete an object.",
 "Attempt to delete an object with retention period bypass.",
 "Attempt to overwrite a file.",
 "View the retention settings for an object.",
 "View the legal hold settings for an object.",
 "Finish the workflow."}

choice := ListAll
for choice != Finish {
 objList := scenario.GetAllObjects(ctx)
 objChoices := scenario.makeObjectChoiceList(objList)
 choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
 switch choice {
 case ListAll:
  log.Println("The current objects in the example buckets are:")
  for _, objChoice := range objChoices {
   log.Println("\t", objChoice)
  }
 case DeleteObject, DeleteRetentionObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
  obj := objList[objChoice]
  deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Println("Nothing to delete.")
   default:
    panic(err)
   }
  } else if deleted {
   log.Printf("Object %s deleted.\n", obj.key)
  }
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
```

```
    case *types.NoSuchBucket:
     log.Println("Couldn't upload to nonexistent bucket.")
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded new content to object %s.\n", obj.key)
   }
  case ViewRetention:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
   obj := objList[objChoice]
   retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket, obj.key)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get retention configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if retention != nil {
    log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
   } else {
    log.Printf("Object %s does not have object retention configured.\n", obj.key)
   }
  case ViewLegalHold:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
   obj := objList[objChoice]
   legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket, obj.key,
obj.versionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if legalHold != nil {
    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
   } else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
   }
```

```
  case Finish:
   log.Println("Let's clean up.")
  }
  log.Println(strings.Repeat("-", 88))
 }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns them
 in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
```

```
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete S3-Aktionen umschließt.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "log"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
```

```
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
```

```
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3 object.
```

```
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
```

```
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an existing
 bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }
```

```
 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)
 error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
 contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else {
   err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
   }, time.Minute)
   if err != nil {
    log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
   } else {
    outKey = *output.Key
   }
  }
  return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

```
// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,
 versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
 time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
 key, bucket)
  } else {
   deleted = true
```

```
  }
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 } else {
  for _, delObjs := range delOut.Deleted {
   err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
    ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
 time.Minute)
```

```
    if err != nil {
     log.Printf("Failed attempt to wait for object %s to be deleted.\n",
 *delObjs.Key)
    } else {
     log.Printf("Deleted %s.\n", *delObjs.Key)
    }
   }
  }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "log"
 "s3_object_lock/actions"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario and
 handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
```

```
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
```

```
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucket.name)
  } else {
   log.Printf("Deleted %s.\n", bucket.name)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx, bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("No objects to get from %s.\n", bucket.name)
  default:
   panic(err)
  }
 }
 delObjects := make([]types.ObjectIdentifier, len(versions))
 for i, version := range versions {
  if lockConfig != nil && lockConfig.ObjectLockEnabled ==
 types.ObjectLockEnabledEnabled {
   status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
 *version.Key, *version.VersionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
```

```
     log.Printf("Couldn't determine legal hold status for %s in %s.\n",
 *version.Key, bucket.name)
    default:
     panic(err)
    }
   } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
    err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
 *version.VersionId, types.ObjectLockLegalHoldStatusOff)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
 bucket.name)
     default:
      panic(err)
     }
    }
   }
  }
  delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
 version.VersionId}
 }
 err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
 bucket.retentionEnabled)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold

- PutObjectLockConfiguration
- PutObjectRetention

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für Go V2

> **Note**
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die Upload- und Download-Manager verwenden, um die Daten in Teile zu zerlegen und sie gleichzeitig zu übertragen.

```
import (
 "bytes"
 "context"
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
 objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}
```

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie die Upload- und Download-Manager im Kontext verwenden.

```
import (
 "context"
 "crypto/rand"
 "log"
 "strings"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use Amazon
```

```
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
 that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := questioner.(*demotools.MockQuestioner)
   if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
    log.Println(r)
   }
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 large object demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
 bucket:",
  demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
```

```
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting object.")
 err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
 if err != nil {
  panic(err)
 }
 log.Println("Deleting bucket.")
 err = bucketBasics.DeleteBucket(ctx, bucketName)
 if err != nil {
  panic(err)
```

```
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
```

```
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Beispiele für Amazon SNS unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon SNS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SNS.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- API-Details finden Sie ListTopics in der AWS SDK für Go API-Referenz.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
```

```
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- API-Details finden Sie CreateTopic in der AWS SDK für Go API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- API-Details finden Sie DeleteTopic in der AWS SDK für Go API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main
```

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- API-Details finden Sie ListTopics in der AWS SDK für Go API-Referenz.

Veröffentlichen

Das folgende Codebeispiel zeigt die VerwendungPublish.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
```

```
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
 return err
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für Go -API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
```

```
    return "", err
   }
   attributes = map[string]string{"FilterPolicy": string(filterBytes)}
  }
  output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
   Protocol:              aws.String("sqs"),
   TopicArn:              aws.String(topicArn),
   Attributes:            attributes,
   Endpoint:              aws.String(queueArn),
   ReturnSubscriptionArn: true,
  })
  if err != nil {
   log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
    queueArn, topicArn, err)
  } else {
   subscriptionArn = *output.SubscriptionArn
  }

  return subscriptionArn, err
 }
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für Go -API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
```

```
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
 bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
 +
   "Deduplication IDs are either set in the message or are automatically generated
\n" +
   "from content using a hash function. If a message is successfully published to\n"
 +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted\n"
 +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
```

```
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
 ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
 string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))
```

```
var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n" +
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
 panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription
ARN %v.\n",
 queueName, topicName, subscriptionArn)

return subscriptionArn, filterPolicy != nil
```

```
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
 isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another messsage?
 (y/n) ", "y")
```

```
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}
```

```
// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the
\n"+
  "topic. You can select from several options for configuring the topic and the\n"+
  "subscriptions for the queues. You can then post to the topic and see the results
\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
```

```
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
 queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
 for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SNS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
```

```
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
```

```
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
```

```
 return err
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SQS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
```

```
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
```

```
    Resource:  aws.String(queueArn),
    Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

```
// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
```

```
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management Console
\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
```

```
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst

wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Beispiele für Amazon SQS unter Verwendung von SDK für Go V2

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS SDK für Go V2 mit Amazon SQS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
```

```
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- API-Details finden Sie ListQueues in der AWS SDK für Go API-Referenz.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
```

```
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}
```

- API-Details finden Sie CreateQueue in der AWS SDK für Go API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
}
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK für Go API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- API-Details finden Sie DeleteQueue in der AWS SDK für Go API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
```

```
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}
```

- API-Details finden Sie GetQueueAttributes in der AWS SDK für Go API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 sqsClient := sqs.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the queues for your account.")
 var queueUrls []string
 paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get queues. Here's why: %v\n", err)
   break
  } else {
   queueUrls = append(queueUrls, output.QueueUrls...)
  }
 }
 if len(queueUrls) == 0 {
  fmt.Println("You don't have any queues!")
 } else {
  for _, queueUrl := range queueUrls {
   fmt.Printf("\t%v\n", queueUrl)
  }
 }
}
```

- API-Details finden Sie ListQueues in der AWS SDK für Go API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
```

```
 }
 return messages, err
}
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für Go API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
```

```
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
```

```
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK für Go API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Go V2

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
import (
 "context"
 "encoding/json"
 "fmt"
 "log"
```

```
 "strings"
 "topics_and_queues/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
 bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
 +
```

```
   "Deduplication IDs are either set in the message or are automatically generated
\n" +
   "from content using a hash function. If a message is successfully published to\n"
 +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted\n"
 +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
 ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
```

```
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
 string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
    "If you add a filter to this subscription, then only the filtered messages\n" +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
```

```
    log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

    var toneSelections []string
    askAboutTones := true
    for askAboutTones {
     toneIndex := runner.questioner.AskChoice(
      "Enter the number of the tone you want to filter by:\n", ToneChoices)
     toneSelections = append(toneSelections, ToneChoices[toneIndex])
     askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
    }
    log.Printf("Your subscription will be filtered to only pass the following tones:
 %v\n", toneSelections)
    filterPolicy = map[string][]string{TONE_KEY: toneSelections}
   }
  }

  subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
  filterPolicy)
  if err != nil {
   panic(err)
  }
  log.Printf("The queue %v is now subscribed to the topic %v with the subscription
  ARN %v.\n",
   queueName, topicName, subscriptionArn)

  return subscriptionArn, filterPolicy != nil
 }

 func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
  isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
  var message string
  var groupId string
  var dedupId string
  var toneSelection string
  publishMore := true
  for publishMore {
   groupId = ""
   dedupId = ""
   toneSelection = ""
   message = runner.questioner.Ask("Enter a message to publish: ")
   if isFifoTopic {
```

```
   log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another messsage?
 (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
```

```
    }
    messages = append(messages, currentMsgs...)
   }
   if len(messages) == 0 {
    log.Printf("No messages were received by queue %v.\n", queueUrl)
   } else if len(messages) == 1 {
    log.Printf("One message was received by queue %v:\n", queueUrl)

   } else {
    log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
   }
   for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
     panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
   }

   if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
     panic(err)
    }
   }
  }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
```

```
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the
\n"+
  "topic. You can select from several options for configuring the topic and the\n"+
  "subscriptions for the queues. You can then post to the topic and see the results
\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
 queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
```

```
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
 for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SNS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)
```

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
```

```
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent to
 all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered according
 to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string) error
 {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
 }
 return err
}
```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete Amazon-SQS-Aktionen umschließt.

```
import (
 "context"
 "encoding/json"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/sqs"
 "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
 error) {
```

```
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
 an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
 the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
```

```
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  messages = result.Messages
```

```
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
 from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

Bereinigen von Ressourcen.

```
import (
 "context"
 "fmt"
 "log"
 "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 topicArn  string
 queueUrls []string
 snsActor  *actions.SnsActions
 sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong during cleanup. Use the AWS Management Console
\n" +
    "to remove any remaining resources that were created for this scenario.")
  }
 }()

 var err error
 if resources.topicArn != "" {
  log.Printf("Deleting topic %v.\n", resources.topicArn)
  err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
  if err != nil {
   panic(err)
  }
 }

 for _, queueUrl := range resources.queueUrls {
  log.Printf("Deleting queue %v.\n", queueUrl)
  err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
  if err != nil {
   panic(err)
  }
 }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
 for _, record := range event.Records {
  err := processMessage(record)
  if err != nil {
   return err
  }
 }
 fmt.Println("done")
 return nil
}

func processMessage(record events.SQSMessage) error {
 fmt.Printf("Processed message %s\n", record.Body)
 // TODO: Do interesting work based on the new message
 return nil
}

func main() {
 lambda.Start(handler)
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Go V2

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"
 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
 error) {
 batchItemFailures := []map[string]interface{}{}

 for _, message := range sqsEvent.Records {
  if len(message.Body) > 0 {
   // Your message processing condition here
   fmt.Printf("Successfully processed message: %s\n", message.Body)
  } else {
   // Message processing failed
   fmt.Printf("Failed to process message %s\n", message.MessageId)
   batchItemFailures = append(batchItemFailures, map[string]interface{}
{"itemIdentifier": message.MessageId})
  }
 }

 sqsBatchResponse := map[string]interface{}{
  "batchItemFailures": batchItemFailures,
 }
 return sqsBatchResponse, nil
}
```

```
func main() {
 lambda.Start(handler)
}
```

Codebeispiele für SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für JavaScript (v2) mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for JavaScript (v2) -Entwicklerhandbuch — Weitere Informationen zur Verwendung JavaScript mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- CloudWatch Beispiele zur Verwendung des SDK für JavaScript (v2)
- CloudWatch Beispiele für Ereignisse, die SDK für JavaScript (v2) verwenden
- CloudWatch Protokolliert Beispiele mit SDK für JavaScript (v2)
- DynamoDB-Beispiele, die SDK für JavaScript (v2) verwenden

- AWS Entity Resolution Beispiele zur Verwendung des SDK für JavaScript (v2)
- EventBridge Beispiele zur Verwendung des SDK für JavaScript (v2)
- Amazon Glacier-Beispiele, die das SDK für JavaScript (v2) verwenden
- IAM-Beispiele, die SDK für JavaScript (v2) verwenden
- Lambda-Beispiele, die SDK für JavaScript (v2) verwenden
- Amazon Pinpoint-Beispiele, die das SDK für JavaScript (v2) verwenden
- Amazon Pinpoint SMS- und Sprach-API-Beispiele unter Verwendung des SDK für JavaScript (v2)
- Amazon SNS-Beispiele, die das SDK für JavaScript (v2) verwenden
- Amazon SQS-Beispiele mit SDK für JavaScript (v2)
- AWS STS Beispiele zur Verwendung des SDK für JavaScript (v2)

CloudWatch Beispiele zur Verwendung des SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAlarms in der AWS SDK für JavaScript API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK für JavaScript API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DisableAlarmActions in der AWS SDK für JavaScript API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie EnableAlarmActions in der AWS SDK für JavaScript API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
```

```
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListMetrics in der AWS SDK für JavaScript API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
```

```
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutMetricAlarm in der AWS SDK für JavaScript API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutMetricData in der AWS SDK für JavaScript API-Referenz.

CloudWatch Beispiele für Ereignisse, die SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit CloudWatch Ereignissen Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
```

```
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

cwevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutEvents in der AWS SDK für JavaScript API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });
```

```
var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

cwevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutRule in der AWS SDK für JavaScript API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
```

```
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

cwevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutTargets in der AWS SDK für JavaScript API-Referenz.

CloudWatch Protokolliert Beispiele mit SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit CloudWatch Protokollen Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  filterName: "FILTER",
  logGroupName: "LOG_GROUP",
};

cwl.deleteSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteSubscriptionFilter in der AWS SDK für JavaScript API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  logGroupName: "GROUP_NAME",
  limit: 5,
};

cwl.describeSubscriptionFilters(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DescribeSubscriptionFilters in der AWS SDK für JavaScript API-Referenz.

`PutSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`PutSubscriptionFilter`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({ apiVersion: "2014-03-28" });

var params = {
  destinationArn: "LAMBDA_FUNCTION_ARN",
  filterName: "FILTER_NAME",
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP",
};

cwl.putSubscriptionFilter(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutSubscriptionFilter in der AWS SDK für JavaScript API-Referenz.

DynamoDB-Beispiele, die SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung. `BatchGetItem`

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: {
      Keys: [
```

```
        { KEY_NAME: { N: "KEY_VALUE_1" } },
        { KEY_NAME: { N: "KEY_VALUE_2" } },
        { KEY_NAME: { N: "KEY_VALUE_3" } },
      ],
      ProjectionExpression: "KEY_NAME, ATTRIBUTE",
    },
  },
};

ddb.batchGetItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function (element, index, array) {
      console.log(element);
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie BatchGetItem in der AWS SDK für JavaScript API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object
```

```
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  RequestItems: {
    TABLE_NAME: [
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
      {
        PutRequest: {
          Item: {
            KEY: { N: "KEY_VALUE" },
            ATTRIBUTE_1: { S: "ATTRIBUTE_1_VALUE" },
            ATTRIBUTE_2: { N: "ATTRIBUTE_2_VALUE" },
          },
        },
      },
    ],
  },
};

ddb.batchWriteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie BatchWriteItem in der AWS SDK für JavaScript API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  AttributeDefinitions: [
    {
      AttributeName: "CUSTOMER_ID",
      AttributeType: "N",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      AttributeType: "S",
    },
  ],
  KeySchema: [
    {
      AttributeName: "CUSTOMER_ID",
      KeyType: "HASH",
    },
    {
      AttributeName: "CUSTOMER_NAME",
      KeyType: "RANGE",
    },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1,
  },
  TableName: "CUSTOMER_LIST",
```

```
  StreamSpecification: {
    StreamEnabled: false,
  },
};

// Call DynamoDB to create the table
ddb.createTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Table Created", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateTable in der AWS SDK für JavaScript API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Element aus einer Tabelle.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
```

```
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "VALUE" },
  },
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Löschen Sie ein Element mithilfe des DynamoDB-Dokument-Clients aus einer Tabelle.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  Key: {
    HASH_KEY: VALUE,
  },
  TableName: "TABLE",
};

docClient.delete(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- API-Details finden Sie DeleteItem in der AWS SDK für JavaScript API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function (err, data) {
  if (err && err.code === "ResourceNotFoundException") {
    console.log("Error: Table not found");
  } else if (err && err.code === "ResourceInUseException") {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteTable in der AWS SDK für JavaScript API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: process.argv[2],
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DescribeTable in der AWS SDK für JavaScript API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Element aus einer Tabelle ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Key: {
    KEY_NAME: { N: "001" },
  },
  ProjectionExpression: "ATTRIBUTE_NAME",
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Rufen Sie ein Element mithilfe des DynamoDB-Dokument-Clients aus einer Tabelle ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "EPISODES_TABLE",
  Key: { KEY_NAME: VALUE },
};

docClient.get(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetItem in der AWS SDK für JavaScript API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });
```

```
// Call DynamoDB to retrieve the list of tables
ddb.listTables({ Limit: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListTables in der AWS SDK für JavaScript API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Element in eine Tabelle ein.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

var params = {
  TableName: "CUSTOMER_LIST",
  Item: {
    CUSTOMER_ID: { N: "001" },
    CUSTOMER_NAME: { S: "Richard Roe" },
  },
```

```
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Fügen Sie ein Element mithilfe des DynamoDB-Dokument-Clients in eine Tabelle ein.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  TableName: "TABLE",
  Item: {
    HASHKEY: VALUE,
    ATTRIBUTE_1: "STRING_VALUE",
    ATTRIBUTE_2: VALUE_2,
  },
};

docClient.put(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutItem in der AWS SDK für JavaScript API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({ apiVersion: "2012-08-10" });

var params = {
  ExpressionAttributeValues: {
    ":s": 2,
    ":e": 9,
    ":topic": "PHRASE",
  },
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  TableName: "EPISODES_TABLE",
};

docClient.query(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Items);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
  // Specify which items in the results are returned.
  FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
  // Define the expression attribute value, which are substitutes for the values you
 want to compare.
  ExpressionAttributeValues: {
    ":topic": { S: "SubTitle2" },
    ":s": { N: 1 },
    ":e": { N: 2 },
  },
  // Set the projection expression, which are the attributes that you want.
  ProjectionExpression: "Season, Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Success", data);
    data.Items.forEach(function (element, index, array) {
      console.log(
        "printing",
        element.Title.S + " (" + element.Subtitle.S + ")"
      );
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für JavaScript -API-Referenz.

Szenarien

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

SDK für JavaScript (v2)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion zum Aktualisieren einer Amazon DynamoDB-Tabelle mit Benutzerauswahlen verwendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

AWS Entity Resolution Beispiele zur Verwendung des SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) with Aktionen ausführen und gängige Szenarien implementieren AWS Entity Resolution.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Schemazuordnung.
- Erstellen Sie einen AWS Entity Resolution Arbeitsablauf.
- Starten Sie den passenden Auftrag für den Workflow.
- Rufen Sie Details für den passenden Auftrag ab.
- Rufen Sie die Schemazuordnung ab.
- Listen Sie alle Schemazuordnungen auf.
- Markieren Sie die Schemazuordnungsressource.
- Löschen Sie die AWS Entity Resolution Assets.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Entity Resolution Funktionen demonstriert.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
} from "@aws-sdk/client-cloudformation";
import {
  EntityResolutionClient,
  CreateSchemaMappingCommand,
  CreateMatchingWorkflowCommand,
  GetMatchingJobCommand,
  StartMatchingJobCommand,
  GetSchemaMappingCommand,
  ListSchemaMappingsCommand,
  TagResourceCommand,
  DeleteMatchingWorkflowCommand,
  DeleteSchemaMappingCommand,
  ConflictException,
  ValidationException,
} from "@aws-sdk/client-entityresolution";
import {
  DeleteObjectsCommand,
  DeleteBucketCommand,
  PutObjectCommand,
  S3Client,
  ListObjectsCommand,
} from "@aws-sdk/client-s3";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { readFile } from "node:fs/promises";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
```

```
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = `${data.inputs.entityResolutionStack}`;

/*The inputs for this example can be edited in the ../input.json.*/
import data from "../inputs.json" with { type: "json" };
const skipWhenErrors = (state) => state.errors.length > 0;
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  verbose: "false",
  skipWhen: skipWhenErrors,
});

const region = "eu-west-1";

const entityResolutionClient = new EntityResolutionClient({ region: region });
const cloudFormationClient = new CloudFormationClient({ region: region });
const s3Client = new S3Client({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "AWS Entity Resolution is a fully-managed machine learning service provided by " +
    "Amazon Web Services (AWS) that helps organizations extract, link, and " +
    "organize information from multiple data sources. It leverages natural " +
    "language processing and deep learning models to identify and resolve " +
    "entities, such as people, places, organizations, and products, " +
    "across structured and unstructured data.\n" +
    "\n" +
    "With Entity Resolution, customers can build robust data integration " +
    "pipelines to combine and reconcile data from multiple systems, databases, " +
    "and documents. The service can handle ambiguous, incomplete, or conflicting " +
    "information, and provide a unified view of entities and their relationships. "
 +
    "This can be particularly valuable in applications such as customer 360, " +
    "fraud detection, supply chain management, and knowledge management, where " +
    "accurate entity identification is crucial.\n" +
    "\n" +
```

```
    "The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x " +
    "provides a set of methods to programmatically interact with the AWS Entity " +
    "Resolution service. This allows developers to automate the entity extraction, "
 +
    "linking, and deduplication process as part of their data processing workflows.
 " +
    "With Entity Resolution, organizations can unlock the value of their data, " +
    "improve decision-making, and enhance customer experiences by having a reliable,
 " +
    "comprehensive view of their key entities.",

  { header: true },
);
const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "To prepare the AWS resources needed for this scenario application, the next step
 uploads " +
    "a CloudFormation template whose resulting stack creates the following
 resources:\n" +
    "- An AWS Glue Data Catalog table \n" +
    "- An AWS IAM role \n" +
    "- An AWS S3 bucket \n" +
    "- An AWS Entity Resolution Schema \n" +
    "It can take a couple minutes for the Stack to finish creating the resources.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/entity-resolution-basics/entity-
resolution-basics-template.yml`,
        "utf8",
      );
      await cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: cloudFormationClient },
```

```
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );

      state.entityResolutionRole = stack.Stacks[0].Outputs[1];
      state.jsonGlueTable = stack.Stacks[0].Outputs[2];
      state.CSVGlueTable = stack.Stacks[0].Outputs[3];
      state.glueDataBucket = stack.Stacks[0].Outputs[0];
      state.stackName = stack.StackName;
      console.log(state.glueDataBucket);
      console.log(
        `The  ARN of the EntityResolution Role is
${state.entityResolutionRole.OutputValue}`,
      );
      console.log(
        `The ARN of the Json Glue Table is ${state.jsonGlueTable.OutputValue}`,
      );
      console.log(
        `The ARN of the CSV Glue Table is ${state.CSVGlueTable.OutputValue}`,
      );
      console.log(
        `The name of the Glue Data Bucket is ${state.glueDataBucket.OutputValue}\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the following JSON in ../data.json to the
${state.glueDataBucket.OutputValue} S3 bucket...`,
      );
      const bucketName = state.glueDataBucket.OutputValue;

      const putObjectParams = {
        Bucket: bucketName,
```

```
        Key: "jsonData/data.json",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.json`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.json file data uploaded to the ${state.glueDataBucket.OutputValue}
 S3 bucket.\n`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
    try {
      console.log(
        `Uploading the CSV data in ../data.csv to the
 ${state.glueDataBucket.OutputValue} S3 bucket...`,
      );

      const bucketName = state.glueDataBucket.OutputValue;
      const putObjectParams = {
        Bucket: bucketName,
        Key: "csvData/data.csv",
        Body: await readFileSync(
          `${__dirname}/../../../../javascriptv3/example_code/entityresolution/
data.csv`,
        ),
      };
      const command = new PutObjectCommand(putObjectParams);
      const response = await s3Client.send(command);
      console.log(
        `../data.csv file data uploaded to the ${state.glueDataBucket.OutputValue}
 S3 bucket.`,
      );
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);
```

```
const displayCreateSchemaMapping = new ScenarioOutput(
  "displayCreateSchemaMapping",
  "1. Create Schema Mapping" +
    "Entity Resolution schema mapping aligns and integrates data from " +
    "multiple sources by identifying and matching corresponding entities " +
    "like customers or products. It unifies schemas, resolves conflicts, " +
    "and uses machine learning to link related entities, enabling a " +
    "consolidated, accurate view for improved data quality and decision-making." +
    "\n" +
    "In this example, the schema mapping lines up with the fields in the JSON and
 CSV objects. That is, " +
    " it contains these fields: id, name, and email. ",
);

const sdkCreateSchemaMapping = new ScenarioAction(
  "sdkCreateSchemaMapping",
  async (/** @type {State} */ state) => {
    const createSchemaMappingParamsJson = {
      schemaName: `${data.inputs.schemaNameJson}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
      ],
    };
    const createSchemaMappingParamsCSV = {
      schemaName: `${data.inputs.schemaNameCSV}`,
      mappedInputFields: [
        {
          fieldName: "id",
          type: "UNIQUE_ID",
        },
        {
          fieldName: "name",
          type: "NAME",
```

```
        },
        {
          fieldName: "email",
          type: "EMAIL_ADDRESS",
        },
        {
          fieldName: "phone",
          type: "PROVIDER_ID",
          subType: "STRING",
        },
      ],
    };
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsJson,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameJson = response.schemaName;
      state.schemaArn = response.schemaArn;
      state.idOutputAttribute = response.mappedInputFields[0].fieldName;
      state.nameOutputAttribute = response.mappedInputFields[1].fieldName;
      state.emailOutputAttribute = response.mappedInputFields[2].fieldName;

      console.log("The JSON schema mapping name is ", state.schemaNameJson);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema mapping already exists: ${caught.message} \n Exiting
program.`,
        );
        return;
      }
    }
    try {
      const command = new CreateSchemaMappingCommand(
        createSchemaMappingParamsCSV,
      );
      const response = await entityResolutionClient.send(command);
      state.schemaNameCSV = response.schemaName;
      state.phoneOutputAttribute = response.mappedInputFields[3].fieldName;
      console.log("The CSV schema mapping name is ", state.schemaNameCSV);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
```

```
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateMatchingWorkflow = new ScenarioOutput(
  "displayCreateMatchingWorkflow",
  "2. Create an AWS Entity Resolution Workflow. " +
    "An Entity Resolution matching workflow identifies and links records " +
    "across datasets that represent the same real-world entity, such as " +
    "customers or products. Using techniques like schema mapping, " +
    "data profiling, and machine learning algorithms, " +
    "it evaluates attributes like names or emails to detect duplicates " +
    "or relationships, even with variations or inconsistencies. " +
    "The workflow outputs consolidated, de-duplicated data." +
    "\n" +
    "We will use the machine learning-based matching technique.",
);

const sdkCreateMatchingWorkflow = new ScenarioAction(
  "sdkCreateMatchingWorkflow",
  async (/** @type {State} */ state) => {
    const createMatchingWorkflowParams = {
      roleArn: `${state.entityResolutionRole.OutputValue}`,
      workflowName: `${data.inputs.workflowName}`,
      description: "Created by using the AWS SDK for JavaScript (v3).",
      inputSourceConfig: [
        {
          inputSourceARN: `${state.jsonGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameJson}`,
          applyNormalization: false,
        },
        {
          inputSourceARN: `${state.CSVGlueTable.OutputValue}`,
          schemaName: `${data.inputs.schemaNameCSV}`,
          applyNormalization: false,
        },
      ],
      outputSourceConfig: [
        {
          outputS3Path: `s3://${state.glueDataBucket.OutputValue}/eroutput`,
```

```
          output: [
            {
              name: state.idOutputAttribute,
            },
            {
              name: state.nameOutputAttribute,
            },
            {
              name: state.emailOutputAttribute,
            },
            {
              name: state.phoneOutputAttribute,
            },
          ],
          applyNormalization: false,
        },
      ],
      resolutionTechniques: { resolutionType: "ML_MATCHING" },
    };
    try {
      const command = new CreateMatchingWorkflowCommand(
        createMatchingWorkflowParams,
      );
      const response = await entityResolutionClient.send(command);
      state.workflowArn = response.workflowArn;
      console.log(
        `Workflow created successfully.\n The workflow ARN is:
${response.workflowArn}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
program.`,
        );
        return;
      }
      if (caught instanceof ValidationException) {
        console.error(
          `There was a validation exception: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
```

```
  },
);
const displayMatchingJobOfWorkflow = new ScenarioOutput(
  "displayMatchingJobOfWorkflow",
  "3. Start the matching job of the workflow",
);

const sdkMatchingJobOfWorkflow = new ScenarioAction(
  "sdk",
  async (/** @type {State} */ state) => {
    const matchingJobOfWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    try {
      const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
      const response = await entityResolutionClient.send(command);
      state.jobID = response.jobId;
      console.log(`Job ID: ${state.jobID} \n
The matching job was successfully started.`);
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The matching workflow already exists: ${caught.message} \n Exiting
 program.`,
        );
        return;
      }
    }
  },
);

const displayGetDetailsforJob = new ScenarioOutput(
  "displayGetDetailsforJob",
  `4. While the matching job is running, let's look at other API methods. First,
 let's get details for the job `,
);

const sdkGetDetailsforJob = new ScenarioAction(
  "sdkGetDetailsforJob",
  async (/** @type {State} */ state) => {
    const getDetailsforJobParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${state.jobID}`,
    };
```

```
    try {
      const command = new GetMatchingJobCommand(getDetailsforJobParams);
      const response = await entityResolutionClient.send(command);
      state.Status = response.status;
      state.response = response;
      console.log(`Job status: ${state.Status} `);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetSchemaMappingJson = new ScenarioOutput(
  "displayGetSchemaMappingJson",
  "5. Get the schema mapping for the JSON data.",
);

const sdkGetSchemaMappingJson = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const getSchemaMappingJsonParams = {
      schemaName: `${data.inputs.schemaNameJson}`,
    };
    try {
      const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
      const response = await entityResolutionClient.send(command);
      console.log("Schema·mapping·ARN·is:·", response.schemaArn);
      const resultMappings = response.mappedInputFields;
      const noOfResultMappings = resultMappings.length;
      for (let i = 0; i < noOfResultMappings; i++) {
        console.log(
          `Attribute name: ${resultMappings[i].fieldName} `,
          `Attribute type: ${resultMappings[i].type}`,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayListSchemaMappings = new ScenarioOutput(
```

```
  "displayListSchemaMappings",
  "6. List Schema Mappings.",
);

const sdkListSchemaMappings = new ScenarioAction(
  "sdkListSchemaMappings",
  async (/** @type {State} */ state) => {
    try {
      const command = new ListSchemaMappingsCommand({});
      const response = await entityResolutionClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayTagTheJsonSchema = new ScenarioOutput(
  "display",
  "7. Tag the resource. \n" +
    "Tags can help you organize and categorize your Entity Resolution resources. " +
    "You can also use them to scope user permissions by granting a user permission "
 +
    "to access or change only resources with certain tag values. " +
    "In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged. For
 this example, " +
    "the SchemaMapping is tagged.",
);

const sdkTagTheJsonSchema = new ScenarioAction(
  "sdkGetSchemaMappingJson",
  async (/** @type {State} */ state) => {
    const tagResourceCommandParams = {
      resourceArn: state.schemaArn,
      tags: {
        tag1: "tag1Value",
        tag2: "tag2Value",
      },
```

```
    };
    try {
      const command = new TagResourceCommand(tagResourceCommandParams);
      const response = await entityResolutionClient.send(command);
      console.log("Successfully tagged the resource.");
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayGetJobInfo = new ScenarioOutput(
  "displayGetJobInfo",
  "8. View the results of the AWS Entity Resolution Workflow.\n " +
    "Please perform this task manually in the AWS Management Console. ",
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "9. Delete the resources \n" +
    "You cannot delete a workflow that is in a running state. So this will take ~30
 minutes.\n" +
    "If you don't want to delete the resources, simply exit this application.",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    console.log(
      "You selected to delete the resources. This will take about 30 minutes.",
    );
    await wait(1800);
    const bucketName = state.glueDataBucket.OutputValue;
    try {
      const emptyBucket = async ({ bucketName }) => {
        const listObjectsCommand = new ListObjectsCommand({
          Bucket: bucketName,
        });
        const { Contents } = await s3Client.send(listObjectsCommand);
        const keys = Contents.map((c) => c.Key);

        const deleteObjectsCommand = new DeleteObjectsCommand({
          Bucket: bucketName,
```

```
          Delete: { Objects: keys.map((key) => ({ Key: key })) },
        });
        await s3Client.send(deleteObjectsCommand);
        console.log(`Bucket ${bucketName} emptied successfully.\n`);
      };
      await emptyBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteBucket = async ({ bucketName }) => {
        const command = new DeleteBucketCommand({ Bucket: bucketName });
        await s3Client.send(command);
        console.log(`Bucket ${bucketName} deleted successfully.\n`);
      };
      await deleteBucket({ bucketName });
    } catch (error) {
      console.log("error ", error);
    }
    try {
      console.log(
        "Now we will delete the CloudFormation stack, which deletes the resources
that were created at the beginning of the scenario.",
      );
      const deleteStackParams = { StackName: `${state.stackName}` };
      const command = new DeleteStackCommand(deleteStackParams);
      const response = await cloudFormationClient.send(command);
      console.log("CloudFormation stack deleted successfully.");
    } catch (error) {
      console.log("error ", error);
    }
    try {
      const deleteWorkflowParams = {
        workflowName: `${data.inputs.workflowName}`,
      };
      const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
      const response = await entityResolutionClient.send(command);
      console.log("Workflow deleted successfully!");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `Job associated with workflow ${data.inputs.workflowName} is still
running, so can't be deleted.
```

```
          Neither can schemas ${data.inputs.schemaNameJson} and
${data.inputs.schemaNameCSV} associated with it. Please confirm this workflow is
finished in the AWS Management Console, then delete it manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteJSONschemaMapping = {
        schemaName: `${data.inputs.schemaNameJson}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteJSONschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully. ");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameJson} can't be deleted because it is
associated with workflow
           ${data.inputs.workflowName}, which is still running. Please confirm this
workflow is finished in the AWS Management Console, then delete it manually.`,
        );
        throw caught;
      }
    }
    try {
      const deleteCSVschemaMapping = {
        schemaName: `${data.inputs.schemaNameCSV}`,
      };
      const command = new DeleteSchemaMappingCommand(deleteCSVschemaMapping);
      const response = await entityResolutionClient.send(command);
      console.log("Schema mapping deleted successfully.");
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `The schema ${data.inputs.schemaNameCSV} can't be deleted because it
is associated with workflow ${data.inputs.workflowName}, which is still running.
Please confirm this workflow is finished in the AWS Management Console, then delete
it manually.`,
        );
        throw caught;
      }
    }
  },
```

```
  {
    skipWhen: (/** @type {State} */ state) =>
      state.confirmDeleteResources === "",
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you " +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming languages,
 have a look at: " +
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Entity Resolution Basics Scenario", [
  greet,
  pressEnter,
  displayBuildCloudFormationStack,
  sdkBuildCloudFormationStack,
  pressEnter,
  displayCreateSchemaMapping,
  sdkCreateSchemaMapping,
  pressEnter,
  displayCreateMatchingWorkflow,
  sdkCreateMatchingWorkflow,
  pressEnter,
  displayMatchingJobOfWorkflow,
  sdkMatchingJobOfWorkflow,
  pressEnter,
  displayGetDetailsforJob,
  sdkGetDetailsforJob,
  pressEnter,
  displayGetSchemaMappingJson,
  sdkGetSchemaMappingJson,
  pressEnter,
  displayListSchemaMappings,
  sdkListSchemaMappings,
  pressEnter,
  displayTagTheJsonSchema,
  sdkTagTheJsonSchema,
  pressEnter,
  displayGetJobInfo,
```

```
  pressEnter,
  displayDeleteResources,
  pressEnter,
  sdkDeleteResources,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows
  - ListSchemaMappings
  - StartMatchingJob
  - TagResource

EventBridge Beispiele zur Verwendung des SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
```

```
      DetailType: "appRequestSubmitted",
      Resources: ["RESOURCE_ARN"],
      Source: "com.company.app",
    },
  ],
};

ebevents.putEvents(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- API-Details finden Sie PutEvents in der AWS SDK für JavaScript API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN",
  ScheduleExpression: "rate(5 minutes)",
```

```
  State: "ENABLED",
};

ebevents.putRule(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- API-Details finden Sie PutRule in der AWS SDK für JavaScript API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({ apiVersion: "2015-10-07" });

var params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN",
      Id: "myEventBridgeTarget",
    },
  ],
```

```
};

ebevents.putTargets(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- API-Details finden Sie PutTargets in der AWS SDK für JavaScript API-Referenz.

Amazon Glacier-Beispiele, die das SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit Amazon Glacier Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVault`

Das folgende Codebeispiel zeigt die Verwendung`CreateVault`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateVault in der AWS SDK für JavaScript API-Referenz.

`UploadArchive`

Das folgende Codebeispiel zeigt die Verwendung`UploadArchive`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
// Call Glacier to upload the archive.
```

```
glacier.uploadArchive(params, function (err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UploadArchive in der AWS SDK für JavaScript API-Referenz.

`UploadMultipartPart`

Das folgende Codebeispiel zeigt die Verwendung`UploadMultipartPart`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines mehrteiligen Uploads mit 1-Megabyte-Blöcken eines Buffer-Objekts.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
```

```
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
      };

    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };

      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
```

```
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UploadMultipartPart in der AWS SDK für JavaScript API-Referenz.

IAM-Beispiele, die SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit IAM Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie AttachRolePolicy in der AWS SDK für JavaScript API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateAccessKey in der AWS SDK für JavaScript API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateAccountAlias in der AWS SDK für JavaScript API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreatePolicy in der AWS SDK für JavaScript API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
```

```
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateUser in der AWS SDK für JavaScript API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAccessKey in der AWS SDK für JavaScript API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAccountAlias in der AWS SDK für JavaScript API-Referenz.

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteServerCertificate in der AWS SDK für JavaScript API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteUser in der AWS SDK für JavaScript API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
```

```
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DetachRolePolicy in der AWS SDK für JavaScript API-Referenz.

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung`GetAccessKeyLastUsed`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetAccessKeyLastUsed in der AWS SDK für JavaScript API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetPolicy in der AWS SDK für JavaScript API-Referenz.

`GetServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetServerCertificate`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetServerCertificate in der AWS SDK für JavaScript API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListAccessKeys in der AWS SDK für JavaScript API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListAccountAliases in der AWS SDK für JavaScript API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListServerCertificates in der AWS SDK für JavaScript API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
```

```
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListUsers in der AWS SDK für JavaScript API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
```

```
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateAccessKey in der AWS SDK für JavaScript API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateServerCertificate`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateServerCertificate in der AWS SDK für JavaScript API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateUser in der AWS SDK für JavaScript API-Referenz.

Lambda-Beispiele, die SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

SDK für JavaScript (v2)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion zum Aktualisieren einer Amazon DynamoDB-Tabelle mit Benutzerauswahlen verwendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

Amazon Pinpoint-Beispiele, die das SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
"use strict";

const AWS = require("aws-sdk");

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2";
```

```
// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https:\/\/aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";

// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
```

```
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: senderAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

//Try to send the email.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
  } else {
    console.log(
      "Email sent! Message ID: ",
```

```
      data["MessageResponse"]["Result"][toAddress]["MessageId"]
    );
  }
});
```

Senden Sie eine SMS-Nachricht.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number.  For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";
```

```
// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
var senderId = "MySenderID";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: applicationId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

//Try to send the message.
pinpoint.sendMessages(params, function (err, data) {
  // If something goes wrong, print an error message.
```

```
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
    console.log(
      "Message sent! " +
        data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
    );
  }
});
```

- API-Details finden Sie SendMessages in der AWS SDK für JavaScript API-Referenz.

Amazon Pinpoint SMS- und Sprach-API-Beispiele unter Verwendung des SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für JavaScript (v2) mit Amazon Pinpoint SMS and Voice API Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendVoiceMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendVoiceMessage`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"use strict";

var AWS = require("aws-sdk");

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best results,
 you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage =
  "<speak>" +
  "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> " +
```

```
  "using the <break strength='weak'/>AWS SDK for JavaScript in Node.js. " +
  "<amazon:effect phonation='soft'>Thank you for listening." +
  "</amazon:effect>" +
  "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({ profile: "default" });
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({ region: aws_region });

//Create a new Pinpoint object.
var pinpointsmsvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId,
    },
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber,
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function (err, data) {
  // If something goes wrong, print an error message.
  if (err) {
    console.log(err.message);
    // Otherwise, show the unique ID for the message.
  } else {
```

```
    console.log("Message sent! Message ID: " + data["MessageId"]);
  }
});
```

- API-Details finden Sie SendVoiceMessage in der AWS SDK für JavaScript API-Referenz.

Amazon SNS-Beispiele, die das SDK für JavaScript (v2) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetTopicAttributes in der AWS SDK für JavaScript API-Referenz.

Amazon SQS-Beispiele mit SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung`ChangeMessageVisibility`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Amazon-SQS-Nachricht und ändern Sie deren Timeout-Sichtbarkeit.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region to us-west-2
AWS.config.update({ region: "us-west-2" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else {
    // Make sure we have a message
    if (data.Messages != null) {
      var visibilityParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
        VisibilityTimeout: 20, // 20 second timeout
```

```
      };
      sqs.changeMessageVisibility(visibilityParams, function (err, data) {
        if (err) {
          console.log("Delete Error", err);
        } else {
          console.log("Timeout Changed", data);
        }
      });
    } else {
      console.log("No messages to change");
    }
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ChangeMessageVisibility in der AWS SDK für JavaScript API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
```

```
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    DelaySeconds: "60",
    MessageRetentionPeriod: "86400",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Erstellen Sie eine Amazon-SQS-Warteschlange, die auf eine Nachricht wartet.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};

sqs.createQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- API-Details finden Sie CreateQueue in der AWS SDK für JavaScript API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Amazon-SQS-Nachrichten ab und löschen Sie sie.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle,
```

```
    };
    sqs.deleteMessage(deleteParams, function (err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteMessage in der AWS SDK für JavaScript API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueUrl: "SQS_QUEUE_URL",
};
```

```
sqs.deleteQueue(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteQueue in der AWS SDK für JavaScript API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die URL für eine Amazon-SQS-Warteschlange ab.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  QueueName: "SQS_QUEUE_NAME",
};

sqs.getQueueUrl(params, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetQueueUrl in der AWS SDK für JavaScript API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Ihre Amazon-SQS-Warteschlangen auf.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {};

sqs.listQueues(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListQueues in der AWS SDK für JavaScript API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
```

```
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ReceiveMessage in der AWS SDK für JavaScript API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Nachricht an eine Amazon-SQS-Warteschlange.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create an SQS service object
var sqs = new AWS.SQS({ apiVersion: "2012-11-05" });

var params = {
  // Remove DelaySeconds parameter and value for FIFO queues
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
```

```
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
    "Information about current NY Times fiction bestseller for week of 12/11/2016.",
  // MessageDeduplicationId: "TheWhistler",  // Required for FIFO queues
  // MessageGroupId: "Group1",  // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL",
};

sqs.sendMessage(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.MessageId);
  }
});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie SendMessage in der AWS SDK für JavaScript API-Referenz.

AWS STS Beispiele zur Verwendung des SDK für JavaScript (v2)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v2) with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für JavaScript (v2)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});
```

```
//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- API-Details finden Sie AssumeRole in der AWS SDK für JavaScript API-Referenz.

Codebeispiele für SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für JavaScript (v3) mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for JavaScript (v3) -Entwicklerhandbuch — Weitere Informationen zur Verwendung JavaScript mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.

- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- Beispiele für API-Gateways, die SDK für JavaScript (v3) verwenden
- Aurora-Beispiele, die SDK für JavaScript (v3) verwenden
- Beispiele für Auto Scaling mit SDK für JavaScript (v3)
- Amazon Bedrock-Beispiele, die SDK für JavaScript (v3) verwenden
- Beispiele für Amazon Bedrock Runtime mit SDK für JavaScript (v3)
- Beispiele für Amazon Bedrock Agents, die SDK für JavaScript (v3) verwenden
- Amazon Bedrock Agents Runtime-Beispiele, die das SDK für JavaScript (v3) verwenden
- CloudWatch Beispiele zur Verwendung des SDK für JavaScript (v3)
- CloudWatch Beispiele für Ereignisse, die SDK für JavaScript (v3) verwenden
- CloudWatch Protokolliert Beispiele mithilfe des SDK für JavaScript (v3)
- CodeBuild Beispiele mit SDK für JavaScript (v3)
- Beispiele für Amazon Cognito Identity mit SDK für JavaScript (v3)
- Beispiele für Amazon Cognito Identity Provider, die SDK für JavaScript (v3) verwenden
- Amazon Comprehend-Beispiele mit SDK für JavaScript (v3)
- Amazon DocumentDB-Beispiele, die das SDK für JavaScript (v3) verwenden
- DynamoDB-Beispiele, die SDK für (v3) verwenden JavaScript
- Amazon EC2-Beispiele mit SDK für (v3) JavaScript
- Elastic Load Balancing — Beispiele für Version 2 mit SDK für JavaScript (v3)
- AWS Entity Resolution Beispiele mit SDK für JavaScript (v3)
- EventBridge Beispiele zur Verwendung des SDK für JavaScript (v3)
- Amazon Glacier-Beispiele mit SDK für JavaScript (v3)
- AWS Glue Beispiele zur Verwendung des SDK für JavaScript (v3)
- HealthImaging Beispiele zur Verwendung des SDK für JavaScript (v3)
- IAM-Beispiele, die SDK für JavaScript (v3) verwenden
- AWS IoT SiteWise Beispiele mit SDK für JavaScript (v3)

- Kinesis-Beispiele, die SDK für JavaScript (v3) verwenden
- Lambda-Beispiele mit SDK für JavaScript (v3)
- Amazon Lex: Beispiele für die Verwendung des SDK für JavaScript (v3)
- Beispiele für Amazon Locations mit SDK für JavaScript (v3)
- AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für JavaScript (v3)
- Amazon MSK-Beispiele, die das SDK für JavaScript (v3) verwenden
- Amazon Personalisieren Sie Beispiele mithilfe des SDK für JavaScript (v3)
- Beispiele für Amazon Personalize Events mithilfe des SDK für JavaScript (v3)
- Amazon Personalize Runtime-Beispiele mithilfe des SDK für JavaScript (v3)
- Amazon Pinpoint-Beispiele mit SDK für JavaScript (v3)
- Beispiele für Amazon Polly, die SDK für JavaScript (v3) verwenden
- Amazon RDS-Beispiele, die SDK für JavaScript (v3) verwenden
- Beispiele für Amazon RDS Data Service mit SDK für JavaScript (v3)
- Amazon Redshift-Beispiele mit SDK für JavaScript (v3)
- Beispiele für Amazon Rekognition mit SDK für JavaScript (v3)
- Amazon S3-Beispiele mit SDK für JavaScript (v3)
- SageMaker KI-Beispiele, die SDK für JavaScript (v3) verwenden
- Secrets Manager-Beispiele, die SDK für JavaScript (v3) verwenden
- Amazon SES-Beispiele mit SDK für JavaScript (v3)
- Amazon SNS-Beispiele, die das SDK für JavaScript (v3) verwenden
- Amazon SQS-Beispiele, die SDK für JavaScript (v3) verwenden
- Beispiele für Step Functions unter Verwendung des SDK für JavaScript (v3)
- AWS STS Beispiele zur Verwendung des SDK für JavaScript (v3)
- Support Beispiele zur Verwendung des SDK für JavaScript (v3)
- Beispiele für Systems Manager, die SDK für JavaScript (v3) verwenden
- Amazon Textract-Beispiele mit SDK für JavaScript (v3)
- Amazon Transcribe-Beispiele mit SDK für (v3) JavaScript
- Beispiele für Amazon Translate, die das SDK für JavaScript (v3) verwenden

Beispiele für API-Gateways, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Aurora-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Aurora Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für JavaScript (v3)

Zeigt, wie Sie AWS SDK für JavaScript (v3) verwenden, um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte per E-Mail mithilfe von Amazon Simple Email Service (Amazon SES) versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Auto Scaling mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Auto Scaling Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AttachLoadBalancerTargetGroups`

Das folgende Codebeispiel zeigt die Verwendung`AttachLoadBalancerTargetGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new AutoScalingClient({});
await client.send(
  new AttachLoadBalancerTargetGroupsCommand({
    AutoScalingGroupName: NAMES.autoScalingGroupName,
    TargetGroupARNs: [state.targetGroupArn],
  }),
);
```

- API-Details finden Sie AttachLoadBalancerTargetGroups in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
```

```
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
```

```
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
```

```
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
```

```
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
```

```
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
```

```
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
```

```
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
```

```
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
```

```
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
```

```
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
```

```
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
```

```
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
```

```
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }
```

```
      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
```

```
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
```

```
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
```

```
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
```

```
    new ScenarioOutput("testBrokenDependency", (state) =>
      MESSAGES.demoTestBrokenDependency.replace(
        "${TABLE_NAME}",
        state.badTableName,
      ),
    ),
    ...statusSteps,
    new ScenarioInput(
      "staticResponseConfirmation",
      MESSAGES.demoStaticResponseConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("staticResponse", async (state) => {
      if (!state.staticResponseConfirmation) {
        process.exit();
      } else {
        const client = new SSMClient({});
        await client.send(
          new PutParameterCommand({
            Name: NAMES.ssmFailureResponseKey,
            Value: "static",
            Overwrite: true,
            Type: "String",
          }),
        );
      }
    }),
    new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
    ...statusSteps,
    new ScenarioInput(
      "badCredentialsConfirmation",
      MESSAGES.demoBadCredentialsConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("badCredentialsExit", (state) => {
      if (!state.badCredentialsConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction("fixDynamoDBName", async () => {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
```

```
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
```

```
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
```

```
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
```

```
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
```

```
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
```

```
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
```

```
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
```

```
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);
```

```
      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
```

```
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
```

```
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
```

```
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
```

```
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
```

```
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
```

```
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
```

```
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
```

```
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
```

```
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
```

```
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers

- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Amazon Bedrock-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock beginnen können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

const REGION = "us-east-1";
const client = new BedrockClient({ region: REGION });

export const main = async () => {
  const command = new ListFoundationModelsCommand({});

  const response = await client.send(command);
  const models = response.modelSummaries;

  console.log("Listing the available Bedrock foundation models:");

  for (const model of models) {
    console.log("=".repeat(42));
    console.log(` Model: ${model.modelId}`);
    console.log("-".repeat(42));
    console.log(` Name: ${model.modelName}`);
    console.log(` Provider: ${model.providerName}`);
    console.log(` Model ARN: ${model.modelArn}`);
    console.log(` Input modalities: ${model.inputModalities}`);
    console.log(` Output modalities: ${model.outputModalities}`);
    console.log(` Supported customizations: ${model.customizationsSupported}`);
    console.log(` Supported inference types: ${model.inferenceTypesSupported}`);
    console.log(` Lifecycle status: ${model.modelLifecycle.status}`);
    console.log(`${"=".repeat(42)}\n`);
  }
```

```
  const active = models.filter(
    (m) => m.modelLifecycle.status === "ACTIVE",
  ).length;
  const legacy = models.filter(
    (m) => m.modelLifecycle.status === "LEGACY",
  ).length;

  console.log(
    `There are ${active} active and ${legacy} legacy foundation models in
 ${REGION}.`,
  );

  return response;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für JavaScript API-Referenz.

Aktionen

`GetFoundationModel`

Das folgende Codebeispiel zeigt die Verwendung`GetFoundationModel`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Details über ein Basismodell ab.

```
import { fileURLToPath } from "node:url";
```

```
import {
  BedrockClient,
  GetFoundationModelCommand,
} from "@aws-sdk/client-bedrock";

/**
 * Get details about an Amazon Bedrock foundation model.
 *
 * @return {FoundationModelDetails} - The list of available bedrock foundation
 models.
 */
export const getFoundationModel = async () => {
  const client = new BedrockClient();

  const command = new GetFoundationModelCommand({
    modelIdentifier: "amazon.titan-embed-text-v1",
  });

  const response = await client.send(command);

  return response.modelDetails;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const model = await getFoundationModel();
  console.log(model);
}
```

- API-Details finden Sie GetFoundationModel in der AWS SDK für JavaScript API-Referenz.

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die verfügbaren Basismodelle auf.

```
import { fileURLToPath } from "node:url";

import {
  BedrockClient,
  ListFoundationModelsCommand,
} from "@aws-sdk/client-bedrock";

/**
 * List the available Amazon Bedrock foundation models.
 *
 * @return {FoundationModelSummary[]} - The list of available bedrock foundation
 models.
 */
export const listFoundationModels = async () => {
  const client = new BedrockClient();

  const input = {
    // byProvider: 'STRING_VALUE',
    // byCustomizationType: 'FINE_TUNING' || 'CONTINUED_PRE_TRAINING',
    // byOutputModality: 'TEXT' || 'IMAGE' || 'EMBEDDING',
    // byInferenceType: 'ON_DEMAND' || 'PROVISIONED',
  };

  const command = new ListFoundationModelsCommand(input);

  const response = await client.send(command);

  return response.modelSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const models = await listFoundationModels();
  console.log(models);
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Bedrock Runtime mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Szenarien
- Amazon Nova
- Amazon Nova Canvas
- Anthropic Claude
- Cohere Command
- Meta Llama
- Mistral AI

Erste Schritte

Hallo Amazon Bedrock Runtime

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock Runtime beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
```

```
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Send a prompt to Amazon Bedrock using the Converse API.

const AWS_REGION = "us-east-1";

// Set the model ID, e.g., Claude Haiku.
// The "global." prefix enables cross-region inference, allowing the request
// to be routed to the nearest available region for the specified model.
const MODEL_ID = "global.anthropic.claude-haiku-4-5-20251001-v1:0";
const PROMPT = "Hi. In a short paragraph, explain what you can do.";

const hello = async () => {
  console.log("=".repeat(35));
  console.log("Welcome to the Amazon Bedrock demo!");
  console.log("=".repeat(35));

  console.log("Model: Anthropic Claude Haiku");
  console.log(`Prompt: ${PROMPT}\n`);
  console.log("Invoking model...\n");

  // Create a new Bedrock Runtime client instance.
  const client = new BedrockRuntimeClient({ region: AWS_REGION });

  // Create the command with the model ID, the user message, and a basic
 configuration.
  const command = new ConverseCommand({
    modelId: MODEL_ID,
    messages: [
      {
        role: "user",
        content: [{ text: PROMPT }],
      },
    ],
  });

  // Send the command to the model and wait for the response.
  try {
    const response = await client.send(command);

    // Extract and print the response text.
    const responseText = response.output.message.content[0].text;
```

```
    console.log(`Response: ${responseText}`);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "BedrockRuntimeServiceException"
    ) {
      console.error(
        `ERROR: Can't invoke '${MODEL_ID}'. Reason: ${caught.message}`,
      );
      throw caught;
    }
    throw caught;
  }
};

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await hello();
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Szenarien

Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/* Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer to
 it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
```

```
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each step.
\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information. \n"
 +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before sending
 the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon Bedrock,
 and returns the response.
// param "messages" - The conversation history including the next message to send.
// return - The response from Amazon Bedrock.
```

```
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
```

```
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the tool's
 response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is sent
 back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
```

```
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
```

```
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or coordinates."
 +
    "Weather information will be provided using a custom Tool and open-meteo API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);
```

```
const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look at:
\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);
```

```
const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure and send a request
// - Process the response

import {
  BedrockRuntimeClient,
  ConversationRole,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use:
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one line.";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the request
// Optional parameters to control the model's response:
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
  },
};

// Step 5: Send and process the request
// - Send the request to the model
// - Extract and return the generated text from the response
try {
  const response = await client.send(new ConverseCommand(request));
  console.log(response.output.message.content[0].text);
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  throw error;
}
```

Senden Sie mithilfe der Converse-API von Bedrock und einer Tool-Konfiguration eine Konversation mit Nachrichten an Amazon Nova.

```
// This example demonstrates how to send a conversation of messages to Amazon Nova
 using Bedrock's Converse API with a tool configuration.
// It shows how to:
// - 1. Set up the Amazon Bedrock runtime client
// - 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response (model ID, user input, system prompt, and the tool spec)
// - 3. Send the request to Amazon Bedrock, and returns the response.
// - 4. Add the tool response to the conversation, and send it back to Amazon
 Bedrock.
// - 5. Publish the response.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client

// Credentials will be automatically loaded from the environment
const bedRockRuntimeClient = new BedrockRuntimeClient({
  region: "us-east-1",
});

// Step 2. Define the parameters required enable Amazon Bedrock to use a tool when
 formulating its response.

// The Bedrock Model ID.
const modelId = "amazon.nova-lite-v1:0";

// The system prompt to help Amazon Bedrock craft it's response.
const system_prompt = [
  {
    text:
      "You are a music expert that provides the most popular song played on a radio
 station, using only the\n" +
      "the top_song tool, which he call sign for the radio station for which you
 want the most popular song. " +
      "Example calls signs are WZPZ and WKRP. \n" +
      "- Only use the top_song tool. Never guess or make up information. \n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
```

```
      "- Only respond to queries about the most popular song played on a radio
 station\n" +
      "Remind off-topic users of your purpose. \n" +
      "- Never claim to search online, access external data, or use tools besides
 the top_song tool.\n",
  },
];
//  The user's question.
const message = [
  {
    role: "user",
    content: [{ text: "What is the most popular song on WZPZ?" }],
  },
];
// The tool specification. In this case, it uses an example schema for
// a tool that gets the most popular song played on a radio station.
const tool_config = {
  tools: [
    {
      toolSpec: {
        name: "top_song",
        description: "Get the most popular song played on a radio station.",
        inputSchema: {
          json: {
            type: "object",
            properties: {
              sign: {
                type: "string",
                description:
                  "The call sign for the radio station for which you want the most
 popular song. Example calls signs are WZPZ and WKRP.",
              },
            },
            required: ["sign"],
          },
        },
      },
    },
  ],
};

// Helper function to return the song and artist from top_song tool.
async function get_top_song(call_sign) {
  try {
```

```
    if (call_sign === "WZPZ") {
      const song = "Elemental Hotel";
      const artist = "8 Storey Hike";
      return { song, artist };
    }
  } catch (error) {
    console.log(`${error.message}`);
  }
}

// 3. Send the request to Amazon Bedrock, and returns the response.
export async function SendConversationtoBedrock(
  modelId,
  message,
  system_prompt,
  tool_config,
) {
  try {
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: message,
        system: system_prompt,
        toolConfig: tool_config,
      }),
    );
    if (response.stopReason === "tool_use") {
      const toolResultFinal = [];
      try {
        const output_message = response.output.message;
        message.push(output_message);
        const toolRequests = output_message.content;
        const toolMessage = toolRequests[0].text;
        console.log(toolMessage.replace(/<[^>]+>/g, ""));
        for (const toolRequest of toolRequests) {
          if (Object.hasOwn(toolRequest, "toolUse")) {
            const toolUse = toolRequest.toolUse;
            const sign = toolUse.input.sign;
            const toolUseID = toolUse.toolUseId;
            console.log(
              `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
            );
            if (toolUse.name === "top_song") {
              const toolResult = [];
```

```
              try {
                const top_song = await get_top_song(toolUse.input.sign).then(
                  (top_song) => top_song,
                );
                const toolResult = {
                  toolResult: {
                    toolUseId: toolUseID,
                    content: [
                      {
                        json: { song: top_song.song, artist: top_song.artist },
                      },
                    ],
                  },
                };
                toolResultFinal.push(toolResult);
              } catch (err) {
                const toolResult = {
                  toolUseId: toolUseID,
                  content: [{ json: { text: err.message } }],
                  status: "error",
                };
              }
            }
          }
        }
        const toolResultMessage = {
          role: "user",
          content: toolResultFinal,
        };
        // Step 4. Add the tool response to the conversation, and send it back to
Amazon Bedrock.

        message.push(toolResultMessage);
        await SendConversationtoBedrock(
          modelId,
          message,
          system_prompt,
          tool_config,
        );
      } catch (caught) {
        console.error(`${caught.message}`);
        throw caught;
      }
    }
```

```
    // 4. Publish the response.
    if (response.stopReason === "end_turn") {
      const finalMessage = response.output.message.content[0].text;
      const messageToPrint = finalMessage.replace(/<[^>]+>/g);
      console.log(messageToPrint.replace(/<[^>]+>/g));
      return messageToPrint;
    }
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        `${caught.name} - Model not ready, please wait and try again.`,
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        `${caught.name} - Error occurred while sending Converse request`,
      );
      throw caught;
    }
  }
}
await SendConversationtoBedrock(modelId, message, system_prompt, tool_config);
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// This example demonstrates how to use the Amazon Nova foundation models
// to generate streaming text responses.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Create a message
// - Configure a streaming request
// - Process the streaming response

import {
  BedrockRuntimeClient,
  ConversationRole,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Step 1: Create the Amazon Bedrock runtime client
// Credentials will be automatically loaded from the environment
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Step 2: Specify which model to use
// Available Amazon Nova models and their characteristics:
// - Amazon Nova Micro: Text-only model optimized for lowest latency and cost
// - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video, and text
// - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed, and
 cost
//
// For the most current model IDs, see:
// https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
const modelId = "amazon.nova-lite-v1:0";

// Step 3: Create the message
// The message includes the text prompt and specifies that it comes from the user
const inputText =
  "Describe the purpose of a 'hello world' program in one paragraph";
const message = {
  content: [{ text: inputText }],
  role: ConversationRole.USER,
};

// Step 4: Configure the streaming request
// Optional parameters to control the model's response:
```

```
// - maxTokens: maximum number of tokens to generate
// - temperature: randomness (max: 1.0, default: 0.7)
//   OR
// - topP: diversity of word choice (max: 1.0, default: 0.9)
// Note: Use either temperature OR topP, but not both
const request = {
  modelId,
  messages: [message],
  inferenceConfig: {
    maxTokens: 500, // The maximum response length
    temperature: 0.5, // Using temperature for randomness control
    //topP: 0.9,        // Alternative: use topP instead of temperature
  },
};

// Step 5: Send and process the streaming request
// - Send the request to the model
// - Process each chunk of the streaming response
try {
  const response = await client.send(new ConverseStreamCommand(request));

  for await (const chunk of response.stream) {
    if (chunk.contentBlockDelta) {
      // Print each text chunk as it arrives
      process.stdout.write(chunk.contentBlockDelta.delta?.text || "");
    }
  }
} catch (error) {
  console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
  process.exitCode = 1;
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für JavaScript API-Referenz.

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/* Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";
import data from "./questions.json" with { type: "json" };
import toolConfig from "./tool_config.json" with { type: "json" };

const __filename = fileURLToPath(import.meta.url);

const systemPrompt = [
  {
    text:
```

```
      "You are a weather assistant that provides current weather data for user-
specified locations using only\n" +
      "the Weather_Tool, which expects latitude and longitude. Infer the coordinates
 from the location yourself.\n" +
      "If the user provides coordinates, infer the approximate location and refer to
 it in your response.\n" +
      "To use the tool, you strictly apply the provided tool specification.\n" +
      "If the user specifies a state, country, or region, infer the locations of
 cities within that state.\n" +
      "\n" +
      "- Explain your step-by-step process, and give brief updates before each step.
\n" +
      "- Only use the Weather_Tool for data. Never guess or make up information. \n"
 +
      "- Repeat the tool use for subsequent requests if necessary.\n" +
      "- If the tool errors, apologize, explain weather is unavailable, and suggest
 other options.\n" +
      "- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use\n" +
      "  emojis where appropriate.\n" +
      "- Only respond to weather queries. Remind off-topic users of your purpose.
 \n" +
      "- Never claim to search online, access external data, or use tools besides
 Weather_Tool.\n" +
      "- Complete the entire process until you have all required data before sending
 the complete response.",
  },
];
const tools_config = toolConfig;

/// Starts the conversation with the user and handles the interaction with Bedrock.
async function askQuestion(userMessage) {
  // The maximum number of recursive calls allowed in the tool use function.
  // This helps prevent infinite loops and potential performance issues.
  const max_recursions = 5;
  const messages = [
    {
      role: "user",
      content: [{ text: userMessage }],
    },
  ];
  try {
    const response = await SendConversationtoBedrock(messages);
    await ProcessModelResponseAsync(response, messages, max_recursions);
```

```
  } catch (error) {
    console.log("error ", error);
  }
}

// Sends the conversation, the system prompt, and the tool spec to Amazon Bedrock,
 and returns the response.
// param "messages" - The conversation history including the next message to send.
// return - The response from Amazon Bedrock.
async function SendConversationtoBedrock(messages) {
  const bedRockRuntimeClient = new BedrockRuntimeClient({
    region: "us-east-1",
  });
  try {
    const modelId = "amazon.nova-lite-v1:0";
    const response = await bedRockRuntimeClient.send(
      new ConverseCommand({
        modelId: modelId,
        messages: messages,
        system: systemPrompt,
        toolConfig: tools_config,
      }),
    );
    return response;
  } catch (caught) {
    if (caught.name === "ModelNotReady") {
      console.log(
        "`${caught.name}` - Model not ready, please wait and try again.",
      );
      throw caught;
    }
    if (caught.name === "BedrockRuntimeException") {
      console.log(
        '`${caught.name}` - "Error occurred while sending Converse request.',
      );
      throw caught;
    }
  }
}

// Processes the response received via Amazon Bedrock and performs the necessary
 actions based on the stop reason.
// param "response" - The model's response returned via Amazon Bedrock.
// param "messages" - The conversation history.
```

```
// param "max_recursions" - The maximum number of recursive calls allowed.
async function ProcessModelResponseAsync(response, messages, max_recursions) {
  if (max_recursions <= 0) {
    await HandleToolUseAsync(response, messages);
  }
  if (response.stopReason === "tool_use") {
    await HandleToolUseAsync(response, messages, max_recursions - 1);
  }
  if (response.stopReason === "end_turn") {
    const messageToPrint = response.output.message.content[0].text;
    console.log(messageToPrint.replace(/<[^>]+>/g, ""));
  }
}
// Handles the tool use case by invoking the specified tool and sending the tool's
 response back to Bedrock.
// The tool response is appended to the conversation, and the conversation is sent
 back to Amazon Bedrock for further processing.
// param "response" - the model's response containing the tool use request.
// param "messages" - the conversation history.
// param "max_recursions" - The maximum number of recursive calls allowed.
async function HandleToolUseAsync(response, messages, max_recursions) {
  const toolResultFinal = [];
  try {
    const output_message = response.output.message;
    messages.push(output_message);
    const toolRequests = output_message.content;
    const toolMessage = toolRequests[0].text;
    console.log(toolMessage.replace(/<[^>]+>/g, ""));
    for (const toolRequest of toolRequests) {
      if (Object.hasOwn(toolRequest, "toolUse")) {
        const toolUse = toolRequest.toolUse;
        const latitude = toolUse.input.latitude;
        const longitude = toolUse.input.longitude;
        const toolUseID = toolUse.toolUseId;
        console.log(
          `Requesting tool ${toolUse.name}, Tool use id ${toolUseID}`,
        );
        if (toolUse.name === "Weather_Tool") {
          try {
            const current_weather = await callWeatherTool(
              longitude,
              latitude,
            ).then((current_weather) => current_weather);
            const currentWeather = current_weather;
```

```
            const toolResult = {
              toolResult: {
                toolUseId: toolUseID,
                content: [{ json: currentWeather }],
              },
            };
            toolResultFinal.push(toolResult);
          } catch (err) {
            console.log("An error occurred. ", err);
          }
        }
      }
    }

    const toolResultMessage = {
      role: "user",
      content: toolResultFinal,
    };
    messages.push(toolResultMessage);
    // Send the conversation to Amazon Bedrock
    await ProcessModelResponseAsync(
      await SendConversationtoBedrock(messages),
      messages,
    );
  } catch (error) {
    console.log("An error occurred. ", error);
  }
}
// Call the Weathertool.
// param = longitude of location
// param = latitude of location
async function callWeatherTool(longitude, latitude) {
  // Open-Meteo API endpoint
  const apiUrl = `https://api.open-meteo.com/v1/forecast?latitude=
${latitude}&longitude=${longitude}&current_weather=true`;

  // Fetch the weather data.
  return fetch(apiUrl)
    .then((response) => {
      return response.json().then((current_weather) => {
        return current_weather;
      });
    })
    .catch((error) => {
```

```
      console.error("Error fetching weather data:", error);
    });
}
/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "input",
  default: "",
});

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Bedrock Tool Use demo! \n" +
    "This assistant provides current weather information for user-specified
 locations. " +
    "You can ask for weather details by providing the location name or coordinates."
 +
    "Weather information will be provided using a custom Tool and open-meteo API." +
    "For the purposes of this example, we'll use in order the questions in ./
questions.json :\n" +
    "What's the weather like in Seattle? " +
    "What's the best kind of cat? " +
    "Where is the warmest city in Washington State right now? " +
    "What's the warmest city in California right now?\n" +
    "To exit the program, simply type 'x' and press Enter.\n" +
    "Have fun and experiment with the app by editing the questions in ./
questions.json! " +
    "P.S.: You're not limited to single locations, or even to using English! ",

  { header: true },
);
const displayAskQuestion1 = new ScenarioOutput(
  "displayAskQuestion1",
  "Press enter to ask question number 1 (default is 'What's the weather like in
 Seattle?')",
);

const askQuestion1 = new ScenarioAction(
  "askQuestion1",
  async (/** @type {State} */ state) => {
    const userMessage1 = data.questions["question-1"];
    await askQuestion(userMessage1);
```

```
  },
);

const displayAskQuestion2 = new ScenarioOutput(
  "displayAskQuestion2",
  "Press enter to ask question number 2 (default is 'What's the best kind of
 cat?')",
);

const askQuestion2 = new ScenarioAction(
  "askQuestion2",
  async (/** @type {State} */ state) => {
    const userMessage2 = data.questions["question-2"];
    await askQuestion(userMessage2);
  },
);
const displayAskQuestion3 = new ScenarioOutput(
  "displayAskQuestion3",
  "Press enter to ask question number 3 (default is 'Where is the warmest city in
 Washington State right now?')",
);

const askQuestion3 = new ScenarioAction(
  "askQuestion3",
  async (/** @type {State} */ state) => {
    const userMessage3 = data.questions["question-3"];
    await askQuestion(userMessage3);
  },
);

const displayAskQuestion4 = new ScenarioOutput(
  "displayAskQuestion4",
  "Press enter to ask question number 4 (default is 'What's the warmest city in
 California right now?')",
);

const askQuestion4 = new ScenarioAction(
  "askQuestion4",
  async (/** @type {State} */ state) => {
    const userMessage4 = data.questions["question-4"];
    await askQuestion(userMessage4);
  },
);
```

```
const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Bedrock Tool Use demo. We hope you\n" +
    "learned something new, or got some inspiration for your own apps today!\n" +
    "For more Bedrock examples in different programming languages, have a look at:
\n" +
    "https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html",
);

const myScenario = new Scenario("Converse Tool Scenario", [
  greet,
  pressEnter,
  displayAskQuestion1,
  askQuestion1,
  pressEnter,
  displayAskQuestion2,
  askQuestion2,
  pressEnter,
  displayAskQuestion3,
  askQuestion3,
  pressEnter,
  displayAskQuestion4,
  askQuestion4,
  pressEnter,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
```

```
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

Amazon Nova Canvas

InvokeModel

Das folgende Codebeispiel zeigt, wie Amazon Nova Canvas auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
import {
  BedrockRuntimeClient,
  InvokeModelCommand,
} from "@aws-sdk/client-bedrock-runtime";
import { saveImage } from "../../utils/image-creation.js";
import { fileURLToPath } from "node:url";

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 *
 * @returns {Promise<string>} Base64-encoded image data
 */
export const invokeModel = async () => {
```

```
  // Step 1: Create the Amazon Bedrock runtime client
  // Credentials will be automatically loaded from the environment
  const client = new BedrockRuntimeClient({ region: "us-east-1" });

  // Step 2: Specify which model to use
  // For the latest available models, see:
  // https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html
  const modelId = "amazon.nova-canvas-v1:0";

  // Step 3: Configure the request payload
  // First, set the main parameters:
  // - prompt: Text description of the image to generate
  // - seed: Random number for reproducible generation (0 to 858,993,459)
  const prompt = "A stylized picture of a cute old steampunk robot";
  const seed = Math.floor(Math.random() * 858993460);

  // Then, create the payload using the following structure:
  // - taskType: TEXT_IMAGE (specifies text-to-image generation)
  // - textToImageParams: Contains the text prompt
  // - imageGenerationConfig: Contains optional generation settings (seed, quality,
 etc.)
  // For a list of available request parameters, see:
  // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
  const payload = {
    taskType: "TEXT_IMAGE",
    textToImageParams: {
      text: prompt,
    },
    imageGenerationConfig: {
      seed,
      quality: "standard",
    },
  };

  // Step 4: Send and process the request
  // - Embed the payload in a request object
  // - Send the request to the model
  // - Extract and return the generated image data from the response
  try {
    const request = {
      modelId,
      body: JSON.stringify(payload),
    };
```

```
    const response = await client.send(new InvokeModelCommand(request));

    const decodedResponseBody = new TextDecoder().decode(response.body);
    // The response includes an array of base64-encoded PNG images
    /** @type {{images: string[]}} */
    const responseBody = JSON.parse(decodedResponseBody);
    return responseBody.images[0]; // Base64-encoded image data
  } catch (error) {
    console.error(`ERROR: Can't invoke '${modelId}'. Reason: ${error.message}`);
    throw error;
  }
};

// If run directly, execute the example and save the generated image
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("Generating image. This may take a few seconds...");
  invokeModel()
    .then(async (imageData) => {
      const imagePath = await saveImage(imageData, "nova-canvas");
      // Example path: javascriptv3/example_code/bedrock-runtime/output/nova-canvas/
image-01.png
      console.log(`Image saved to: ${imagePath}`);
    })
    .catch((error) => {
      console.error("Execution failed:", error);
      process.exitCode = 1;
    });
}
```

- API-Details finden Sie InvokeModel in der AWS SDK für JavaScript API-Referenz.

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);
```

```
  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Conversation API to send a text message to Anthropic Claude.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Claude 3 Haiku.
const modelId = "anthropic.claude-3-haiku-20240307-v1:0";
```

```
// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für JavaScript API-Referenz.

Cohere Command

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);
```

```
  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Cohere Command.

import {
  BedrockRuntimeClient,
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Command R.
```

```
const modelId = "cohere.command-r-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für JavaScript API-Referenz.

Meta Llama

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
```

```
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Details zur API finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Meta Llama.

import {
  BedrockRuntimeClient,
```

```
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Llama 3 8b Instruct.
const modelId = "meta.llama3-8b-instruct-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für JavaScript API-Referenz.

Mistral AI

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
  ConverseCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseCommand({
  modelId,
```

```
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the response text.
  const responseText = response.output.message.content[0].text;
  console.log(responseText);
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK für JavaScript -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Conversation API to send a text message to Mistral.

import {
  BedrockRuntimeClient,
```

```
  ConverseStreamCommand,
} from "@aws-sdk/client-bedrock-runtime";

// Create a Bedrock Runtime client in the AWS Region you want to use.
const client = new BedrockRuntimeClient({ region: "us-east-1" });

// Set the model ID, e.g., Mistral Large.
const modelId = "mistral.mistral-large-2402-v1:0";

// Start a conversation with the user message.
const userMessage =
  "Describe the purpose of a 'hello world' program in one line.";
const conversation = [
  {
    role: "user",
    content: [{ text: userMessage }],
  },
];

// Create a command with the model ID, the message, and a basic configuration.
const command = new ConverseStreamCommand({
  modelId,
  messages: conversation,
  inferenceConfig: { maxTokens: 512, temperature: 0.5, topP: 0.9 },
});

try {
  // Send the command to the model and wait for the response
  const response = await client.send(command);

  // Extract and print the streamed response text in real-time.
  for await (const item of response.stream) {
    if (item.contentBlockDelta) {
      process.stdout.write(item.contentBlockDelta.delta?.text);
    }
  }
} catch (err) {
  console.log(`ERROR: Can't invoke '${modelId}'. Reason: ${err}`);
  process.exit(1);
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Bedrock Agents, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Bedrock Agents Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Agenten für Amazon Bedrock

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Agenten für Amazon Bedrock.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  GetAgentCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";
```

```
/**
 * @typedef {Object} AgentSummary
 */

/**
 * A simple scenario to demonstrate basic setup and interaction with the Bedrock
 Agents Client.
 *
 * This function first initializes the Amazon Bedrock Agents client for a specific
 region.
 * It then retrieves a list of existing agents using the streamlined paginator
 approach.
 * For each agent found, it retrieves detailed information using a command object.
 *
 * Demonstrates:
 * - Use of the Bedrock Agents client to initialize and communicate with the AWS
 service.
 * - Listing resources in a paginated response pattern.
 * - Accessing an individual resource using a command object.
 *
 * @returns {Promise<void>} A promise that resolves when the function has completed
 execution.
 */
export const main = async () => {
  const region = "us-east-1";

  console.log("=".repeat(68));

  console.log(`Initializing Amazon Bedrock Agents client for ${region}...`);
  const client = new BedrockAgentClient({ region });

  console.log("Retrieving the list of existing agents...");
  const paginatorConfig = { client };
  const pages = paginateListAgents(paginatorConfig, {});

  /** @type {AgentSummary[]} */
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  console.log(`Found ${agentSummaries.length} agents in ${region}.`);

  if (agentSummaries.length > 0) {
```

```
    for (const agentSummary of agentSummaries) {
      const agentId = agentSummary.agentId;
      console.log("=".repeat(68));
      console.log(`Retrieving agent with ID: ${agentId}:`);
      console.log("-".repeat(68));

      const command = new GetAgentCommand({ agentId });
      const response = await client.send(command);
      const agent = response.agent;

      console.log(` Name: ${agent.agentName}`);
      console.log(` Status: ${agent.agentStatus}`);
      console.log(` ARN: ${agent.agentArn}`);
      console.log(` Foundation model: ${agent.foundationModel}`);
    }
  }
  console.log("=".repeat(68));
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  await main();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - GetAgent
  - ListAgents

Aktionen

`CreateAgent`

Das folgende Codebeispiel zeigt die Verwendung`CreateAgent`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  CreateAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Creates an Amazon Bedrock Agent.
 *
 * @param {string} agentName - A name for the agent that you create.
 * @param {string} foundationModel - The foundation model to be used by the agent
 you create.
 * @param {string} agentResourceRoleArn - The ARN of the IAM role with permissions
 required by the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing details of the created agent.
 */
export const createAgent = async (
  agentName,
  foundationModel,
  agentResourceRoleArn,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  const command = new CreateAgentCommand({
    agentName,
    foundationModel,
    agentResourceRoleArn,
```

```
  });
  const response = await client.send(command);

  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentName and accountId, and roleName with a
 unique name for the new agent,
  // the id of your AWS account, and the name of an existing execution role that the
 agent can use inside your account.
  // For foundationModel, specify the desired model. Ensure to remove the brackets
 '[]' before adding your data.

  // A string (max 100 chars) that can include letters, numbers, dashes '-', and
 underscores '_'.
  const agentName = "[your-bedrock-agent-name]";

  // Your AWS account id.
  const accountId = "[123456789012]";

  // The name of the agent's execution role. It must be prefixed by
 `AmazonBedrockExecutionRoleForAgents_`.
  const roleName = "[AmazonBedrockExecutionRoleForAgents_your-role-name]";

  // The ARN for the agent's execution role.
  // Follow the ARN format: 'arn:aws:iam::account-id:role/role-name'
  const roleArn = `arn:aws:iam::${accountId}:role/${roleName}`;

  // Specify the model for the agent. Change if a different model is preferred.
  const foundationModel = "anthropic.claude-v2";

  // Check for unresolved placeholders in agentName and roleArn.
  checkForPlaceholders([agentName, roleArn]);

  console.log("Creating a new agent...");

  const agent = await createAgent(agentName, foundationModel, roleArn);
  console.log(agent);
}
```

- API-Details finden Sie CreateAgent in der AWS SDK für JavaScript API-Referenz.

`DeleteAgent`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAgent`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  DeleteAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Deletes an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent to delete.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-
agent").DeleteAgentCommandOutput>} An object containing the agent id, the status,
 and some additional metadata.
 */
export const deleteAgent = (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });
  const command = new DeleteAgentCommand({ agentId });
  return client.send(command);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets (`[]`) before adding your data.
```

```
  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Deleting agent with ID ${agentId}...`);

  const response = await deleteAgent(agentId);
  console.log(response);
}
```

- API-Details finden Sie DeleteAgent in der AWS SDK für JavaScript API-Referenz.

`GetAgent`

Das folgende Codebeispiel zeigt die Verwendung`GetAgent`.

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  GetAgentCommand,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves the details of an Amazon Bedrock Agent.
 *
 * @param {string} agentId - The unique identifier of the agent.
```

```
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").Agent>} An object
 containing the agent details.
 */
export const getAgent = async (agentId, region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  const command = new GetAgentCommand({ agentId });
  const response = await client.send(command);
  return response.agent;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId with an existing agent's id.
  // Ensure to remove the brackets '[]' before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // Check for unresolved placeholders in agentId.
  checkForPlaceholders([agentId]);

  console.log(`Retrieving agent with ID ${agentId}...`);

  const agent = await getAgent(agentId);
  console.log(agent);
}
```

- API-Details finden Sie GetAgent in der AWS SDK für JavaScript API-Referenz.

`ListAgentActionGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListAgentActionGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aktionsgruppen für einen Agenten.

```
import { fileURLToPath } from "node:url";
import { checkForPlaceholders } from "../lib/utils.js";

import {
  BedrockAgentClient,
  ListAgentActionGroupsCommand,
  paginateListAgentActionGroups,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves a list of Action Groups of an agent utilizing the paginator function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of` loop.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithPaginator = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  // Create a paginator configuration
  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const params = { agentId, agentVersion };

  const pages = paginateListAgentActionGroups(paginatorConfig, params);

  // Paginate until there are no more results
  const actionGroupSummaries = [];
  for await (const page of pages) {
```

```
    actionGroupSummaries.push(...page.actionGroupSummaries);
  }

  return actionGroupSummaries;
};

/**
 * Retrieves a list of Action Groups of an agent utilizing the
 ListAgentActionGroupsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the client
 and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentActionGroupsWithPaginator()` example below.
 *
 * @param {string} agentId - The unique identifier of the agent.
 * @param {string} agentVersion - The version of the agent.
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<ActionGroupSummary[]>} An array of action group summaries.
 */
export const listAgentActionGroupsWithCommandObject = async (
  agentId,
  agentVersion,
  region = "us-east-1",
) => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const actionGroupSummaries = [];
  do {
    const command = new ListAgentActionGroupsCommand({
      agentId,
      agentVersion,
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{actionGroupSummaries: ActionGroupSummary[], nextToken?: string}} */
    const response = await client.send(command);

    for (const actionGroup of response.actionGroupSummaries || []) {
      actionGroupSummaries.push(actionGroup);
    }
```

```
    nextToken = response.nextToken;
  } while (nextToken);

  return actionGroupSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  // Replace the placeholders for agentId and agentVersion with an existing agent's
 id and version.
  // Ensure to remove the brackets '[]' before adding your data.

  // The agentId must be an alphanumeric string with exactly 10 characters.
  const agentId = "[ABC123DE45]";

  // A string either containing `DRAFT` or a number with 1-5 digits (e.g., '123' or
 'DRAFT').
  const agentVersion = "[DRAFT]";

  // Check for unresolved placeholders in agentId and agentVersion.
  checkForPlaceholders([agentId, agentVersion]);

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using ListAgentActionGroupsCommand:",
  );

  for (const actionGroup of await listAgentActionGroupsWithCommandObject(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
  }

  console.log("=".repeat(68));
  console.log(
    "Listing agent action groups using the paginateListAgents function:",
  );
  for (const actionGroup of await listAgentActionGroupsWithPaginator(
    agentId,
    agentVersion,
  )) {
    console.log(actionGroup);
```

```
  }
}
```

- API-Details finden Sie ListAgentActionGroups in der AWS SDK für JavaScript API-Referenz.

`ListAgents`

Das folgende Codebeispiel zeigt die Verwendung`ListAgents`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Agenten, die zu einem Konto gehören.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentClient,
  ListAgentsCommand,
  paginateListAgents,
} from "@aws-sdk/client-bedrock-agent";

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the paginator
 function.
 *
 * This function leverages a paginator, which abstracts the complexity of
 pagination, providing
 * a straightforward way to handle paginated results inside a `for await...of` loop.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithPaginator = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });
```

```
  const paginatorConfig = {
    client,
    pageSize: 10, // optional, added for demonstration purposes
  };

  const pages = paginateListAgents(paginatorConfig, {});

  // Paginate until there are no more results
  const agentSummaries = [];
  for await (const page of pages) {
    agentSummaries.push(...page.agentSummaries);
  }

  return agentSummaries;
};

/**
 * Retrieves a list of available Amazon Bedrock agents utilizing the
 ListAgentsCommand.
 *
 * This function demonstrates the manual approach, sending a command to the client
 and processing the response.
 * Pagination must manually be managed. For a simplified approach that abstracts
 away pagination logic, see
 * the `listAgentsWithPaginator()` example below.
 *
 * @param {string} [region='us-east-1'] - The AWS region in use.
 * @returns {Promise<AgentSummary[]>} An array of agent summaries.
 */
export const listAgentsWithCommandObject = async (region = "us-east-1") => {
  const client = new BedrockAgentClient({ region });

  let nextToken;
  const agentSummaries = [];
  do {
    const command = new ListAgentsCommand({
      nextToken,
      maxResults: 10, // optional, added for demonstration purposes
    });

    /** @type {{agentSummaries: AgentSummary[], nextToken?: string}} */
    const paginatedResponse = await client.send(command);
```

```
    agentSummaries.push(...(paginatedResponse.agentSummaries || []));

    nextToken = paginatedResponse.nextToken;
  } while (nextToken);

  return agentSummaries;
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  console.log("=".repeat(68));
  console.log("Listing agents using ListAgentsCommand:");
  for (const agent of await listAgentsWithCommandObject()) {
    console.log(agent);
  }

  console.log("=".repeat(68));
  console.log("Listing agents using the paginateListAgents function:");
  for (const agent of await listAgentsWithPaginator()) {
    console.log(agent);
  }
}
```

- API-Details finden Sie ListAgents in der AWS SDK für JavaScript API-Referenz.

Amazon Bedrock Agents Runtime-Beispiele, die das SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Amazon Bedrock Agents Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`InvokeAgent`

Das folgende Codebeispiel zeigt die Verwendung`InvokeAgent`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  BedrockAgentRuntimeClient,
  InvokeAgentCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * @typedef {Object} ResponseBody
 * @property {string} completion
 */

/**
 * Invokes a Bedrock agent to run an inference using the input
 * provided in the request body.
 *
 * @param {string} prompt - The prompt that you want the Agent to complete.
 * @param {string} sessionId - An arbitrary identifier for the session.
 */
export const invokeBedrockAgent = async (prompt, sessionId) => {
  const client = new BedrockAgentRuntimeClient({ region: "us-east-1" });
  // const client = new BedrockAgentRuntimeClient({
  //   region: "us-east-1",
  //   credentials: {
  //     accessKeyId: "accessKeyId", // permission to invoke agent
  //     secretAccessKey: "accessKeySecret",
  //   },
  // });

  const agentId = "AJBHXXILZN";
```

```
  const agentAliasId = "AVKP1ITZAA";

  const command = new InvokeAgentCommand({
    agentId,
    agentAliasId,
    sessionId,
    inputText: prompt,
  });

  try {
    let completion = "";
    const response = await client.send(command);

    if (response.completion === undefined) {
      throw new Error("Completion is undefined");
    }

    for await (const chunkEvent of response.completion) {
      const chunk = chunkEvent.chunk;
      console.log(chunk);
      const decodedResponse = new TextDecoder("utf-8").decode(chunk.bytes);
      completion += decodedResponse;
    }

    return { sessionId: sessionId, completion };
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const result = await invokeBedrockAgent("I need help.", "123");
  console.log(result);
}
```

- API-Details finden Sie InvokeAgent in der AWS SDK für JavaScript API-Referenz.

`InvokeFlow`

Das folgende Codebeispiel zeigt die Verwendung`InvokeFlow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";

import {
  BedrockAgentRuntimeClient,
  InvokeFlowCommand,
} from "@aws-sdk/client-bedrock-agent-runtime";

/**
 * Invokes an alias of a flow to run the inputs that you specify and return
 * the output of each node as a stream.
 *
 * @param {{
 *  flowIdentifier: string,
 *  flowAliasIdentifier: string,
 *  prompt?: string,
 *  region?: string
 * }} options
 * @returns {Promise<import("@aws-sdk/client-bedrock-agent").FlowNodeOutput>} An
 object containing information about the output from flow invocation.
 */
export const invokeBedrockFlow = async ({
  flowIdentifier,
  flowAliasIdentifier,
  prompt = "Hi, how are you?",
  region = "us-east-1",
}) => {
  const client = new BedrockAgentRuntimeClient({ region });

  const command = new InvokeFlowCommand({
    flowIdentifier,
    flowAliasIdentifier,
    inputs: [
      {
```

```
        content: {
          document: prompt,
        },
        nodeName: "FlowInputNode",
        nodeOutputName: "document",
      },
    ],
  });

  let flowResponse = {};
  const response = await client.send(command);

  for await (const chunkEvent of response.responseStream) {
    const { flowOutputEvent, flowCompletionEvent } = chunkEvent;

    if (flowOutputEvent) {
      flowResponse = { ...flowResponse, ...flowOutputEvent };
      console.log("Flow output event:", flowOutputEvent);
    } else if (flowCompletionEvent) {
      flowResponse = { ...flowResponse, ...flowCompletionEvent };
      console.log("Flow completion event:", flowCompletionEvent);
    }
  }

  return flowResponse;
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    flowIdentifier: {
      type: "string",
      required: true,
    },
    flowAliasIdentifier: {
      type: "string",
      required: true,
    },
```

```
    prompt: {
      type: "string",
    },
    region: {
      type: "string",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    invokeBedrockFlow(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie InvokeFlow in der AWS SDK für JavaScript API-Referenz.

CloudWatch Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAlarms in der AWS SDK für JavaScript API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK für JavaScript API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
```

```
export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DisableAlarmActions in der AWS SDK für JavaScript API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie EnableAlarmActions in der AWS SDK für JavaScript API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  CloudWatchServiceException,
  ListMetricsCommand,
} from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = async () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
```

```
  });

  try {
    const response = await client.send(command);
    console.log(`Metrics count: ${response.Metrics?.length}`);
    return response;
  } catch (caught) {
    if (caught instanceof CloudWatchServiceException) {
      console.error(`Error from CloudWatch. ${caught.name}: ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListMetrics in der AWS SDK für JavaScript API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutMetricAlarm in der AWS SDK für JavaScript API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
```

```
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutMetricData in der AWS SDK für JavaScript API-Referenz.

CloudWatch Beispiele für Ereignisse, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit CloudWatch Ereignissen Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutEventsCommand({
    // The list of events to send to Amazon CloudWatch Events.
    Entries: [
      {
        // The name of the application or service that is sending the event.
        Source: "my.app",

        // The name of the event that is being sent.
        DetailType: "My Custom Event",

        // The data that is sent with the event.
        Detail: JSON.stringify({ timeOfEvent: new Date().toISOString() }),
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};
```

```
export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutEvents in der AWS SDK für JavaScript API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { client } from "../libs/client.js";

const run = async () => {
  // Request parameters for PutRule.
  // https://docs.aws.amazon.com/eventbridge/latest/APIReference/
API_PutRule.html#API_PutRule_RequestParameters
  const command = new PutRuleCommand({
    Name: process.env.CLOUDWATCH_EVENTS_RULE,

    // The event pattern for the rule.
    //  Example: {"source": ["my.app"]}
    EventPattern: process.env.CLOUDWATCH_EVENTS_RULE_PATTERN,
```

```
    // The state of the rule. Valid values: ENABLED, DISABLED
    State: "ENABLED",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutRule in der AWS SDK für JavaScript API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
```

```
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutTargetsCommand({
    // The name of the Amazon CloudWatch Events rule.
    Rule: process.env.CLOUDWATCH_EVENTS_RULE,

    // The targets to add to the rule.
    Targets: [
      {
        Arn: process.env.CLOUDWATCH_EVENTS_TARGET_ARN,
        // The ID of the target. Choose a unique ID for each target.
        Id: process.env.CLOUDWATCH_EVENTS_TARGET_ID,
      },
    ],
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";

export const client = new CloudWatchEventsClient({});
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutTargets in der AWS SDK für JavaScript API-Referenz.

CloudWatch Protokolliert Beispiele mithilfe des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit CloudWatch Protokollen Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateLogGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateLogGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new CreateLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
```

```
    console.error(err);
  }
};

export default run();
```

- API-Details finden Sie CreateLogGroup in der AWS SDK für JavaScript API-Referenz.

`DeleteLogGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLogGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteLogGroupCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteLogGroupCommand({
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API-Details finden Sie DeleteLogGroup in der AWS SDK für JavaScript API-Referenz.

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteSubscriptionFilterCommand({
    // The name of the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,
    // The name of the log group.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API-Details finden Sie DeleteSubscriptionFilter in der AWS SDK für JavaScript API-Referenz.

`DescribeLogGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeLogGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateDescribeLogGroups,
  CloudWatchLogsClient,
} from "@aws-sdk/client-cloudwatch-logs";

const client = new CloudWatchLogsClient({});

export const main = async () => {
  const paginatedLogGroups = paginateDescribeLogGroups({ client }, {});
  const logGroups = [];

  for await (const page of paginatedLogGroups) {
    if (page.logGroups?.every((lg) => !!lg)) {
      logGroups.push(...page.logGroups);
    }
  }

  console.log(logGroups);
  return logGroups;
};
```

- API-Details finden Sie DescribeLogGroups in der AWS SDK für JavaScript API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-
logs";
import { client } from "../libs/client.js";

const run = async () => {
  // This will return a list of all subscription filters in your account
  // matching the log group name.
  const command = new DescribeSubscriptionFiltersCommand({
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
    limit: 1,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API-Details finden Sie DescribeSubscriptionFilters in der AWS SDK für JavaScript API-Referenz.

`GetQueryResults`

Das folgende Codebeispiel zeigt die Verwendung`GetQueryResults`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Simple wrapper for the GetQueryResultsCommand.
 * @param {string} queryId
 */
_getQueryResults(queryId) {
  return this.client.send(new GetQueryResultsCommand({ queryId }));
}
```

- API-Details finden Sie GetQueryResults in der AWS SDK für JavaScript API-Referenz.

`PutSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`PutSubscriptionFilter`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutSubscriptionFilterCommand } from "@aws-sdk/client-cloudwatch-logs";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new PutSubscriptionFilterCommand({
    // An ARN of a same-account Kinesis stream, Kinesis Firehose
    // delivery stream, or Lambda function.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
SubscriptionFilters.html
```

```
    destinationArn: process.env.CLOUDWATCH_LOGS_DESTINATION_ARN,

    // A name for the filter.
    filterName: process.env.CLOUDWATCH_LOGS_FILTER_NAME,

    // A filter pattern for subscribing to a filtered stream of log events.
    // https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/
FilterAndPatternSyntax.html
    filterPattern: process.env.CLOUDWATCH_LOGS_FILTER_PATTERN,

    // The name of the log group. Messages in this group matching the filter pattern
    // will be sent to the destination ARN.
    logGroupName: process.env.CLOUDWATCH_LOGS_LOG_GROUP,
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

- API-Details finden Sie PutSubscriptionFilter in der AWS SDK für JavaScript API-Referenz.

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für JavaScript (v3)

Binden Sie die erforderlichen Dateien ein.

```
import { CloudWatchLogsClient, StartLiveTailCommand } from "@aws-sdk/client-
cloudwatch-logs";
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
async function handleResponseAsync(response) {
```

```
    try {
      for await (const event of response.responseStream) {
        if (event.sessionStart !== undefined) {
          console.log(event.sessionStart);
        } else if (event.sessionUpdate !== undefined) {
          for (const logEvent of event.sessionUpdate.sessionResults) {
            const timestamp = logEvent.timestamp;
            const date = new Date(timestamp);
            console.log("[" + date + "] " + logEvent.message);
          }
        } else {
            console.error("Unknown event type");
        }
      }
    } catch (err) {
        // On-stream exceptions are captured here
        console.error(err)
    }
}
```

Starten Sie die Live-Tail-Sitzung.

```
    const client = new CloudWatchLogsClient();

    const command = new StartLiveTailCommand({
        logGroupIdentifiers: logGroupIdentifiers,
        logStreamNames: logStreamNames,
        logEventFilterPattern: filterPattern
    });
    try{
        const response = await client.send(command);
        handleResponseAsync(response);
    } catch (err){
        // Pre-stream exceptions are captured here
        console.log(err);
    }
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
    /* Set a timeout to close the client. This will stop the Live Tail session. */
    setTimeout(function() {
```

```
        console.log("Client timeout");
        client.destroy();
      }, 10000);
```

- API-Details finden Sie StartLiveTail in der AWS SDK für JavaScript API-Referenz.

StartQuery

Das folgende Codebeispiel zeigt die VerwendungStartQuery.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Wrapper for the StartQueryCommand. Uses a static query string
 * for consistency.
 * @param {[Date, Date]} dateRange
 * @param {number} maxLogs
 * @returns {Promise<{ queryId: string }>}
 */
async _startQuery([startDate, endDate], maxLogs = 10000) {
  try {
    return await this.client.send(
      new StartQueryCommand({
        logGroupNames: this.logGroupNames,
        queryString: "fields @timestamp, @message | sort @timestamp asc",
        startTime: startDate.valueOf(),
        endTime: endDate.valueOf(),
        limit: maxLogs,
      }),
    );
  } catch (err) {
    /** @type {string} */
    const message = err.message;
    if (message.startsWith("Query's end date and time")) {
      // This error indicates that the query's start or end date occur
```

```
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }
```

- API-Details finden Sie StartQuery in der AWS SDK für JavaScript API-Referenz.

Szenarien

Ausführen einer umfangreichen Abfrage

Das folgende Codebeispiel zeigt, wie CloudWatch Logs verwendet werden, um mehr als 10.000 Datensätze abzufragen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
import { CloudWatchQuery } from "./cloud-watch-query.js";

console.log("Starting a recursive query...");

if (!process.env.QUERY_START_DATE || !process.env.QUERY_END_DATE) {
  throw new Error(
    "QUERY_START_DATE and QUERY_END_DATE environment variables are required.",
  );
}
```

```
const cloudWatchQuery = new CloudWatchQuery(new CloudWatchLogsClient({}), {
  logGroupNames: ["/workflows/cloudwatch-logs/large-query"],
  dateRange: [
    new Date(Number.parseInt(process.env.QUERY_START_DATE)),
    new Date(Number.parseInt(process.env.QUERY_END_DATE)),
  ],
});

await cloudWatchQuery.run();

console.log(
  `Queries finished in ${cloudWatchQuery.secondsElapsed} seconds.\nTotal logs found:
 ${cloudWatchQuery.results.length}`,
);
```

Dies ist eine Klasse, die Abfragen bei Bedarf in mehrere Schritte aufteilt.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  StartQueryCommand,
  GetQueryResultsCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { splitDateRange } from "@aws-doc-sdk-examples/lib/utils/util-date.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

class DateOutOfBoundsError extends Error {}

export class CloudWatchQuery {
  /**
   * Run a query for all CloudWatch Logs within a certain date range.
   * CloudWatch logs return a max of 10,000 results. This class
   * performs a binary search across all of the logs in the provided
   * date range if a query returns the maximum number of results.
   *
   * @param {import('@aws-sdk/client-cloudwatch-logs').CloudWatchLogsClient} client
   * @param {{ logGroupNames: string[], dateRange: [Date, Date], queryConfig:
 { limit: number } }} config
   */
  constructor(client, { logGroupNames, dateRange, queryConfig }) {
    this.client = client;
    /**
```

```
     * All log groups are queried.
     */
    this.logGroupNames = logGroupNames;

    /**
     * The inclusive date range that is queried.
     */
    this.dateRange = dateRange;

    /**
     * CloudWatch Logs never returns more than 10,000 logs.
     */
    this.limit = queryConfig?.limit ?? 10000;

    /**
     * @type {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]}
     */
    this.results = [];
  }

  /**
   * Run the query.
   */
  async run() {
    this.secondsElapsed = 0;
    const start = new Date();
    this.results = await this._largeQuery(this.dateRange);
    const end = new Date();
    this.secondsElapsed = (end - start) / 1000;
    return this.results;
  }

  /**
   * Recursively query for logs.
   * @param {[Date, Date]} dateRange
   * @returns {Promise<import("@aws-sdk/client-cloudwatch-logs").ResultField[][]>}
   */
  async _largeQuery(dateRange) {
    const logs = await this._query(dateRange, this.limit);

    console.log(
      `Query date range: ${dateRange
        .map((d) => d.toISOString())
        .join(" to ")}. Found ${logs.length} logs.`,
```

```
    );

    if (logs.length < this.limit) {
      return logs;
    }

    const lastLogDate = this._getLastLogDate(logs);
    const offsetLastLogDate = new Date(lastLogDate);
    offsetLastLogDate.setMilliseconds(lastLogDate.getMilliseconds() + 1);
    const subDateRange = [offsetLastLogDate, dateRange[1]];
    const [r1, r2] = splitDateRange(subDateRange);
    const results = await Promise.all([
      this._largeQuery(r1),
      this._largeQuery(r2),
    ]);
    return [logs, ...results].flat();
  }

  /**
   * Find the most recent log in a list of logs.
   * @param {import("@aws-sdk/client-cloudwatch-logs").ResultField[][]} logs
   */
  _getLastLogDate(logs) {
    const timestamps = logs
      .map(
        (log) =>
          log.find((fieldMeta) => fieldMeta.field === "@timestamp")?.value,
      )
      .filter((t) => !!t)
      .map((t) => `${t}Z`)
      .sort();

    if (!timestamps.length) {
      throw new Error("No timestamp found in logs.");
    }

    return new Date(timestamps[timestamps.length - 1]);
  }

  /**
   * Simple wrapper for the GetQueryResultsCommand.
   * @param {string} queryId
   */
  _getQueryResults(queryId) {
```

```
    return this.client.send(new GetQueryResultsCommand({ queryId }));
  }

  /**
   * Starts a query and waits for it to complete.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   */
  async _query(dateRange, maxLogs) {
    try {
      const { queryId } = await this._startQuery(dateRange, maxLogs);
      const { results } = await this._waitUntilQueryDone(queryId);
      return results ?? [];
    } catch (err) {
      /**
       * This error is thrown when StartQuery returns an error indicating
       * that the query's start or end date occur before the log group was
       * created.
       */
      if (err instanceof DateOutOfBoundsError) {
        return [];
      }
      throw err;
    }
  }

  /**
   * Wrapper for the StartQueryCommand. Uses a static query string
   * for consistency.
   * @param {[Date, Date]} dateRange
   * @param {number} maxLogs
   * @returns {Promise<{ queryId: string }>}
   */
  async _startQuery([startDate, endDate], maxLogs = 10000) {
    try {
      return await this.client.send(
        new StartQueryCommand({
          logGroupNames: this.logGroupNames,
          queryString: "fields @timestamp, @message | sort @timestamp asc",
          startTime: startDate.valueOf(),
          endTime: endDate.valueOf(),
          limit: maxLogs,
        }),
      );
```

```
    } catch (err) {
      /** @type {string} */
      const message = err.message;
      if (message.startsWith("Query's end date and time")) {
        // This error indicates that the query's start or end date occur
        // before the log group was created.
        throw new DateOutOfBoundsError(message);
      }

      throw err;
    }
  }

  /**
   * Call GetQueryResultsCommand until the query is done.
   * @param {string} queryId
   */
  _waitUntilQueryDone(queryId) {
    const getResults = async () => {
      const results = await this._getQueryResults(queryId);
      const queryDone = [
        "Complete",
        "Failed",
        "Cancelled",
        "Timeout",
        "Unknown",
      ].includes(results.status);

      return { queryDone, results };
    };

    return retry(
      { intervalInMs: 1000, maxRetries: 60, quiet: true },
      async () => {
        const { queryDone, results } = await getResults();
        if (!queryDone) {
          throw new Error("Query not done.");
        }

        return results;
      },
    );
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - GetQueryResults
  - StartQuery

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Laufzeit-API. JavaScript In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

CodeBuild Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren CodeBuild.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateProject`

Das folgende Codebeispiel zeigt die Verwendung`CreateProject`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ein Projekt zu erstellen.

```
import {
  ArtifactsType,
  CodeBuildClient,
  ComputeType,
  CreateProjectCommand,
  EnvironmentType,
  SourceType,
} from "@aws-sdk/client-codebuild";

// Create the AWS CodeBuild project.
```

```
export const createProject = async (
  projectName = "MyCodeBuilder",
  roleArn = "arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin",
  buildOutputBucket = "xxxx",
  githubUrl = "https://...",
) => {
  const codeBuildClient = new CodeBuildClient({});

  const response = await codeBuildClient.send(
    new CreateProjectCommand({
      artifacts: {
        // The destination of the build artifacts.
        type: ArtifactsType.S3,
        location: buildOutputBucket,
      },
      // Information about the build environment. The combination of "computeType"
 and "type" determines the
      // requirements for the environment such as CPU, memory, and disk space.
      environment: {
        // Build environment compute types.
        // https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
compute-types.html
        computeType: ComputeType.BUILD_GENERAL1_SMALL,
        // Docker image identifier.
        // See https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-
available.html
        image: "aws/codebuild/standard:7.0",
        // Build environment type.
        type: EnvironmentType.LINUX_CONTAINER,
      },
      name: projectName,
      // A role ARN with permission to create a CodeBuild project, write to the
 artifact location, and write CloudWatch logs.
      serviceRole: roleArn,
      source: {
        // The type of repository that contains the source code to be built.
        type: SourceType.GITHUB,
        // The location of the repository that contains the source code to be built.
        location: githubUrl,
      },
    }),
  );
  console.log(response);
  //   {
```

```
//     '$metadata': {
//       httpStatusCode: 200,
//       requestId: 'b428b244-777b-49a6-a48d-5dffedced8e7',
//       extendedRequestId: undefined,
//       cfId: undefined,
//       attempts: 1,
//       totalRetryDelay: 0
//     },
//     project: {
//       arn: 'arn:aws:codebuild:us-east-1:xxxxxxxxxxxx:project/MyCodeBuilder',
//       artifacts: {
//         encryptionDisabled: false,
//         location: 'xxxxxx-xxxxxxx-xxxxxx',
//         name: 'MyCodeBuilder',
//         namespaceType: 'NONE',
//         packaging: 'NONE',
//         type: 'S3'
//       },
//       badge: { badgeEnabled: false },
//       cache: { type: 'NO_CACHE' },
//       created: 2023-08-18T14:46:48.979Z,
//       encryptionKey: 'arn:aws:kms:us-east-1:xxxxxxxxxxxx:alias/aws/s3',
//       environment: {
//         computeType: 'BUILD_GENERAL1_SMALL',
//         environmentVariables: [],
//         image: 'aws/codebuild/standard:7.0',
//         imagePullCredentialsType: 'CODEBUILD',
//         privilegedMode: false,
//         type: 'LINUX_CONTAINER'
//       },
//       lastModified: 2023-08-18T14:46:48.979Z,
//       name: 'MyCodeBuilder',
//       projectVisibility: 'PRIVATE',
//       queuedTimeoutInMinutes: 480,
//       serviceRole: 'arn:aws:iam::xxxxxxxxxxxx:role/CodeBuildAdmin',
//       source: {
//         insecureSsl: false,
//         location: 'https://...',
//         reportBuildStatus: false,
//         type: 'GITHUB'
//       },
//       timeoutInMinutes: 60
//     }
//   }
```

```
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateProject in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Cognito Identity mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Cognito Identity Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für JavaScript (v3)

> Zeigt, wie Sie das verwenden AWS SDK für JavaScript , um eine React-Anwendung zu erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Beispiele für Amazon Cognito Identity Provider, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Cognito Identity Provider Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- API-Details finden Sie ListUserPools in der AWS SDK für JavaScript API-Referenz.

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- API-Details finden Sie AdminGetUser in der AWS SDK für JavaScript API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
```

```
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK für JavaScript API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });
```

```
  return client.send(command);
};
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK für JavaScript API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK für JavaScript API-Referenz.

`ConfirmDevice`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmDevice`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- API-Details finden Sie ConfirmDevice in der AWS SDK für JavaScript API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const confirmSignUp = ({ clientId, username, code }) => {
```

```
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK für JavaScript API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
```

```
    return [null, err];
  }
};
```

- API-Details finden Sie DeleteUser in der AWS SDK für JavaScript API-Referenz.

`InitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`InitiateAuth`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- API-Details finden Sie InitiateAuth in der AWS SDK für JavaScript API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- API-Details finden Sie ListUsers in der AWS SDK für JavaScript API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });
```

```
  return client.send(command);
};
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK für JavaScript API-Referenz.

`RespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`RespondToAuthChallenge`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
```

```
};
```

- API-Details finden Sie RespondToAuthChallenge in der AWS SDK für JavaScript API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- API-Details finden Sie SignUp in der AWS SDK für JavaScript API-Referenz.

`UpdateUserPool`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUserPool`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- API-Details finden Sie UpdateUserPool in der AWS SDK für JavaScript API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK für JavaScript API-Referenz.

Szenarien

Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie bekannte Amazon-Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie einen interaktiven „Szenario“-Lauf. Die JavaScript (v3) -Beispiele teilen sich einen Szenario-Runner, um komplexe Beispiele zu rationalisieren. Der komplette Quellcode ist aktiviert GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
```

```
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Dieses Szenario demonstriert die automatische Bestätigung eines bekannten Benutzers. Es orchestriert die Beispielschritte.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
```

```
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
```

```
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
```

```
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
```

```
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });
```

```
    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
```

```
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);
```

```
const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with: ${state.token.slice(0,
 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
```

```
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Dies sind Schritte, die mit anderen Szenarien geteilt werden.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
```

```
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

Ein Handler für den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
```

```
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName '${event.userName}'.
 Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Modul für CloudWatch Logs-Aktionen.

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";
```

```
/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[] |
 null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
```

```
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für Amazon-Cognito-Aktionen.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
```

```
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
```

```
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password: string }}
 config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
```

```
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für DynamoDB-Aktionen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
```

```
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string, unknown>[] }}
 config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.

- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Für ein optimales Erlebnis klonen Sie das GitHub Repository und führen Sie dieses Beispiel aus. Der folgende Code ist ein Teil der vollständigen Beispielanwendung.

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-up'
 command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;
```

```
  try {
    validateUser(username, password, email);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
```

```
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the 'confirm-
sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};
```

```
export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-software-
token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};
```

```
const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-initiate-
auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
```

```
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-auth-
challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
```

```
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
```

```
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };
```

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- API-Details finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Amazon Comprehend-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

SDK für JavaScript (v3)

> Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.
>
> In diesem Beispiel verwendete Dienste
>
> - Amazon Comprehend
> - Amazon SES
> - Amazon Transcribe
> - Amazon Translate

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel Building an Amazon Lex-Chatbot im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie das innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
```

```
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
```

```
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Amazon DocumentDB-Beispiele, die das SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon DocumentDB Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Amazon DocumentDB-Ereignisses unter Verwendung von Lambda. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Ein Amazon DocumentDB-Ereignis mit Lambda wird ausgeführt TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';
```

```
console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null, 2));
};
```

DynamoDB-Beispiele, die SDK für (v3) verwenden JavaScript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit DynamoDB Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Weitere Informationen zur Arbeit mit DynamoDB finden Sie unter DynamoDB programmieren mit. AWS SDK für JavaScript JavaScript

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response.TableNames.join("\n"));
  return response;
};
```

- API-Details finden Sie in der API-Referenz. ListTables AWS SDK für JavaScript

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { readFileSync } from "node:fs";
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";

/**
 * This module is a convenience library. It abstracts Amazon DynamoDB's data type
 * descriptors (such as S, N, B, and BOOL) by marshalling JavaScript objects into
 * AttributeValue shapes.
 */
import {
  BatchWriteCommand,
  DeleteCommand,
  DynamoDBDocumentClient,
  GetCommand,
```

```
  PutCommand,
  UpdateCommand,
  paginateQuery,
  paginateScan,
} from "@aws-sdk/lib-dynamodb";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);
const tableName = getUniqueName("Movies");
const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);

export const main = async () => {
  /**
   * Create a table.
   */

  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "year",
        // 'N' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "N",
      },
      { AttributeName: "title", AttributeType: "S" },
    ],
    // The KeySchema defines the primary key. The primary key can be
```

```
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [
      // The way your data is accessed determines how you structure your keys.
      // The movies table will be queried for movies by year. It makes sense
      // to make year our partition (HASH) key.
      { AttributeName: "year", KeyType: "HASH" },
      { AttributeName: "title", KeyType: "RANGE" },
    ],
  });

  log("Creating a table.");
  const createTableResponse = await client.send(createTableCommand);
  log(`Table created: ${JSON.stringify(createTableResponse.TableDescription)}`);

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Add a movie to the table.
   */

  log("Adding a single movie to the table.");
  // PutCommand is the first example usage of 'lib-dynamodb'.
  const putCommand = new PutCommand({
    TableName: tableName,
    Item: {
      // In 'client-dynamodb', the AttributeValue would be required (`year: { N:
 1981 }`)
      // 'lib-dynamodb' simplifies the usage ( `year: 1981` )
      year: 1981,
      // The preceding KeySchema defines 'title' as our sort (RANGE) key, so 'title'
      // is required.
      title: "The Evil Dead",
      // Every other attribute is optional.
      info: {
        genres: ["Horror"],
      },
    },
```

```
  });
  await docClient.send(putCommand);
  log("The movie was added.");

  /**
   * Get a movie from the table.
   */

  log("Getting a single movie from the table.");
  const getCommand = new GetCommand({
    TableName: tableName,
    // Requires the complete primary key. For the movies table, the primary key
    // is only the id (partition key).
    Key: {
      year: 1981,
      title: "The Evil Dead",
    },
    // Set this to make sure that recent writes are reflected.
    // For more information, see https://docs.aws.amazon.com/amazondynamodb/latest/
developerguide/HowItWorks.ReadConsistency.html.
    ConsistentRead: true,
  });
  const getResponse = await docClient.send(getCommand);
  log(`Got the movie: ${JSON.stringify(getResponse.Item)}`);

  /**
   * Update a movie in the table.
   */

  log("Updating a single movie in the table.");
  const updateCommand = new UpdateCommand({
    TableName: tableName,
    Key: { year: 1981, title: "The Evil Dead" },
    // This update expression appends "Comedy" to the list of genres.
    // For more information on update expressions, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.UpdateExpressions.html
    UpdateExpression: "set #i.#g = list_append(#i.#g, :vals)",
    ExpressionAttributeNames: { "#i": "info", "#g": "genres" },
    ExpressionAttributeValues: {
      ":vals": ["Comedy"],
    },
    ReturnValues: "ALL_NEW",
  });
```

```
const updateResponse = await docClient.send(updateCommand);
log(`Movie updated: ${JSON.stringify(updateResponse.Attributes)}`);

/**
 * Delete a movie from the table.
 */

log("Deleting a single movie from the table.");
const deleteCommand = new DeleteCommand({
  TableName: tableName,
  Key: { year: 1981, title: "The Evil Dead" },
});
await docClient.send(deleteCommand);
log("Movie deleted.");

/**
 * Upload a batch of movies.
 */

log("Adding movies from local JSON file.");
const file = readFileSync(
  `${dirname}../../../../resources/sample_files/movies.json`,
);
const movies = JSON.parse(file.toString());
// chunkArray is a local convenience function. It takes an array and returns
// a generator function. The generator function yields every N items.
const movieChunks = chunkArray(movies, 25);
// For every chunk of 25 movies, make one BatchWrite request.
for (const chunk of movieChunks) {
  const putRequests = chunk.map((movie) => ({
    PutRequest: {
      Item: movie,
    },
  }));

  const command = new BatchWriteCommand({
    RequestItems: {
      [tableName]: putRequests,
    },
  });

  await docClient.send(command);
}
log("Movies added.");
```

```
  /**
   * Query for movies by year.
   */

  log("Querying for all movies from 1981.");
  const paginatedQuery = paginateQuery(
    { client: docClient },
    {
      TableName: tableName,
      //For more information about query expressions, see
      // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Query.html#Query.KeyConditionExpressions
      KeyConditionExpression: "#y = :y",
      // 'year' is a reserved word in DynamoDB. Indicate that it's an attribute
      // name by using an expression attribute name.
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y": 1981 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1981 = [];
  for await (const page of paginatedQuery) {
    movies1981.push(...page.Items);
  }
  log(`Movies: ${movies1981.map((m) => m.title).join(", ")}`);

  /**
   * Scan the table for movies between 1980 and 1990.
   */

  log("Scan for movies released between 1980 and 1990");
  // A 'Scan' operation always reads every item in the table. If your design
 requires
  // the use of 'Scan', consider indexing your table or changing your design.
  // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-query-
scan.html
  const paginatedScan = paginateScan(
    { client: docClient },
    {
      TableName: tableName,
```

```
      // Scan uses a filter expression instead of a key condition expression. Scan
will
      // read the entire table and then apply the filter.
      FilterExpression: "#y between :y1 and :y2",
      ExpressionAttributeNames: { "#y": "year" },
      ExpressionAttributeValues: { ":y1": 1980, ":y2": 1990 },
      ConsistentRead: true,
    },
  );
  /**
   * @type { Record<string, any>[] };
   */
  const movies1980to1990 = [];
  for await (const page of paginatedScan) {
    movies1980to1990.push(...page.Items);
  }
  log(
    `Movies: ${movies1980to1990
      .map((m) => `${m.title} (${m.year})`)
      .join(", ")}`,
  );

  /**
   * Delete the table.
   */

  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  log(`Deleting table ${tableName}.`);
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const breakfastFoods = ["Eggs", "Bacon", "Sausage"];
  const command = new BatchExecuteStatementCommand({
    Statements: breakfastFoods.map((food) => ({
      Statement: `INSERT INTO BreakfastFoods value {'Name':?}`,
      Parameters: [food],
    })),
  });
```

```
  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Rufen Sie mithilfe von PartiQL einen Stapel von Elementen ab.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Teaspoons"],
        ConsistentRead: true,
      },
      {
        Statement: "SELECT * FROM PepperMeasurements WHERE Unit=?",
        Parameters: ["Grams"],
        ConsistentRead: true,
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Aktualisieren Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const eggUpdates = [
    ["duck", "fried"],
    ["chicken", "omelette"],
  ];
  const command = new BatchExecuteStatementCommand({
    Statements: eggUpdates.map((change) => ({
      Statement: "UPDATE Eggs SET Style=? where Variety=?",
      Parameters: [change[1], change[0]],
    })),
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Löschen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchExecuteStatementCommand({
    Statements: [
      {
```

```
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Grape"],
      },
      {
        Statement: "DELETE FROM Flavors where Name=?",
        Parameters: ["Strawberry"],
      },
    ],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für JavaScript API-Referenz.

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter BatchGet.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { BatchGetCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new BatchGetCommand({
    // Each key in this object is the name of a table. This example refers
```

```
    // to a Books table.
    RequestItems: {
      Books: {
        // Each entry in Keys is an object that specifies a primary key.
        Keys: [
          {
            Title: "How to AWS",
          },
          {
            Title: "DynamoDB for DBAs",
          },
        ],
        // Only return the "Title" and "PageCount" attributes.
        ProjectionExpression: "Title, PageCount",
      },
    },
  });

  const response = await docClient.send(command);
  console.log(response.Responses.Books);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie BatchGetItem in der AWS SDK für JavaScript API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter BatchWrite.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";
import { readFileSync } from "node:fs";

// These modules are local to our GitHub repository. We recommend cloning
// the project from GitHub if you want to run this example.
// For more information, see https://github.com/awsdocs/aws-doc-sdk-examples.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { chunkArray } from "@aws-doc-sdk-examples/lib/utils/util-array.js";

const dirname = dirnameFromMetaUrl(import.meta.url);

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const file = readFileSync(
    `${dirname}../../../../../resources/sample_files/movies.json`,
  );

  const movies = JSON.parse(file.toString());

  // chunkArray is a local convenience function. It takes an array and returns
  // a generator function. The generator function yields every N items.
  const movieChunks = chunkArray(movies, 25);

  // For every chunk of 25 movies, make one BatchWrite request.
  for (const chunk of movieChunks) {
    const putRequests = chunk.map((movie) => ({
      PutRequest: {
        Item: movie,
      },
    }));

    const command = new BatchWriteCommand({
      RequestItems: {
        // An existing table is required. A composite key of 'title' and 'year' is
 recommended
        // to account for duplicate titles.
        BatchWriteMoviesTable: putRequests,
```

```
      },
    });

    await docClient.send(command);
  }
};
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für JavaScript API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new CreateTableCommand({
    TableName: "EspressoDrinks",
    // For more information about data types,
    // see https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
HowItWorks.NamingRulesDataTypes.html#HowItWorks.DataTypes and
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
    AttributeDefinitions: [
      {
        AttributeName: "DrinkName",
        AttributeType: "S",
      },
    ],
    KeySchema: [
      {
```

```
        AttributeName: "DrinkName",
        KeyType: "HASH",
      },
    ],
    BillingMode: "PAY_PER_REQUEST",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateTable in der AWS SDK für JavaScript API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter DeleteCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, DeleteCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new DeleteCommand({
    TableName: "Sodas",
    Key: {
```

```
      Flavor: "Cola",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteItem in der AWS SDK für JavaScript API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DeleteTableCommand({
    TableName: "DecafCoffees",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie DeleteTable in der AWS SDK für JavaScript API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeTableCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new DescribeTableCommand({
    TableName: "Pastries",
  });

  const response = await client.send(command);
  console.log(`TABLE NAME: ${response.Table.TableName}`);
  console.log(`TABLE ITEM COUNT: ${response.Table.ItemCount}`);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DescribeTable in der AWS SDK für JavaScript API-Referenz.

`DescribeTimeToLive`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTimeToLive`.

SDK für JavaScript (v3)

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK für JavaScript.

```
import { DynamoDBClient, DescribeTimeToLiveCommand } from "@aws-sdk/client-
dynamodb";

export const describeTTL = async (tableName, region) => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    try {
        const ttlDescription = await client.send(new
 DescribeTimeToLiveCommand({ TableName: tableName }));

        if (ttlDescription.TimeToLiveDescription.TimeToLiveStatus === 'ENABLED') {
            console.log("TTL is enabled for table %s.", tableName);
        } else {
            console.log("TTL is not enabled for table %s.", tableName);
        }

        return ttlDescription;
    } catch (e) {
        console.error(`Error describing table: ${e}`);
        throw e;
    }
}

// Example usage (commented out for testing)
// describeTTL('your-table-name', 'us-east-1');
```

- API-Details finden Sie DescribeTimeToLive in der AWS SDK für JavaScript API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: `INSERT INTO Flowers value {'Name':?}`,
    Parameters: ["Rose"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Rufen Sie ein Element mithilfe von PartiQL ab.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "SELECT * FROM CloudTypes WHERE IsStorm=?",
    Parameters: [false],
    ConsistentRead: true,
  });
```

```
  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Aktualisieren Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ExecuteStatementCommand({
    Statement: "UPDATE EyeColors SET IsRecessive=? where Color=?",
    Parameters: [true, "blue"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

Löschen Sie ein Element mithilfe von PartiQL.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";

import {
  ExecuteStatementCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
```

```
  const command = new ExecuteStatementCommand({
    Statement: "DELETE FROM PaintColors where Name=?",
    Parameters: ["Purple"],
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für JavaScript API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter GetCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new GetCommand({
    TableName: "AngryAnimals",
    Key: {
      CommonName: "Shoebill",
    },
  });
```

```
  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie GetItem in der AWS SDK für JavaScript API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListTablesCommand, DynamoDBClient } from "@aws-sdk/client-dynamodb";

const client = new DynamoDBClient({});

export const main = async () => {
  const command = new ListTablesCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListTables in der AWS SDK für JavaScript API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter PutCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { PutCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new PutCommand({
    TableName: "HappyAnimals",
    Item: {
      CommonName: "Shiba Inu",
    },
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie PutItem in der AWS SDK für JavaScript API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter QueryCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { QueryCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new QueryCommand({
    TableName: "CoffeeCrop",
    KeyConditionExpression:
      "OriginCountry = :originCountry AND RoastDate > :roastDate",
    ExpressionAttributeValues: {
      ":originCountry": "Ethiopia",
      ":roastDate": "2023-05-01",
    },
    ConsistentRead: true,
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter ScanCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, ScanCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new ScanCommand({
    ProjectionExpression: "#Name, Color, AvgLifeSpan",
    ExpressionAttributeNames: { "#Name": "Name" },
    TableName: "Birds",
  });

  const response = await docClient.send(command);
  for (const bird of response.Items) {
    console.log(`${bird.Name} - (${bird.Color}, ${bird.AvgLifeSpan})`);
  }
  return response;
};
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für JavaScript -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Dokument-Client verwendet, um die Arbeit mit Elementen in DynamoDB zu vereinfachen. API-Details finden Sie unter UpdateCommand.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, UpdateCommand } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

export const main = async () => {
  const command = new UpdateCommand({
    TableName: "Dogs",
    Key: {
      Breed: "Labrador",
    },
    UpdateExpression: "set Color = :color",
    ExpressionAttributeValues: {
      ":color": "black",
    },
    ReturnValues: "ALL_NEW",
  });

  const response = await docClient.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

`UpdateTimeToLive`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTimeToLive`.

SDK für JavaScript (v3)

Aktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-dynamodb";

export const enableTTL = async (tableName, ttlAttribute, region = 'us-east-1') => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: true,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL enabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to enable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error enabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// enableTTL('ExampleTable', 'exampleTtlAttribute');
```

Deaktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTimeToLiveCommand } from "@aws-sdk/client-dynamodb";

export const disableTTL = async (tableName, ttlAttribute, region = 'us-east-1') => {

    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const params = {
        TableName: tableName,
        TimeToLiveSpecification: {
            Enabled: false,
            AttributeName: ttlAttribute
        }
    };

    try {
        const response = await client.send(new UpdateTimeToLiveCommand(params));
        if (response.$metadata.httpStatusCode === 200) {
            console.log(`TTL disabled successfully for table ${tableName}, using
 attribute name ${ttlAttribute}.`);
        } else {
            console.log(`Failed to disable TTL for table ${tableName}, response
 object: ${response}`);
        }
        return response;
    } catch (e) {
        console.error(`Error disabling TTL: ${e}`);
        throw e;
    }
};

// Example usage (commented out for testing)
// disableTTL('ExampleTable', 'exampleTtlAttribute');
```

- API-Details finden Sie UpdateTimeToLive in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Codebeispiele zeigen, wie eine Anwendung erstellt wird, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Vergleichen mehrerer Werte mit einem einzigen Attribut

Im folgenden Codebeispiel wird gezeigt, wie Sie mehrere Werte mit einem einzigen Attribut in DynamoDB vergleichen.

- Verwenden des IN-Operators, um mehrere Werte mit einem einzigen Attribut zu vergleichen
- Vergleichen des IN-Operators mit mehreren OR-Bedingungen
- Verständnis der Vorteilen von IN hinsichtlich Leistung und Ausdruckskomplexität

SDK für JavaScript (v3)

Vergleichen Sie mehrere Werte mit einem einzigen Attribut mithilfe von AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  ScanCommand,
  QueryCommand
```

```
} = require("@aws-sdk/lib-dynamodb");

/**
 * Query or scan a DynamoDB table to find items where an attribute matches any value
 from a list.
 *
 * This function demonstrates the use of the IN operator to compare a single
 attribute
 * against multiple possible values, which is more efficient than using multiple OR
 conditions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} attributeName - The name of the attribute to compare against the
 values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key attribute
 for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the matching
 items
 */
async function compareMultipleValues(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the filter expression using the IN operator
  const filterExpression = `${attributeName} IN (${valuesList.map((_, index) =>
 `:val${index}`).join(', ')})`;

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});
```

```
  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
```

```
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}

/**
 * Alternative implementation using multiple OR conditions instead of the IN
 operator.
 *
 * This function is provided for comparison to show why using the IN operator is
 preferable.
 * With many values, this approach becomes verbose and less efficient.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {string} attributeName - The name of the attribute to compare against the
 values list
 * @param {Array} valuesList - List of values to compare the attribute against
 * @param {string} [partitionKeyName] - Optional name of the partition key attribute
 for query operations
 * @param {string} [partitionKeyValue] - Optional value of the partition key to
 query
 * @returns {Promise<Object>} - The response from DynamoDB containing the matching
 items
 */
async function compareWithOrConditions(
  config,
  tableName,
  attributeName,
  valuesList,
  partitionKeyName,
  partitionKeyValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // If no values provided, return empty result
  if (!valuesList || valuesList.length === 0) {
    return {
      Items: [],
      Count: 0
    };
  }

  // Create the filter expression using multiple OR conditions
  const filterConditions = valuesList.map((_, index) => `${attributeName} = :val
${index}`);
  const filterExpression = filterConditions.join(' OR ');

  // Create expression attribute values for the values list
  const expressionAttributeValues = valuesList.reduce((acc, val, index) => {
    acc[`:val${index}`] = val;
    return acc;
  }, {});

  // If partition key is provided, perform a query operation
  if (partitionKeyName && partitionKeyValue) {
    const keyCondition = `${partitionKeyName} = :partitionKey`;
```

```
    expressionAttributeValues[':partitionKey'] = partitionKeyValue;

    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
      const params = {
        TableName: tableName,
        KeyConditionExpression: keyCondition,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new QueryCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  } else {
    // Otherwise, perform a scan operation
    // Initialize array to collect all items
    let allItems = [];
    let lastEvaluatedKey;

    // Use pagination to get all results
    do {
```

```
      const params = {
        TableName: tableName,
        FilterExpression: filterExpression,
        ExpressionAttributeValues: expressionAttributeValues
      };

      // Add ExclusiveStartKey if we have a lastEvaluatedKey from a previous scan
      if (lastEvaluatedKey) {
        params.ExclusiveStartKey = lastEvaluatedKey;
      }

      const response = await docClient.send(new ScanCommand(params));

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems = [...allItems, ...response.Items];
      }

      // Get the key for the next page of results
      lastEvaluatedKey = response.LastEvaluatedKey;
    } while (lastEvaluatedKey);

    // Return the complete result
    return {
      Items: allItems,
      Count: allItems.length
    };
  }
}
```

Anwendungsbeispiel für den Vergleich mehrerer Werte mit AWS SDK für JavaScript.

```
/**
 * Example of how to use the compareMultipleValues function.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const attributeName = "Category";
  const valuesList = ["Electronics", "Computers", "Accessories"];
```

```
 console.log(`Searching for products in any of these categories:
${valuesList.join(', ')}`);

 try {
   // Using the IN operator (recommended approach)
   console.log("\nApproach 1: Using the IN operator");
   const response = await compareMultipleValues(
     config,
     tableName,
     attributeName,
     valuesList
   );

   console.log(`Found ${response.Count} products in the specified categories`);

   // Using multiple OR conditions (alternative approach)
   console.log("\nApproach 2: Using multiple OR conditions");
   const response2 = await compareWithOrConditions(
     config,
     tableName,
     attributeName,
     valuesList
   );

   console.log(`Found ${response2.Count} products in the specified categories`);

   // Example with a query operation
   console.log("\nQuerying a specific manufacturer's products in multiple
categories");
   const partitionKeyName = "Manufacturer";
   const partitionKeyValue = "Acme";

   const response3 = await compareMultipleValues(
     config,
     tableName,
     attributeName,
     valuesList,
     partitionKeyName,
     partitionKeyValue
   );

   console.log(`Found ${response3.Count} Acme products in the specified
categories`);
```

```
    // Explain the benefits of using the IN operator
    console.log("\nBenefits of using the IN operator:");
    console.log("1. More concise expression compared to multiple OR conditions");
    console.log("2. Better readability and maintainability");
    console.log("3. Potentially better performance with large value lists");
    console.log("4. Simpler code that's less prone to errors");
    console.log("5. Easier to modify when adding or removing values");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - Query
  - Scan

Bedingtes Aktualisieren der TTL eines Elements

Im folgenden Codebeispiel wird gezeigt, wie die TTL eines Elements bedingungsabhängig aktualisiert wird.

SDK für JavaScript (v3)

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItemConditional = async (tableName, partitionKey, sortKey, region
 = 'us-east-1', newAttribute = 'default-value') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
```

```
        TableName: tableName,
        Key: marshall({
            artist: partitionKey,
            album: sortKey
        }),
        UpdateExpression: "SET newAttribute = :newAttribute",
        ConditionExpression: "expireAt > :expiration",
        ExpressionAttributeValues: marshall({
            ':newAttribute': newAttribute,
            ':expiration': currentTime
        }),
        ReturnValues: "ALL_NEW"
    };

    try {
        const response = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(response.Attributes);
        console.log("Item updated successfully: ", responseData);
        return responseData;
    } catch (error) {
        if (error.name === "ConditionalCheckFailedException") {
            console.log("Condition check failed: Item's 'expireAt' is expired.");
        } else {
            console.error("Error updating item: ", error);
        }
        throw error;
    }
};

// Example usage (commented out for testing)
// updateItemConditional('your-table-name', 'your-partition-key-value', 'your-sort-
key-value');
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Zählen von Ausdrucksoperatoren

Im folgenden Codebeispiel wird gezeigt, wie Ausdrucksoperatoren in DynamoDB gezählt werden.

- Machen Sie sich mit der Obergrenze von 300 Operatoren in DynamoDB vertraut.
- Zählen Sie Operatoren in komplexen Ausdrücken.

- Optimieren Sie Ausdrücke, um innerhalb der Grenzwerte zu bleiben.

SDK für JavaScript (v3)

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  QueryCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Create a complex filter expression with a specified number of conditions.
 *
 * This function demonstrates how to generate a complex expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} conditionsCount - Number of conditions to include
 * @param {boolean} useAnd - Whether to use AND (true) or OR (false) between
 conditions
 * @returns {Object} - Object containing the filter expression and attribute values
 */
function createComplexFilterExpression(conditionsCount, useAnd = true) {
  // Initialize the expression parts and attribute values
  const conditions = [];
  const expressionAttributeValues = {};

  // Generate the specified number of conditions
  for (let i = 0; i < conditionsCount; i++) {
    // Alternate between different comparison operators for variety
    let condition;
    const valueKey = `:val${i}`;

    switch (i % 5) {
      case 0:
        condition = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        condition = `attribute${i} > ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
```

```
        break;
      case 2:
        condition = `attribute${i} < ${valueKey}`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
      case 3:
        condition = `contains(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = `substring${i}`;
        break;
      case 4:
        condition = `attribute_exists(attribute${i})`;
        break;
    }

    conditions.push(condition);
  }

  // Join the conditions with AND or OR
  const operator = useAnd ? " AND " : " OR ";
  const filterExpression = conditions.join(operator);

  // Calculate the operator count
  // Each condition has 1 operator (=, >, <, contains, attribute_exists)
  // Each AND or OR between conditions is 1 operator
  const operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount -
 1 : 0);

  return {
    filterExpression,
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Create a complex update expression with a specified number of operations.
 *
 * This function demonstrates how to generate a complex update expression with
 * a specific number of operators to test the 300 operator limit.
 *
 * @param {number} operationsCount - Number of operations to include
 * @returns {Object} - Object containing the update expression and attribute values
 */
function createComplexUpdateExpression(operationsCount) {
```

```
  // Initialize the expression parts and attribute values
  const setOperations = [];
  const expressionAttributeValues = {};

  // Generate the specified number of SET operations
  for (let i = 0; i < operationsCount; i++) {
    // Alternate between different types of SET operations
    let operation;
    const valueKey = `:val${i}`;

    switch (i % 3) {
      case 0:
        // Simple assignment (1 operator: =)
        operation = `attribute${i} = ${valueKey}`;
        expressionAttributeValues[valueKey] = `value${i}`;
        break;
      case 1:
        // Addition (2 operators: = and +)
        operation = `attribute${i} = attribute${i} + ${valueKey}`;
        expressionAttributeValues[valueKey] = i;
        break;
      case 2:
        // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
        operation = `attribute${i} = if_not_exists(attribute${i}, ${valueKey})`;
        expressionAttributeValues[valueKey] = i * 10;
        break;
    }

    setOperations.push(operation);
  }

  // Create the update expression
  const updateExpression = `SET ${setOperations.join(", ")}`;

  // Calculate the operator count
  // Each operation has 1-2 operators as noted above
  let operatorCount = 0;
  for (let i = 0; i < operationsCount; i++) {
    operatorCount += (i % 3 === 0) ? 1 : 2;
  }

  return {
    updateExpression,
```

```
    expressionAttributeValues,
    operatorCount
  };
}

/**
 * Test the operator limit by attempting an operation with a complex expression.
 *
 * This function demonstrates what happens when an expression approaches or
 * exceeds the 300 operator limit.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} operatorCount - Target number of operators to include
 * @returns {Promise<Object>} - Result of the operation attempt
 */
async function testOperatorLimit(
  config,
  tableName,
  key,
  operatorCount
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create a complex update expression with the specified operator count
  const { updateExpression, expressionAttributeValues, operatorCount: actualCount }
 =
    createComplexUpdateExpression(Math.ceil(operatorCount / 1.5)); // Adjust to get
 close to target count

  console.log(`Generated update expression with approximately ${actualCount}
 operators`);

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };
```

```
  try {
    // Attempt the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      message: `Operation succeeded with ${actualCount} operators`,
      data: response
    };
  } catch (error) {
    // Check if the error is due to exceeding the operator limit
    if (error.name === "ValidationException" &&
        error.message.includes("too many operators")) {
      return {
        success: false,
        message: `Operation failed: ${error.message}`,
        operatorCount: actualCount
      };
    }

    // Return other errors
    return {
      success: false,
      message: `Operation failed: ${error.message}`,
      error
    };
  }
}

/**
 * Break down a complex expression into multiple simpler operations.
 *
 * This function demonstrates how to handle expressions that would exceed
 * the 300 operator limit by breaking them into multiple operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {number} totalOperations - Total number of operations to perform
 * @returns {Promise<Object>} - Result of the operations
 */
async function breakDownComplexExpression(
  config,
  tableName,
```

```
  key,
  totalOperations
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Calculate how many operations we can safely include in each batch
  // Using 150 as a conservative limit (well below 300)
  const operationsPerBatch = 100;
  const batchCount = Math.ceil(totalOperations / operationsPerBatch);

  console.log(`Breaking down ${totalOperations} operations into ${batchCount}
 batches`);

  const results = [];

  // Process each batch
  for (let batch = 0; batch < batchCount; batch++) {
    // Calculate the operations for this batch
    const batchStart = batch * operationsPerBatch;
    const batchEnd = Math.min(batchStart + operationsPerBatch, totalOperations);
    const batchSize = batchEnd - batchStart;

    console.log(`Processing batch ${batch + 1}/${batchCount} with ${batchSize}
 operations`);

    // Create an update expression for this batch
    const { updateExpression, expressionAttributeValues, operatorCount } =
      createComplexUpdateExpression(batchSize);

    // Define the update parameters
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: updateExpression,
      ExpressionAttributeValues: expressionAttributeValues,
      ReturnValues: "UPDATED_NEW"
    };

    try {
      // Perform the update operation for this batch
      const response = await docClient.send(new UpdateCommand(params));
```

```
      results.push({
        batch: batch + 1,
        success: true,
        operatorCount,
        attributes: response.Attributes
      });
    } catch (error) {
      results.push({
        batch: batch + 1,
        success: false,
        operatorCount,
        error: error.message
      });

      // Stop processing if an error occurs
      break;
    }
  }

  return {
    totalBatches: batchCount,
    results
  };
}

/**
 * Count operators in a DynamoDB expression based on the rules in the documentation.
 *
 * This function demonstrates how operators are counted according to the
 * DynamoDB documentation.
 *
 * @param {string} expression - The DynamoDB expression to analyze
 * @returns {Object} - Breakdown of operator counts
 */
function countOperatorsInExpression(expression) {
  // Initialize counters for different operator types
  const counts = {
    comparisonOperators: 0,
    logicalOperators: 0,
    functions: 0,
    arithmeticOperators: 0,
    specialOperators: 0,
    total: 0
  };
```

```
  // Count comparison operators (=, <>, <, <=, >, >=)
  const comparisonRegex = /[^<>]=[^=]|<>|<=|>=|[^<]>[^=]|[^>]<[^=]/g;
  const comparisonMatches = expression.match(comparisonRegex) || [];
  counts.comparisonOperators = comparisonMatches.length;

  // Count logical operators (AND, OR, NOT)
  const andMatches = expression.match(/\bAND\b/g) || [];
  const orMatches = expression.match(/\bOR\b/g) || [];
  const notMatches = expression.match(/\bNOT\b/g) || [];
  counts.logicalOperators = andMatches.length + orMatches.length +
 notMatches.length;

  // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
  const functionRegex = /\b(attribute_exists|attribute_not_exists|attribute_type|
begins_with|contains|size|if_not_exists)\(/g;
  const functionMatches = expression.match(functionRegex) || [];
  counts.functions = functionMatches.length;

  // Count arithmetic operators (+ and -)
  const arithmeticMatches = expression.match(/[a-zA-Z0-9_)\]]\s*[\+\-]\s*[a-zA-
Z0-9_(:]/g) || [];
  counts.arithmeticOperators = arithmeticMatches.length;

  // Count special operators (BETWEEN, IN)
  const betweenMatches = expression.match(/\bBETWEEN\b/g) || [];
  const inMatches = expression.match(/\bIN\b/g) || [];
  counts.specialOperators = betweenMatches.length + inMatches.length;

  // Add extra operators for BETWEEN (each BETWEEN includes an AND)
  counts.logicalOperators += betweenMatches.length;

  // Calculate total
  counts.total = counts.comparisonOperators +
                 counts.logicalOperators +
                 counts.functions +
                 counts.arithmeticOperators +
                 counts.specialOperators;

  return counts;
}
```

Beispiel für die Verwendung des Ausdrucksoperators Zählung mit AWS SDK für JavaScript.

```
/**
 * Example of how to work with expression operator counting.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating DynamoDB expression operator counting and the 300
 operator limit");

  try {
    // Example 1: Analyze a simple expression
    console.log("\nExample 1: Analyzing a simple expression");
    const simpleExpression = "Price = :price AND Rating > :rating AND Category IN
 (:cat1, :cat2, :cat3)";
    const simpleCount = countOperatorsInExpression(simpleExpression);

    console.log(`Expression: ${simpleExpression}`);
    console.log("Operator count breakdown:");
    console.log(`- Comparison operators: ${simpleCount.comparisonOperators}`);
    console.log(`- Logical operators: ${simpleCount.logicalOperators}`);
    console.log(`- Functions: ${simpleCount.functions}`);
    console.log(`- Arithmetic operators: ${simpleCount.arithmeticOperators}`);
    console.log(`- Special operators: ${simpleCount.specialOperators}`);
    console.log(`- Total operators: ${simpleCount.total}`);

    // Example 2: Analyze a complex expression
    console.log("\nExample 2: Analyzing a complex expression");
    const complexExpression =
      "(attribute_exists(Category) AND Size BETWEEN :min AND :max) OR " +
      "(Price > :price AND contains(Description, :keyword) AND " +
      "(Rating >= :minRating OR Reviews > :minReviews))";
    const complexCount = countOperatorsInExpression(complexExpression);

    console.log(`Expression: ${complexExpression}`);
    console.log("Operator count breakdown:");
    console.log(`- Comparison operators: ${complexCount.comparisonOperators}`);
    console.log(`- Logical operators: ${complexCount.logicalOperators}`);
    console.log(`- Functions: ${complexCount.functions}`);
    console.log(`- Arithmetic operators: ${complexCount.arithmeticOperators}`);
```

```
    console.log(`- Special operators: ${complexCount.specialOperators}`);
    console.log(`- Total operators: ${complexCount.total}`);

    // Example 3: Test approaching the operator limit
    console.log("\nExample 3: Testing an expression approaching the operator
limit");
    const approachingLimit = await testOperatorLimit(config, tableName, key, 290);
    console.log(approachingLimit.message);

    // Example 4: Test exceeding the operator limit
    console.log("\nExample 4: Testing an expression exceeding the operator limit");
    const exceedingLimit = await testOperatorLimit(config, tableName, key, 310);
    console.log(exceedingLimit.message);

    // Example 5: Breaking down a complex expression
    console.log("\nExample 5: Breaking down a complex expression into multiple
operations");
    const breakdownResult = await breakDownComplexExpression(config, tableName, key,
500);
    console.log(`Processed ${breakdownResult.results.length} of
${breakdownResult.totalBatches} batches`);

    // Explain the operator counting rules
    console.log("\nKey points about DynamoDB expression operator counting:");
    console.log("1. The maximum number of operators in any expression is 300");
    console.log("2. Each comparison operator (=, <>, <, <=, >, >=) counts as 1
operator");
    console.log("3. Each logical operator (AND, OR, NOT) counts as 1 operator");
    console.log("4. Each function call (attribute_exists, contains, etc.) counts as
1 operator");
    console.log("5. Each arithmetic operator (+ or -) counts as 1 operator");
    console.log("6. BETWEEN counts as 2 operators (BETWEEN itself and the AND within
it)");
    console.log("7. IN counts as 1 operator regardless of the number of values");
    console.log("8. Parentheses for grouping and attribute paths don't count as
operators");
    console.log("9. When you exceed the limit, the error always reports '301
operators'");
    console.log("10. For complex operations, break them into multiple smaller
operations");

  } catch (error) {
    console.error("Error:", error);
  }
```

```
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Tabelle mit aktiviertem Warmdurchsatz

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit aktiviertem Warmdurchsatz erstellt wird.

SDK für JavaScript (v3)

Erstellen Sie eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung mithilfe von AWS SDK für JavaScript.

```
import { DynamoDBClient, CreateTableCommand } from "@aws-sdk/client-dynamodb";
```

```
export async function createDynamoDBTableWithWarmThroughput(
  tableName,
  partitionKey,
  sortKey,
  miscKeyAttr,
  nonKeyAttr,
  tableProvisionedReadUnits,
  tableProvisionedWriteUnits,
  tableWarmReads,
  tableWarmWrites,
  indexName,
  indexProvisionedReadUnits,
  indexProvisionedWriteUnits,
  indexWarmReads,
  indexWarmWrites,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });
    const command = new CreateTableCommand({
      TableName: tableName,
      AttributeDefinitions: [
          { AttributeName: partitionKey, AttributeType: "S" },
          { AttributeName: sortKey, AttributeType: "S" },
          { AttributeName: miscKeyAttr, AttributeType: "N" },
      ],
      KeySchema: [
          { AttributeName: partitionKey, KeyType: "HASH" },
          { AttributeName: sortKey, KeyType: "RANGE" },
      ],
      ProvisionedThroughput: {
          ReadCapacityUnits: tableProvisionedReadUnits,
          WriteCapacityUnits: tableProvisionedWriteUnits,
      },
      WarmThroughput: {
          ReadUnitsPerSecond: tableWarmReads,
          WriteUnitsPerSecond: tableWarmWrites,
      },
      GlobalSecondaryIndexes: [
          {
            IndexName: indexName,
            KeySchema: [
                { AttributeName: sortKey, KeyType: "HASH" },
```

```
                { AttributeName: miscKeyAttr, KeyType: "RANGE" },
            ],
            Projection: {
                ProjectionType: "INCLUDE",
                NonKeyAttributes: [nonKeyAttr],
            },
            ProvisionedThroughput: {
                ReadCapacityUnits: indexProvisionedReadUnits,
                WriteCapacityUnits: indexProvisionedWriteUnits,
            },
            WarmThroughput: {
                ReadUnitsPerSecond: indexWarmReads,
                WriteUnitsPerSecond: indexWarmWrites,
            },
          },
      ],
    });
    const response = await ddbClient.send(command);
    console.log(response);
    return response;
  } catch (error) {
    console.error(`Error creating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
createDynamoDBTableWithWarmThroughput(
  'example-table',
  'pk',
  'sk',
  'gsiKey',
  'data',
  10, 10, 5, 5,
  'example-index',
  5, 5, 2, 2
);
*/
```

- API-Details finden Sie CreateTable in der AWS SDK für JavaScript API-Referenz.

Erstellen eines Elements mit einer TTL

Das folgende Codebeispiel zeigt, wie Sie ein Element mit TTL erstellen.

SDK für JavaScript (v3)

```
import { DynamoDBClient, PutItemCommand } from "@aws-sdk/client-dynamodb";

export function createDynamoDBItem(table_name, region, partition_key, sort_key) {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    // Get the current time in epoch second format
    const current_time = Math.floor(new Date().getTime() / 1000);

    // Calculate the expireAt time (90 days from now) in epoch second format
    const expire_at = Math.floor((new Date().getTime() + 90 * 24 * 60 * 60 * 1000) /
 1000);

    // Create DynamoDB item
    const item = {
        'partitionKey': {'S': partition_key},
        'sortKey': {'S': sort_key},
        'createdAt': {'N': current_time.toString()},
        'expireAt': {'N': expire_at.toString()}
    };

    const putItemCommand = new PutItemCommand({
        TableName: table_name,
        Item: item,
        ProvisionedThroughput: {
            ReadCapacityUnits: 1,
            WriteCapacityUnits: 1,
        },
    });

    client.send(putItemCommand, function(err, data) {
        if (err) {
            console.log("Exception encountered when creating item %s, here's what
 happened: ", data, err);
            throw err;
        } else {
```

```
            console.log("Item created successfully: %s.", data);
            return data;
        }
    });
}

// Example usage (commented out for testing)
// createDynamoDBItem('your-table-name', 'us-east-1', 'your-partition-key-value',
 'your-sort-key-value');
```

- API-Details finden Sie PutItem in der AWS SDK für JavaScript API-Referenz.

Löschen von Daten mit PartiQL DELETE

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-DELETE-Anweisungen löschen.

SDK für JavaScript (v3)

Löschen Sie Elemente aus einer DynamoDB-Tabelle mithilfe von PartiQL DELETE-Anweisungen mit. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to delete items from a DynamoDB table using
 PartiQL.
 * It shows different ways to delete documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Delete a single item by its partition key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByPartitionKey = async (
```

```
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
```

```
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item deleted successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item:", err);
    throw err;
  }
};

/**
 * Delete an item with a condition to ensure the delete only happens if a condition
 is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${conditionAttribute} = ?`,
    Parameters: [partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
```

```
    console.log("Item deleted with condition successfully");
    return data;
  } catch (err) {
    console.error("Error deleting item with condition:", err);
    throw err;
  }
};

/**
 * Batch delete multiple items using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param keys - Array of objects containing key information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchDeleteItems = async (
  tableName: string,
  keys: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    sortKeyName?: string;
    sortKeyValue?: string | number;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each delete
  const statements = keys.map((key) => {
    if (key.sortKeyName && key.sortKeyValue !== undefined) {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ? AND
 ${key.sortKeyName} = ?`,
        Parameters: [key.partitionKeyValue, key.sortKeyValue],
      };
    } else {
      return {
        Statement: `DELETE FROM "${tableName}" WHERE ${key.partitionKeyName} = ?`,
        Parameters: [key.partitionKeyValue],
      };
    }
  });

  const params = {
```

```
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch deleted successfully");
    return data;
  } catch (err) {
    console.error("Error batch deleting items:", err);
    throw err;
  }
};

/**
 * Delete multiple items that match a filter condition.
 * Note: This performs a scan operation which can be expensive on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const deleteItemsByFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `DELETE FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items deleted by filter successfully");
    return data;
  } catch (err) {
    console.error("Error deleting items by filter:", err);
    throw err;
  }
};
```

```
/**
 * Example usage showing how to delete items with different index types
 */
export const deleteExamples = async () => {
  // Delete an item by partition key (simple primary key)
  await deleteItemByPartitionKey("UsersTable", "userId", "user123");

  // Delete an item by composite key (partition key + sort key)
  await deleteItemByCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789"
  );

  // Delete with a condition
  await deleteItemWithCondition(
    "UsersTable",
    "userId",
    "user789",
    "userStatus",
    "inactive"
  );

  // Batch delete multiple items
  await batchDeleteItems("UsersTable", [
    { partitionKeyName: "userId", partitionKeyValue: "user234" },
    { partitionKeyName: "userId", partitionKeyValue: "user345" },
  ]);

  // Batch delete items with composite keys
  await batchDeleteItems("OrdersTable", [
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order567",
      sortKeyName: "productId",
      sortKeyValue: "prod123",
    },
    {
      partitionKeyName: "orderId",
      partitionKeyValue: "order678",
      sortKeyName: "productId",
```

```
      sortKeyValue: "prod456",
    },
  ]);

  // Delete items by filter (use with caution)
  await deleteItemsByFilter("UsersTable", "userStatus", "deleted");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Einfügen von Daten mithilfe von PartiQL INSERT

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-INSERT-Anweisungen einfügen.

SDK für JavaScript (v3)

Fügen Sie Elemente mithilfe von PartiQL INSERT-Anweisungen mit in eine DynamoDB-Tabelle ein. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to insert items into a DynamoDB table using
 PartiQL.
 * It shows different ways to insert documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Insert a single item into a DynamoDB table using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @returns The response from the ExecuteStatementCommand
```

```
 */
export const insertItem = async (tableName: string, item: Record<string, any>) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Convert the item to a string representation for PartiQL
  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted successfully");
    return data;
  } catch (err) {
    console.error("Error inserting item:", err);
    throw err;
  }
};

/**
 * Insert multiple items into a DynamoDB table using PartiQL batch operation.
 * This is more efficient than inserting items one by one.
 *
 * @param tableName - The name of the DynamoDB table
 * @param items - Array of items to insert
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchInsertItems = async (tableName: string, items: Record<string,
 any>[]) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each item
  const statements = items.map((item) => {
    const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
    return {
      Statement: `INSERT INTO "${tableName}" VALUE ${itemString}`,
    };
  });

  const params = {
```

```
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items inserted successfully");
    return data;
  } catch (err) {
    console.error("Error batch inserting items:", err);
    throw err;
  }
};

/**
 * Insert an item with a condition to prevent overwriting existing items.
 * This is useful for ensuring you don't accidentally overwrite data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param item - The item to insert
 * @param partitionKeyName - The name of the partition key attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const insertItemWithCondition = async (
  tableName: string,
  item: Record<string, any>,
  partitionKeyName: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const itemString = JSON.stringify(item).replace(/"([^"]+)":/g, '$1:');
  const partitionKeyValue = JSON.stringify(item[partitionKeyName]);

  const params = {
    Statement: `INSERT INTO "${tableName}" VALUE ${itemString} WHERE
 attribute_not_exists(${partitionKeyName})`,
    Parameters: [{ S: partitionKeyValue }],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item inserted with condition successfully");
    return data;
  } catch (err) {
```

```
    console.error("Error inserting item with condition:", err);
    throw err;
  }
};

/**
 * Example usage showing how to insert items with different index types
 */
export const insertExamples = async () => {
  // Example table with a simple primary key (just partition key)
  const simpleKeyItem = {
    userId: "user123",
    name: "John Doe",
    email: "john@example.com",
  };
  await insertItem("UsersTable", simpleKeyItem);

  // Example table with composite key (partition key + sort key)
  const compositeKeyItem = {
    orderId: "order456",
    productId: "prod789",
    quantity: 2,
    price: 29.99,
  };
  await insertItem("OrdersTable", compositeKeyItem);

  // Example with Global Secondary Index (GSI)
  // The GSI might be on the email attribute
  const gsiItem = {
    userId: "user789",
    email: "jane@example.com",
    name: "Jane Smith",
    userType: "premium",  // This could be part of a GSI
  };
  await insertItem("UsersTable", gsiItem);

  // Example with Local Secondary Index (LSI)
  // LSI uses the same partition key but different sort key
  const lsiItem = {
    orderId: "order567",  // Partition key
    productId: "prod123", // Sort key for the table
    orderDate: "2023-11-15", // Potential sort key for an LSI
    quantity: 1,
    price: 19.99,
```

```
  };
  await insertItem("OrdersTable", lsiItem);

  // Batch insert example with multiple items
  const batchItems = [
    {
      userId: "user234",
      name: "Alice Johnson",
      email: "alice@example.com",
    },
    {
      userId: "user345",
      name: "Bob Williams",
      email: "bob@example.com",
    },
  ];
  await batchInsertItems("UsersTable", batchItems);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

SDK für JavaScript (v3)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion zum Aktualisieren einer Amazon DynamoDB-Tabelle mit Benutzerauswahlen verwendet. Diese App verwendet v3. AWS SDK für JavaScript

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB

- Lambda

Ausführen erweiterter Abfrageoperationen

Im folgenden Codebeispiel wird gezeigt, wie Sie erweiterte Abfrageoperationen in DynamoDB ausführen.

- Abfragen von Tabellen mithilfe verschiedener Filter- und Bedingungstechniken
- Implementieren der Paginierung für große Ergebnismengen
- Verwenden globaler sekundärer Indizes für alternative Zugriffsmuster
- Anwenden von Konsistenzkontrollen auf Grundlage der Anwendungsanforderungen

SDK für JavaScript (v3)

Abfrage mit stark konsistenten Lesevorgängen mit AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
```

```
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

Abfrage unter Verwendung eines globalen sekundären Index mit AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
```

```
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}

/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
```

```
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

Abfrage mit Paginierung unter Verwendung von AWS SDK für JavaScript.

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
```

```
 try {
   // Create DynamoDB client
   const client = new DynamoDBClient(config);

   // Initialize variables for pagination
   let lastEvaluatedKey = undefined;
   const allItems = [];
   let pageCount = 0;

   // Loop until all pages are retrieved
   do {
     // Construct the query input
     const input = {
       TableName: tableName,
       KeyConditionExpression: "#pk = :pkValue",
       Limit: pageSize,
       ExpressionAttributeNames: {
         "#pk": partitionKeyName
       },
       ExpressionAttributeValues: {
         ":pkValue": { S: partitionKeyValue }
       }
     };

     // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous query
     if (lastEvaluatedKey) {
       input.ExclusiveStartKey = lastEvaluatedKey;
     }

     // Execute the query
     const command = new QueryCommand(input);
     const response = await client.send(command);

     // Process the current page of results
     pageCount++;
     console.log(`Processing page ${pageCount} with ${response.Items.length}
items`);

     // Add the items from this page to our collection
     if (response.Items && response.Items.length > 0) {
       allItems.push(...response.Items);
     }

     // Get the LastEvaluatedKey for the next page
```

```
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in ${pageCount}
 pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
```

```
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
```

```
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }

    // Add minimum views filter if provided
```

```
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Ausführen von Listenoperationen

Das folgende Codebeispiel zeigt, wie Listenoperationen für DynamoDB-Tabellen ausgeführt werden.

- Hinzufügen von Elementen zu einem Listenattribut
- Entfernen von Elementen aus einem Listenattribut
- Aktualisieren von bestimmten Elementen in einer Liste nach Index
- Verwenden von Funktionen zum Anfügen und zum Auflisten von Indizes

SDK für JavaScript (v3)

Demonstrieren Sie Listenoperationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Append elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the end of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to append to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function appendToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
```

```
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} =
 list_append(if_not_exists(${listName}, :empty_list), :values)`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Prepend elements to a list attribute.
 *
 * This function demonstrates how to use the list_append function to add elements
 * to the beginning of a list.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} values - The values to prepend to the list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function prependToList(
  config,
  tableName,
  key,
  listName,
  values
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  // Note: To prepend, we put the new values first in the list_append function
```

```
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = list_append(:values,
 if_not_exists(${listName}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": [],
      ":values": values
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update a specific element in a list by index.
 *
 * This function demonstrates how to update a specific element in a list
 * using the index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to update
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateListElement(
  config,
  tableName,
  key,
  listName,
  index,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Define the update parameters using index notation
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${index}] = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove an element from a list by index.
 *
 * This function demonstrates how to remove a specific element from a list
 * using the REMOVE action with index notation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} index - The index of the element to remove
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeListElement(
  config,
  tableName,
  key,
  listName,
  index
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using REMOVE with index notation
  const params = {
    TableName: tableName,
```

```
    Key: key,
    UpdateExpression: `REMOVE ${listName}[${index}]`,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Concatenate two lists.
 *
 * This function demonstrates how to concatenate two lists using the list_append
 function.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName1 - The name of the first list attribute
 * @param {string} listName2 - The name of the second list attribute
 * @param {string} resultListName - The name of the attribute to store the
 concatenated list
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function concatenateLists(
  config,
  tableName,
  key,
  listName1,
  listName2,
  resultListName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using list_append
  const params = {
    TableName: tableName,
    Key: key,
```

```
    UpdateExpression: `SET ${resultListName} =
 list_append(if_not_exists(${listName1}, :empty_list),
 if_not_exists(${listName2}, :empty_list))`,
    ExpressionAttributeValues: {
      ":empty_list": []
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a nested list structure.
 *
 * This function demonstrates how to create and work with nested lists.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {Array} nestedLists - An array of arrays to create a nested list structure
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createNestedList(
  config,
  tableName,
  key,
  listName,
  nestedLists
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters to create a nested list
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName} = :nested_lists`,
    ExpressionAttributeValues: {
```

```
      ":nested_lists": nestedLists
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an element in a nested list.
 *
 * This function demonstrates how to update an element in a nested list
 * using multiple index notations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} listName - The name of the list attribute
 * @param {number} outerIndex - The index in the outer list
 * @param {number} innerIndex - The index in the inner list
 * @param {any} value - The new value for the element
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedListElement(
  config,
  tableName,
  key,
  listName,
  outerIndex,
  innerIndex,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using multiple index notations
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${listName}[${outerIndex}][${innerIndex}] = :value`,
```

```
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to work with lists in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "UserProfiles";
  const key = { UserId: "U12345" };

  console.log("Demonstrating list operations in DynamoDB");

  try {
    // Example 1: Append elements to a list
    console.log("\nExample 1: Appending elements to a list");
    const response1 = await appendToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["laptop", "headphones", "monitor"]
    );

    console.log("Appended to list:", response1.Attributes);

    // Example 2: Prepend elements to a list
    console.log("\nExample 2: Prepending elements to a list");
    const response2 = await prependToList(
      config,
      tableName,
      key,
      "RecentSearches",
      ["keyboard", "mouse"]
    );

    console.log("Prepended to list:", response2.Attributes);

    // Get the current state of the item
    let currentItem = await getItem(config, tableName, key);
    console.log("\nCurrent state of RecentSearches:", currentItem?.RecentSearches);
```

```
// Example 3: Update a specific element in a list
console.log("\nExample 3: Updating a specific element in a list");
const response3 = await updateListElement(
  config,
  tableName,
  key,
  "RecentSearches",
  0, // Update the first element
  "mechanical keyboard" // New value
);

console.log("Updated list element:", response3.Attributes);

// Example 4: Remove an element from a list
console.log("\nExample 4: Removing an element from a list");
const response4 = await removeListElement(
  config,
  tableName,
  key,
  "RecentSearches",
  2 // Remove the third element
);

console.log("List after removing element:", response4.Attributes);

// Example 5: Create and concatenate lists
console.log("\nExample 5: Creating and concatenating lists");

// First, create two separate lists
await updateWithMultipleActions(
  config,
  tableName,
  key,
  "SET WishList = :wishlist, SavedItems = :saveditems",
  null,
  {
    ":wishlist": ["gaming laptop", "wireless earbuds"],
    ":saveditems": ["smartphone", "tablet"]
  }
);

// Then, concatenate them
const response5 = await concatenateLists(
```

```
    config,
    tableName,
    key,
    "WishList",
    "SavedItems",
    "AllItems"
  );

  console.log("Concatenated lists:", response5.Attributes);

  // Example 6: Create a nested list structure
  console.log("\nExample 6: Creating a nested list structure");
  const response6 = await createNestedList(
    config,
    tableName,
    key,
    "Categories",
    [
      ["Electronics", "Computers", "Accessories"],
      ["Books", "Magazines", "E-books"],
      ["Clothing", "Shoes", "Watches"]
    ]
  );

  console.log("Created nested list:", response6.Attributes);

  // Example 7: Update an element in a nested list
  console.log("\nExample 7: Updating an element in a nested list");
  const response7 = await updateNestedListElement(
    config,
    tableName,
    key,
    "Categories",
    0, // First inner list
    1, // Second element in that list
    "Laptops" // New value
  );

  console.log("Updated nested list element:", response7.Attributes);

  // Get the final state of the item
  currentItem = await getItem(config, tableName, key);
  console.log("\nFinal state of the item:", JSON.stringify(currentItem, null, 2));
```

```
    // Explain list operations
    console.log("\nKey points about list operations in DynamoDB:");
    console.log("1. Use list_append to add elements to a list");
    console.log("2. To append elements, use list_append(existingList,
 newElements)");
    console.log("3. To prepend elements, use list_append(newElements,
 existingList)");
    console.log("4. Use if_not_exists to handle cases where the list might not exist
 yet");
    console.log("5. Use index notation (list[0]) to access or update specific
 elements");
    console.log("6. Use REMOVE with index notation to remove elements from a list");
    console.log("7. Lists can contain elements of different types");
    console.log("8. Lists can be nested (lists of lists)");
    console.log("9. Use multiple index notations (list[0][1]) to access nested list
 elements");

  } catch (error) {
    console.error("Error:", error);
  }
}

/**
 * Helper function for the examples.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} updateExpression - The update expression
 * @param {Object} expressionAttributeNames - Expression attribute name placeholders
 * @param {Object} expressionAttributeValues - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
```

```
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Ausführen von Map-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Map-Operationen in DynamoDB ausgeführt werden.

- Fügen Sie verschachtelte Attribute in Map-Strukturen hinzu und aktualisieren Sie sie.
- Entfernen bestimmter Felder aus Maps
- Arbeiten mit tief verschachtelten Map-Attributen

SDK für JavaScript (v3)

Demonstrieren Sie Kartenoperationen mit AWS SDK für JavaScript.

```
/**
 * Example of updating map attributes in DynamoDB.
 *
 * This module demonstrates how to update map attributes that may not exist,
 * how to update nested attributes, and how to handle various map update scenarios.
 */

const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update a map attribute safely, handling the case where the map might not exist.
 *
 * This function demonstrates using the if_not_exists function to safely update
 * a map attribute that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeSafe(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using SET with if_not_exists
  const params = {
```

```
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return response;
  } catch (error) {
    // If the error is because the map doesn't exist, create it
    if (error.name === "ValidationException" &&
        error.message.includes("The document path provided in the update expression
 is invalid")) {

      // Create the map with the specified key-value pair
      const createParams = {
        TableName: tableName,
        Key: key,
        UpdateExpression: `SET ${mapName} = :map`,
        ExpressionAttributeValues: {
          ":map": { [mapKey]: value }
        },
        ReturnValues: "UPDATED_NEW"
      };

      return await docClient.send(new UpdateCommand(createParams));
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Update a map attribute using the if_not_exists function.
 *
 * This function demonstrates a more elegant approach using if_not_exists
 * to handle the case where the map doesn't exist yet.
 *
```

```
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {string} mapKey - The key within the map to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMapAttributeWithIfNotExists(
  config,
  tableName,
  key,
  mapName,
  mapKey,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // A single UpdateExpression can't reference both a map and a path inside that
  // same map (for example `${mapName}` and `${mapName}.${mapKey}`): DynamoDB
  // rejects overlapping document paths with a ValidationException. Instead,
  // perform the update in two steps.

  // Step 1: create the map with an empty value if it doesn't already exist.
  await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`,
    ExpressionAttributeValues: {
      ":emptyMap": {}
    }
  }));

  // Step 2: set the key within the now-guaranteed-to-exist map.
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${mapName}.${mapKey} = :value`,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
```

```
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Add a value to a deeply nested map, creating parent maps if they don't exist.
 *
 * This function demonstrates how to update a deeply nested attribute,
 * creating any parent maps that don't exist along the way.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} path - The path to the nested attribute as an array of keys
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToNestedMap(
  config,
  tableName,
  key,
  path,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET";
  const expressionAttributeValues = {};

  // For each level in the path, create a map if it doesn't exist
  for (let i = 0; i < path.length; i++) {
    const currentPath = path.slice(0, i + 1).join(".");
    const parentPath = i > 0 ? path.slice(0, i).join(".") : null;

    if (parentPath) {
      updateExpression += ` ${parentPath} = if_not_exists(${parentPath}, :emptyMap
${i}),`;
```

```
      expressionAttributeValues[`:emptyMap${i}`] = {};
    }
  }

  // Set the final value
  const fullPath = path.join(".");
  updateExpression += ` ${fullPath} = :value`;
  expressionAttributeValues[":value"] = value;

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update multiple fields in a map attribute in a single operation.
 *
 * This function demonstrates how to update multiple fields in a map
 * in a single DynamoDB operation.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} mapName - The name of the map attribute
 * @param {Object} updates - Object containing key-value pairs to update
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateMultipleMapFields(
  config,
  tableName,
  key,
  mapName,
  updates
) {
```

```
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = `SET ${mapName} = if_not_exists(${mapName}, :emptyMap)`;
  const expressionAttributeValues = {
    ":emptyMap": {}
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([field, value], index) => {
    updateExpression += `, ${mapName}.${field} = :val${index}`;
    expressionAttributeValues[`:val${index}`] = value;
  });

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
```

```
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}

/**
 * Example of how to use the map attribute update functions.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating different approaches to update map attributes in
 DynamoDB");

  try {
    // Example 1: Update a map attribute that might not exist (two-step approach)
    console.log("\nExample 1: Updating a map attribute that might not exist (two-
step approach)");
    const response1 = await updateMapAttributeSafe(
      config,
      tableName,
      key,
      "Preferences",
      "Theme",
      "Dark"
    );
```

```
    console.log("Updated preferences:", response1.Attributes);

    // Example 2: Update a map attribute using if_not_exists (elegant approach)
    console.log("\nExample 2: Updating a map attribute using if_not_exists (elegant
approach)");
    const response2 = await updateMapAttributeWithIfNotExists(
      config,
      tableName,
      key,
      "Settings",
      "NotificationsEnabled",
      true
    );

    console.log("Updated settings:", response2.Attributes);

    // Example 3: Update a deeply nested attribute
    console.log("\nExample 3: Updating a deeply nested attribute");
    const response3 = await addToNestedMap(
      config,
      tableName,
      key,
      ["Profile", "Address", "City"],
      "Seattle"
    );

    console.log("Updated nested attribute:", response3.Attributes);

    // Example 4: Update multiple fields in a map
    console.log("\nExample 4: Updating multiple fields in a map");
    const response4 = await updateMultipleMapFields(
      config,
      tableName,
      key,
      "ContactInfo",
      {
        Email: "user@example.com",
        Phone: "555-123-4567",
        PreferredContact: "Email"
      }
    );

    console.log("Updated multiple fields:", response4.Attributes);
```

```
    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Explain the benefits of different approaches
    console.log("\nKey points about updating map attributes:");
    console.log("1. Use if_not_exists to handle maps that might not exist");
    console.log("2. Multiple updates can be combined in a single operation");
    console.log("3. Deeply nested attributes require creating parent maps");
    console.log("4. DynamoDB expressions are atomic - the entire update succeeds or
 fails");
    console.log("5. Using a single operation is more efficient than multiple
 separate updates");

  } catch (error) {
    console.error("Error:", error);
  }
}

// Export the functions
module.exports = {
  updateMapAttributeSafe,
  updateMapAttributeWithIfNotExists,
  addToNestedMap,
  updateMultipleMapFields,
  getItem,
  exampleUsage
};

// Run the example if this file is executed directly
if (require.main === module) {
  exampleUsage();
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Ausführen von Set-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Set-Operationen in DynamoDB ausgeführt werden.

- Hinzufügen von Elementen zu einem Set-Attribut

- Entfernen von Elementen aus einem Set-Attribut
- Verwenden der Operationen ADD und DELETE mit Sätzen

SDK für JavaScript (v3)

Demonstrieren Sie Satzoperationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Add elements to a set attribute.
 *
 * This function demonstrates using the ADD operation to add elements to a set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to add to the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function addToSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
```

```
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Remove elements from a set attribute.
 *
 * This function demonstrates using the DELETE operation to remove elements from a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The values to remove from the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function removeFromSet(
  config,
  tableName,
  key,
```

```
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using DELETE
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `DELETE ${setName} :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Create a new set attribute with initial values.
 *
 * This function demonstrates using the SET operation to create a new set attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The initial values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function createSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create the appropriate set type
  let setValues;
  if (setType === 'string') {
    setValues = new Set(values.map(String));
  } else if (setType === 'number') {
    setValues = new Set(values.map(Number));
  } else if (setType === 'binary') {
    setValues = new Set(values);
  } else {
    throw new Error(`Unsupported set type: ${setType}`);
  }

  // Define the update parameters using SET
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${setName} = :values`,
    ExpressionAttributeValues: {
      ":values": setValues
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));
```

```
  return response;
}

/**
 * Replace an entire set attribute with a new set of values.
 *
 * This function demonstrates using the SET operation to replace an entire set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @param {Array} values - The new values for the set
 * @param {string} setType - The type of set ('string', 'number', or 'binary')
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function replaceSet(
  config,
  tableName,
  key,
  setName,
  values,
  setType = 'string'
) {
  // This is the same as createSet, but included for clarity of intent
  return await createSet(config, tableName, key, setName, values, setType);
}

/**
 * Remove the last element from a set and handle the empty set case.
 *
 * This function demonstrates what happens when you delete the last element of a
 set.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} setName - The name of the set attribute
 * @returns {Promise<Object>} - The result of the operation
 */
async function removeLastElementFromSet(
  config,
  tableName,
  key,
```

```
  setName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // First, get the current item to check the set
  const currentItem = await getItem(config, tableName, key);

  // Check if the set exists and has elements
  if (!currentItem || !currentItem[setName] || currentItem[setName].size === 0) {
    return {
      success: false,
      message: "Set doesn't exist or is already empty",
      item: currentItem
    };
  }

  // Get the set values
  const setValues = Array.from(currentItem[setName]);

  // If there's only one element left, remove the attribute entirely
  if (setValues.length === 1) {
    // Define the update parameters to remove the attribute
    const params = {
      TableName: tableName,
      Key: key,
      UpdateExpression: `REMOVE ${setName}`,
      ReturnValues: "UPDATED_NEW"
    };

    // Perform the update operation
    await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Last element removed, attribute has been deleted",
      removedValue: setValues[0]
    };
  } else {
    // Otherwise, remove just the last element
    // Create a set with just the last element
    const lastElement = setValues[setValues.length - 1];
    const setType = typeof lastElement === 'number' ? 'number' : 'string';
```

```
    // Remove the last element
    const response = await removeFromSet(
      config,
      tableName,
      key,
      setName,
      [lastElement],
      setType
    );

    return {
      success: true,
      message: "Last element removed, set still contains elements",
      removedValue: lastElement,
      remainingSet: response.Attributes[setName]
    };
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };
```

```
  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Set-Operationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to work with sets in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Users";
  const key = { UserId: "U12345" };

  console.log("Demonstrating set operations in DynamoDB");

  try {
    // Example 1: Create a string set
    console.log("\nExample 1: Creating a string set");
    const response1 = await createSet(
      config,
      tableName,
      key,
      "Interests",
      ["Reading", "Hiking", "Cooking"],
      "string"
    );

    console.log("Created set:", response1.Attributes);

    // Example 2: Add elements to a set
    console.log("\nExample 2: Adding elements to a set");
    const response2 = await addToSet(
      config,
      tableName,
      key,
      "Interests",
```

```
    ["Photography", "Travel"],
    "string"
  );

  console.log("Updated set after adding elements:", response2.Attributes);

  // Example 3: Remove elements from a set
  console.log("\nExample 3: Removing elements from a set");
  const response3 = await removeFromSet(
    config,
    tableName,
    key,
    "Interests",
    ["Cooking"],
    "string"
  );

  console.log("Updated set after removing elements:", response3.Attributes);

  // Example 4: Create a number set
  console.log("\nExample 4: Creating a number set");
  const response4 = await createSet(
    config,
    tableName,
    key,
    "FavoriteNumbers",
    [7, 42, 99],
    "number"
  );

  console.log("Created number set:", response4.Attributes);

  // Example 5: Replace an entire set
  console.log("\nExample 5: Replacing an entire set");
  const response5 = await replaceSet(
    config,
    tableName,
    key,
    "Interests",
    ["Gaming", "Movies", "Music"],
    "string"
  );

  console.log("Replaced set:", response5.Attributes);
```

```
   // Example 6: Remove the last element from a set
   console.log("\nExample 6: Removing the last element from a set");

   // First, create a set with just one element
   await createSet(
     config,
     tableName,
     { UserId: "U67890" },
     "Tags",
     ["LastTag"],
     "string"
   );

   // Then, remove the last element
   const response6 = await removeLastElementFromSet(
     config,
     tableName,
     { UserId: "U67890" },
     "Tags"
   );

   console.log(response6.message);
   console.log("Removed value:", response6.removedValue);

   // Get the final state of the items
   console.log("\nFinal state of the items:");
   const item1 = await getItem(config, tableName, key);
   console.log("User U12345:", JSON.stringify(item1, null, 2));

   const item2 = await getItem(config, tableName, { UserId: "U67890" });
   console.log("User U67890:", JSON.stringify(item2, null, 2));

   // Explain set operations
   console.log("\nKey points about set operations in DynamoDB:");
   console.log("1. Use ADD to add elements to a set (duplicates are automatically
removed)");
   console.log("2. Use DELETE to remove elements from a set");
   console.log("3. Use SET to create a new set or replace an existing one");
   console.log("4. DynamoDB supports three types of sets: string sets, number sets,
and binary sets");
   console.log("5. When you delete the last element from a set, the attribute
remains as an empty set");
   console.log("6. To remove an empty set, use the REMOVE operation");
```

```
    console.log("7. Sets automatically maintain unique values (no duplicates)");
    console.log("8. You cannot mix data types within a set");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie Batch-PartiQL-Anweisungen aus.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
```

```
  DynamoDBDocumentClient,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "Cities";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({}, { confirmAll });
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
    } else {
      throw caught;
    }
  }
```

```
  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "name",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "name", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
```

```
   * Insert items.
   */

  log("Inserting cities into the table.");
  const addItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statements: [
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["Alachua", 10712],
      },
      {
        Statement: `INSERT INTO ${tableName} value {'name':?, 'population':?}`,
        Parameters: ["High Springs", 6415],
      },
    ],
  });
  await docClient.send(addItemsStatementCommand);
  log("Cities inserted.");

  /**
   * Select items.
   */

  log("Selecting cities from the table.");
  const selectItemsStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statements: [
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `SELECT * FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
  const selectItemResponse = await docClient.send(selectItemsStatementCommand);
  log(
    `Got cities: ${selectItemResponse.Responses.map(
      (r) => `${r.Item.name} (${r.Item.population})`,
```

```
    ).join(", ")}`,
  );

  /**
   * Update items.
   */

  log("Modifying the populations.");
  const updateItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statements: [
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [10, "Alachua"],
      },
      {
        Statement: `UPDATE ${tableName} SET population=? WHERE name=?`,
        Parameters: [5, "High Springs"],
      },
    ],
  });
  await docClient.send(updateItemStatementCommand);
  log("Updated cities.");

  /**
   * Delete the items.
   */

  log("Deleting the cities.");
  const deleteItemStatementCommand = new BatchExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statements: [
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["Alachua"],
      },
      {
        Statement: `DELETE FROM ${tableName} WHERE name=?`,
        Parameters: ["High Springs"],
      },
    ],
  });
```

```
  await docClient.send(deleteItemStatementCommand);
  log("Cities deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für JavaScript API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie einzelne PartiQL-Anweisungen aus.

```
import {
  BillingMode,
  CreateTableCommand,
  DeleteTableCommand,
  DescribeTableCommand,
```

```
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario";

const client = new DynamoDBClient({});
const docClient = DynamoDBDocumentClient.from(client);

const log = (msg) => console.log(`[SCENARIO] ${msg}`);
const tableName = "SingleOriginCoffees";

export const main = async (confirmAll = false) => {
  /**
   * Delete table if it exists.
   */
  try {
    await client.send(new DescribeTableCommand({ TableName: tableName }));
    // If no error was thrown, the table exists.
    const input = new ScenarioInput(
      "deleteTable",
      `A table named ${tableName} already exists. If you choose not to delete
this table, the scenario cannot continue. Delete it?`,
      { type: "confirm", confirmAll },
    );
    const deleteTable = await input.handle({});
    if (deleteTable) {
      await client.send(new DeleteTableCommand({ tableName }));
    } else {
      console.warn(
        "Scenario could not run. Either delete ${tableName} or provide a unique
 table name.",
      );
      return;
    }
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceNotFoundException"
    ) {
      // Do nothing. This means the table is not there.
```

```
    } else {
      throw caught;
    }
  }

  /**
   * Create a table.
   */

  log("Creating a table.");
  const createTableCommand = new CreateTableCommand({
    TableName: tableName,
    // This example performs a large write to the database.
    // Set the billing mode to PAY_PER_REQUEST to
    // avoid throttling the large write.
    BillingMode: BillingMode.PAY_PER_REQUEST,
    // Define the attributes that are necessary for the key schema.
    AttributeDefinitions: [
      {
        AttributeName: "varietal",
        // 'S' is a data type descriptor that represents a number type.
        // For a list of all data type descriptors, see the following link.
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Programming.LowLevelAPI.html#Programming.LowLevelAPI.DataTypeDescriptors
        AttributeType: "S",
      },
    ],
    // The KeySchema defines the primary key. The primary key can be
    // a partition key, or a combination of a partition key and a sort key.
    // Key schema design is important. For more info, see
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-
practices.html
    KeySchema: [{ AttributeName: "varietal", KeyType: "HASH" }],
  });
  await client.send(createTableCommand);
  log(`Table created: ${tableName}.`);

  /**
   * Wait until the table is active.
   */

  // This polls with DescribeTableCommand until the requested table is 'ACTIVE'.
  // You can't write to a table before it's active.
  log("Waiting for the table to be active.");
```

```
  await waitUntilTableExists({ client }, { TableName: tableName });
  log("Table active.");

  /**
   * Insert an item.
   */

  log("Inserting a coffee into the table.");
  const addItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.insert.html
    Statement: `INSERT INTO ${tableName} value {'varietal':?, 'profile':?}`,
    Parameters: ["arabica", ["chocolate", "floral"]],
  });
  await client.send(addItemStatementCommand);
  log("Coffee inserted.");

  /**
   * Select an item.
   */

  log("Selecting the coffee from the table.");
  const selectItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.select.html
    Statement: `SELECT * FROM ${tableName} WHERE varietal=?`,
    Parameters: ["arabica"],
  });
  const selectItemResponse = await docClient.send(selectItemStatementCommand);
  log(`Got coffee: ${JSON.stringify(selectItemResponse.Items[0])}`);

  /**
   * Update the item.
   */

  log("Add a flavor profile to the coffee.");
  const updateItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.update.html
    Statement: `UPDATE ${tableName} SET profile=list_append(profile, ?) WHERE
 varietal=?`,
    Parameters: [["fruity"], "arabica"],
  });
  await client.send(updateItemStatementCommand);
```

```
  log("Updated coffee");

  /**
   * Delete the item.
   */

  log("Deleting the coffee.");
  const deleteItemStatementCommand = new ExecuteStatementCommand({
    // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ql-
reference.delete.html
    Statement: `DELETE FROM ${tableName} WHERE varietal=?`,
    Parameters: ["arabica"],
  });
  await docClient.send(deleteItemStatementCommand);
  log("Coffee deleted.");

  /**
   * Delete the table.
   */

  log("Deleting the table.");
  const deleteTableCommand = new DeleteTableCommand({ TableName: tableName });
  await client.send(deleteTableCommand);
  log("Table deleted.");
};
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für JavaScript API-Referenz.

Abfragen einer Tabelle mit einem globalen sekundären Index

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem globalen sekundären Index abgefragt wird.

- Abfragen einer DynamoDB-Tabelle mit ihrem Primärschlüssel
- Abfragen eines globalen sekundären Index (GSI) für alternative Zugriffsmuster
- Vergleichen von Tabellenabfragen und GSI-Abfragen

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit dem Primärschlüssel mit ab. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table using the primary key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} userId - The user ID to query by (partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryTable(
  config,
  tableName,
  userId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the base table
    const input = {
      TableName: tableName,
      KeyConditionExpression: "user_id = :userId",
      ExpressionAttributeValues: {
        ":userId": { S: userId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying table: ${error}`);
    throw error;
  }
}
```

Fragen Sie einen DynamoDB Global Secondary Index (GSI) mit ab. AWS SDK für JavaScript

```
/**
 * Queries a DynamoDB Global Secondary Index (GSI)
 *
```

```
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} indexName - The name of the GSI to query
 * @param {string} gameId - The game ID to query by (GSI partition key)
 * @returns {Promise<Object>} - The query response
 */
async function queryGSI(
  config,
  tableName,
  indexName,
  gameId
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input for the GSI
    const input = {
      TableName: tableName,
      IndexName: indexName,
      KeyConditionExpression: "game_id = :gameId",
      ExpressionAttributeValues: {
        ":gameId": { S: gameId }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying GSI: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mithilfe einer „begins_with“-Bedingung

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einer „begins_with“-Bedingung abgefragt wird.

- Verwenden der Funktion „begins_with“ in einem Schlüsselbedingungsausdruck
- Filtern der Elemente auf Grundlage eines Präfixmusters für den Sortierschlüssel

SDK für JavaScript (v3)

Fragen Sie mit AWS SDK für JavaScript eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung im Sortierschlüssel ab.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items where the sort key begins with a specific
 prefix
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key
 * @param {string} prefix - The prefix to match at the beginning of the sort key
 * @returns {Promise<Object>} - The query response
 */
async function queryWithBeginsWith(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  prefix
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND begins_with(#sk, :prefix)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
```

```
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":prefix": { S: prefix }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with begins_with: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit einem Datumsbereich

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mithilfe eines Datumsbereichs im Sortierschlüssel abgefragt wird.

- Abfragen von Elementen innerhalb eines bestimmten Datumsbereichs
- Verwenden von Vergleichsoperatoren für Sortierschlüssel im Datumsformat

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range on the sort key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
```

```
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
```

```
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit einem komplexen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem komplexen Filterausdruck abgefragt wird.

- Anwenden von komplexen Filterausdrücken auf Abfrageergebnisse
- Kombinieren von mehreren Bedingungen über logische Operatoren
- Filtern von Elementen auf der Grundlage von Nicht-Schlüsselattributen.

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with a complex filter expression
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number|string} minViews - Minimum number of views for filtering
 * @param {number|string} minReplies - Minimum number of replies for filtering
 * @param {string} requiredTag - Tag that must be present in the item's tags set
 * @returns {Promise<Object>} - The query response
 */
async function queryWithComplexFilter(
  config,
  tableName,
  partitionKeyName,
```

```
  partitionKeyValue,
  minViews,
  minReplies,
  requiredTag
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      FilterExpression: "views >= :minViews AND replies >= :minReplies AND
 contains(tags, :tag)",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":minViews": { N: minViews.toString() },
        ":minReplies": { N: minReplies.toString() },
        ":tag": { S: requiredTag }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with complex filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit einem dynamischen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem dynamischen Filterausdruck abgefragt wird.

- Erstellen von Filterausdrücken dynamisch zur Laufzeit
- Konstruieren von Filterbedingungen auf der Grundlage von Benutzereingaben oder Anwendungsstatus
- Hinzufügen oder Entfernen von Filterkriterien unter bestimmten Bedingungen

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

async function queryWithDynamicFilter(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  sortKeyValue,
  filterParams = {}
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize filter expression components
    let filterExpressions = [];
    const expressionAttributeValues = {
      ":pkValue": { S: partitionKeyValue },
      ":skValue": { S: sortKeyValue }
    };
    const expressionAttributeNames = {
      "#pk": partitionKeyName,
      "#sk": sortKeyName
    };

    // Add status filter if provided
    if (filterParams.status) {
      filterExpressions.push("status = :status");
      expressionAttributeValues[":status"] = { S: filterParams.status };
    }
```

```
    // Add minimum views filter if provided
    if (filterParams.minViews !== undefined) {
      filterExpressions.push("views >= :minViews");
      expressionAttributeValues[":minViews"] = { N:
 filterParams.minViews.toString() };
    }

    // Add author filter if provided
    if (filterParams.author) {
      filterExpressions.push("author = :author");
      expressionAttributeValues[":author"] = { S: filterParams.author };
    }

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue AND #sk = :skValue"
    };

    // Add filter expression if any filters were provided
    if (filterExpressions.length > 0) {
      input.FilterExpression = filterExpressions.join(" AND ");
    }

    // Add expression attribute names and values
    input.ExpressionAttributeNames = expressionAttributeNames;
    input.ExpressionAttributeValues = expressionAttributeValues;

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with dynamic filter: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit verschachtelten Attributen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit verschachtelten Attributen abgefragt wird.

- Zugreifen und Filtern nach verschachtelten Attributen in DynamoDB-Elementen
- Verwenden von Dokumentpfadausdrücken, um auf verschachtelte Elemente zu verweisen

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit verschachtelten Attributen mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table filtering on a nested attribute
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} productId - The product ID to query by (partition key)
 * @param {string} category - The category to filter by (nested attribute)
 * @returns {Promise<Object>} - The query response
 */
async function queryWithNestedAttribute(
  config,
  tableName,
  productId,
  category
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "product_id = :productId",
      FilterExpression: "details.category = :category",
      ExpressionAttributeValues: {
        ":productId": { S: productId },
        ":category": { S: category }
      }
```

```
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with nested attribute: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit Paginierung

Das folgende Codebeispiel zeigt, wie eine Tabelle mit Paginierung abgefragt wird.

- Implementieren der Paginierung für DynamoDB-Abfrageergebnisse
- Verwenden Sie den LastEvaluatedKey , um nachfolgende Seiten abzurufen.
- Steuern der Anzahl der Elemente pro Seite mit dem Parameter „Limit“

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit Paginierung unter Verwendung von ab. AWS SDK für JavaScript

```
/**
 * Example demonstrating how to handle large query result sets in DynamoDB using
 pagination
 *
 * This example shows:
 * - How to use pagination to handle large result sets
 * - How to use LastEvaluatedKey to retrieve the next page of results
 * - How to construct subsequent query requests using ExclusiveStartKey
 */
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
```

```
 * Queries a DynamoDB table with pagination to handle large result sets
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {number} pageSize - Number of items per page
 * @returns {Promise<Array>} - All items from the query
 */
async function queryWithPagination(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  pageSize = 25
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Initialize variables for pagination
    let lastEvaluatedKey = undefined;
    const allItems = [];
    let pageCount = 0;

    // Loop until all pages are retrieved
    do {
      // Construct the query input
      const input = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pkValue",
        Limit: pageSize,
        ExpressionAttributeNames: {
          "#pk": partitionKeyName
        },
        ExpressionAttributeValues: {
          ":pkValue": { S: partitionKeyValue }
        }
      };

      // Add ExclusiveStartKey if we have a LastEvaluatedKey from a previous query
      if (lastEvaluatedKey) {
        input.ExclusiveStartKey = lastEvaluatedKey;
      }
```

```
      // Execute the query
      const command = new QueryCommand(input);
      const response = await client.send(command);

      // Process the current page of results
      pageCount++;
      console.log(`Processing page ${pageCount} with ${response.Items.length}
 items`);

      // Add the items from this page to our collection
      if (response.Items && response.Items.length > 0) {
        allItems.push(...response.Items);
      }

      // Get the LastEvaluatedKey for the next page
      lastEvaluatedKey = response.LastEvaluatedKey;

    } while (lastEvaluatedKey); // Continue until there are no more pages

    console.log(`Query complete. Retrieved ${allItems.length} items in ${pageCount}
 pages.`);
    return allItems;
  } catch (error) {
    console.error(`Error querying with pagination: ${error}`);
    throw error;
  }
}

/**
 * Example usage:
 *
 * // Query all items in the "AWS DynamoDB" forum with pagination
 * const allItems = await queryWithPagination(
 *   { region: "us-west-2" },
 *   "ForumThreads",
 *   "ForumName",
 *   "AWS DynamoDB",
 *   25 // 25 items per page
 * );
 *
 * console.log(`Total items retrieved: ${allItems.length}`);
 *
 * // Notes on pagination:
```

```
 * // - LastEvaluatedKey contains the primary key of the last evaluated item
 * // - When LastEvaluatedKey is undefined/null, there are no more items to retrieve
 * // - ExclusiveStartKey tells DynamoDB where to start the next page
 * // - Pagination helps manage memory usage for large result sets
 * // - Each page requires a separate network request to DynamoDB
 */

module.exports = { queryWithPagination };
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen einer Tabelle mit strikt konsistenten Lesevorgängen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit strikt konsistenten Lesevorgängen abgefragt wird.

- Konfigurieren des Konsistenzniveaus für DynamoDB-Abfragen
- Verwenden der strikt konsistenten Lesevorgänge, um die aktuellen Daten zu erhalten
- Erkennen der Kompromisse zwischen letztendlicher Konsistenz und starker Konsistenz

SDK für JavaScript (v3)

Fragen Sie eine DynamoDB-Tabelle mit konfigurierbarer Lesekonsistenz ab mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table with configurable read consistency
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {boolean} useConsistentRead - Whether to use strongly consistent reads
 * @returns {Promise<Object>} - The query response
 */
async function queryWithConsistentRead(
  config,
```

```
  tableName,
  partitionKeyName,
  partitionKeyValue,
  useConsistentRead = false
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: "#pk = :pkValue",
      ExpressionAttributeNames: {
        "#pk": partitionKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue }
      },
      ConsistentRead: useConsistentRead
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying with consistent read: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen von Daten mit PartiQL SELECT

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-SELECT-Anweisungen abfragen.

SDK für JavaScript (v3)

Fragen Sie Elemente aus einer DynamoDB-Tabelle mithilfe von PartiQL SELECT-Anweisungen mit ab. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to query items from a DynamoDB table using PartiQL.
 * It shows different ways to select data with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Select all items from a DynamoDB table using PartiQL.
 * Note: This should be used with caution on large tables.
 *
 * @param tableName - The name of the DynamoDB table
 * @returns The response from the ExecuteStatementCommand
 */
export const selectAllItems = async (tableName: string) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}"`,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select an item by its primary key using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
```

```
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByPartitionKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ?`,
    Parameters: [partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select an item by its composite key (partition key + sort key) using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemByCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${partitionKeyName} = ? AND
 ${sortKeyName} = ?`,
    Parameters: [partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving item:", err);
    throw err;
  }
};

/**
 * Select items using a filter condition with PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param filterAttribute - The attribute to filter on
 * @param filterValue - The value to filter by
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsWithFilter = async (
  tableName: string,
  filterAttribute: string,
  filterValue: string | number
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${filterAttribute} = ?`,
    Parameters: [filterValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
```

```
    throw err;
  }
};

/**
 * Select items using a begins_with function for prefix matching.
 * This is useful for querying hierarchical data.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for prefix
 * @param prefix - The prefix to match
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByPrefix = async (
  tableName: string,
  attributeName: string,
  prefix: string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE
 begins_with(${attributeName}, ?)`,
    Parameters: [prefix],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Select items using a between condition for range queries.
 *
 * @param tableName - The name of the DynamoDB table
 * @param attributeName - The attribute to check for range
 * @param startValue - The start value of the range
 * @param endValue - The end value of the range
```

```
 * @returns The response from the ExecuteStatementCommand
 */
export const selectItemsByRange = async (
  tableName: string,
  attributeName: string,
  startValue: number | string,
  endValue: number | string
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `SELECT * FROM "${tableName}" WHERE ${attributeName} BETWEEN ? AND ?
`,
    Parameters: [startValue, endValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Items retrieved successfully");
    return data;
  } catch (err) {
    console.error("Error retrieving items:", err);
    throw err;
  }
};

/**
 * Example usage showing how to select items with different index types
 */
export const selectExamples = async () => {
  // Select all items from a table (use with caution on large tables)
  await selectAllItems("UsersTable");

  // Select by partition key (simple primary key)
  await selectItemByPartitionKey("UsersTable", "userId", "user123");

  // Select by composite key (partition key + sort key)
  await selectItemByCompositeKey("OrdersTable", "orderId", "order456", "productId",
 "prod789");

  // Select with a filter condition (can use any attribute)
  await selectItemsWithFilter("UsersTable", "userType", "premium");
```

```
  // Select items with a prefix (useful for hierarchical data)
  await selectItemsByPrefix("ProductsTable", "category", "electronics");

  // Select items within a range (useful for numeric or date ranges)
  await selectItemsByRange("OrdersTable", "orderDate", "2023-01-01", "2023-12-31");
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Abfragen nach TTL-Elementen

Im folgenden Codebeispiel wird gezeigt, wie TTL-Elemente abgefragt werden.

SDK für JavaScript (v3)

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK für JavaScript

```
import { DynamoDBClient, QueryCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const queryFiltered = async (tableName, primaryKey, region = 'us-east-1') =>
 {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);

    const params = {
        TableName: tableName,
        KeyConditionExpression: "#pk = :pk",
        FilterExpression: "#ea > :ea",
        ExpressionAttributeNames: {
            "#pk": "primaryKey",
            "#ea": "expireAt"
        },
```

```
        ExpressionAttributeValues: marshall({
            ":pk": primaryKey,
            ":ea": currentTime
        })
    };

    try {
        const { Items } = await client.send(new QueryCommand(params));
        Items.forEach(item => {
            console.log(unmarshall(item))
        });
        return Items;
    } catch (err) {
        console.error(`Error querying items: ${err}`);
        throw err;
    }
}

// Example usage (commented out for testing)
// queryFiltered('your-table-name', 'your-partition-key-value');
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Abfragen von Tabellen mithilfe von Datums- und Uhrzeitmustern

Im folgenden Codebeispiel wird gezeigt, wie Datums- und Uhrzeitmuster zum Abfragen von Tabellen verwendet werden.

- Speichern und Abfragen von date/time Werten in DynamoDB.
- Implementieren von Datumsbereichsabfragen mithilfe von Sortierschlüsseln
- Formatieren von Datumszeichenfolgen für effektive Abfragen

SDK für JavaScript (v3)

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit AWS SDK für JavaScript.

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");
```

```
/**
 * Queries a DynamoDB table for items within a specific date range on the sort key
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} sortKeyName - The name of the sort key (must be a date/time
 attribute)
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRangeOnSortKey(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  sortKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: '#pk = :pkValue AND #sk BETWEEN :startDate
 AND :endDate',
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#sk": sortKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
```

```
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range on sort key: ${error}`);
    throw error;
  }
}
```

Abfrage mit Datums-/Uhrzeitvariablen mit. AWS SDK für JavaScript

```
const { DynamoDBClient, QueryCommand } = require("@aws-sdk/client-dynamodb");

/**
 * Queries a DynamoDB table for items within a specific date range
 *
 * @param {Object} config - AWS SDK configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key
 * @param {string} partitionKeyValue - The value of the partition key
 * @param {string} dateKeyName - The name of the date attribute to filter on
 * @param {Date} startDate - The start date for the range query
 * @param {Date} endDate - The end date for the range query
 * @returns {Promise<Object>} - The query response
 */
async function queryByDateRange(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  dateKeyName,
  startDate,
  endDate
) {
  try {
    // Create DynamoDB client
    const client = new DynamoDBClient(config);

    // Format dates as ISO strings for DynamoDB
    const formattedStartDate = startDate.toISOString();
```

```
    const formattedEndDate = endDate.toISOString();

    // Construct the query input
    const input = {
      TableName: tableName,
      KeyConditionExpression: `#pk = :pkValue AND #dateAttr BETWEEN :startDate
 AND :endDate`,
      ExpressionAttributeNames: {
        "#pk": partitionKeyName,
        "#dateAttr": dateKeyName
      },
      ExpressionAttributeValues: {
        ":pkValue": { S: partitionKeyValue },
        ":startDate": { S: formattedStartDate },
        ":endDate": { S: formattedEndDate }
      }
    };

    // Execute the query
    const command = new QueryCommand(input);
    return await client.send(command);
  } catch (error) {
    console.error(`Error querying by date range: ${error}`);
    throw error;
  }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für JavaScript -API-Referenz.

Verstehen der Reihenfolge der Aktualisierungsausdrücke

Im folgenden Codebeispiel wird die Reihenfolge der Aktualisierungsausdrücke erklärt.

- Erfahren, wie DynamoDB Aktualisierungsausdrücke verarbeitet
- Verstehen der Reihenfolge der Operationen in Aktualisierungsausdrücken
- Vermeiden von unerwarteten Ergebnissen durch Verständnis der Ausdrucksauswertung

SDK für JavaScript (v3)

Demonstrieren Sie die Aktualisierungsreihenfolge mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an item with multiple actions in a single update expression.
 *
 * This function demonstrates how to use multiple actions in a single update
 expression
 * and how DynamoDB processes these actions.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} updateExpression - The update expression with multiple actions
 * @param {Object} [expressionAttributeNames] - Expression attribute name
 placeholders
 * @param {Object} [expressionAttributeValues] - Expression attribute value
 placeholders
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithMultipleActions(
  config,
  tableName,
  key,
  updateExpression,
  expressionAttributeNames,
  expressionAttributeValues
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Prepare the update parameters
  const updateParams = {
    TableName: tableName,
```

```
    Key: key,
    UpdateExpression: updateExpression,
    ReturnValues: "UPDATED_NEW"
  };

  // Add expression attribute names if provided
  if (expressionAttributeNames) {
    updateParams.ExpressionAttributeNames = expressionAttributeNames;
  }

  // Add expression attribute values if provided
  if (expressionAttributeValues) {
    updateParams.ExpressionAttributeValues = expressionAttributeValues;
  }

  // Execute the update
  const response = await docClient.send(new UpdateCommand(updateParams));

  return response;
}

/**
 * Demonstrate that variables hold copies of existing values before modifications.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that uses the values of attributes before they are modified in the same
 expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateValueCopying(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Step 1: Create an item with initial values
  const initialItem = { ...key, a: 1, b: 2, c: 3 };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with an expression that uses values before they are
 modified
  // This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to the
 value of 'b'
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE a SET b = a, c = b",
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
```

```
 * Demonstrate the order in which different action types are processed.
 *
 * This function creates an item with initial values, then updates it with an
 expression
 * that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the order
 * in which they are processed.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateActionOrder(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    set_attr: new Set(["A", "B", "C"]),
    to_remove: "This will be removed",
    to_modify: "Original value"
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};
```

```
  // Step 3: Update the item with multiple action types
  // The actions will be processed in this order: REMOVE, SET, ADD, DELETE
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: "REMOVE to_remove SET to_modify = :new_value ADD
 counter :increment DELETE set_attr :elements",
    ExpressionAttributeValues: {
      ":new_value": "Updated value",
      ":increment": 5,
      ":elements": new Set(["B"])
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}

/**
 * Update multiple attributes with a single SET action.
 *
 * This function demonstrates how to update multiple attributes in a single SET
 action,
 * which is more efficient than using multiple separate update operations.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {Object} attributes - The attributes to update and their new values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
```

```
async function updateWithMultipleSetActions(
  config,
  tableName,
  key,
  attributes
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression and expression attribute values
  let updateExpression = "SET ";
  const expressionAttributeValues = {};

  // Add each attribute to the update expression
  Object.entries(attributes).forEach(([attrName, attrValue], index) => {
    const valuePlaceholder = `:val${index}`;

    if (index > 0) {
      updateExpression += ", ";
    }
    updateExpression += `${attrName} = ${valuePlaceholder}`;

    expressionAttributeValues[valuePlaceholder] = attrValue;
  });

  // Execute the update
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: updateExpression,
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Update an attribute with a value from another attribute or a default value.
 *
 * This function demonstrates how to use if_not_exists to conditionally copy a value
 * from one attribute to another, or use a default value if the source doesn't
 exist.
```

```
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to update
 * @param {string} sourceAttribute - The attribute to copy the value from
 * @param {string} targetAttribute - The attribute to update
 * @param {any} defaultValue - The default value to use if the source attribute
 doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateWithConditionalValueCopying(
  config,
  tableName,
  key,
  sourceAttribute,
  targetAttribute,
  defaultValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Use if_not_exists to conditionally copy the value
  const response = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${targetAttribute} =
 if_not_exists(${sourceAttribute}, :default)`,
    ExpressionAttributeValues: {
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  }));

  return response;
}

/**
 * Demonstrate complex update expressions with multiple operations on the same
 attribute.
 *
 * This function shows how DynamoDB processes multiple operations on the same
 attribute
 * in a single update expression.
```

```
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The primary key of the item to create and update
 * @returns {Promise<Object>} - A dictionary containing the results of the
 demonstration
 */
async function demonstrateMultipleOperationsOnSameAttribute(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Step 1: Create an item with initial values
  const initialItem = {
    ...key,
    counter: 10,
    list_attr: [1, 2, 3],
    map_attr: {
      nested1: "value1",
      nested2: "value2"
    }
  };

  await docClient.send(new PutCommand({
    TableName: tableName,
    Item: initialItem
  }));

  // Step 2: Get the item to verify initial state
  const responseBefore = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemBefore = responseBefore.Item || {};

  // Step 3: Update the item with multiple operations on the same attributes
  const updateResponse = await docClient.send(new UpdateCommand({
    TableName: tableName,
    Key: key,
```

```
    UpdateExpression: `
      SET counter = counter + :inc1,
          counter = counter + :inc2,
          map_attr.nested1 = :new_val1,
          map_attr.nested3 = :new_val3,
          list_attr[0] = list_attr[1],
          list_attr[1] = list_attr[2]
    `,
    ExpressionAttributeValues: {
      ":inc1": 5,
      ":inc2": 3,
      ":new_val1": "updated_value1",
      ":new_val3": "new_value3"
    },
    ReturnValues: "UPDATED_NEW"
  }));

  // Step 4: Get the item to verify final state
  const responseAfter = await docClient.send(new GetCommand({
    TableName: tableName,
    Key: key
  }));

  const itemAfter = responseAfter.Item || {};

  // Return the results
  return {
    initialState: itemBefore,
    updateResponse: updateResponse,
    finalState: itemAfter
  };
}
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK für JavaScript.

```
/**
 * Example of how to use update expression order of operations in DynamoDB.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "OrderProcessing";
```

```
 console.log("Demonstrating update expression order of operations in DynamoDB");

 try {
   // Example 1: Demonstrating value copying in update expressions
   console.log("\nExample 1: Demonstrating value copying in update expressions");
   const results1 = await demonstrateValueCopying(
     config,
     tableName,
     { OrderId: "order123" }
   );

   console.log("Initial state:", JSON.stringify(results1.initialState, null, 2));
   console.log("Update response:", JSON.stringify(results1.updateResponse, null,
2));
   console.log("Final state:", JSON.stringify(results1.finalState, null, 2));

   console.log("\nExplanation:");
   console.log("1. The initial state had a=1, b=2, c=3");
   console.log("2. The update expression 'REMOVE a SET b = a, c = b' did the
following:");
   console.log("   - Copied the value of 'a' (which was 1) to be used for 'b'");
   console.log("   - Copied the value of 'b' (which was 2) to be used for 'c'");
   console.log("   - Removed the attribute 'a'");
   console.log("3. The final state has b=1, c=2, and 'a' is removed");
   console.log("4. This demonstrates that DynamoDB uses the values of attributes as
they were BEFORE any modifications");

   // Example 2: Demonstrating the order of different action types
   console.log("\nExample 2: Demonstrating the order of different action types");
   const results2 = await demonstrateActionOrder(
     config,
     tableName,
     { OrderId: "order456" }
   );

   console.log("Initial state:", JSON.stringify(results2.initialState, null, 2));
   console.log("Update response:", JSON.stringify(results2.updateResponse, null,
2));
   console.log("Final state:", JSON.stringify(results2.finalState, null, 2));

   console.log("\nExplanation:");
   console.log("1. The update expression contained multiple action types: REMOVE,
SET, ADD, DELETE");
```

```
   console.log("2. DynamoDB processes these actions in this order: REMOVE, SET,
ADD, DELETE");
   console.log("3. First, 'to_remove' was removed");
   console.log("4. Then, 'to_modify' was set to a new value");
   console.log("5. Next, 'counter' was incremented by 5");
   console.log("6. Finally, 'B' was removed from the set attribute");

   // Example 3: Updating multiple attributes in a single SET action
   console.log("\nExample 3: Updating multiple attributes in a single SET action");
   const response3 = await updateWithMultipleSetActions(
     config,
     tableName,
     { OrderId: "order789" },
     {
       Status: "Shipped",
       ShippingDate: "2025-05-28",
       TrackingNumber: "1Z999AA10123456784"
     }
   );

   console.log("Multiple attributes updated successfully:",
JSON.stringify(response3.Attributes, null, 2));

   // Example 4: Conditional value copying with if_not_exists
   console.log("\nExample 4: Conditional value copying with if_not_exists");
   const response4 = await updateWithConditionalValueCopying(
     config,
     tableName,
     { OrderId: "order101" },
     "PreferredShippingMethod",
     "ShippingMethod",
     "Standard"
   );

   console.log("Conditional value copying result:",
JSON.stringify(response4.Attributes, null, 2));

   // Example 5: Multiple operations on the same attribute
   console.log("\nExample 5: Multiple operations on the same attribute");
   const results5 = await demonstrateMultipleOperationsOnSameAttribute(
     config,
     tableName,
     { OrderId: "order202" }
   );
```

```
    console.log("Initial state:", JSON.stringify(results5.initialState, null, 2));
    console.log("Update response:", JSON.stringify(results5.updateResponse, null,
 2));
    console.log("Final state:", JSON.stringify(results5.finalState, null, 2));

    console.log("\nExplanation:");
    console.log("1. The counter was incremented twice (first by 5, then by 3) for a
 total of +8");
    console.log("2. The map attribute had one value updated and a new nested
 attribute added");
    console.log("3. The list attribute had values shifted (value at index 1 moved to
 index 0, value at index 2 moved to index 1)");
    console.log("4. All operations within the SET action are processed from left to
 right");

    // Key points about update expression order of operations
    console.log("\nKey Points About Update Expression Order of Operations:");
    console.log("1. Variables in expressions hold copies of attribute values as they
 existed BEFORE any modifications");
    console.log("2. Multiple actions in an update expression are processed in this
 order: REMOVE, SET, ADD, DELETE");
    console.log("3. Within each action type, operations are processed from left to
 right");
    console.log("4. You can reference the same attribute multiple times in an
 expression");
    console.log("5. You can use if_not_exists() to conditionally set values based on
 attribute existence");
    console.log("6. Using a single update expression with multiple actions is more
 efficient than multiple separate updates");
    console.log("7. The update expression is atomic - either all actions succeed or
 none do");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Aktualisieren der Einstellung für den Warmdurchsatz einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie die Warmdurchsatzeinstellung einer Tabelle aktualisiert wird.

SDK für JavaScript (v3)

Aktualisieren Sie mit AWS SDK für JavaScript die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
import { DynamoDBClient, UpdateTableCommand } from "@aws-sdk/client-dynamodb";

export async function updateDynamoDBTableWarmThroughput(
  tableName,
  tableReadUnits,
  tableWriteUnits,
  gsiName,
  gsiReadUnits,
  gsiWriteUnits,
  region = "us-east-1"
) {
  try {
    const ddbClient = new DynamoDBClient({ region: region });

    // Construct the update table request
    const updateTableRequest = {
      TableName: tableName,
      GlobalSecondaryIndexUpdates: [
        {
            Update: {
                IndexName: gsiName,
                WarmThroughput: {
                    ReadUnitsPerSecond: gsiReadUnits,
                    WriteUnitsPerSecond: gsiWriteUnits,
                },
            },
        },
      ],
      WarmThroughput: {
          ReadUnitsPerSecond: tableReadUnits,
          WriteUnitsPerSecond: tableWriteUnits,
      },
    };
```

```
    const command = new UpdateTableCommand(updateTableRequest);
    const response = await ddbClient.send(command);
    console.log(`Table updated successfully! Response:
 ${JSON.stringify(response)}`);
    return response;
  } catch (error) {
    console.error(`Error updating table: ${error}`);
    throw error;
  }
}

// Example usage (commented out for testing)
/*
updateDynamoDBTableWarmThroughput(
  'example-table',
  5, 5,
  'example-index',
  2, 2
);
*/
```

- API-Details finden Sie UpdateTable in der AWS SDK für JavaScript API-Referenz.

Aktualisieren der TTL eines Elements

Das folgende Codebeispiel zeigt, wie Sie die TTL eines Elements aktualisieren.

SDK für JavaScript (v3)

```
import { DynamoDBClient, UpdateItemCommand } from "@aws-sdk/client-dynamodb";
import { marshall, unmarshall } from "@aws-sdk/util-dynamodb";

export const updateItem = async (tableName, partitionKey, sortKey, region = 'us-
east-1') => {
    const client = new DynamoDBClient({
        region: region,
        endpoint: `https://dynamodb.${region}.amazonaws.com`
    });

    const currentTime = Math.floor(Date.now() / 1000);
```

```
    const expireAt = Math.floor((Date.now() + 90 * 24 * 60 * 60 * 1000) / 1000);

    const params = {
        TableName: tableName,
        Key: marshall({
            partitionKey: partitionKey,
            sortKey: sortKey
        }),
        UpdateExpression: "SET updatedAt = :c, expireAt = :e",
        ExpressionAttributeValues: marshall({
            ":c": currentTime,
            ":e": expireAt
        }),
    };

    try {
        const data = await client.send(new UpdateItemCommand(params));
        const responseData = unmarshall(data.Attributes);
        console.log("Item updated successfully: %s", responseData);
        return responseData;
    } catch (err) {
        console.error("Error updating item:", err);
        throw err;
    }
}

// Example usage (commented out for testing)
// updateItem('your-table-name', 'your-partition-key-value', 'your-sort-key-value');
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Aktualisieren von Daten mit PartiQL UPDATE

Im folgenden Codebeispiel wird gezeigt, wie Sie Daten mit PartiQL-UPDATE-Anweisungen aktualisieren.

SDK für JavaScript (v3)

Aktualisieren Sie Elemente in einer DynamoDB-Tabelle mithilfe von PartiQL UPDATE-Anweisungen mit. AWS SDK für JavaScript

```
/**
 * This example demonstrates how to update items in a DynamoDB table using PartiQL.
 * It shows different ways to update documents with various index types.
 */
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ExecuteStatementCommand,
  BatchExecuteStatementCommand,
} from "@aws-sdk/lib-dynamodb";

/**
 * Update a single attribute of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateSingleAttribute = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
```

```
    console.error("Error updating item:", err);
    throw err;
  }
};

/**
 * Update multiple attributes of an item using PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeUpdates - Object containing attribute names and their new values
 * @returns The response from the ExecuteStatementCommand
 */
export const updateMultipleAttributes = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeUpdates: Record<string, any>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create SET clause for each attribute
  const setClause = Object.keys(attributeUpdates)
    .map((attr, index) => `${attr} = ?`)
    .join(", ");

  // Create parameters array with attribute values followed by the partition key
 value
  const parameters = [...Object.values(attributeUpdates), partitionKeyValue];

  const params = {
    Statement: `UPDATE "${tableName}" SET ${setClause} WHERE ${partitionKeyName} = ?
`,
    Parameters: parameters,
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
```

```
    throw err;
  }
};

/**
 * Update an item identified by a composite key (partition key + sort key) using
 PartiQL.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param sortKeyName - The name of the sort key attribute
 * @param sortKeyValue - The value of the sort key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCompositeKey = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  sortKeyName: string,
  sortKeyValue: string | number,
  attributeName: string,
  attributeValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${sortKeyName} = ?`,
    Parameters: [attributeValue, partitionKeyValue, sortKeyValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated successfully");
    return data;
  } catch (err) {
    console.error("Error updating item:", err);
    throw err;
  }
};
```

```
/**
 * Update an item with a condition to ensure the update only happens if a condition
 is met.
 *
 * @param tableName - The name of the DynamoDB table
 * @param partitionKeyName - The name of the partition key attribute
 * @param partitionKeyValue - The value of the partition key
 * @param attributeName - The name of the attribute to update
 * @param attributeValue - The new value for the attribute
 * @param conditionAttribute - The attribute to check in the condition
 * @param conditionValue - The value to compare against in the condition
 * @returns The response from the ExecuteStatementCommand
 */
export const updateItemWithCondition = async (
  tableName: string,
  partitionKeyName: string,
  partitionKeyValue: string | number,
  attributeName: string,
  attributeValue: any,
  conditionAttribute: string,
  conditionValue: any
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  const params = {
    Statement: `UPDATE "${tableName}" SET ${attributeName} = ? WHERE
 ${partitionKeyName} = ? AND ${conditionAttribute} = ?`,
    Parameters: [attributeValue, partitionKeyValue, conditionValue],
  };

  try {
    const data = await docClient.send(new ExecuteStatementCommand(params));
    console.log("Item updated with condition successfully");
    return data;
  } catch (err) {
    console.error("Error updating item with condition:", err);
    throw err;
  }
};

/**
 * Batch update multiple items using PartiQL.
```

```
 *
 * @param tableName - The name of the DynamoDB table
 * @param updates - Array of objects containing key and update information
 * @returns The response from the BatchExecuteStatementCommand
 */
export const batchUpdateItems = async (
  tableName: string,
  updates: Array<{
    partitionKeyName: string;
    partitionKeyValue: string | number;
    attributeName: string;
    attributeValue: any;
  }>
) => {
  const client = new DynamoDBClient({});
  const docClient = DynamoDBDocumentClient.from(client);

  // Create statements for each update
  const statements = updates.map((update) => {
    return {
      Statement: `UPDATE "${tableName}" SET ${update.attributeName} = ? WHERE
 ${update.partitionKeyName} = ?`,
      Parameters: [update.attributeValue, update.partitionKeyValue],
    };
  });

  const params = {
    Statements: statements,
  };

  try {
    const data = await docClient.send(new BatchExecuteStatementCommand(params));
    console.log("Items batch updated successfully");
    return data;
  } catch (err) {
    console.error("Error batch updating items:", err);
    throw err;
  }
};

/**
 * Example usage showing how to update items with different index types
 */
export const updateExamples = async () => {
```

```
  // Update a single attribute using a simple primary key
  await updateSingleAttribute("UsersTable", "userId", "user123", "email",
"newemail@example.com");

  // Update multiple attributes at once
  await updateMultipleAttributes("UsersTable", "userId", "user123", {
    email: "newemail@example.com",
    name: "John Smith",
    lastLogin: new Date().toISOString(),
  });

  // Update an item with a composite key (partition key + sort key)
  await updateItemWithCompositeKey(
    "OrdersTable",
    "orderId",
    "order456",
    "productId",
    "prod789",
    "quantity",
    5
  );

  // Update with a condition
  await updateItemWithCondition(
    "UsersTable",
    "userId",
    "user123",
    "userStatus",
    "active",
    "userType",
    "premium"
  );

  // Batch update multiple items
  await batchUpdateItems("UsersTable", [
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user123",
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
    {
      partitionKeyName: "userId",
      partitionKeyValue: "user456",
```

```
      attributeName: "lastLogin",
      attributeValue: new Date().toISOString(),
    },
  ]);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchExecuteStatement
  - ExecuteStatement

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Operationen mit unteilbaren Zählern

Im folgenden Codebeispiel wird gezeigt, wie Sie Operationen mit unteilbaren Zählern in DynamoDB verwenden.

- Inkrementieren von unteilbaren Zählern mithilfe von ADD- und SET-Operationen
- Sicheres Erhöhen von Zählern, die möglicherweise nicht existieren
- Implementieren einer optimistischen Sperre für Zähleroperationen

SDK für JavaScript (v3)

Demonstrieren Sie atomare Gegenoperationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Increment a counter using the ADD operation.
 *
 * This function demonstrates using the ADD operation for atomic increments.
 * The ADD operation is atomic and is the recommended way to increment counters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithAdd(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
```

```
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using ADD
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `ADD ${counterName} :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter using the SET operation with an expression.
 *
 * This function demonstrates using the SET operation with an expression for
 increments.
 * While this approach works, it's less idiomatic for simple increments than using
 ADD.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithSet(
  config,
  tableName,
  key,
  counterName,
  incrementValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Define the update parameters using SET with an expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter safely, handling the case where the counter might not exist.
 *
 * This function demonstrates using the if_not_exists function with SET to safely
 * increment a counter that might not exist yet.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} defaultValue - The default value if the counter doesn't exist
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterSafely(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  defaultValue = 0
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);
```

```
  // Define the update parameters using SET with if_not_exists
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = if_not_exists(${counterName}, :default)
 + :increment`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":default": defaultValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Increment a counter with optimistic locking to prevent race conditions.
 *
 * This function demonstrates using a condition expression to implement optimistic
 * locking, which prevents race conditions when multiple processes try to update
 * the same counter.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} counterName - The name of the counter attribute
 * @param {number} incrementValue - The value to increment by
 * @param {number} expectedValue - The expected current value of the counter
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function incrementCounterWithLocking(
  config,
  tableName,
  key,
  counterName,
  incrementValue,
  expectedValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
```

```
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${counterName} = ${counterName} + :increment`,
    ConditionExpression: `${counterName} = :expected`,
    ExpressionAttributeValues: {
      ":increment": incrementValue,
      ":expected": expectedValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));
    return {
      success: true,
      data: response
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        error: "Optimistic locking failed: the counter value has changed"
      };
    }
    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of a counter.
 *
 * Helper function to retrieve the current value of a counter attribute.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @param {string} counterName - The name of the counter attribute
```

```
 * @returns {Promise<number|null>} - The current counter value or null if not found
 */
async function getCounterValue(
  config,
  tableName,
  key,
  counterName
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the counter value if it exists, otherwise null
  return response.Item && counterName in response.Item
    ? response.Item[counterName]
    : null;
}
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use the atomic counter operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };
  const counterName = "ViewCount";
  const incrementValue = 1;

  console.log("Demonstrating different approaches to increment counters in
 DynamoDB");
```

```
 try {
   // Example 1: Using ADD operation (recommended for simple increments)
   console.log("\nExample 1: Incrementing counter with ADD operation");
   const response1 = await incrementCounterWithAdd(
     config,
     tableName,
     key,
     counterName,
     incrementValue
   );

   console.log(`Counter incremented to: ${response1.Attributes[counterName]}`);

   // Example 2: Using SET operation with an expression
   console.log("\nExample 2: Incrementing counter with SET operation");
   const response2 = await incrementCounterWithSet(
     config,
     tableName,
     key,
     counterName,
     incrementValue
   );

   console.log(`Counter incremented to: ${response2.Attributes[counterName]}`);

   // Example 3: Safely incrementing a counter that might not exist
   console.log("\nExample 3: Safely incrementing counter that might not exist");
   const newKey = { ProductId: "P67890" };
   const response3 = await incrementCounterSafely(
     config,
     tableName,
     newKey,
     counterName,
     incrementValue,
     0
   );

   console.log(`Counter initialized and incremented to:
${response3.Attributes[counterName]}`);

   // Example 4: Incrementing with optimistic locking
   console.log("\nExample 4: Incrementing with optimistic locking");
```

```
    // First, get the current counter value
    const currentValue = await getCounterValue(config, tableName, key, counterName);
    console.log(`Current counter value: ${currentValue}`);

    // Then, try to increment with optimistic locking
    const response4 = await incrementCounterWithLocking(
      config,
      tableName,
      key,
      counterName,
      incrementValue,
      currentValue
    );

    if (response4.success) {
      console.log(`Counter successfully incremented to:
 ${response4.data.Attributes[counterName]}`);
    } else {
      console.log(response4.error);
    }

    // Explain the differences between ADD and SET
    console.log("\nKey differences between ADD and SET for counter operations:");
    console.log("1. ADD is more concise and idiomatic for simple increments");
    console.log("2. SET with expressions is more flexible for complex operations");
    console.log("3. Both operations are atomic and safe for concurrent updates");
    console.log("4. SET with if_not_exists is required when the attribute might not
 exist");
    console.log("5. Optimistic locking can be added to either approach for
 additional safety");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für JavaScript API-Referenz.

Verwenden von bedingten Operationen

Im folgenden Codebeispiel wird gezeigt, wie bedingte Operationen in DynamoDB verwendet werden.

- Implementieren von bedingten Schreibvorgängen, um das Überschreiben von Daten zu verhindern
- Verwenden von Bedingungsausdrücken, um Geschäftsregeln durchzusetzen
- Ordnungsgemäßer Umgang mit Fehlern bei bedingten Prüfungen

SDK für JavaScript (v3)

Demonstrieren Sie bedingte Operationen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  DeleteCommand,
  GetCommand,
  PutCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Perform a conditional update operation.
 *
 * This function demonstrates how to update an item only if a condition is met.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
 * @param {string} updateAttribute - The attribute to update
 * @param {any} updateValue - The new value to set
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalUpdate(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue,
  updateAttribute,
  updateValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
```

```
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateAttribute} = :value`,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":value": updateValue,
      ":condition": conditionValue
    },
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Condition was met and update was performed",
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, update was not performed",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Perform a conditional delete operation.
 *
 * This function demonstrates how to delete an item only if a condition is met.
 *
```

```
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to delete
 * @param {string} conditionAttribute - The attribute to check in the condition
 * @param {any} conditionValue - The value to compare against
 * @returns {Promise<Object>} - Result of the operation
 */
async function conditionalDelete(
  config,
  tableName,
  key,
  conditionAttribute,
  conditionValue
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the delete parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    ConditionExpression: `${conditionAttribute} = :condition`,
    ExpressionAttributeValues: {
      ":condition": conditionValue
    },
    ReturnValues: "ALL_OLD"
  };

  try {
    // Perform the delete operation
    const response = await docClient.send(new DeleteCommand(params));

    return {
      success: true,
      message: "Condition was met and item was deleted",
      deletedItem: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Condition was not met, item was not deleted",
```

```
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement optimistic locking with a version number.
 *
 * This function demonstrates how to use a version number for optimistic locking
 * to prevent race conditions when multiple processes update the same item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {Object} updates - The attributes to update
 * @param {number} expectedVersion - The expected current version number
 * @returns {Promise<Object>} - Result of the operation
 */
async function updateWithOptimisticLocking(
  config,
  tableName,
  key,
  updates,
  expectedVersion
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Build the update expression
  const updateExpressions = [];
  const expressionAttributeValues = {
    ":expectedVersion": expectedVersion,
    ":newVersion": expectedVersion + 1
  };

  // Add each update to the expression
  Object.entries(updates).forEach(([attribute, value], index) => {
    updateExpressions.push(`${attribute} = :val${index}`);
    expressionAttributeValues[`:val${index}`] = value;
```

```
  });

  // Add the version update
  updateExpressions.push("version = :newVersion");

  // Define the update parameters with a condition expression
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${updateExpressions.join(", ")}`,
    ConditionExpression: "version = :expectedVersion",
    ExpressionAttributeValues: expressionAttributeValues,
    ReturnValues: "UPDATED_NEW"
  };

  try {
    // Perform the update operation
    const response = await docClient.send(new UpdateCommand(params));

    return {
      success: true,
      message: "Update succeeded with optimistic locking",
      newVersion: expectedVersion + 1,
      updatedAttributes: response.Attributes
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Optimistic locking failed: the item was modified by another
 process",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Implement a conditional write that creates an item only if it doesn't exist.
 *
```

```
 * This function demonstrates how to use attribute_not_exists to create an item
 * only if it doesn't already exist (similar to an "INSERT IF NOT EXISTS"
 operation).
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} item - The item to create
 * @returns {Promise<Object>} - Result of the operation
 */
async function createIfNotExists(
  config,
  tableName,
  item
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Extract the primary key attributes
  const keyAttributes = Object.keys(item).filter(attr =>
    attr === "id" || attr === "ID" || attr === "Id" ||
    attr.endsWith("Id") || attr.endsWith("ID") ||
    attr.endsWith("Key")
  );

  if (keyAttributes.length === 0) {
    throw new Error("Could not determine primary key attributes");
  }

  // Create a condition expression that checks if the item doesn't exist
  const conditionExpression = `attribute_not_exists(${keyAttributes[0]})`;

  // Define the put parameters with a condition expression
  const params = {
    TableName: tableName,
    Item: item,
    ConditionExpression: conditionExpression
  };

  try {
    // Perform the put operation
    await docClient.send(new PutCommand(params));

    return {
```

```
      success: true,
      message: "Item was created because it didn't exist",
      item
    };
  } catch (error) {
    // Check if the error is due to the condition check failing
    if (error.name === "ConditionalCheckFailedException") {
      return {
        success: false,
        message: "Item already exists, creation was skipped",
        error: "ConditionalCheckFailedException"
      };
    }

    // Re-throw other errors
    throw error;
  }
}

/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };
```

```
  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung bedingter Operationen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use conditional operations.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };

  console.log("Demonstrating conditional operations in DynamoDB");

  try {
    // Example 1: Conditional update based on attribute value
    console.log("\nExample 1: Conditional update based on attribute value");
    const updateResult = await conditionalUpdate(
      config,
      tableName,
      key,
      "Category",
      "Electronics",
      "Price",
      299.99
    );

    console.log(`Result: ${updateResult.message}`);
    if (updateResult.success) {
      console.log("Updated attributes:", updateResult.updatedAttributes);
    }

    // Example 2: Conditional delete based on attribute value
    console.log("\nExample 2: Conditional delete based on attribute value");
    const deleteResult = await conditionalDelete(
      config,
```

```
    tableName,
    key,
    "InStock",
    false
  );

  console.log(`Result: ${deleteResult.message}`);
  if (deleteResult.success) {
    console.log("Deleted item:", deleteResult.deletedItem);
  }

  // Example 3: Optimistic locking with version number
  console.log("\nExample 3: Optimistic locking with version number");

  // First, get the current item to check its version
  const currentItem = await getItem(config, tableName, { ProductId: "P67890" });
  const currentVersion = currentItem ? (currentItem.version || 0) : 0;

  console.log(`Current version: ${currentVersion}`);

  // Then, update with optimistic locking
  const lockingResult = await updateWithOptimisticLocking(
    config,
    tableName,
    { ProductId: "P67890" },
    {
      Name: "Updated Product Name",
      Description: "This is an updated description"
    },
    currentVersion
  );

  console.log(`Result: ${lockingResult.message}`);
  if (lockingResult.success) {
    console.log(`New version: ${lockingResult.newVersion}`);
    console.log("Updated attributes:", lockingResult.updatedAttributes);
  }

  // Example 4: Create item only if it doesn't exist
  console.log("\nExample 4: Create item only if it doesn't exist");
  const createResult = await createIfNotExists(
    config,
    tableName,
    {
```

```
        ProductId: "P99999",
        Name: "New Product",
        Category: "Accessories",
        Price: 19.99,
        InStock: true
      }
    );

    console.log(`Result: ${createResult.message}`);
    if (createResult.success) {
      console.log("Created item:", createResult.item);
    }

    // Explain conditional operations
    console.log("\nKey points about conditional operations:");
    console.log("1. Conditional operations only succeed if the condition is met");
    console.log("2. ConditionalCheckFailedException indicates the condition wasn't
 met");
    console.log("3. Optimistic locking prevents race conditions in concurrent
 updates");
    console.log("4. attribute_exists and attribute_not_exists are useful for
 checking if attributes are present");
    console.log("5. Conditional operations are atomic - they either succeed
 completely or fail completely");
    console.log("6. You can use any valid comparison operators and functions in
 condition expressions");
    console.log("7. Conditional operations don't consume write capacity if the
 condition fails");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - DeleteItem
  - PutItem
  - UpdateItem

Verwenden von Ausdrucksattributnamen

Im folgenden Codebeispiel wird gezeigt, wie Sie Ausdrucksattributnamen in DynamoDB verwenden.

- Arbeiten mit reservierten Wörtern in DynamoDB-Ausdrücken
- Verwenden von Platzhaltern für Ausdrucksattributnamen
- Behandeln von Sonderzeichen in Attributnamen

SDK für JavaScript (v3)

Demonstrieren Sie Ausdrucksattributnamen mit AWS SDK für JavaScript.

```
const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
const {
  DynamoDBDocumentClient,
  UpdateCommand,
  GetCommand,
  QueryCommand,
  ScanCommand
} = require("@aws-sdk/lib-dynamodb");

/**
 * Update an attribute that is a reserved word in DynamoDB.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that are reserved words in DynamoDB.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} reservedWordAttribute - The reserved word attribute to update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateReservedWordAttribute(
  config,
  tableName,
  key,
  reservedWordAttribute,
  value
) {
  // Initialize the DynamoDB client
```

```
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Update an attribute that contains special characters.
 *
 * This function demonstrates how to use expression attribute names to update
 * attributes that contain special characters.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string} specialCharAttribute - The attribute with special characters to
 update
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateSpecialCharacterAttribute(
  config,
  tableName,
  key,
  specialCharAttribute,
  value
) {
```

```
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the update parameters using expression attribute names
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: "SET #attr = :value",
    ExpressionAttributeNames: {
      "#attr": specialCharAttribute
    },
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Query items using an attribute that is a reserved word.
 *
 * This function demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {string} partitionKeyName - The name of the partition key attribute
 * @param {any} partitionKeyValue - The value of the partition key
 * @param {string} reservedWordAttribute - The reserved word attribute to filter on
 * @param {any} value - The value to compare against
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function queryWithReservedWordAttribute(
  config,
  tableName,
  partitionKeyName,
  partitionKeyValue,
  reservedWordAttribute,
```

```
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the query parameters using expression attribute names
  const params = {
    TableName: tableName,
    KeyConditionExpression: "#pkName = :pkValue",
    FilterExpression: "#attr = :value",
    ExpressionAttributeNames: {
      "#pkName": partitionKeyName,
      "#attr": reservedWordAttribute
    },
    ExpressionAttributeValues: {
      ":pkValue": partitionKeyValue,
      ":value": value
    }
  };

  // Perform the query operation
  const response = await docClient.send(new QueryCommand(params));

  return response;
}

/**
 * Update a nested attribute with a path that contains reserved words.
 *
 * This function demonstrates how to use expression attribute names to update
 * nested attributes where the path contains reserved words.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to update
 * @param {string[]} attributePath - The path to the nested attribute as an array
 * @param {any} value - The value to set
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function updateNestedReservedWordAttribute(
  config,
  tableName,
  key,
```

```
  attributePath,
  value
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names for each part of the path
  const expressionAttributeNames = {};
  for (let i = 0; i < attributePath.length; i++) {
    expressionAttributeNames[`#attr${i}`] = attributePath[i];
  }

  // Build the attribute path using the expression attribute names
  const attributePathExpression = attributePath
    .map((_, i) => `#attr${i}`)
    .join(".");

  // Define the update parameters
  const params = {
    TableName: tableName,
    Key: key,
    UpdateExpression: `SET ${attributePathExpression} = :value`,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: {
      ":value": value
    },
    ReturnValues: "UPDATED_NEW"
  };

  // Perform the update operation
  const response = await docClient.send(new UpdateCommand(params));

  return response;
}

/**
 * Scan a table with multiple attribute name placeholders.
 *
 * This function demonstrates how to use multiple expression attribute names
 * in a complex filter expression.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
```

```
 * @param {Object} filters - Object mapping attribute names to filter values
 * @returns {Promise<Object>} - The response from DynamoDB
 */
async function scanWithMultipleAttributeNames(
  config,
  tableName,
  filters
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Create expression attribute names and values
  const expressionAttributeNames = {};
  const expressionAttributeValues = {};
  const filterConditions = [];

  // Build the filter expression
  Object.entries(filters).forEach(([attrName, value], index) => {
    const nameKey = `#attr${index}`;
    const valueKey = `:val${index}`;

    expressionAttributeNames[nameKey] = attrName;
    expressionAttributeValues[valueKey] = value;
    filterConditions.push(`${nameKey} = ${valueKey}`);
  });

  // Join the filter conditions with AND
  const filterExpression = filterConditions.join(" AND ");

  // Define the scan parameters
  const params = {
    TableName: tableName,
    FilterExpression: filterExpression,
    ExpressionAttributeNames: expressionAttributeNames,
    ExpressionAttributeValues: expressionAttributeValues
  };

  // Perform the scan operation
  const response = await docClient.send(new ScanCommand(params));

  return response;
}
```

```
/**
 * Get the current value of an item.
 *
 * Helper function to retrieve the current value of an item.
 *
 * @param {Object} config - AWS configuration object
 * @param {string} tableName - The name of the DynamoDB table
 * @param {Object} key - The key of the item to get
 * @returns {Promise<Object|null>} - The item or null if not found
 */
async function getItem(
  config,
  tableName,
  key
) {
  // Initialize the DynamoDB client
  const client = new DynamoDBClient(config);
  const docClient = DynamoDBDocumentClient.from(client);

  // Define the get parameters
  const params = {
    TableName: tableName,
    Key: key
  };

  // Perform the get operation
  const response = await docClient.send(new GetCommand(params));

  // Return the item if it exists, otherwise null
  return response.Item || null;
}
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK für JavaScript.

```
/**
 * Example of how to use expression attribute names.
 */
async function exampleUsage() {
  // Example parameters
  const config = { region: "us-west-2" };
  const tableName = "Products";
  const key = { ProductId: "P12345" };
```

```
console.log("Demonstrating expression attribute names in DynamoDB");

try {
  // Example 1: Update an attribute that is a reserved word
  console.log("\nExample 1: Updating an attribute that is a reserved word");
  const response1 = await updateReservedWordAttribute(
    config,
    tableName,
    key,
    "Size", // "SIZE" is a reserved word in DynamoDB
    "Large"
  );

  console.log("Updated attribute:", response1.Attributes);

  // Example 2: Update an attribute with special characters
  console.log("\nExample 2: Updating an attribute with special characters");
  const response2 = await updateSpecialCharacterAttribute(
    config,
    tableName,
    key,
    "Product-Type", // Contains a hyphen, which is a special character
    "Electronics"
  );

  console.log("Updated attribute:", response2.Attributes);

  // Example 3: Query with a reserved word attribute
  console.log("\nExample 3: Querying with a reserved word attribute");
  const response3 = await queryWithReservedWordAttribute(
    config,
    tableName,
    "Category",
    "Electronics",
    "Count", // "COUNT" is a reserved word in DynamoDB
    10
  );

  console.log(`Found ${response3.Items.length} items`);

  // Example 4: Update a nested attribute with reserved words in the path
  console.log("\nExample 4: Updating a nested attribute with reserved words in the
path");
```

```
    const response4 = await updateNestedReservedWordAttribute(
      config,
      tableName,
      key,
      ["Dimensions", "Size", "Height"], // "SIZE" is a reserved word
      30
    );

    console.log("Updated nested attribute:", response4.Attributes);

    // Example 5: Scan with multiple attribute name placeholders
    console.log("\nExample 5: Scanning with multiple attribute name placeholders");
    const response5 = await scanWithMultipleAttributeNames(
      config,
      tableName,
      {
        "Size": "Large",
        "Count": 10,
        "Product-Type": "Electronics"
      }
    );

    console.log(`Found ${response5.Items.length} items`);

    // Get the final state of the item
    console.log("\nFinal state of the item:");
    const item = await getItem(config, tableName, key);
    console.log(JSON.stringify(item, null, 2));

    // Show some common reserved words
    console.log("\nSome common DynamoDB reserved words:");
    const commonReservedWords = [
      "ABORT", "ABSOLUTE", "ACTION", "ADD", "ALL", "ALTER", "AND", "ANY", "AS",
      "ASC", "BETWEEN", "BY", "CASE", "CAST", "COLUMN", "CONNECT", "COUNT",
      "CREATE", "CURRENT", "DATE", "DELETE", "DESC", "DROP", "ELSE", "EXISTS",
      "FOR", "FROM", "GRANT", "GROUP", "HAVING", "IN", "INDEX", "INSERT", "INTO",
      "IS", "JOIN", "KEY", "LEVEL", "LIKE", "LIMIT", "LOCAL", "MAX", "MIN", "NAME",
      "NOT", "NULL", "OF", "ON", "OR", "ORDER", "OUTER", "REPLACE", "RETURN",
      "SELECT", "SET", "SIZE", "TABLE", "THEN", "TO", "UPDATE", "USER", "VALUES",
      "VIEW", "WHERE"
    ];
    console.log(commonReservedWords.join(", "));

    // Explain expression attribute names
```

```
    console.log("\nKey points about expression attribute names:");
    console.log("1. Use expression attribute names (#name) for reserved words");
    console.log("2. Use expression attribute names for attributes with special
 characters");
    console.log("3. Special characters include: spaces, hyphens, dots, and other
 non-alphanumeric characters");
    console.log("4. Expression attribute names are required for nested attributes
 with reserved words");
    console.log("5. You can use multiple expression attribute names in a single
 expression");
    console.log("6. Expression attribute names are case-sensitive");
    console.log("7. Expression attribute names are only used in expressions, not in
 the actual data");

  } catch (error) {
    console.error("Error:", error);
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - Query
  - UpdateItem

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Laufzeit-API. JavaScript In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines DynamoDB-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};
```

```
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Es wird ein DynamoDB-Ereignis unter Verwendung von Lambda konsumiert. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Meldung von Fehlern bei DynamoDB-Batchelementen mithilfe von Lambda. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
```

```
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier: curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Meldung von Fehlern bei DynamoDB-Batchelementen unter Verwendung von Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;

  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

Amazon EC2-Beispiele mit SDK für (v3) JavaScript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon EC2.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
  const client = new EC2Client();
  try {
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({}),
    );

    const securityGroupList = SecurityGroups.slice(0, 9)
      .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
      .join("\n");

    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.

- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Diese Datei enthält eine Liste häufig verwendeter Aktionen für EC2. Die Schritte sind mit einem Szenario-Framework aufgebaut, das die Ausführung eines interaktiven Beispiels vereinfacht. Den vollständigen Kontext finden Sie im GitHub Repository.

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
import { join } from "node:path";
import { get } from "node:http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
```

```
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

/**
 * @typedef {{
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,
 *   errors: Error[],
 *   keyPairId?: string,
 *   tmpDirectory?: string,
 *   securityGroupId?: string,
 *   ipAddress?: string,
 *   images?: import('@aws-sdk/client-ec2').Image[],
 *   image?: import('@aws-sdk/client-ec2').Image,
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],
 *   instanceId?: string,
 *   instanceIpAddress?: string,
 *   allocationId?: string,
 *   allocatedIpAddress?: string,
 *   associationId?: string,
 * }} State
 */

/**
 * A skip function provided to the `skipWhen` of a Step when you want
 * to ignore that step if any errors have occurred.
 * @param {State} state
 */
const skipWhenErrors = (state) => state.errors.length > 0;

const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;

export const confirm = new ScenarioInput("confirmContinue", "Continue?", {
  type: "confirm",
  skipWhen: skipWhenErrors,
});

export const exitOnNoConfirm = new ScenarioAction(
```

```
  "exitOnConfirmContinueFalse",
  (/** @type { { earlyExit: boolean } & Record<string, any>} */ state) => {
    if (!state[confirm.name]) {
      state.earlyExit = true;
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

export const greeting = new ScenarioOutput(
  "greeting",
  `

Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
 - A key pair - This is for SSH access to your EC2 instance. You only need to
 provide the name.
 - A security group - This is used for configuring access to your instance. Again,
 only the name is needed.
 - An IP address - Your public IP address will be fetched.
 - An Amazon Machine Image (AMI)
 - A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyPairName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
```

```
      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
      await writeFile(
        `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
        KeyMaterial,
        {
          mode: 0o400,
        },
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidKeyPair.Duplicate"
      ) {
        caught.message = `${caught.message}. Try another key name.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
```

```
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(
        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
      ) {
        caught.message = `${caught.message}. Did you provide the required value?`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no key pair to delete or the user chooses
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.keyPairId || !state[confirmDeleteKeyPair.name],
  },
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",
  { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
  "createSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Create a new security group that will be used to configure ingress/egress
 for
      // an EC2 instance.
      const { GroupId } = await state.ec2Client.send(
        new CreateSecurityGroupCommand({
          GroupName: state[provideSecurityGroupName.name],
          Description: "A security group for the Amazon EC2 example.",
        }),
      );
```

```
      state.securityGroupId = GroupId;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {
        caught.message = `${caught.message}. Please provide a different name for
 your security group.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
```

```
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
      state.errors.push(caught);
    }
  },
  {
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (/** @type {State} */ state) =>
      !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
  },
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (/** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
      state.ipAddress = ipAddress;
      // Allow ingress from the IP address above to the security group.
      // This will allow you to SSH into the EC2 instance.
      const command = new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.securityGroupId,
        IpPermissions: [
          {
            IpProtocol: "tcp",
            FromPort: 22,
            ToPort: 22,
            IpRanges: [{ CidrIp: `${ipAddress}/32` }],
          },
        ],
      });
```

```
      await state.ec2Client.send(command);
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS Systems
 Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
 parameters.

    // Create the paginator for getting images. Actions that return multiple pages
 of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
```

```
      },
    );

    try {
      for await (const page of getParametersByPathPaginator) {
        for (const param of page.Parameters) {
          // Filter by Amazon Linux 2
          if (param.Name.includes("amzn2")) {
            AMIs.push(param.Value);
          }
        }
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidFilterValue") {
        caught.message = `${caught.message} Please provide a valid filter value for
paginateGetParametersByPath.`;
      }
      state.errors.push(caught);
      return;
    }

    const imageDetails = [];
    const describeImagesPaginator = paginateDescribeImages(
      { client: state.ec2Client },
      // The images found from the call to SSM.
      { ImageIds: AMIs },
    );

    try {
      // Get more details for the images found above.
      for await (const page of describeImagesPaginator) {
        imageDetails.push(...(page.Images || []));
      }

      // Store the image details for later use.
      state.images = imageDetails;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
        caught.message = `${caught.message}. Please provide a valid image id.`;
      }

      state.errors.push(caught);
    }
  },
```

```
  { skipWhen: skipWhenErrors },
);

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);

export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "*.small"] },
        ],
      },
    );

    const instanceTypes = [];

    try {
      for await (const page of paginator) {
        if (page.InstanceTypes.length) {
          instanceTypes.push(...(page.InstanceTypes || []));
```

```
        }
      }

      if (!instanceTypes.length) {
        state.errors.push(
          "No instance types matched the instance type filters.",
        );
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check the provided values and
 try again.`;
      }

      state.errors.push(caught);
    }

    state.instanceTypes = instanceTypes;
  },
  { skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
```

```
        KeyName: state[provideKeyPairName.name],
        SecurityGroupIds: [state.securityGroupId],
        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your ability
 to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
 least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
 even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and `MaxCount`
 set to the same value.
        // If you want to launch up to a certain number of instances, use `MaxCount`
 and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
      }),
    );

    state.instanceId = Instances[0].InstanceId;

    try {
      // Poll `DescribeInstanceStatus` until status is "ok".
      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);
```

```
export const logRunInstance = new ScenarioOutput(
  "logRunInstance",
  "The next step is to run your EC2 instance for the first time. This can take a few
 minutes.",
  { header: true, skipWhen: skipWhenErrors },
);

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (/** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );

      for await (const page of paginator) {
        for (const reservation of page.Reservations) {
          instances.push(...reservation.Instances);
        }
      }
      if (instances.length !== 1) {
        throw new Error(`Instance ${state.instanceId} not found.`);
      }

      // The only info we need is the IP address for SSH purposes.
      state.instanceIpAddress = instances[0].PublicIpAddress;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check provided values and try
 again.`;
      }

      state.errors.push(caught);
    }
  },
```

```
  { skipWhen: skipWhenErrors },
);

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);

export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStopped(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
```

```
  // Don't try to stop an instance that doesn't exist.
  { skipWhen: (/** @type { State } */ state) => !state.instanceId },
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is
 restarted.",
    "The next step is to stop and restart your instance to demonstrate this
 behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }
```

```
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
  "logIpAllocation",
  [
    "It is possible to have a static IP address.",
    "To demonstrate this, an IP will be allocated and associated to your EC2
 instance.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
  "allocateIp",
  async (/** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours until
 you release it.
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
      state.allocationId = AllocationId;
      state.allocatedIpAddress = PublicIp;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "MissingParameter") {
        caught.message = `${caught.message}. Did you provide these values?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
  "associateIp",
  async (/** @type { State } */ state) => {
    try {
      // Associate an allocated IP address to an EC2 instance. An IP address can be
 allocated
```

```
      // with the AllocateAddress action.
      const { AssociationId } = await state.ec2Client.send(
        new AssociateAddressCommand({
          AllocationId: state.allocationId,
          InstanceId: state.instanceId,
        }),
      );
      state.associationId = AssociationId;
      // Update the IP address that is being tracked to match
      // the one just associated.
      state.instanceIpAddress = state.allocatedIpAddress;
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
        caught.message = `${caught.message}. Did you provide the ID of a valid
 Elastic IP address AllocationId?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(
  "logStaticIpProof",
  "The IP address should remain the same even after stopping and starting the
 instance.",
  { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
  "logCleanUp",
  "That's it! You can choose to clean up the resources now, or clean them up on your
 own later.",
  { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
  "confirmDisassociateAddress",
  "Do you want to disassociate and release the static IP address created earlier?",
  {
    type: "confirm",
```

```
    skipWhen: (/** @type { State } */ state) => !state.associationId,
  },
);

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid association
 ID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
```

```
        caught.message = `${caught.message}. Please provide a valid AllocationID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.allocationId,
  },
);

export const confirmTerminateInstance = new ScenarioInput(
  "confirmTerminateInstance",
  "Do you want to terminate the instance?",
  // Don't do anything when an instance was never run.
  {
    skipWhen: (/** @type { State } */ state) => !state.instanceId,
    type: "confirm",
  },
);

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new TerminateInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );
      await waitUntilInstanceTerminated(
        { client: state.ec2Client },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the
 waiter.`;
      }

      state.errors.push(caught);
    }
  },
  {
```

```
    // Don't do anything when there's no instance to terminate or the
    // use chooses not to terminate.
    skipWhen: (/** @type { State } */ state) =>
      !state.instanceId || !state[confirmTerminateInstance.name],
  },
);

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (/** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair

- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AllocateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Allocates an Elastic IP address to your AWS account.
```

```
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon EC2.
 Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Details finden Sie AllocateAddress in der AWS SDK für JavaScript API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AssociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that are
 in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
 instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
 with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
 ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Did you provide the ID of a valid Elastic IP address
 AllocationId?`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie AssociateAddress in der AWS SDK für JavaScript API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  AuthorizeSecurityGroupIngressCommand,
  EC2Client,
} from "@aws-sdk/client-ec2";

/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });
```

```
  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK für JavaScript API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save to
 a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new CreateKeyPairCommand({
```

```
    KeyName: keyName,
  });

  try {
    const { KeyMaterial, KeyName } = await client.send(command);
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
      console.warn(`${caught.message}. Try another key name.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateKeyPair in der AWS SDK für JavaScript API-Referenz.

`CreateLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateLaunchTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
```

```
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
```

- API-Details finden Sie CreateLaunchTemplate in der AWS SDK für JavaScript API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
  const client = new EC2Client({});
  const command = new CreateSecurityGroupCommand({
    // Up to 255 characters in length. Cannot start with sg-.
    GroupName: groupName,
    // Up to 255 characters in length.
    Description: description,
  });
```

```
  try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK für JavaScript API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
```

```
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK für JavaScript API-Referenz.

`DeleteLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLaunchTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    await client.send(
      new DeleteLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
      }),
    );
```

- API-Details finden Sie DeleteLaunchTemplate in der AWS SDK für JavaScript API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK für JavaScript API-Referenz.

`DescribeAddresses`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAddresses`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeAddressesCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: [allocationId],
  });

  try {
    const { Addresses } = await client.send(command);
    const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
    console.log("Elastic IP addresses:");
    console.log(addressList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationId.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeAddresses in der AWS SDK für JavaScript API-Referenz.

`DescribeIamInstanceProfileAssociations`

Das folgende Codebeispiel zeigt die Verwendung`DescribeIamInstanceProfileAssociations`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
```

- API-Details finden Sie DescribeIamInstanceProfileAssociations in der AWS SDK für JavaScript API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";

/**
```

```
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all of
 the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize is
 just sugar for
    // the MaxResults property in the base command.
    { client, pageSize },
    {
      // There are almost 70,000 images available. Be specific with your filtering
      // to increase efficiency.
      // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
ec2/interfaces/describeimagescommandinput.html#filters
      Filters: [{ Name: "architecture", Values: [architecture] }],
    },
  );

  /**
   * @type {import('@aws-sdk/client-ec2').Image[]}
   */
  const images = [];
  let recordsScanned = 0;

  try {
    for await (const page of paginator) {
      recordsScanned += pageSize;
      if (page.Images.length) {
        images.push(...page.Images);
        break;
      }
      console.log(
        `No matching image found yet. Searched ${recordsScanned} records.`,
      );
    }

    if (images.length) {
      console.log(
```

```
        `Found ${images.length} images:\n\n${images.map((image) =>
 image.Name).join("\n")}\n`,
      );
    } else {
      console.log(
        `No matching images found. Searched ${recordsScanned} records.\n`,
      );
    }

    return images;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};
```

- API-Details finden Sie DescribeImages in der AWS SDK für JavaScript API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstanceTypes } from "@aws-sdk/client-ec2";

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }} options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
```

```
 pageSize = Number.parseInt(pageSize);
 const client = new EC2Client({});

 // The paginate function is a wrapper around the underlying command.
 const paginator = paginateDescribeInstanceTypes(
   // Without limiting the page size, this call can take a long time. pageSize is
just sugar for
   // the MaxResults property in the underlying command.
   { client, pageSize },
   {
     Filters: [
       {
         Name: "processor-info.supported-architecture",
         Values: supportedArch,
       },
       { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
     ],
   },
 );

 try {
   /**
    * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
    */
   const instanceTypes = [];

   for await (const page of paginator) {
     if (page.InstanceTypes.length) {
       instanceTypes.push(...page.InstanceTypes);

       // When we have at least 1 result, we can stop.
       if (instanceTypes.length >= 1) {
         break;
       }
     }
   }
   console.log(
     `Memory size in MiB for matching instance types:\n\n${instanceTypes.map((it)
=> `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB} MiB`).join("\n")}`,
   );
 } catch (caught) {
   if (caught instanceof Error && caught.name === "InvalidParameterValue") {
     console.warn(`${caught.message}`);
     return [];
```

```
    }
    throw caught;
  }
};
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK für JavaScript API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstances } from "@aws-sdk/client-ec2";

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;

  const paginator = paginateDescribeInstances(
    {
      client,
      pageSize,
    },
    {
```

```
      Filters: [
        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
          Name: "launch-time",
          Values: [launchTimePattern],
        },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').Instance[]}
     */
    const instanceList = [];
    for await (const page of paginator) {
      const { Reservations } = page;
      for (const reservation of Reservations) {
        instanceList.push(...reservation.Instances);
      }
    }
    console.log(
      `Running instances launched this month:\n\n${instanceList.map((instance) =>
 instance.InstanceId).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeInstances in der AWS SDK für JavaScript API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeKeyPairsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK für JavaScript API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeRegionsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
 returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
            // You can specify multiple values for a filter.
            // You can also use '*' as a wildcard. This will return all
            // of the regions that start with `us-east-`.
            Values: regionNames,
          },
        ]
      : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
```

```
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeRegions in der AWS SDK für JavaScript API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
export const main = async ({ groupIds = [] }) => {
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
```

```
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {
      console.warn(caught.message);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für JavaScript API-Referenz.

`DescribeSubnets`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubnets`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
```

- API-Details finden Sie DescribeSubnets in der AWS SDK für JavaScript API-Referenz.

`DescribeVpcs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVpcs`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new EC2Client({});
const { Vpcs } = await client.send(
  new DescribeVpcsCommand({
    Filters: [{ Name: "is-default", Values: ["true"] }],
  }),
);
```

- API-Details finden Sie DescribeVpcs in der AWS SDK für JavaScript API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";
```

```
/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DisassociateAddress in der AWS SDK für JavaScript API-Referenz.

`MonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`MonitorInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, MonitorInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie MonitorInstances in der AWS SDK für JavaScript API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, RebootInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Please provide the InstanceId of a valid instance to
 reboot.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie RebootInstances in der AWS SDK für JavaScript API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ReleaseAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
  const client = new EC2Client({});
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AllocationId: allocationId,
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie ReleaseAddress in der AWS SDK für JavaScript API-Referenz.

`ReplaceIamInstanceProfileAssociation`

Das folgende Codebeispiel zeigt die Verwendung`ReplaceIamInstanceProfileAssociation`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );
```

- API-Details finden Sie ReplaceIamInstanceProfileAssociation in der AWS SDK für JavaScript API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, RunInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Create new EC2 instances.
 * @param {{
 *  keyName: string,
 *  securityGroupIds: string[],
 *  imageId: string,
 *  instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *  minCount?: number,
 *  maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,
  imageId,
  instanceType,
  minCount = "1",
  maxCount = "1",
}) => {
  const client = new EC2Client({});
  minCount = Number.parseInt(minCount);
  maxCount = Number.parseInt(maxCount);
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: keyName,
    // Your security group.
    SecurityGroupIds: securityGroupIds,
    // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
 For more information, see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
    // An instance type describing the resources provided to your instance. There
 are multiple
    // ways to search for instance types. For more information see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-discovery.html
    InstanceType: instanceType,
    // Availability Zones have capacity limitations that may impact your ability to
 launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at least
 the `MinCount` of instances.
    // However, EC2 will attempt to launch up to the `MaxCount` of instances, even
 if the full request cannot be satisfied.
```

```
    // If you need a specific number of instances, use `MinCount` and `MaxCount` set
 to the same value.
    // If you want to launch up to a certain number of instances, use `MaxCount` and
 let EC2 provision as many as possible.
    // If you require a minimum number of instances, but do not want to exceed a
 maximum, use both `MinCount` and `MaxCount`.
    MinCount: minCount,
    MaxCount: maxCount,
  });

  try {
    const { Instances } = await client.send(command);
    const instanceList = Instances.map(
      (instance) => `• ${instance.InstanceId}`,
    ).join("\n");
    console.log(`Launched instances:\n${instanceList}`);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie RunInstances in der AWS SDK für JavaScript API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
```

```
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie StartInstances in der AWS SDK für JavaScript API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie StopInstances in der AWS SDK für JavaScript API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
```

```
      throw caught;
    }
  }
};
```

- API-Details finden Sie TerminateInstances in der AWS SDK für JavaScript API-Referenz.

`UnmonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`UnmonitorInstances`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
 ${im.Monitoring.State}.`,
    );
```

```
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie UnmonitorInstances in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
```

```
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
```

```
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
```

```
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
```

```
    new ScenarioOutput(
      "populatingTable",
      MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("populateTable", () => {
      const client = new DynamoDBClient({});
      /**
       * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
       */
      const recommendations = JSON.parse(
        readFileSync(join(RESOURCES_PATH, "recommendations.json")),
      );

      return client.send(
        new BatchWriteItemCommand({
          RequestItems: {
            [NAMES.tableName]: recommendations.map((item) => ({
              PutRequest: { Item: item },
            })),
          },
        }),
      );
    }),
    new ScenarioOutput(
      "populatedTable",
      MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioOutput(
      "creatingKeyPair",
      MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
    ),
    new ScenarioAction("createKeyPair", async () => {
      const client = new EC2Client({});
      const { KeyMaterial } = await client.send(
        new CreateKeyPairCommand({
          KeyName: NAMES.keyPairName,
        }),
      );

      writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
    }),
    new ScenarioOutput(
      "createdKeyPair",
      MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
```

```
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
```

```
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
```

```
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
```

```
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
```

```
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
```

```
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
```

```
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
```

```
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
```

```
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
```

```
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
```

```
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
```

```
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
```

```
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
```

```
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
```

```
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
```

```
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
```

```
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
```

```
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
 state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
```

```
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
```

```
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
```

```
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );
```

```
  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
```

```
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
```

```
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
```

```
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
```

```
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
```

```
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
```

```
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
```

```
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
```

```
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
```

```
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
```

```
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate

- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Elastic Load Balancing — Beispiele für Version 2 mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) zusammen mit Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Elastic Load Balancing

Das folgende Codebeispiel zeigt die ersten Schritte mit Elastic Load Balancing.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
```

```
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Details finden Sie DescribeLoadBalancers in der AWS SDK für JavaScript API-Referenz.

Aktionen

`CreateListener`

Das folgende Codebeispiel zeigt die Verwendung`CreateListener`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
```

- API-Details finden Sie CreateListener in der AWS SDK für JavaScript API-Referenz.

`CreateLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`CreateLoadBalancer`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { LoadBalancers } = await client.send(
  new CreateLoadBalancerCommand({
    Name: NAMES.loadBalancerName,
    Subnets: state.subnets,
  }),
);
state.loadBalancerDns = LoadBalancers[0].DNSName;
state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
await waitUntilLoadBalancerAvailable(
  { client },
  { Names: [NAMES.loadBalancerName] },
);
```

- API-Details finden Sie CreateLoadBalancer in der AWS SDK für JavaScript API-Referenz.

`CreateTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateTargetGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { TargetGroups } = await client.send(
  new CreateTargetGroupCommand({
    Name: NAMES.loadBalancerTargetGroupName,
    Protocol: "HTTP",
    Port: 80,
    HealthCheckPath: "/healthcheck",
    HealthCheckIntervalSeconds: 10,
    HealthCheckTimeoutSeconds: 5,
    HealthyThresholdCount: 2,
    UnhealthyThresholdCount: 2,
    VpcId: state.defaultVpc,
  }),
);
```

- API-Details finden Sie CreateTargetGroup in der AWS SDK für JavaScript API-Referenz.

`DeleteLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLoadBalancer`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  const client = new ElasticLoadBalancingV2Client({});
  const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
```

```
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
```

- API-Details finden Sie DeleteLoadBalancer in der AWS SDK für JavaScript API-Referenz.

`DeleteTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTargetGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  const client = new ElasticLoadBalancingV2Client({});
  try {
    const { TargetGroups } = await client.send(
      new DescribeTargetGroupsCommand({
        Names: [NAMES.loadBalancerTargetGroupName],
      }),
    );

    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      client.send(
        new DeleteTargetGroupCommand({
          TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
      ),
    );
```

```
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
```

- API-Details finden Sie DeleteTargetGroup in der AWS SDK für JavaScript API-Referenz.

`DescribeLoadBalancers`

Das folgende Codebeispiel zeigt die Verwendung`DescribeLoadBalancers`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ElasticLoadBalancingV2Client,
  DescribeLoadBalancersCommand,
} from "@aws-sdk/client-elastic-load-balancing-v2";

export async function main() {
  const client = new ElasticLoadBalancingV2Client({});
  const { LoadBalancers } = await client.send(
    new DescribeLoadBalancersCommand({}),
  );
  const loadBalancersList = LoadBalancers.map(
    (lb) => `• ${lb.LoadBalancerName}: ${lb.DNSName}`,
  ).join("\n");
  console.log(
    "Hello, Elastic Load Balancing! Let's list some of your load balancers:\n",
    loadBalancersList,
  );
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
```

```
  main();
}
```

- API-Details finden Sie DescribeLoadBalancers in der AWS SDK für JavaScript API-Referenz.

`DescribeTargetGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTargetGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const client = new ElasticLoadBalancingV2Client({});
const { TargetGroups } = await client.send(
  new DescribeTargetGroupsCommand({
    Names: [NAMES.loadBalancerTargetGroupName],
  }),
);
```

- API-Details finden Sie DescribeTargetGroups in der AWS SDK für JavaScript API-Referenz.

`DescribeTargetHealth`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTargetHealth`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const { TargetHealthDescriptions } = await client.send(
  new DescribeTargetHealthCommand({
    TargetGroupArn: TargetGroups[0].TargetGroupArn,
  }),
);
```

- API-Details finden Sie DescribeTargetHealth in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
```

```
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
```

```
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
```

```
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
     */
```

```
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
```

```
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
```

```
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
```

```
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
```

```
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
```

```
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
```

```
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
```

```
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
```

```
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
```

```
      */
    const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
    state.myIp = ipResponse.trim();
    const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
      ({ IpRanges }) =>
        IpRanges.some(
          ({ CidrIp }) =>
            CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
        ),
    )
      .filter(({ IpProtocol }) => IpProtocol === "tcp")
      .filter(({ FromPort }) => FromPort === 80);

    state.myIpRules = myIpRules;
  },
),
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    }
    return MESSAGES.noIpRules;
  },
),
new ScenarioInput(
  "shouldAddInboundRule",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return false;
    }
    return MESSAGES.noIpRules;
  },
  { type: "confirm" },
),
```

```
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
```

```
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
```

```
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);
```

```
const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
```

```
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
```

```
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
```

```
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
```

```
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
```

```
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
```

```
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
```

```
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
```

```
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";
```

```
import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
```

```
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
```

```
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
```

```
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
```

```
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
```

```
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
```

```
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
```

```
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
```

```
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
```

```
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
```

```
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
```

```
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
```

```
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
```

```
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

AWS Entity Resolution Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren AWS Entity Resolution.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo AWS Entity Resolution

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Entity Resolution beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  EntityResolutionClient,
  ListMatchingWorkflowsCommand,
} from "@aws-sdk/client-entityresolution";

export const main = async () => {
  const region = "eu-west-1";
  const erClient = new EntityResolutionClient({ region: region });
  try {
    const command = new ListMatchingWorkflowsCommand({});
    const response = await erClient.send(command);
    const workflowSummaries = response.workflowSummaries;
    for (const workflowSummary of workflowSummaries) {
      console.log(`Attribute name: ${workflowSummaries[0].workflowName} `);
    }
    if (workflowSummaries.length === 0) {
      console.log("No matching workflows found.");
    }
  } catch (error) {
    console.error(
      `An error occurred in listing the workflow summaries: ${error.message} \n
 Exiting program.`,
    );
    return;
  }
};
```

- API-Details finden Sie ListMatchingWorkflows in der AWS SDK für JavaScript API-Referenz.

Aktionen

`CreateMatchingWorkflow`

Das folgende Codebeispiel zeigt die Verwendung`CreateMatchingWorkflow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createMatchingWorkflowParams = {
    roleArn: `${data.inputs.roleArn}`,
    workflowName: `${data.inputs.workflowName}`,
    description: "Created by using the AWS SDK for JavaScript (v3).",
    inputSourceConfig: [
      {
        inputSourceARN: `${data.inputs.JSONinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameJson}`,
```

```
        applyNormalization: false,
      },
      {
        inputSourceARN: `${data.inputs.CSVinputSourceARN}`,
        schemaName: `${data.inputs.schemaNameCSV}`,
        applyNormalization: false,
      },
    ],
    outputSourceConfig: [
      {
        outputS3Path: `s3://${data.inputs.myBucketName}/eroutput`,
        output: [
          {
            name: "id",
          },
          {
            name: "name",
          },
          {
            name: "email",
          },
          {
            name: "phone",
          },
        ],
        applyNormalization: false,
      },
    ],
    resolutionTechniques: { resolutionType: "ML_MATCHING" },
  };
  try {
    const command = new CreateMatchingWorkflowCommand(
      createMatchingWorkflowParams,
    );
    const response = await erClient.send(command);

    console.log(
      `Workflow created successfully.\n The workflow ARN is:
${response.workflowArn}`,
    );
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
```

```
};
```

- API-Details finden Sie CreateMatchingWorkflow in der AWS SDK für JavaScript API-Referenz.

`CreateSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchemaMapping`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  CreateSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const createSchemaMappingParamsJson = {
    schemaName: `${data.inputs.schemaNameJson}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
```

```
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
    ],
  };
  const createSchemaMappingParamsCSV = {
    schemaName: `${data.inputs.schemaNameCSV}`,
    mappedInputFields: [
      {
        fieldName: "id",
        type: "UNIQUE_ID",
      },
      {
        fieldName: "name",
        type: "NAME",
      },
      {
        fieldName: "email",
        type: "EMAIL_ADDRESS",
      },
      {
        fieldName: "phone",
        type: "PROVIDER_ID",
        subType: "STRING",
      },
    ],
  };
  try {
    const command = new CreateSchemaMappingCommand(
      createSchemaMappingParamsJson,
    );
    const response = await erClient.send(command);
    console.log("The JSON schema mapping name is ", response.schemaName);
  } catch (error) {
    console.log("error ", error.message);
  }
};
```

- API-Details finden Sie CreateSchemaMapping in der AWS SDK für JavaScript API-Referenz.

`DeleteMatchingWorkflow`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMatchingWorkflow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteMatchingWorkflowCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  try {
    const deleteWorkflowParams = {
      workflowName: `${data.inputs.workflowName}`,
    };
    const command = new DeleteMatchingWorkflowCommand(deleteWorkflowParams);
    const response = await erClient.send(command);
    console.log("Workflow deleted successfully!", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- API-Details finden Sie DeleteMatchingWorkflow in der AWS SDK für JavaScript API-Referenz.

`DeleteSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSchemaMapping`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  DeleteSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const deleteSchemaMapping = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new DeleteSchemaMappingCommand(deleteSchemaMapping);
    const response = await erClient.send(command);
    console.log("Schema mapping deleted successfully. ", response);
  } catch (error) {
    console.log("error ", error);
  }
};
```

- API-Details finden Sie DeleteSchemaMapping in der AWS SDK für JavaScript API-Referenz.

GetMatchingJob

Das folgende Codebeispiel zeigt die VerwendungGetMatchingJob.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const getJobInfoParams = {
      workflowName: `${data.inputs.workflowName}`,
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new GetMatchingJobCommand(getJobInfoParams);
      const response = await erClient.send(command);
      console.log(`Job status: ${response.status}`);
    } catch (error) {
      console.log("error ", error.message);
    }
  }
};
```

- API-Details finden Sie GetMatchingJob in der AWS SDK für JavaScript API-Referenz.

`GetSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`GetSchemaMapping`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  GetSchemaMappingCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const getSchemaMappingJsonParams = {
    schemaName: `${data.inputs.schemaNameJson}`,
  };
  try {
    const command = new GetSchemaMappingCommand(getSchemaMappingJsonParams);
    const response = await erClient.send(command);
    console.log(response);
    console.log(
      `Schema mapping for the JSON data:\n ${response.mappedInputFields[0]}`,
    );
```

```
    console.log("Schema mapping ARN is: ", response.schemaArn);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- API-Details finden Sie GetSchemaMapping in der AWS SDK für JavaScript API-Referenz.

`ListSchemaMappings`

Das folgende Codebeispiel zeigt die Verwendung`ListSchemaMappings`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";

import {
  ListSchemaMappingsCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  async function getInfo() {
    const listSchemaMappingsParams = {
      workflowName: `${data.inputs.workflowName}`,
```

```
      jobId: `${data.inputs.jobId}`,
    };
    try {
      const command = new ListSchemaMappingsCommand(listSchemaMappingsParams);
      const response = await erClient.send(command);
      const noOfSchemas = response.schemaList.length;
      for (let i = 0; i < noOfSchemas; i++) {
        console.log(
          `Schema Mapping Name: ${response.schemaList[i].schemaName} `,
        );
      }
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  }
```

- API-Details finden Sie ListSchemaMappings in der AWS SDK für JavaScript API-Referenz.

`StartMatchingJob`

Das folgende Codebeispiel zeigt die Verwendung`StartMatchingJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
import {
  StartMatchingJobCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
```

```
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const matchingJobOfWorkflowParams = {
    workflowName: `${data.inputs.workflowName}`,
  };
  try {
    const command = new StartMatchingJobCommand(matchingJobOfWorkflowParams);
    const response = await erClient.send(command);
    console.log(`Job ID: ${response.jobID} \n
The matching job was successfully started.`);
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- API-Details finden Sie StartMatchingJob in der AWS SDK für JavaScript API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//The default inputs for this demo are read from the ../inputs.json.

import { fileURLToPath } from "node:url";
```

```
import {
  TagResourceCommand,
  EntityResolutionClient,
} from "@aws-sdk/client-entityresolution";
import data from "../inputs.json" with { type: "json" };

const region = "eu-west-1";
const erClient = new EntityResolutionClient({ region: region });

export const main = async () => {
  const tagResourceCommandParams = {
    resourceArn: `${data.inputs.schemaArn}`,
    tags: {
      tag1: "tag1Value",
      tag2: "tag2Value",
    },
  };
  try {
    const command = new TagResourceCommand(tagResourceCommandParams);
    const response = await erClient.send(command);
    console.log("Successfully tagged the resource.");
  } catch (caught) {
    console.error(caught.message);
    throw caught;
  }
};
```

- API-Details finden Sie TagResource in der AWS SDK für JavaScript API-Referenz.

EventBridge Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  EventBridgeClient,
  PutEventsCommand,
} from "@aws-sdk/client-eventbridge";

export const putEvents = async (
  source = "eventbridge.integration.test",
  detailType = "greeting",
  resources = [],
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutEventsCommand({
      Entries: [
        {
          Detail: JSON.stringify({ greeting: "Hello there." }),
          DetailType: detailType,
```

```
          Resources: resources,
          Source: source,
        },
      ],
    }),
  );

  console.log("PutEvents response:");
  console.log(response);
  // PutEvents response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3d0df73d-dcea-4a23-ae0d-f5556a3ac109',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Entries: [ { EventId: '51620841-5af4-6402-d9bc-b77734991eb5' } ],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- API-Details finden Sie PutEvents in der AWS SDK für JavaScript API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { EventBridgeClient, PutRuleCommand } from "@aws-sdk/client-eventbridge";

export const putRule = async (
  ruleName = "some-rule",
  source = "some-source",
) => {
  const client = new EventBridgeClient({});

  const response = await client.send(
    new PutRuleCommand({
      Name: ruleName,
      EventPattern: JSON.stringify({ source: [source] }),
      State: "ENABLED",
      EventBusName: "default",
    }),
  );

  console.log("PutRule response:");
  console.log(response);
  // PutRule response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd7292ced-1544-421b-842f-596326bc7072',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   RuleArn: 'arn:aws:events:us-east-1:xxxxxxxxxxxx:rule/
EventBridgeTestRule-1696280037720'
  // }
  return response;
};
```

- API-Details finden Sie PutRule in der AWS SDK für JavaScript API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import {
  EventBridgeClient,
  PutTargetsCommand,
} from "@aws-sdk/client-eventbridge";

export const putTarget = async (
  existingRuleName = "some-rule",
  targetArn = "arn:aws:lambda:us-east-1:000000000000:function:test-func",
  uniqueId = Date.now().toString(),
) => {
  const client = new EventBridgeClient({});
  const response = await client.send(
    new PutTargetsCommand({
      Rule: existingRuleName,
      Targets: [
        {
          Arn: targetArn,
          Id: uniqueId,
        },
      ],
    }),
  );

  console.log("PutTargets response:");
  console.log(response);
  // PutTargets response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'f5b23b9a-2c17-45c1-ad5c-f926c3692e3d',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   FailedEntries: [],
  //   FailedEntryCount: 0
  // }

  return response;
};
```

- API-Details finden Sie PutTargets in der AWS SDK für JavaScript API-Referenz.

Szenarien

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Laufzeit-API. JavaScript In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Amazon Glacier-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Glacier Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVault`

Das folgende Codebeispiel zeigt die Verwendung`CreateVault`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Erstellen des Tresors.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateVault in der AWS SDK für JavaScript API-Referenz.

`UploadArchive`

Das folgende Codebeispiel zeigt die Verwendung`UploadArchive`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Hochladen des Archivs.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UploadArchive in der AWS SDK für JavaScript API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
```

```
  return JobNames;
};
```

- API-Details finden Sie ListJobs in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und starten Sie einen Crawler, der einen öffentlichen Amazon Simple Storage Service (Amazon S3) -Bucket crawlt und eine Metadaten-Datenbank generiert, die die gefundenen CSV-formatted Daten beschreibt.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
```

```
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
```

```
const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }

  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
```

```
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };
```

```
/**
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Erstellen und führen Sie einen Job aus, der CSV-Daten aus dem Amazon S3-Quell-Bucket extrahiert, sie durch Entfernen und Umbenennen von Feldern transformiert und die JSON-formatted Ausgabe in einen anderen Amazon S3-Bucket lädt.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
```

```
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
    case "ERROR":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
```

```
    case "SUCCEEDED":
      return;
    default:
      break;
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }} context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
```

```
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Listet Informationen über Auftragsausführungen auf und zeigt einige der transformierten Daten an.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};

/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-glue').GetJobRunCommandOutput>}
 getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-glue').GetJobRunsCommandOutput}
 getJobRuns
 */
```

```
/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
  };
```

Löscht alle Ressourcen, die von der Demo erstellt wurden.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
```

```
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
```

```
    * @type {{ selectedJobNames: string[] }}
    */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
```

```
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
/**
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };
```

```
const cleanUpCrawlerStep = async (context) => {
  log("Deleting crawler.");

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log("Crawler is already deleted.");
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCrawler`

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- API-Details finden Sie CreateCrawler in der AWS SDK für JavaScript API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- API-Details finden Sie CreateJob in der AWS SDK für JavaScript API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- API-Details finden Sie DeleteCrawler in der AWS SDK für JavaScript API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- API-Details finden Sie DeleteDatabase in der AWS SDK für JavaScript API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API-Details finden Sie DeleteJob in der AWS SDK für JavaScript API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});
```

```
  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- API-Details finden Sie DeleteTable in der AWS SDK für JavaScript API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetCrawler in der AWS SDK für JavaScript API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetDatabase in der AWS SDK für JavaScript API-Referenz.

`GetDatabases`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabases`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- API-Details finden Sie GetDatabases in der AWS SDK für JavaScript API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetJob in der AWS SDK für JavaScript API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetJobRun in der AWS SDK für JavaScript API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetJobRuns in der AWS SDK für JavaScript API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetTables in der AWS SDK für JavaScript API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- API-Details finden Sie ListJobs in der AWS SDK für JavaScript API-Referenz.

StartCrawler

Das folgende Codebeispiel zeigt die VerwendungStartCrawler.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- API-Details finden Sie StartCrawler in der AWS SDK für JavaScript API-Referenz.

StartJobRun

Das folgende Codebeispiel zeigt die VerwendungStartJobRun.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- API-Details finden Sie StartJobRun in der AWS SDK für JavaScript API-Referenz.

HealthImaging Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo HealthImaging

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von HealthImaging beginnen.

SDK für JavaScript (v3)

```
import {
  ListDatastoresCommand,
  MedicalImagingClient,
} from "@aws-sdk/client-medical-imaging";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new MedicalImagingClient({});

export const helloMedicalImaging = async () => {
  const command = new ListDatastoresCommand({});

  const { datastoreSummaries } = await client.send(command);
  console.log("Datastores: ");
  console.log(datastoreSummaries.map((item) => item.datastoreName).join("\n"));
  return datastoreSummaries;
};
```

- API-Details finden Sie ListDatastores in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktionen

`CopyImageSet`

Das folgende Codebeispiel zeigt die Verwendung`CopyImageSet`.

SDK für JavaScript (v3)

Hilfsfunktion zum Kopieren eines Imagesatzes.

```
import { CopyImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The source image set ID.
 * @param {string} sourceVersionId - The source version ID.
 * @param {string} destinationImageSetId - The optional ID of the destination image
 set.
 * @param {string} destinationVersionId - The optional version ID of the destination
 image set.
 * @param {boolean} force - Force the copy action.
 * @param {[string]} copySubsets - A subset of instance IDs to copy.
 */
export const copyImageSet = async (
  datastoreId = "xxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
  sourceVersionId = "1",
  destinationImageSetId = "",
  destinationVersionId = "",
  force = false,
  copySubsets = [],
) => {
  try {
    const params = {
      datastoreId: datastoreId,
      sourceImageSetId: imageSetId,
      copyImageSetInformation: {
        sourceImageSet: { latestVersionId: sourceVersionId },
      },
      force: force,
    };
    if (destinationImageSetId !== "" && destinationVersionId !== "") {
      params.copyImageSetInformation.destinationImageSet = {
        imageSetId: destinationImageSetId,
        latestVersionId: destinationVersionId,
      };
    }

    if (copySubsets.length > 0) {
```

```
      let copySubsetsJson;
      copySubsetsJson = {
        SchemaVersion: 1.1,
        Study: {
          Series: {
            imageSetId: {
              Instances: {},
            },
          },
        },
      };

      for (let i = 0; i < copySubsets.length; i++) {
        copySubsetsJson.Study.Series.imageSetId.Instances[copySubsets[i]] = {};
      }

      params.copyImageSetInformation.dicomCopies = copySubsetsJson;
    }

    const response = await medicalImagingClient.send(
      new CopyImageSetCommand(params),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //         httpStatusCode: 200,
    //         requestId: 'd9b219ce-cc48-4a44-a5b2-c5c3068f1ee8',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    //      },
    //       datastoreId: 'xxxxxxxxxxxxx',
    //       destinationImageSetProperties: {
    //             createdAt: 2023-09-27T19:46:21.824Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING',
    //             latestVersionId: '1',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //       },
    //       sourceImageSetProperties: {
```

```
    //             createdAt: 2023-09-22T14:49:26.427Z,
    //             imageSetArn: 'arn:aws:medical-imaging:us-
east-1:xxxxxxxxxxx:datastore/xxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxx',
    //             imageSetId: 'xxxxxxxxxxxxxxx',
    //             imageSetState: 'LOCKED',
    //             imageSetWorkflowStatus: 'COPYING_WITH_READ_ONLY_ACCESS',
    //             latestVersionId: '4',
    //             updatedAt: 2023-09-27T19:46:21.824Z
    //      }
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Kopieren Sie einen Imagesatz ohne Ziel.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
  );
```

Kopieren Sie einen Imagesatz mit Ziel.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    false,
  );
```

Kopieren Sie eine Teilmenge eines Imagesatzes mit einem Ziel und erzwingen Sie den Kopiervorgang.

```
  await copyImageSet(
    "12345678901234567890123456789012",
    "12345678901234567890123456789012",
    "1",
    "12345678901234567890123456789012",
    "1",
    true,
    ["12345678901234567890123456789012", "11223344556677889900112233445566"],
  );
```

- API-Details finden Sie CopyImageSet in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`CreateDatastore`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatastore`.

SDK für JavaScript (v3)

```
import { CreateDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreName - The name of the data store to create.
 */
export const createDatastore = async (datastoreName = "DATASTORE_NAME") => {
  const response = await medicalImagingClient.send(
    new CreateDatastoreCommand({ datastoreName: datastoreName }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: 'a71cd65f-2382-49bf-b682-f9209d8d399b',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //    datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //    datastoreStatus: 'CREATING'
  // }
  return response;
};
```

- API-Details finden Sie CreateDatastore in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatastore`.

SDK für JavaScript (v3)

```
import { DeleteDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store to delete.
 */
export const deleteDatastore = async (datastoreId = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new DeleteDatastoreCommand({ datastoreId }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //           httpStatusCode: 200,
```

```
  //           requestId: 'f5beb409-678d-48c9-9173-9a001ee1ebb1',
  //           extendedRequestId: undefined,
  //           cfId: undefined,
  //           attempts: 1,
  //           totalRetryDelay: 0
  //         },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     datastoreStatus: 'DELETING'
  // }

  return response;
};
```

- API-Details finden Sie DeleteDatastore in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteImageSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteImageSet`.

SDK für JavaScript (v3)

```
import { DeleteImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store ID.
 * @param {string} imageSetId - The image set ID.
 */
export const deleteImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new DeleteImageSetCommand({
      datastoreId: datastoreId,
```

```
      imageSetId: imageSetId,
    }),
  );
  console.log(response);
  // {
  //    '$metadata': {
  //        httpStatusCode: 200,
  //        requestId: '6267bbd2-eaa5-4a50-8ee8-8fddf535cf73',
  //        extendedRequestId: undefined,
  //        cfId: undefined,
  //        attempts: 1,
  //        totalRetryDelay: 0
  //     },
  //     datastoreId: 'xxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'LOCKED',
  //     imageSetWorkflowStatus: 'DELETING'
  // }
  return response;
};
```

- API-Details finden Sie DeleteImageSet in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`GetDICOMImportJob`.

SDK für JavaScript (v3)

```
import { GetDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} jobId - The ID of the import job.
```

```
 */
export const getDICOMImportJob = async (
  datastoreId = "xxxxxxxxxxxxxxxxxxxx",
  jobId = "xxxxxxxxxxxxxxxxxxxx",
) => {
  const response = await medicalImagingClient.send(
    new GetDICOMImportJobCommand({ datastoreId: datastoreId, jobId: jobId }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: 'a2637936-78ea-44e7-98b8-7a87d95dfaee',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobProperties: {
  //         dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-19T17:29:21.753Z,
  //             inputS3Uri: 's3://healthimaging-source/CTStudy/',
  //             jobId: ''xxxxxxxxxxxxxxxxxxxxxxxxx'',
  //             jobName: 'job_1',
  //             jobStatus: 'COMPLETED',
  //             outputS3Uri: 's3://health-imaging-dest/
ouput_ct/'xxxxxxxxxxxxxxxxxxxxxxxxx'-DicomImport-'xxxxxxxxxxxxxxxxxxxxxxxxx'/',
  //             submittedAt: 2023-09-19T17:27:25.143Z
  //     }
  // }

  return response;
};
```

- API-Details finden Sie GetDICOMImportJob in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

GetDatastore

Das folgende Codebeispiel zeigt die VerwendungGetDatastore.

SDK für JavaScript (v3)

```
import { GetDatastoreCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreID - The ID of the data store.
 */
export const getDatastore = async (datastoreID = "DATASTORE_ID") => {
  const response = await medicalImagingClient.send(
    new GetDatastoreCommand({ datastoreId: datastoreID }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '55ea7d2e-222c-4a6a-871e-4f591f40cadb',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //    },
  //   datastoreProperties: {
  //        createdAt: 2023-08-04T18:50:36.239Z,
  //         datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //         datastoreName: 'my_datastore',
  //         datastoreStatus: 'ACTIVE',
  //         updatedAt: 2023-08-04T18:50:36.239Z
  //   }
  // }
  return response.datastoreProperties;
};
```

- API-Details finden Sie GetDatastore in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt die Verwendung`GetImageFrame`.

SDK für JavaScript (v3)

```
import { GetImageFrameCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} imageFrameFileName - The name of the file for the HTJ2K-encoded
 image frame.
 * @param {string} datastoreID - The data store's ID.
 * @param {string} imageSetID - The image set's ID.
 * @param {string} imageFrameID - The image frame's ID.
 */
export const getImageFrame = async (
  imageFrameFileName = "image.jph",
  datastoreID = "DATASTORE_ID",
  imageSetID = "IMAGE_SET_ID",
  imageFrameID = "IMAGE_FRAME_ID",
) => {
  const response = await medicalImagingClient.send(
    new GetImageFrameCommand({
      datastoreId: datastoreID,
      imageSetId: imageSetID,
      imageFrameInformation: { imageFrameId: imageFrameID },
    }),
  );
  const buffer = await response.imageFrameBlob.transformToByteArray();
  writeFileSync(imageFrameFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'e4ab42a5-25a3-4377-873f-374ecf4380e1',
```

```
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     contentType: 'application/octet-stream',
  //     imageFrameBlob: <ref *1> IncomingMessage {}
  // }
  return response;
};
```

- API-Details finden Sie GetImageFrame in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSet`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSet`.

SDK für JavaScript (v3)

```
import { GetImageSetCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 * @param {string} imageSetVersion - The optional version of the image set.
 *
 */
export const getImageSet = async (
  datastoreId = "xxxxxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxxxxx",
  imageSetVersion = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imageSetId };
  if (imageSetVersion !== "") {
```

```
    params.imageSetVersion = imageSetVersion;
  }
  const response = await medicalImagingClient.send(
    new GetImageSetCommand(params),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '0615c161-410d-4d06-9d8c-6e1241bb0a5a',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     createdAt: 2023-09-22T14:49:26.427Z,
  //     datastoreId: 'xxxxxxxxxxxxxxx',
  //     imageSetArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxx/imageset/xxxxxxxxxxxxxxxxxxx',
  //     imageSetId: 'xxxxxxxxxxxxxxx',
  //     imageSetState: 'ACTIVE',
  //     imageSetWorkflowStatus: 'CREATED',
  //     updatedAt: 2023-09-22T14:49:26.427Z,
  //     versionId: '1'
  // }

  return response;
};
```

- API-Details finden Sie GetImageSet in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSetMetadata`.

SDK für JavaScript (v3)

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
import { GetImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";
import { writeFileSync } from "node:fs";

/**
 * @param {string} metadataFileName - The name of the file for the gzipped metadata.
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imagesetId - The ID of the image set.
 * @param {string} versionID - The optional version ID of the image set.
 */
export const getImageSetMetadata = async (
  metadataFileName = "metadata.json.gzip",
  datastoreId = "xxxxxxxxxxxxxx",
  imagesetId = "xxxxxxxxxxxxxx",
  versionID = "",
) => {
  const params = { datastoreId: datastoreId, imageSetId: imagesetId };

  if (versionID) {
    params.versionID = versionID;
  }

  const response = await medicalImagingClient.send(
    new GetImageSetMetadataCommand(params),
  );
  const buffer = await response.imageSetMetadataBlob.transformToByteArray();
  writeFileSync(metadataFileName, buffer);

  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '5219b274-30ff-4986-8cab-48753de3a599',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     contentType: 'application/json',
  //     contentEncoding: 'gzip',
```

```
  //     imageSetMetadataBlob: <ref *1> IncomingMessage {}
  // }

  return response;
};
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
  try {
    await getImageSetMetadata(
      "metadata.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
  try {
    await getImageSetMetadata(
      "metadata2.json.gzip",
      "12345678901234567890123456789012",
      "12345678901234567890123456789012",
      "1",
    );
  } catch (err) {
    console.log("Error", err);
  }
```

- API-Details finden Sie GetImageSetMetadata in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDICOMImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListDICOMImportJobs`.

SDK für JavaScript (v3)

```
import { paginateListDICOMImportJobs } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 */
export const listDICOMImportJobs = async (
  datastoreId = "xxxxxxxxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId: datastoreId };
  const paginator = paginateListDICOMImportJobs(paginatorConfig, commandParams);

  const jobSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    jobSummaries.push(...page.jobSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '3c20c66e-0797-446a-a1d8-91b742fd15a0',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     jobSummaries: [
  //         {
  //             dataAccessRoleArn: 'arn:aws:iam::xxxxxxxxxxxx:role/dicom_import',
  //             datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxx',
  //             endedAt: 2023-09-22T14:49:51.351Z,
```

```
  //             jobId: 'xxxxxxxxxxxxxxxxxxxxxxxx',
  //             jobName: 'test-1',
  //             jobStatus: 'COMPLETED',
  //             submittedAt: 2023-09-22T14:48:45.767Z
  // }
  // ]}

  return jobSummaries;
};
```

- API-Details finden Sie ListDICOMImportJobs in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListDatastores`.

SDK für JavaScript (v3)

```
import { paginateListDatastores } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

export const listDatastores = async () => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = {};
  const paginator = paginateListDatastores(paginatorConfig, commandParams);

  /**
   * @type {import("@aws-sdk/client-medical-imaging").DatastoreSummary[]}
   */
  const datastoreSummaries = [];
  for await (const page of paginator) {
```

```
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    datastoreSummaries.push(...page.datastoreSummaries);
    console.log(page);
  }
  // {
  //   '$metadata': {
  //       httpStatusCode: 200,
  //       requestId: '6aa99231-d9c2-4716-a46e-edb830116fa3',
  //       extendedRequestId: undefined,
  //       cfId: undefined,
  //       attempts: 1,
  //       totalRetryDelay: 0
  //   },
  //   datastoreSummaries: [
  //     {
  //       createdAt: 2023-08-04T18:49:54.429Z,
  //       datastoreArn: 'arn:aws:medical-imaging:us-east-1:xxxxxxxxx:datastore/
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //       datastoreName: 'my_datastore',
  //       datastoreStatus: 'ACTIVE',
  //       updatedAt: 2023-08-04T18:49:54.429Z
  //     }
  //     ...
  //   ]
  // }

  return datastoreSummaries;
};
```

- API-Details finden Sie ListDatastores in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListImageSetVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListImageSetVersions`.

SDK für JavaScript (v3)

```
import { paginateListImageSetVersions } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} imageSetId - The ID of the image set.
 */
export const listImageSetVersions = async (
  datastoreId = "xxxxxxxxxxxx",
  imageSetId = "xxxxxxxxxxxx",
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
  };

  const commandParams = { datastoreId, imageSetId };
  const paginator = paginateListImageSetVersions(
    paginatorConfig,
    commandParams,
  );

  const imageSetPropertiesList = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetPropertiesList.push(...page.imageSetPropertiesList);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '74590b37-a002-4827-83f2-3c590279c742',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetPropertiesList: [
  //         {
  //             ImageSetWorkflowStatus: 'CREATED',
  //             createdAt: 2023-09-22T14:49:26.427Z,
```

```
  //             imageSetId: 'xxxxxxxxxxxxxxxxxxxxxx',
  //             imageSetState: 'ACTIVE',
  //             versionId: '1'
  //         }]
  // }
  return imageSetPropertiesList;
};
```

- API-Details finden Sie ListImageSetVersions in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für JavaScript (v3)

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
```

```
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

- API-Details finden Sie ListTagsForResource in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt die Verwendung`SearchImageSets`.

SDK für JavaScript (v3)

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
import { paginateSearchImageSets } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The data store's ID.
 * @param { import('@aws-sdk/client-medical-imaging').SearchFilter[] } filters - The
 search criteria filters.
 * @param { import('@aws-sdk/client-medical-imaging').Sort } sort - The search
 criteria sort.
 */
export const searchImageSets = async (
  datastoreId = "xxxxxxxx",
  searchCriteria = {},
) => {
  const paginatorConfig = {
    client: medicalImagingClient,
    pageSize: 50,
```

```
  };

  const commandParams = {
    datastoreId: datastoreId,
    searchCriteria: searchCriteria,
  };

  const paginator = paginateSearchImageSets(paginatorConfig, commandParams);

  const imageSetsMetadataSummaries = [];
  for await (const page of paginator) {
    // Each page contains a list of `jobSummaries`. The list is truncated if is
 larger than `pageSize`.
    imageSetsMetadataSummaries.push(...page.imageSetsMetadataSummaries);
    console.log(page);
  }
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: 'f009ea9c-84ca-4749-b5b6-7164f00a5ada',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     imageSetsMetadataSummaries: [
  //         {
  //             DICOMTags: [Object],
  //             createdAt: "2023-09-19T16:59:40.551Z",
  //             imageSetId: '7f75e1b5c0f40eac2b24cf712f485f50',
  //             updatedAt: "2023-09-19T16:59:40.551Z",
  //             version: 1
  //         }]
  // }

  return imageSetsMetadataSummaries;
};
```

Anwendungsfall 1: EQUAL-Operator.

```
  const datastoreId = "12345678901234567890123456789012";
```

```
try {
  const searchCriteria = {
    filters: [
      {
        values: [{ DICOMPatientId: "1234567" }],
        operator: "EQUAL",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudyDate und DICOMStudyTime.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "19900101",
              DICOMStudyTime: "000000",
            },
          },
          {
            DICOMStudyDateAndTime: {
              DICOMStudyDate: "20230901",
              DICOMStudyTime: "000000",
            },
          },
        ],
        operator: "BETWEEN",
      },
    ],
  };
```

```
  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { createdAt: new Date("1985-04-12T23:20:50.52Z") },
          { createdAt: new Date() },
        ],
        operator: "BETWEEN",
      },
    ],
  };

  await searchImageSets(datastoreId, searchCriteria);
} catch (err) {
  console.error(err);
}
```

Anwendungsfall #4: EQUAL-Operator für DICOMSeriesInstanceUID und BETWEEN für updatedAt und Sortierung der Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
const datastoreId = "12345678901234567890123456789012";

try {
  const searchCriteria = {
    filters: [
      {
        values: [
          { updatedAt: new Date("1985-04-12T23:20:50.52Z") },
          { updatedAt: new Date() },
        ],
        operator: "BETWEEN",
```

```
        },
        {
          values: [
            {
              DICOMSeriesInstanceUID:
                "1.1.123.123456.1.12.1.1234567890.1234.12345678.123",
            },
          ],
          operator: "EQUAL",
        },
      ],
      sort: {
        sortOrder: "ASC",
        sortField: "updatedAt",
      },
    };

    await searchImageSets(datastoreId, searchCriteria);
  } catch (err) {
    console.error(err);
  }
```

- API-Details finden Sie in der API-Referenz. SearchImageSets AWS SDK für JavaScript

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartDICOMImportJob`.

SDK für JavaScript (v3)

```
import { StartDICOMImportJobCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} jobName - The name of the import job.
```

```
 * @param {string} datastoreId - The ID of the data store.
 * @param {string} dataAccessRoleArn - The Amazon Resource Name (ARN) of the role
 that grants permission.
 * @param {string} inputS3Uri - The URI of the S3 bucket containing the input files.
 * @param {string} outputS3Uri - The URI of the S3 bucket where the output files are
 stored.
 */
export const startDicomImportJob = async (
  jobName = "test-1",
  datastoreId = "12345678901234567890123456789012",
  dataAccessRoleArn = "arn:aws:iam::xxxxxxxxxxxx:role/ImportJobDataAccessRole",
  inputS3Uri = "s3://medical-imaging-dicom-input/dicom_input/",
  outputS3Uri = "s3://medical-imaging-output/job_output/",
) => {
  const response = await medicalImagingClient.send(
    new StartDICOMImportJobCommand({
      jobName: jobName,
      datastoreId: datastoreId,
      dataAccessRoleArn: dataAccessRoleArn,
      inputS3Uri: inputS3Uri,
      outputS3Uri: outputS3Uri,
    }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //     httpStatusCode: 200,
  //         requestId: '6e81d191-d46b-4e48-a08a-cdcc7e11eb79',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  // },
  //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
  //     jobStatus: 'SUBMITTED',
  //     submittedAt: 2023-09-22T14:48:45.767Z
  // }
  return response;
};
```

- API-Details finden Sie StartDICOMImportJob in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für JavaScript (v3)

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- API-Details finden Sie TagResource in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für JavaScript (v3)

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }
```

```
  return response;
};
```

- API-Details finden Sie UntagResource in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UpdateImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`UpdateImageSetMetadata`.

SDK für JavaScript (v3)

```
import { UpdateImageSetMetadataCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} datastoreId - The ID of the HealthImaging data store.
 * @param {string} imageSetId - The ID of the HealthImaging image set.
 * @param {string} latestVersionId - The ID of the HealthImaging image set version.
 * @param {{}} updateMetadata - The metadata to update.
 * @param {boolean} force - Force the update.
 */
export const updateImageSetMetadata = async (
  datastoreId = "xxxxxxxxxx",
  imageSetId = "xxxxxxxxxx",
  latestVersionId = "1",
  updateMetadata = "{}",
  force = false,
) => {
  try {
    const response = await medicalImagingClient.send(
      new UpdateImageSetMetadataCommand({
        datastoreId: datastoreId,
        imageSetId: imageSetId,
        latestVersionId: latestVersionId,
        updateImageSetMetadataUpdates: updateMetadata,
```

```
        force: force,
      }),
    );
    console.log(response);
    // {
    //     '$metadata': {
    //     httpStatusCode: 200,
    //         requestId: '7966e869-e311-4bff-92ec-56a61d3003ea',
    //         extendedRequestId: undefined,
    //         cfId: undefined,
    //         attempts: 1,
    //         totalRetryDelay: 0
    // },
    //     createdAt: 2023-09-22T14:49:26.427Z,
    //     datastoreId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetId: 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
    //     imageSetState: 'LOCKED',
    //     imageSetWorkflowStatus: 'UPDATING',
    //     latestVersionId: '4',
    //     updatedAt: 2023-09-27T19:41:43.494Z
    // }
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es und erzwingen Sie das Update.

```
    const insertAttributes = JSON.stringify({
      SchemaVersion: 1.1,
      Study: {
        DICOM: {
          StudyDescription: "CT CHEST",
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        updatableAttributes: new TextEncoder().encode(insertAttributes),
```

```
    },
  };

  await updateImageSetMetadata(
    datastoreID,
    imageSetID,
    versionID,
    updateMetadata,
    true,
  );
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
  // Attribute key and value must match the existing attribute.
  const remove_attribute = JSON.stringify({
    SchemaVersion: 1.1,
    Study: {
      DICOM: {
        StudyDescription: "CT CHEST",
      },
    },
  });

  const updateMetadata = {
    DICOMUpdates: {
      removableAttributes: new TextEncoder().encode(remove_attribute),
    },
  };

  await updateImageSetMetadata(
    datastoreID,
    imageSetID,
    versionID,
    updateMetadata,
  );
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
  const remove_instance = JSON.stringify({
    SchemaVersion: 1.1,
    Study: {
```

```
        Series: {
          "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
            Instances: {
              "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {},
            },
          },
        },
      },
    });

    const updateMetadata = {
      DICOMUpdates: {
        removableAttributes: new TextEncoder().encode(remove_instance),
      },
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
    );
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
    const updateMetadata = {
      revertToVersionId: "1",
    };

    await updateImageSetMetadata(
      datastoreID,
      imageSetID,
      versionID,
      updateMetadata,
    );
```

- API-Details finden Sie UpdateImageSetMetadata in der AWS SDK für JavaScript API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien

Erste Schritte mit Bildsätzen und Bild-Frames

Das folgende Codebeispiel zeigt, wie DICOM-Dateien importiert und Bildrahmen heruntergeladen werden. HealthImaging

Die Implementierung ist als Befehlszeilenanwendung strukturiert.

- Richten Sie Ressourcen für einen DICOM-Import ein.
- Importieren Sie DICOM-Dateien in einen Datenspeicher.
- Rufen Sie die Bildsatz-IDs für den Importauftrag ab.
- Rufen Sie die Bild-Frame-IDs für den Importauftrag ab.
- Laden Sie die Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.
- Bereinigen Sie die Ressourcen.

SDK für JavaScript (v3)

Orchestrieren Sie die Schritte (index.js).

```
import {
  parseScenarioArgs,
  Scenario,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  saveState,
  loadState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import {
  createStack,
  deployStack,
  getAccountId,
```

```
  getDatastoreName,
  getStackName,
  outputState,
  waitForStackCreation,
} from "./deploy-steps.js";
import {
  doCopy,
  selectDataset,
  copyDataset,
  outputCopiedObjects,
} from "./dataset-steps.js";
import {
  doImport,
  outputImportJobStatus,
  startDICOMImport,
  waitForImportJobCompletion,
} from "./import-steps.js";
import {
  getManifestFile,
  outputImageSetIds,
  parseManifestFile,
} from "./image-set-steps.js";
import {
  getImageSetMetadata,
  outputImageFrameIds,
} from "./image-frame-steps.js";
import { decodeAndVerifyImages, doVerify } from "./verify-steps.js";
import {
  confirmCleanup,
  deleteImageSets,
  deleteStack,
} from "./clean-up-steps.js";

const context = {};

const scenarios = {
  deploy: new Scenario(
    "Deploy Resources",
    [
      deployStack,
      getStackName,
      getDatastoreName,
      getAccountId,
      createStack,
```

```
      waitForStackCreation,
      outputState,
      saveState,
    ],
    context,
  ),
  demo: new Scenario(
    "Run Demo",
    [
      loadState,
      doCopy,
      selectDataset,
      copyDataset,
      outputCopiedObjects,
      doImport,
      startDICOMImport,
      waitForImportJobCompletion,
      outputImportJobStatus,
      getManifestFile,
      parseManifestFile,
      outputImageSetIds,
      getImageSetMetadata,
      outputImageFrameIds,
      doVerify,
      decodeAndVerifyImages,
      saveState,
    ],
    context,
  ),
  destroy: new Scenario(
    "Clean Up Resources",
    [loadState, confirmCleanup, deleteImageSets, deleteStack],
    context,
  ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Health Imaging Workflow",
    description:
      "Work with DICOM images using an AWS Health Imaging data store.",
    synopsis:
```

```
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Stellen Sie Ressourcen bereit (deploy-steps.js).

```
import fs from "node:fs/promises";
import path from "node:path";

import {
  CloudFormationClient,
  CreateStackCommand,
  DescribeStacksCommand,
} from "@aws-sdk/client-cloudformation";
import { STSClient, GetCallerIdentityCommand } from "@aws-sdk/client-sts";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const cfnClient = new CloudFormationClient({});
const stsClient = new STSClient({});

const __dirname = path.dirname(new URL(import.meta.url).pathname);
const cfnTemplatePath = path.join(
  __dirname,
  "../../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml",
);

export const deployStack = new ScenarioInput(
  "deployStack",
  "Do you want to deploy the CloudFormation stack?",
  { type: "confirm" },
);

export const getStackName = new ScenarioInput(
  "getStackName",
```

```
  "Enter a name for the CloudFormation stack:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getDatastoreName = new ScenarioInput(
  "getDatastoreName",
  "Enter a name for the HealthImaging datastore:",
  { type: "input", skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const getAccountId = new ScenarioAction(
  "getAccountId",
  async (/** @type {{}} */ state) => {
    const command = new GetCallerIdentityCommand({});
    const response = await stsClient.send(command);
    state.accountId = response.Account;
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const createStack = new ScenarioAction(
  "createStack",
  async (/** @type {{}} */ state) => {
    const stackName = state.getStackName;
    const datastoreName = state.getDatastoreName;
    const accountId = state.accountId;

    const command = new CreateStackCommand({
      StackName: stackName,
      TemplateBody: await fs.readFile(cfnTemplatePath, "utf8"),
      Capabilities: ["CAPABILITY_IAM"],
      Parameters: [
        {
          ParameterKey: "datastoreName",
          ParameterValue: datastoreName,
        },
        {
          ParameterKey: "userAccountID",
          ParameterValue: accountId,
        },
      ],
    });
```

```
    const response = await cfnClient.send(command);
    state.stackId = response.StackId;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);

export const waitForStackCreation = new ScenarioAction(
  "waitForStackCreation",
  async (/** @type {{}} */ state) => {
    const command = new DescribeStacksCommand({
      StackName: state.stackId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await cfnClient.send(command);
      const stack = response.Stacks?.find(
        (s) => s.StackName === state.getStackName,
      );
      if (!stack || stack.StackStatus === "CREATE_IN_PROGRESS") {
        throw new Error("Stack creation is still in progress");
      }
      if (stack.StackStatus === "CREATE_COMPLETE") {
        state.stackOutputs = stack.Outputs?.reduce((acc, output) => {
          acc[output.OutputKey] = output.OutputValue;
          return acc;
        }, {});
      } else {
        throw new Error(
          `Stack creation failed with status: ${stack.StackStatus}`,
        );
      }
    });
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.deployStack,
  },
);

export const outputState = new ScenarioOutput(
  "outputState",
  (/** @type {{}} */ state) => {
    /**
```

```
     * @type {{ stackOutputs: { DatastoreID: string, BucketName: string, RoleArn:
 string }}}
     */
    const { stackOutputs } = state;
    return `Stack creation completed. Output values:
Datastore ID: ${stackOutputs?.DatastoreID}
Bucket Name: ${stackOutputs?.BucketName}
Role ARN: ${stackOutputs?.RoleArn}
    `;
  },
  { skipWhen: (/** @type {{}} */ state) => !state.deployStack },
);
```

Kopieren Sie DICOM-Dateien (dataset-steps.js).

```
import {
  S3Client,
  CopyObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const s3Client = new S3Client({});

const datasetOptions = [
  {
    name: "CT of chest (2 images)",
    value: "00029d25-fb18-4d42-aaa5-a0897d1ac8f7",
  },
  {
    name: "CT of pelvis (57 images)",
    value: "00025d30-ef8f-4135-a35a-d83eff264fc1",
  },
  {
    name: "MRI of head (192 images)",
    value: "0002d261-8a5d-4e63-8e2e-0cbfac87b904",
  },
```

```
  {
    name: "MRI of breast (92 images)",
    value: "0002dd07-0b7f-4a68-a655-44461ca34096",
  },
];

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   doCopy: boolean
 * }}} State
 */

export const selectDataset = new ScenarioInput(
  "selectDataset",
  (state) => {
    if (!state.doCopy) {
      process.exit(0);
    }
    return "Select a DICOM dataset to import:";
  },
  {
    type: "select",
    choices: datasetOptions,
  },
);

export const doCopy = new ScenarioInput(
  "doCopy",
  "Do you want to copy images from the public dataset into your bucket?",
  {
    type: "confirm",
  },
);

export const copyDataset = new ScenarioAction(
  "copyDataset",
  async (/** @type { State } */ state) => {
    const inputBucket = state.stackOutputs.BucketName;
    const inputPrefix = "input/";
    const selectedDatasetId = state.selectDataset;

    const sourceBucket = "idc-open-data";
```

```
    const sourcePrefix = `${selectedDatasetId}`;

    const listObjectsCommand = new ListObjectsV2Command({
      Bucket: sourceBucket,
      Prefix: sourcePrefix,
    });

    const objects = await s3Client.send(listObjectsCommand);

    const copyPromises = objects.Contents.map((object) => {
      const sourceKey = object.Key;
      const destinationKey = `${inputPrefix}${sourceKey
        .split("/")
        .slice(1)
        .join("/")}`;

      const copyCommand = new CopyObjectCommand({
        Bucket: inputBucket,
        CopySource: `/${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });

      return s3Client.send(copyCommand);
    });

    const results = await Promise.all(copyPromises);
    state.copiedObjects = results.length;
  },
);

export const outputCopiedObjects = new ScenarioOutput(
  "outputCopiedObjects",
  (state) => `${state.copiedObjects} DICOM files were copied.`,
);
```

Starten Sie den Import in den Datenspeicher (import-steps.js).

```
import {
  MedicalImagingClient,
  StartDICOMImportJobCommand,
  GetDICOMImportJobCommand,
} from "@aws-sdk/client-medical-imaging";
```

```
import {
  ScenarioAction,
  ScenarioOutput,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }}} State
 */

export const doImport = new ScenarioInput(
  "doImport",
  "Do you want to import DICOM images into your datastore?",
  {
    type: "confirm",
    default: true,
  },
);

export const startDICOMImport = new ScenarioAction(
  "startDICOMImport",
  async (/** @type {State} */ state) => {
    if (!state.doImport) {
      process.exit(0);
    }
    const medicalImagingClient = new MedicalImagingClient({});
    const inputS3Uri = `s3://${state.stackOutputs.BucketName}/input/`;
    const outputS3Uri = `s3://${state.stackOutputs.BucketName}/output/`;

    const command = new StartDICOMImportJobCommand({
      dataAccessRoleArn: state.stackOutputs.RoleArn,
      datastoreId: state.stackOutputs.DatastoreID,
      inputS3Uri,
      outputS3Uri,
    });

    const response = await medicalImagingClient.send(command);
    state.importJobId = response.jobId;
```

```
  },
);

export const waitForImportJobCompletion = new ScenarioAction(
  "waitForImportJobCompletion",
  async (/** @type {State} */ state) => {
    const medicalImagingClient = new MedicalImagingClient({});
    const command = new GetDICOMImportJobCommand({
      datastoreId: state.stackOutputs.DatastoreID,
      jobId: state.importJobId,
    });

    await retry({ intervalInMs: 10000, maxRetries: 60 }, async () => {
      const response = await medicalImagingClient.send(command);
      const jobStatus = response.jobProperties?.jobStatus;
      if (!jobStatus || jobStatus === "IN_PROGRESS") {
        throw new Error("Import job is still in progress");
      }
      if (jobStatus === "COMPLETED") {
        state.importJobOutputS3Uri = response.jobProperties.outputS3Uri;
      } else {
        throw new Error(`Import job failed with status: ${jobStatus}`);
      }
    });
  },
);

export const outputImportJobStatus = new ScenarioOutput(
  "outputImportJobStatus",
  (state) =>
    `DICOM import job completed. Output location: ${state.importJobOutputS3Uri}`,
);
```

Rufen Sie die Imagesatz-IDs ab (image-set-steps.js -).

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, importJobId: string,
 * importJobOutputS3Uri: string,
 * imageSetIds: string[],
 * manifestContent: { jobSummary: { imageSetsSummary: { imageSetId: string }[] } }
 * }} State
 */

const s3Client = new S3Client({});

export const getManifestFile = new ScenarioAction(
  "getManifestFile",
  async (/** @type {State} */ state) => {
    const bucket = state.stackOutputs.BucketName;
    const prefix = `output/${state.stackOutputs.DatastoreID}-DicomImport-
${state.importJobId}/`;
    const key = `${prefix}job-output-manifest.json`;

    const command = new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    });

    const response = await s3Client.send(command);
    const manifestContent = await response.Body.transformToString();
    state.manifestContent = JSON.parse(manifestContent);
  },
);

export const parseManifestFile = new ScenarioAction(
  "parseManifestFile",
  (/** @type {State} */ state) => {
    const imageSetIds =
      state.manifestContent.jobSummary.imageSetsSummary.reduce((ids, next) => {
        return Object.assign({}, ids, {
          [next.imageSetId]: next.imageSetId,
        });
      }, {});
    state.imageSetIds = Object.keys(imageSetIds);
  },
```

```
);

export const outputImageSetIds = new ScenarioOutput(
  "outputImageSetIds",
  (/** @type {State} */ state) =>
    `The image sets created by this import job are: \n${state.imageSetIds
      .map((id) => `Image set: ${id}`)
      .join("\n")}`,
);
```

Rufen Sie die Image-Rahmen-IDs ab (image-frame-steps.js).

```
import {
  MedicalImagingClient,
  GetImageSetMetadataCommand,
} from "@aws-sdk/client-medical-imaging";
import { gunzip } from "node:zlib";
import { promisify } from "node:util";

import {
  ScenarioAction,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

const gunzipAsync = promisify(gunzip);

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */
```

```
/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetIds: string[] }} State
 */

const medicalImagingClient = new MedicalImagingClient({});
```

```
export const getImageSetMetadata = new ScenarioAction(
  "getImageSetMetadata",
  async (/** @type {State} */ state) => {
    const outputMetadata = [];

    for (const imageSetId of state.imageSetIds) {
      const command = new GetImageSetMetadataCommand({
        datastoreId: state.stackOutputs.DatastoreID,
        imageSetId,
      });

      const response = await medicalImagingClient.send(command);
      const compressedMetadataBlob =
        await response.imageSetMetadataBlob.transformToByteArray();
      const decompressedMetadata = await gunzipAsync(compressedMetadataBlob);
      const imageSetMetadata = JSON.parse(decompressedMetadata.toString());

      outputMetadata.push(imageSetMetadata);
    }

    state.imageSetMetadata = outputMetadata;
  },
);

export const outputImageFrameIds = new ScenarioOutput(
  "outputImageFrameIds",
  (/** @type {State & { imageSetMetadata: ImageSetMetadata[] }} */ state) => {
    let output = "";

    for (const metadata of state.imageSetMetadata) {
      const imageSetId = metadata.ImageSetID;
      /** @type {DICOMMetadata[]} */
      const instances = Object.values(metadata.Study.Series).flatMap(
        (series) => {
          return Object.values(series.Instances);
        },
      );
      const imageFrameIds = instances.flatMap((instance) =>
        instance.ImageFrames.map((frame) => frame.ID),
      );

      output += `Image set ID: ${imageSetId}\nImage frame IDs:\n
${imageFrameIds.join(
        "\n",
```

```
      )}\n\n`;
    }

    return output;
  },
);
```

Überprüfen Sie die Image-Rahmen (verify-steps.js). Die Bibliothek zur Überprüfung von AWS HealthImaging Pixeldaten wurde zur Überprüfung verwendet.

```
import { spawn } from "node:child_process";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
```

```
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

export const doVerify = new ScenarioInput(
  "doVerify",
  "Do you want to verify the imported images?",
  {
    type: "confirm",
    default: true,
  },
);

export const decodeAndVerifyImages = new ScenarioAction(
  "decodeAndVerifyImages",
```

```
 async (/** @type {State} */ state) => {
   if (!state.doVerify) {
     process.exit(0);
   }
   const verificationTool = "./pixel-data-verification/index.js";

   for (const metadata of state.imageSetMetadata) {
     const datastoreId = state.stackOutputs.DatastoreID;
     const imageSetId = metadata.ImageSetID;

     for (const [seriesInstanceUid, series] of Object.entries(
       metadata.Study.Series,
     )) {
       for (const [sopInstanceUid, _] of Object.entries(series.Instances)) {
         console.log(
           `Verifying image set ${imageSetId} with series ${seriesInstanceUid} and
sop ${sopInstanceUid}`,
         );
         const child = spawn(
           "node",
           [
             verificationTool,
             datastoreId,
             imageSetId,
             seriesInstanceUid,
             sopInstanceUid,
           ],
           { stdio: "inherit" },
         );

         await new Promise((resolve, reject) => {
           child.on("exit", (code) => {
             if (code === 0) {
               resolve();
             } else {
               reject(
                 new Error(
                   `Verification tool exited with code ${code} for image set
${imageSetId}`,
                 ),
               );
             }
           });
         });
```

```
        }
      }
    }
  },
);
```

Vernichten Sie Ressourcen (clean-up-steps.js).

```
import {
  CloudFormationClient,
  DeleteStackCommand,
} from "@aws-sdk/client-cloudformation";
import {
  MedicalImagingClient,
  DeleteImageSetCommand,
} from "@aws-sdk/client-medical-imaging";

import {
  ScenarioAction,
  ScenarioInput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * @typedef {Object} DICOMValueRepresentation
 * @property {string} name
 * @property {string} type
 * @property {string} value
 */

/**
 * @typedef {Object} ImageFrameInformation
 * @property {string} ID
 * @property {Array<{ Checksum: number, Height: number, Width: number }>}
 PixelDataChecksumFromBaseToFullResolution
 * @property {number} MinPixelValue
 * @property {number} MaxPixelValue
 * @property {number} FrameSizeInBytes
 */

/**
 * @typedef {Object} DICOMMetadata
 * @property {Object} DICOM
```

```
 * @property {DICOMValueRepresentation[]} DICOMVRs
 * @property {ImageFrameInformation[]} ImageFrames
 */

/**
 * @typedef {Object} Series
 * @property {{ [key: string]: DICOMMetadata }} Instances
 */

/**
 * @typedef {Object} Study
 * @property {Object} DICOM
 * @property {Series[]} Series
 */

/**
 * @typedef {Object} Patient
 * @property {Object} DICOM
 */

/**
 * @typedef {{
 *  SchemaVersion: string,
 *  DatastoreID: string,
 *  ImageSetID: string,
 *  Patient: Patient,
 *  Study: Study
 * }} ImageSetMetadata
 */

/**
 * @typedef {{ stackOutputs: {
 *   BucketName: string,
 *   DatastoreID: string,
 *   RoleArn: string
 * }, imageSetMetadata: ImageSetMetadata[] }} State
 */

const cfnClient = new CloudFormationClient({});
const medicalImagingClient = new MedicalImagingClient({});

export const confirmCleanup = new ScenarioInput(
  "confirmCleanup",
  "Do you want to delete the created resources?",
```

```
  { type: "confirm" },
);

export const deleteImageSets = new ScenarioAction(
  "deleteImageSets",
  async (/** @type {State} */ state) => {
    const datastoreId = state.stackOutputs.DatastoreID;

    for (const metadata of state.imageSetMetadata) {
      const command = new DeleteImageSetCommand({
        datastoreId,
        imageSetId: metadata.ImageSetID,
      });

      try {
        await medicalImagingClient.send(command);
        console.log(`Successfully deleted image set ${metadata.ImageSetID}`);
      } catch (e) {
        if (e instanceof Error) {
          if (e.name === "ConflictException") {
            console.log(`Image set ${metadata.ImageSetID} already deleted`);
          }
        }
      }
    }
  },
  {
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);

export const deleteStack = new ScenarioAction(
  "deleteStack",
  async (/** @type {State} */ state) => {
    const stackName = state.getStackName;

    const command = new DeleteStackCommand({
      StackName: stackName,
    });

    await cfnClient.send(command);
    console.log(`Stack ${stackName} deletion initiated`);
  },
  {
```

```
    skipWhen: (/** @type {{}} */ state) => !state.confirmCleanup,
  },
);
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Taggen eines Datenspeichers

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Datenspeicher taggen.

SDK für JavaScript (v3)

So taggen Sie einen Datenspeicher.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(datastoreArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

So listen Sie Tags für einen Datenspeicher auf.

```
  try {
    const datastoreArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
```

```
    const { tags } = await listTagsForResource(datastoreArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

So entfernen Sie Tags von einem Datenspeicher.

```
  try {
    const datastoreArn =
```

```
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(datastoreArn, keys);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Taggen eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Bildsatz taggen.

SDK für JavaScript (v3)

So taggen Sie einen Imagesatz.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const tags = {
      Deployment: "Development",
    };
    await tagResource(imagesetArn, tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
import { TagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
```

```
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {Record<string,string>} tags - The tags to add to the resource as JSON.
 *                      - For example: {"Deployment" : "Development"}
 */
export const tagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tags = {},
) => {
  const response = await medicalImagingClient.send(
    new TagResourceCommand({ resourceArn: resourceArn, tags: tags }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

So listen Sie Tags für einen Imagesatz auf.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const { tags } = await listTagsForResource(imagesetArn);
    console.log(tags);
  } catch (e) {
    console.log(e);
  }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
import { ListTagsForResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 */
export const listTagsForResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:abc:datastore/def/imageset/ghi",
) => {
  const response = await medicalImagingClient.send(
    new ListTagsForResourceCommand({ resourceArn: resourceArn }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //         httpStatusCode: 200,
  //         requestId: '008fc6d3-abec-4870-a155-20fa3631e645',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //     },
  //     tags: { Deployment: 'Development' }
  // }

  return response;
};
```

So entfernen Sie Tags von einem Imagesatz.

```
  try {
    const imagesetArn =
      "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
    const keys = ["Deployment"];
    await untagResource(imagesetArn, keys);
  } catch (e) {
    console.log(e);
```

```
  }
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
import { UntagResourceCommand } from "@aws-sdk/client-medical-imaging";
import { medicalImagingClient } from "../libs/medicalImagingClient.js";

/**
 * @param {string} resourceArn - The Amazon Resource Name (ARN) for the data store
 or image set.
 * @param {string[]} tagKeys - The keys of the tags to remove.
 */
export const untagResource = async (
  resourceArn = "arn:aws:medical-imaging:us-east-1:xxxxxx:datastore/xxxxx/imageset/
xxx",
  tagKeys = [],
) => {
  const response = await medicalImagingClient.send(
    new UntagResourceCommand({ resourceArn: resourceArn, tagKeys: tagKeys }),
  );
  console.log(response);
  // {
  //     '$metadata': {
  //        httpStatusCode: 204,
  //         requestId: '8a6de9a3-ec8e-47ef-8643-473518b19d45',
  //         extendedRequestId: undefined,
  //         cfId: undefined,
  //         attempts: 1,
  //         totalRetryDelay: 0
  //    }
  // }

  return response;
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit IAM Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von IAM beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      for (const policy of page.Policies) {
        console.log(`${policy.PolicyName}`);
        policyCount++;
      }
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- API-Details finden Sie ListPolicies in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
import {
  CreateUserCommand,
  GetUserCommand,
```

```
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
```

```
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service (AWS
 STS).
  // It's not best practice to use access keys. For more information, see https://
aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;
```

```
  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
```

```
        RoleName: roleName,
      }),
    ),
);

if (!Role) {
  throw new Error("Role not created");
}

// Create a policy that allows the user to list S3 buckets.
const { Policy: listBucketPolicy } = await iamClient.send(
  new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: ["s3:ListAllMyBuckets"],
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
```

```
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);
```

```
  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Anfügen einer Richtlinie.

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
```

```
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie AttachRolePolicy in der AWS SDK für JavaScript API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Zugriffsschlüssel.

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateAccessKey in der AWS SDK für JavaScript API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Konto-Alias.

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateAccountAlias in der AWS SDK für JavaScript API-Referenz.

`CreateGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie CreateGroup in der AWS SDK für JavaScript API-Referenz.

`CreateInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung`CreateInstanceProfile`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- API-Details finden Sie CreateInstanceProfile in der AWS SDK für JavaScript API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie die Richtlinie.

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreatePolicy in der AWS SDK für JavaScript API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie die Rolle.

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- API-Details finden Sie CreateRole in der AWS SDK für JavaScript API-Referenz.

`CreateSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung`CreateSAMLProvider`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
import { readFileSync } from "node:fs";
import * as path from "node:path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter CreateSAMLProvider in der API-Referenz für AWS SDK für JavaScript .

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer serviceverknüpften Rolle.

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-
that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/general/
latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in this
 account",
```

```
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    }
    throw caught;
  }
};
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK für JavaScript API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Benutzer.

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateUser in der AWS SDK für JavaScript API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Zugriffsschlüssel.

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAccessKey in der AWS SDK für JavaScript API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Konto-Alias.

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteAccountAlias in der AWS SDK für JavaScript API-Referenz.

`DeleteGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie DeleteGroup in der AWS SDK für JavaScript API-Referenz.

`DeleteInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung`DeleteInstanceProfile`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    const client = new IAMClient({});
    await client.send(
      new DeleteInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
```

- API-Details finden Sie DeleteInstanceProfile in der AWS SDK für JavaScript API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Richtlinie.

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- API-Details finden Sie DeletePolicy in der AWS SDK für JavaScript API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Rolle.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API-Details finden Sie DeleteRole in der AWS SDK für JavaScript API-Referenz.

`DeleteRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRolePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- API-Details finden Sie DeleteRolePolicy in der AWS SDK für JavaScript API-Referenz.

`DeleteSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSAMLProvider`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
```

```
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter DeleteSAMLProvider in der API-Referenz für AWS SDK für JavaScript .

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Serverzertifikat.

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });
```

```
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteServerCertificate in der AWS SDK für JavaScript API-Referenz.

`DeleteServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServiceLinkedRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API-Details finden Sie DeleteServiceLinkedRole in der AWS SDK für JavaScript API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Benutzer.

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteUser in der AWS SDK für JavaScript API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Trennen Sie die Richtlinie.

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DetachRolePolicy in der AWS SDK für JavaScript API-Referenz.

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung`GetAccessKeyLastUsed`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie den Zugriffsschlüssel ab.

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });

  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetAccessKeyLastUsed in der AWS SDK für JavaScript API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Passwort-Richtlinie des Kontos ab.

```
import {
  GetAccountPasswordPolicyCommand,
```

```
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK für JavaScript API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Richtlinie ab.

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
```

```
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetPolicy in der AWS SDK für JavaScript API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Rolle ab.

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetRole in der AWS SDK für JavaScript API-Referenz.

`GetServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetServerCertificate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Serverzertifikat ab.

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetServerCertificate in der AWS SDK für JavaScript API-Referenz.

`GetServiceLinkedRoleDeletionStatus`

Das folgende Codebeispiel zeigt die Verwendung`GetServiceLinkedRoleDeletionStatus`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- API-Details finden Sie GetServiceLinkedRoleDeletionStatus in der AWS SDK für JavaScript API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Zugriffsschlüssel auf.

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
```

```
      );
    } else {
      break;
    }
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListAccessKeys in der AWS SDK für JavaScript API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Konto-Aliase auf.

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
```

```
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListAccountAliases in der AWS SDK für JavaScript API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Richtlinien auf, die an eine Rolle angefügt sind.

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK für JavaScript API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Gruppen auf.

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
```

```
}
```

- API-Details finden Sie ListGroups in der AWS SDK für JavaScript API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet Sie die Richtlinien auf.

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services account.
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
```

```
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- API-Details finden Sie ListPolicies in der AWS SDK für JavaScript API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet Sie die Richtlinien auf.

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
```

```
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API-Details finden Sie ListRolePolicies in der AWS SDK für JavaScript API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Rollen auf.

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
```

```
      break;
    }
  }
}
```

- API-Details finden Sie ListRoles in der AWS SDK für JavaScript API-Referenz.

`Listet SAML-Anbieter auf`

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die SAML-Anbieter auf.

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für JavaScript - API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Zertifikate auf.

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to simplify
 this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
    } else {
      break;
    }
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListServerCertificates in der AWS SDK für JavaScript API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Benutzer auf.

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);

  for (const { UserName, CreateDate } of response.Users) {
    console.log(`${UserName} created on: ${CreateDate}`);
  }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListUsers in der AWS SDK für JavaScript API-Referenz.

`PutRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutRolePolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::amzn-s3-demo-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie PutRolePolicy in der AWS SDK für JavaScript API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie den Zugriffsschlüssel.

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateAccessKey in der AWS SDK für JavaScript API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateServerCertificate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie ein Serverzertifikat.

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
```

```
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateServerCertificate in der AWS SDK für JavaScript API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie den Benutzer.

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });
```

```
  return client.send(command);
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie UpdateUser in der AWS SDK für JavaScript API-Referenz.

`UploadServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UploadServerCertificate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "node:path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command, or
 the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
```

```
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });

  return client.send(command);
};
```

- API-Details finden Sie UploadServerCertificate in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
```

```
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-
v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";
```

```
import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
```

```
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
```

```
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item'][] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
```

```
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
```

```
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
```

```
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
```

```
    new ScenarioOutput(
      "addedRoleToInstanceProfile",
      MESSAGES.addedRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
    ),
    ...initParamsSteps,
    new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
    new ScenarioAction("createLaunchTemplate", async () => {
      const ssmClient = new SSMClient({});
      const { Parameter } = await ssmClient.send(
        new GetParameterCommand({
          Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        }),
      );
      const ec2Client = new EC2Client({});
      await ec2Client.send(
        new CreateLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
          LaunchTemplateData: {
            InstanceType: "t3.micro",
            ImageId: Parameter.Value,
            IamInstanceProfile: { Name: NAMES.instanceProfileName },
            UserData: readFileSync(
              join(RESOURCES_PATH, "server_startup_script.sh"),
            ).toString("base64"),
            KeyName: NAMES.keyPairName,
          },
        }),
      );
    }),
    new ScenarioOutput(
      "createdLaunchTemplate",
      MESSAGES.createdLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      ),
    ),
    new ScenarioOutput(
      "creatingAutoScalingGroup",
      MESSAGES.creatingAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      ),
```

```
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
```

```
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
```

```
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
```

```
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
```

```
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}}
state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
```

```
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
```

```
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";
```

```
import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
```

```
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
```

```
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);
```

```
const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
```

```
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-auto-scaling').Instance }}
state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
```

```
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-ssm').InstanceInformation}}
state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
```

```
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
```

```
    new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
    healthCheckLoop,
    loadBalancerLoop,
    new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
      type: "confirm",
    }),
    new ScenarioAction("failOpenExit", (state) => {
      if (!state.failOpenConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction("failOpen", () => {
      const client = new SSMClient({});
      return client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: `fake-table-${Date.now()}`,
          Overwrite: true,
          Type: "String",
        }),
      );
    }),
    new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
    healthCheckLoop,
    loadBalancerLoop,
    new ScenarioInput(
      "resetTableConfirmation",
      MESSAGES.demoResetTableConfirmation,
      { type: "confirm" },
    ),
    new ScenarioAction("resetTableExit", (state) => {
      if (!state.resetTableConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction("resetTable", async () => {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: NAMES.tableName,
          Overwrite: true,
          Type: "String",
        }),
```

```
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
```

```
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
```

```
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
```

```
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
```

```
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
```

```
      NAMES.instancePolicyName,
    );
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
```

```
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
```

```
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
```

```
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
```

```
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
```

```
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
```

```
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
```

```
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
```

```
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
```

```
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

AWS IoT SiteWise Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT SiteWise.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT SiteWise

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT SiteWise beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
```

```
  paginateListAssetModels,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new IoTSiteWiseClient();
  const listAssetModelsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
    for await (const page of paginator) {
      listAssetModelsPaginated.push(...page.assetModelSummaries);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
    throw caught;
  }
  for (const { name, creationDate } of listAssetModelsPaginated) {
    console.log(`${name} - ${creationDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Details finden Sie ListAssetModels in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein AWS IoT SiteWise Asset-Modell.

- Erstellen Sie ein AWS IoT SiteWise Asset.
- Rufen Sie die Eigenschafts-ID-Werte ab.
- Daten an ein AWS IoT SiteWise Asset senden.
- Ruft den Wert der AWS IoT SiteWise Asset-Eigenschaft ab.
- Erstellen Sie ein AWS IoT SiteWise Portal.
- Erstellen Sie ein AWS IoT SiteWise Gateway.
- Beschreiben Sie das AWS IoT SiteWise Gateway.
- Löschen Sie die AWS IoT SiteWise Vermögenswerte.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
```

```
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
```

```
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
 that makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes. It is designed to help industrial and manufacturing
 organizations collect data from their equipment and processes, and use that data to
 make informed decisions about their operations.
One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial equipment and systems, including programmable logic controllers
 (PLCs), sensors, and other industrial devices. It can collect data from these
 devices and organize it into a unified data model, making it easier to analyze and
 gain insights from the data. AWS IoT SiteWise also provides tools for visualizing
 the data, setting up alarms and alerts, and generating reports.
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data. It can collect and store data from thousands of devices and
 process millions of data points per second, making it suitable for large-scale
 industrial operations. Additionally, AWS IoT SiteWise is designed to be secure
 and compliant, with features like role-based access controls, data encryption,
 and integration with other AWS services for additional security and compliance
 features.

Let's get started...`,
  { header: true },
);

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "This scenario uses AWS CloudFormation to create an IAM role that is required for
 this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-
template.yml`,
        "utf8",
      );
```

```
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
  `1. Create an AWS SiteWise Asset Model
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment, processes, and systems, that exist in an industrial environment.
 This model provides a structured and hierarchical representation of these assets,
 allowing users to define the relationships and properties of each asset.

This scenario creates two asset model properties: temperature and humidity.`,
);

const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
```

```
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
            {
              name: "Humidity",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
          ],
        }),
      );
      state.assetModel.assetModelId = assetModelResponse.assetModelId;
      console.log(
        `Asset Model successfully created. Asset Model ID:
 ${state.assetModel.assetModelId}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceAlreadyExistsException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} already exists.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
```

```
The IoT SiteWise model that we just created defines the structure and metadata for
 your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});

const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        }),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values

To send data to an asset, we need to get the property ID values. In this scenario,
 we access the temperature and humidity property ID values.`,
);
```

```
const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      for (const retrieveResponseKey in
 retrieveResponse.assetModelPropertySummaries) {
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Humidity"
        ) {
          state.propertyIds.Humidity =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Temperature"
        ) {
          state.propertyIds.Temperature =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
      }
      console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
      console.log(
        `The Temperature propertyId is ${state.propertyIds.Temperature}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem retrieving the properties: ${caught.message}`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
```

```
  },
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
  "displaySendDataToIoTSiteWiseAsset",
  `4. Send data to an AWS IoT SiteWise Asset

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple
 sources, normalize the data into a standard format, and store it in a centralized
 location. This makes it easier to analyze and gain insights from the data.

In this example, we generate sample temperature and humidity data and send it to the
 AWS IoT SiteWise asset.`,
);

const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(
  "sdkSendDataToIoTSiteWiseAsset",
  async (state) => {
    try {
      const sendResponse = await state.iotSiteWiseClient.send(
        new BatchPutAssetPropertyValueCommand({
          entries: [
            {
              entryId: "entry-3",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Humidity,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.humidity,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
            {
              entryId: "entry-4",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Temperature,
              propertyValues: [
                {
                  value: {
```

```
                    doubleValue: state.sampleData.temperature,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
          ],
        }),
      );
      console.log("The data was sent successfully.");
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data from connected equipment and sensors. One of the key benefits of
 reading an IoT SiteWise property is the ability to gain valuable insights from your
 industrial data.`,
);

const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (/** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
```

```
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
        }),
      );
      console.log(
        `The property value for Temperature is
 ${temperatureResponse.propertyValue.value.doubleValue}`,
      );
      console.log(
        `The property value for Humidity is
 ${humidityResponse.propertyValue.value.doubleValue}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources, such as sensors, equipment, and control systems, into a centralized
 platform.`,
);

const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (/** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
```

```
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
        `Portal created successfully. Portal ID ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribePortal = new ScenarioOutput(
  "displayDescribePortal",
  `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`,
);

const sdkDescribePortal = new ScenarioAction(
  "sdkDescribePortal",
  async (/** @type {State} */ state) => {
    try {
      const describePortalResponse = await state.iotSiteWiseClient.send(
        new DescribePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
```

```
);

const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
  "displayCreateIoTSiteWiseGateway",
  `8. Create an IoT SiteWise Gateway

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors, and
 the cloud-based IoT SiteWise service. It is responsible for securely collecting,
 processing, and transmitting data from various industrial assets to the IoT
 SiteWise platform, enabling real-time monitoring, analysis, and optimization of
 industrial operations.`,
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        }),
      );
      console.log(
        `Gateway creation completed successfully. ID is
 ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);
```

```
const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform: ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
```

```
  "displayConfirmDeleteResources",
  async (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
    return "The resources will not be deleted. Please delete them manually to avoid
 charges.";
  },
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    await wait(10); // Give the portal status time to catch up.
    try {
      await state.iotSiteWiseClient.send(
        new DeletePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(
        `Portal ${state.portal.portalName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
      } else {
        console.log(`When trying to delete the portal: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log(
        `Gateway ${state.gateway.gatewayName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
      } else {
```

```
        console.log(`When trying to delete the gateway: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetCommand({
          assetId: state.asset.assetId,
        }),
      );
      await wait(5); // Allow the delete to finish.
      console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
      } else {
        console.log(`When deleting the asset: ${caught.message}`);
      }
    }

    await wait(30); // Allow asset deletion to finish.
    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetModelCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      console.log(
        `Asset Model ${state.assetModel.assetModelName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} was not found.`,
        );
      } else {
        console.log(`When deleting the asset model: ${caught.message}`);
      }
    }

    try {
      await state.cloudFormationClient.send(
        new DeleteStackCommand({
          StackName: stackName,
```

```
        }),
      );
      await waitUntilStackDeleteComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the resources
 manually.`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
```

```
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
```

```
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Aktionen

`BatchPutAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchPutAssetPropertyValue`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  BatchPutAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Batch put asset property values.
 * @param {{ entries : array }}
 */
export const main = async ({ entries }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new BatchPutAssetPropertyValueCommand({
        entries: entries,
      }),
    );
    console.log("Asset properties batch put successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(`${caught.message}. A resource could not be found.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie BatchPutAssetPropertyValue in der AWS SDK für JavaScript API-Referenz.

`CreateAsset`

Das folgende Codebeispiel zeigt die Verwendung`CreateAsset`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetCommand({
        assetName: assetName, // The name to give the Asset.
        assetModelId: assetModelId, // The ID of the asset model from which to
 create the asset.
      }),
    );
    console.log("Asset created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- API-Details finden Sie CreateAsset in der AWS SDK für JavaScript API-Referenz.

`CreateAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`CreateAssetModel`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset Model.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetModelName, assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateAssetModelCommand({
        assetModelName: assetModelName, // The name to give the Asset Model.
      }),
    );
    console.log("Asset model created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the asset model.`,
```

```
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateAssetModel in der AWS SDK für JavaScript API-Referenz.

`CreateGateway`

Das folgende Codebeispiel zeigt die Verwendung`CreateGateway`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create a Gateway.
 * @param {{  }}
 */
export const main = async ({ gatewayName }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateGatewayCommand({
        gatewayName: gatewayName, // The name to give the created Gateway.
      }),
    );
    console.log("Gateway created successfully.");
    return result;
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the Gateway.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateGateway in der AWS SDK für JavaScript API-Referenz.

`DeleteAsset`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAsset`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset.
 * @param {{ assetId : string }}
 */
export const main = async ({ assetId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetCommand({
        assetId: assetId, // The model id to delete.
```

```
      }),
    );
    console.log("Asset deleted successfully.");
    return { assetDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteAsset in der AWS SDK für JavaScript API-Referenz.

`DeleteAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAssetModel`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
```

```
  try {
    await client.send(
      new DeleteAssetModelCommand({
        assetModelId: assetModelId, // The model id to delete.
      }),
    );
    console.log("Asset model deleted successfully.");
    return { assetModelDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset model.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteAssetModel in der AWS SDK für JavaScript API-Referenz.

`DeleteGateway`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGateway`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DeleteGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
```

```
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway deleted successfully.");
    return { gatewayDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the Gateway
 Id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteGateway in der AWS SDK für JavaScript API-Referenz.

`DescribeAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAssetModel`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DescribeAssetModelCommand,
  IoTSiteWiseClient,
```

```
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { assetModelDescription } = await client.send(
      new DescribeAssetModelCommand({
        assetModelId: assetModelId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset model information retrieved successfully.");
    return { assetModelDescription: assetModelDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the
 asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeAssetModel in der AWS SDK für JavaScript API-Referenz.

`DescribeGateway`

Das folgende Codebeispiel zeigt die Verwendung`DescribeGateway`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  DescribeGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { gatewayDescription } = await client.send(
      new DescribeGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway information retrieved successfully.");
    return { gatewayDescription: gatewayDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the Gateway
 Id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DescribeGateway in der AWS SDK für JavaScript API-Referenz.

GetAssetPropertyValue

Das folgende Codebeispiel zeigt die Verwendung`GetAssetPropertyValue`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset property value.
 * @param {{ entryId : string }}
 */
export const main = async ({ entryId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new GetAssetPropertyValueCommand({
        entryId: entryId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset property information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset property entry could not be found. Please
 check the entry id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie GetAssetPropertyValue in der AWS SDK für JavaScript API-Referenz.

`ListAssetModels`

Das folgende Codebeispiel zeigt die Verwendung`ListAssetModels`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ListAssetModelsCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ assetModelTypes : array }}
 */
export const main = async ({ assetModelTypes = [] }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new ListAssetModelsCommand({
        assetModelTypes: assetModelTypes, // The model types to list
      }),
    );
    console.log("Asset model types retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem listing the asset model types.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie ListAssetModels in der AWS SDK für JavaScript API-Referenz.

Kinesis-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Kinesis Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`PutRecords`

Das folgende Codebeispiel zeigt die Verwendung`PutRecords`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { PutRecordsCommand, KinesisClient } from "@aws-sdk/client-kinesis";

/**
 * Put multiple records into a Kinesis stream.
```

```
 * @param {{ streamArn: string }} config
 */
export const main = async ({ streamArn }) => {
  const client = new KinesisClient({});
  try {
    await client.send(
      new PutRecordsCommand({
        StreamARN: streamArn,
        Records: [
          {
            Data: new Uint8Array(),
            /**
             * Determines which shard in the stream the data record is assigned to.
             * Partition keys are Unicode strings with a maximum length limit of 256
             * characters for each key. Amazon Kinesis Data Streams uses the
 partition
             * key as input to a hash function that maps the partition key and
             * associated data to a specific shard.
             */
            PartitionKey: "TEST_KEY",
          },
          {
            Data: new Uint8Array(),
            PartitionKey: "TEST_KEY",
          },
        ],
      }),
    );
  } catch (caught) {
    if (caught instanceof Error) {
      //
    } else {
      throw caught;
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    streamArn: {
```

```
      type: "string",
      description: "The ARN of the stream.",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- API-Details finden Sie PutRecords in der AWS SDK für JavaScript API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Kinesis-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
```

```
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Es wird ein Kinesis-Ereignis mit Lambda ausgeführt. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
```

```
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
```

```
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Meldung von Kinesis-Batch-Item-Fehlern mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
```

```
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Lambda-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Lambda Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { LambdaClient, paginateListFunctions } from "@aws-sdk/client-lambda";

const client = new LambdaClient({});

export const helloLambda = async () => {
  const paginator = paginateListFunctions({ client }, {});
  const functions = [];
```

```
  for await (const page of paginator) {
    const funcNames = page.Functions.map((f) => f.FunctionName);
    functions.push(...funcNames);
  }

  console.log("Functions:");
  console.log(functions.join("\n"));
  return functions;
};
```

- API-Details finden Sie ListFunctions in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine AWS Identity and Access Management (IAM-) Rolle, die Lambda die Berechtigung erteilt, in Protokolle zu schreiben.

```
    logger.log(`Creating role (${NAME_ROLE_LAMBDA})...`);
    const response = await createRole(NAME_ROLE_LAMBDA);

import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

Erstellen Sie eine Lambda-Funktion und laden Sie Handlercode hoch.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

Aktualisieren Sie den Funktionscode und konfigurieren Sie seine Lambda-Umgebung mit einer Umgebungsvariablen.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
```

```
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

Listen Sie die Funktionen für Ihr Konto auf.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Löschen Sie die IAM-Rolle für Ihre Lambda-Funktion.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};

/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.

- CreateFunction
- DeleteFunction
- GetFunction
- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFunction`

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const createFunction = async (funcName, roleArn) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${funcName}.zip`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- API-Details finden Sie CreateFunction in der AWS SDK für JavaScript API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * @param {string} funcName
 */
const deleteFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- API-Details finden Sie DeleteFunction in der AWS SDK für JavaScript API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const getFunction = (funcName) => {
  const client = new LambdaClient({});
  const command = new GetFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- API-Details finden Sie GetFunction in der AWS SDK für JavaScript API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const invoke = async (funcName, payload) => {
  const client = new LambdaClient({});
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für JavaScript -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const listFunctions = () => {
  const client = new LambdaClient({});
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- API-Details finden Sie ListFunctions in der AWS SDK für JavaScript API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = new LambdaClient({});
  const code = await readFile(`${dirname}../functions/${newFunc}.zip`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
```

```
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für JavaScript API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
const updateFunctionConfiguration = async (funcName) => {
  const client = new LambdaClient({});
  const config = readFileSync(`${dirname}../functions/config.json`).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  const result = await client.send(command);
  await waitForFunctionUpdated({ FunctionName: funcName });
  return result;
};
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für JavaScript API-Referenz.

Szenarien

Bestätigen Sie automatisch bekannte Benutzer mit einer Lambda-Funktion

Die folgenden Codebeispiele zeigen, wie bekannte Amazon-Cognito-Benutzer mit einer Lambda-Funktion automatisch bestätigt werden können.

- Konfigurieren Sie einen Benutzerpool, um eine Lambda-Funktion für den `PreSignUp`-Trigger aufzurufen.
- Registrieren eines Benutzers bei Amazon Cognito.
- Die Lambda-Funktion scannt eine DynamoDB-Tabelle und bestätigt automatisch bekannte Benutzer.
- Melden Sie sich als neuer Benutzer an und bereinigen Sie dann die Ressourcen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie einen interaktiven „Szenario“-Lauf. Die JavaScript (v3) -Beispiele teilen sich einen Szenario-Runner, um komplexe Beispiele zu rationalisieren. Der komplette Quellcode ist aktiviert GitHub.

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
```

```
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

Dieses Szenario demonstriert die automatische Bestätigung eines bekannten Benutzers. Es orchestriert die Beispielschritte.

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
```

```
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
```

```
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
```

```
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });
```

```
    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
```

```
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
```

```
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }

    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
```

```
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with: ${state.token.slice(0,
 11)}`,
  { skipWhen: skipWhenErrors },
);

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
```

```
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

Dies sind Schritte, die mit anderen Szenarien geteilt werden.

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
```

```
  { skipWhen: skipWhenErrors },
);
```

Ein Handler für den `PreSignUp`-Trigger mit einer Lambda-Funktion.

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user '${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
```

```
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
 '${storedUserInfo.UserName}' does not match supplied UserName '${event.userName}'.
 Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
 ${storedUserInfo.UserName}. User is confirmed.`,
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

Modul für CloudWatch Logs-Aktionen.

```
import {
```

```
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[] |
 null, unknown]>}
```

```
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für Amazon-Cognito-Aktionen.

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
```

```
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
```

```
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password: string }}
 config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
```

```
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

Modul für DynamoDB-Aktionen.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
```

```
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string, unknown>[] }}
 config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie das innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);
```

```
  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
```

```
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Aufrufen einer Lambda-Funktion von einem Browser aus

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion von einem Browser aus aufgerufen wird.

SDK für JavaScript (v3)

Sie können eine browserbasierte Anwendung erstellen, die eine AWS Lambda Funktion zum Aktualisieren einer Amazon DynamoDB-Tabelle mit Benutzerauswahlen verwendet. Diese App verwendet v3. AWS SDK für JavaScript

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Laufzeit-API. JavaScript In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
```

```
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that the
 DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!


async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
```

```
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number; body:
 string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
```

```
    };
};
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Kinesis-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      throw err;
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Es wird ein Kinesis-Ereignis mit Lambda ausgeführt. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});

export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      throw err;
    }
    logger.info(`Successfully processed ${event.Records.length} records.`);
  }
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
```

```
  return data;
}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines DynamoDB-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
        logDynamoDBRecord(record);
    });
};

const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Es wird ein DynamoDB-Ereignis unter Verwendung von Lambda konsumiert. TypeScript

```
export const handler = async (event, context) => {
    console.log(JSON.stringify(event, null, 2));
    event.Records.forEach(record => {
```

```
        logDynamoDBRecord(record);
    });
}
const logDynamoDBRecord = (record) => {
    console.log(record.eventID);
    console.log(record.eventName);
    console.log(`DynamoDB Record: ${JSON.stringify(record.dynamodb)}`);
};
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Amazon DocumentDB-Ereignisses unter Verwendung von Lambda. JavaScript

```
console.log('Loading function');
exports.handler = async (event, context) => {
    event.events.forEach(record => {
        logDocumentDBEvent(record);
    });
    return 'OK';
};

const logDocumentDBEvent = (record) => {
    console.log('Operation type: ' + record.event.operationType);
    console.log('db: ' + record.event.ns.db);
    console.log('collection: ' + record.event.ns.coll);
    console.log('Full document:', JSON.stringify(record.event.fullDocument, null,
 2));
};
```

Ein Amazon DocumentDB-Ereignis mit Lambda wird ausgeführt TypeScript

```
import { DocumentDBEventRecord, DocumentDBEventSubscriptionContext } from 'aws-
lambda';

console.log('Loading function');

export const handler = async (
  event: DocumentDBEventSubscriptionContext,
  context: any
): Promise<string> => {
  event.events.forEach((record: DocumentDBEventRecord) => {
    logDocumentDBEvent(record);
  });
  return 'OK';
};

const logDocumentDBEvent = (record: DocumentDBEventRecord): void => {
  console.log('Operation type: ' + record.event.operationType);
  console.log('db: ' + record.event.ns.db);
  console.log('collection: ' + record.event.ns.coll);
  console.log('Full document:', JSON.stringify(record.event.fullDocument, null, 2));
};
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Amazon MSK-Ereignisses unter Verwendung von Lambda. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
        console.log('Message:', msg)
      })
    }
}
```

Es wird ein Amazon MSK-Ereignis mit Lambda ausgeführt. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
```

```
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines S3-Ereignisses mit Lambda. JavaScript

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
```

```
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Verbrauch eines S3-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
```

```
    console.log(message);
    throw new Error(message);
  }
};
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines SNS-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Verarbeiten eines SNS-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines SQS-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Verarbeiten eines SQS-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
```

```
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Javascript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    try {
      console.log(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
```

```
      console.log(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      console.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  console.log(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(payload) {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Meldung von Kinesis-Batch-Item-Fehlern mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  KinesisStreamEvent,
  Context,
  KinesisStreamHandler,
  KinesisStreamRecordPayload,
  KinesisStreamBatchResponse,
} from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "kinesis-stream-handler-sample",
});
```

```
export const functionHandler: KinesisStreamHandler = async (
  event: KinesisStreamEvent,
  context: Context
): Promise<KinesisStreamBatchResponse> => {
  for (const record of event.Records) {
    try {
      logger.info(`Processed Kinesis Event - EventID: ${record.eventID}`);
      const recordData = await getRecordDataAsync(record.kinesis);
      logger.info(`Record Data: ${recordData}`);
      // TODO: Do interesting work based on the new data
    } catch (err) {
      logger.error(`An error occurred ${err}`);
      /* Since we are working with streams, we can return the failed item
 immediately.
           Lambda will immediately begin to retry processing from this failed item
 onwards. */
      return {
        batchItemFailures: [{ itemIdentifier: record.kinesis.sequenceNumber }],
      };
    }
  }
  logger.info(`Successfully processed ${event.Records.length} records.`);
  return { batchItemFailures: [] };
};

async function getRecordDataAsync(
  payload: KinesisStreamRecordPayload
): Promise<string> {
  var data = Buffer.from(payload.data, "base64").toString("utf-8");
  await Promise.resolve(1); //Placeholder for actual async work
  return data;
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Meldung von Fehlern bei DynamoDB-Batchelementen mithilfe von Lambda. JavaScript

```
export const handler = async (event) => {
  const records = event.Records;
  let curRecordSequenceNumber = "";

  for (const record of records) {
    try {
      // Process your record
      curRecordSequenceNumber = record.dynamodb.SequenceNumber;
    } catch (e) {
      // Return failed record's sequence number
      return { batchItemFailures: [{ itemIdentifier: curRecordSequenceNumber }] };
    }
  }

  return { batchItemFailures: [] };
};
```

Meldung von Fehlern bei DynamoDB-Batchelementen unter Verwendung von Lambda. TypeScript

```
import {
  DynamoDBBatchResponse,
  DynamoDBBatchItemFailure,
  DynamoDBStreamEvent,
} from "aws-lambda";

export const handler = async (
  event: DynamoDBStreamEvent
): Promise<DynamoDBBatchResponse> => {
  const batchItemFailures: DynamoDBBatchItemFailure[] = [];
  let curRecordSequenceNumber;
```

```
  for (const record of event.Records) {
    curRecordSequenceNumber = record.dynamodb?.SequenceNumber;

    if (curRecordSequenceNumber) {
      batchItemFailures.push({
        itemIdentifier: curRecordSequenceNumber,
      });
    }
  }

  return { batchItemFailures: batchItemFailures };
};
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Meldung von SQS-Batch-Elementfehlern mithilfe von Lambda. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
```

```
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Meldung von Fehlern bei SQS-Batch-Elementen unter Verwendung von Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord } from
 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};

async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

Amazon Lex: Beispiele für die Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Lex Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel Building an Amazon Lex-Chatbot im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Beispiele für Amazon Locations mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Location Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Location

Das folgende Codebeispiel zeigt, wie Sie mit der Nutzung von Amazon Location Service beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  LocationClient,
  ListGeofenceCollectionsCommand,
} from "@aws-sdk/client-location";

/**
 * Lists geofences from a specified geofence collection asynchronously.
 */
export const main = async () => {
```

```
  const region = "eu-west-1";
  const locationClient = new LocationClient({ region: region });
  const listGeofenceCollParams = {
    MaxResults: 100,
  };
  try {
    const command = new ListGeofenceCollectionsCommand(listGeofenceCollParams);
    const response = await locationClient.send(command);
    const geofenceEntries = response.Entries;
    if (geofenceEntries.length === 0) {
      console.log("No Geofences were found in the collection.");
    } else {
      for (const geofenceEntry of geofenceEntries) {
        console.log(`Geofence ID: ${geofenceEntry.CollectionName}`);
      }
    }
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - ListGeofenceCollections
  - ListGeofences

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Location-Karte.
- Erstellen Sie einen API-Schlüssel für Amazon Location.
- Zeigen Sie die Karten-URL an.

- Erstellen Sie eine Geofence-Sammlung.
- Speichern einer Geofence-Geometrie
- Erstellen einer Tracker-Ressource
- Aktualisieren der Position eines Geräts
- Rufen Sie die neueste Positionsaktualisierung für ein bestimmtes Gerät ab.
- Erstellen Sie einen Routenrechner.
- Ermitteln der Entfernung zwischen Seattle und Vancouver
- Verwenden Sie Amazon-Location-APIs höherer Ebene.
- Löschen Sie die Amazon-Location-Komponenten.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
Before running this JavaScript code example, set up your development environment,
 including your credentials.
This demo illustrates how to use the AWS SDK for JavaScript (v3) to work with Amazon
 Location Service.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-
started.html
*/

import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
import {
  CreateMapCommand,
  CreateGeofenceCollectionCommand,
  PutGeofenceCommand,
  CreateTrackerCommand,
  BatchUpdateDevicePositionCommand,
  GetDevicePositionCommand,
  CreateRouteCalculatorCommand,
  CalculateRouteCommand,
  LocationClient,
  ConflictException,
  ResourceNotFoundException,
  DeleteGeofenceCollectionCommand,
  DeleteRouteCalculatorCommand,
  DeleteTrackerCommand,
  DeleteMapCommand,
} from "@aws-sdk/client-location";

import {
  GeoPlacesClient,
  ReverseGeocodeCommand,
  SearchNearbyCommand,
  SearchTextCommand,
  GetPlaceCommand,
  ValidationException,
} from "@aws-sdk/client-geo-places";

import { parseArgs } from "node:util";
import { fileURLToPath } from "node:url";

/*The inputs for this example can be edited in the ./input.json.*/
import data from "./inputs.json" with { type: "json" };

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
/* v8 ignore next 3 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
  verbose: "false",
});

const pressEnterConfirm = new ScenarioInput(
```

```
  "confirm",
  "Press Enter to continue",
  {
    type: "confirm",
    verbose: "false",
  },
);

const region = "eu-west-1";

const locationClient = new LocationClient({ region: region });

const greet = new ScenarioOutput(
  "greet",
  "Welcome to the Amazon Location Use demo! \n" +
    "AWS Location Service is a fully managed service offered by Amazon Web Services
 (AWS) that " +
    "provides location-based services for developers. This service simplifies " +
    "the integration of location-based features into applications, making it " +
    "Maps: The service provides access to high-quality maps, satellite imagery, " +
    "and geospatial data from various providers, allowing developers to " +
    "easily embed maps into their applications:\n" +
    "Tracking: The Location Service enables real-time tracking of mobile devices, "
 +
    "assets, or other entities, allowing developers to build applications " +
    "that can monitor the location of people, vehicles, or other objects.\n" +
    "Geocoding: The service provides the ability to convert addresses or " +
    "location names into geographic coordinates (latitude and longitude), " +
    "and vice versa, enabling developers to integrate location-based search " +
    "and routing functionality into their applications. " +
    "Please define values ./inputs.json for each user-defined variable used in this
 app. Otherwise the default is used:\n" +
    "- mapName: The name of the map to be create (default is 'AWSMap').\n" +
    "- keyName: The name of the API key to create (default is ' AWSApiKey')\n" +
    "- collectionName: The name of the geofence collection (default is
 'AWSLocationCollection')\n" +
    "- geoId: The geographic identifier used for the geofence or map (default is
 'geoId')\n" +
    "- trackerName: The name of the tracker (default is 'geoTracker')\n" +
    "- calculatorName: The name of the route calculator (default is
 'AWSRouteCalc')\n" +
    "- deviceId: The ID of the device (default is 'iPhone-112356')",

  { header: true },
```

```
);
const displayCreateAMap = new ScenarioOutput(
  "displayCreateAMap",
  "1. Create a map\n" +
    "An AWS Location map can enhance the user experience of your " +
    " application by providing accurate and personalized location-based " +
    " features. For example, you could use the geocoding capabilities to " +
    " allow users to search for and locate businesses, landmarks, or " +
    " other points of interest within a specific region.",
);

const sdkCreateAMap = new ScenarioAction(
  "sdkCreateAMap",
  async (/** @type {State} */ state) => {
    const createMapParams = {
      MapName: `${data.inputs.mapName}`,
      Configuration: { style: "VectorEsriNavigation" },
    };
    try {
      const command = new CreateMapCommand(createMapParams);
      const response = await locationClient.send(command);
      state.MapName = response.MapName;
      console.log("Map created. Map ARN is: ", state.MapName);
    } catch (error) {
      console.error("Error creating map: ", error);
      throw error;
    }
  },
);

const displayMapUrl = new ScenarioOutput(
  "displayMapUrl",
  "2. Display Map URL\n" +
    "When you embed a map in a web app or website, the API key is " +
    "included in the map tile URL to authenticate requests. You can " +
    "restrict API keys to specific AWS Location operations (e.g., only " +
    "maps, not geocoding). API keys can expire, ensuring temporary " +
    "access control.\n" +
    "In order to get the MAP URL you need to create and get the API Key value. " +
    "You can create and get the key value using the AWS Management Console under " +
    "Location Services. These operations cannot be completed using the " +
    "AWS SDK. For more information about getting the key value, see " +
    "the AWS Location Documentation.",
);
```

```
const sdkDisplayMapUrl = new ScenarioAction(
  "sdkDisplayMapUrl",
  async (/** @type {State} */ state) => {
    const mapURL = `https://maps.geo.aws.amazon.com/maps/v0/maps/${state.MapName}/
tiles/{z}/{x}/{y}?key=API_KEY_VALUE`;
    state.mapURL = mapURL;
    console.log(
      `Replace \'API_KEY_VALUE\' in the following URL with the value for the API key
 you create and get from the AWS Management Console under Location Services. This is
 then the Map URL you can embed this URL in your Web app:\n
${state.mapURL}`,
    );
  },
);
const displayCreateGeoFenceColl = new ScenarioOutput(
  "displayCreateGeoFenceColl",
  "3. Create a geofence collection, which manages and stores geofences.",
);

const sdkCreateGeoFenceColl = new ScenarioAction(
  "sdkCreateGeoFenceColl",
  async (/** @type {State} */ state) => {
    // Creates a new geofence collection.
    const geoFenceCollParams = {
      CollectionName: `${data.inputs.collectionName}`,
    };
    try {
      const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
      const response = await locationClient.send(command);
      state.CollectionName = response.CollectionName;
      console.log(
        `The geofence collection was successfully created: ${state.CollectionName}`,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An unexpected error occurred while creating the geofence collection:
 ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
```

```
);
const displayStoreGeometry = new ScenarioOutput(
  "displayStoreGeometry",
  "4. Store a geofence geometry in a given geofence collection. " +
    "An AWS Location geofence is a virtual boundary that defines a geographic area "
 +
    "on a map. It is a useful feature for tracking the location of " +
    "assets or monitoring the movement of objects within a specific region. " +
    "To define a geofence, you need to specify the coordinates of a " +
    "polygon that represents the area of interest. The polygon must be " +
    "defined in a counter-clockwise direction, meaning that the points of " +
    "the polygon must be listed in a counter-clockwise order. " +
    "This is a requirement for the AWS Location service to correctly " +
    "interpret the geofence and ensure that the location data is " +
    "accurately processed within the defined area.",
);

const sdkStoreGeometry = new ScenarioAction(
  "sdkStoreGeometry",
  async (/** @type {State} */ state) => {
    const geoFenceGeoParams = {
      CollectionName: `${data.inputs.collectionName}`,
      GeofenceId: `${data.inputs.geoId}`,
      Geometry: {
        Polygon: [
          [
            [-122.3381, 47.6101],
            [-122.3281, 47.6101],
            [-122.3281, 47.6201],
            [-122.3381, 47.6201],
            [-122.3381, 47.6101],
          ],
        ],
      },
    };
    try {
      const command = new PutGeofenceCommand(geoFenceGeoParams);
      const response = await locationClient.send(command);
      state.GeoFencId = response.GeofenceId;
      console.log("GeoFence created. GeoFence ID is: ", state.GeoFencId);
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
```

```
          `A validation error occurred while creating geofence: ${caught.message} \n
 Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayCreateTracker = new ScenarioOutput(
  "displayCreateTracker",
  "5. Create a tracker resource which lets you retrieve current and historical
 location of devices.",
);

const sdkCreateTracker = new ScenarioAction(
  "sdkCreateTracker",
  async (/** @type {State} */ state) => {
    //Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
    const createTrackerParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Description: "Created using the JavaScript V3 SDK",
      PositionFiltering: "TimeBased",
    };
    try {
      const command = new CreateTrackerCommand(createTrackerParams);
      const response = await locationClient.send(command);
      state.trackerName = response.TrackerName;
      console.log("Tracker created. Tracker name is : ", state.trackerName);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while creating geofence: ${caught.message} \n
 Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
      }
      return;
    }
  },
);
const displayUpdatePosition = new ScenarioOutput(
  "displayUpdatePosition",
```

```
  "6. Update the position of a device in the location tracking system." +
    "The AWS Location Service does not enforce a strict format for deviceId, but it
 must:\n " +
    "- Be a string (case-sensitive).\n" +
    "- Be 1–100 characters long.\n" +
    "- Contain only: Alphanumeric characters (A-Z, a-z, 0-9); Underscores (_);
 Hyphens (-); and be the same ID used when sending and retrieving positions.",
);

const sdkUpdatePosition = new ScenarioAction(
  "sdkUpdatePosition",
  async (/** @type {State} */ state) => {
    // Updates the position of a device in the location tracking system.

    const updateDevicePosParams = {
      TrackerName: `${data.inputs.trackerName}`,
      Updates: [
        {
          DeviceId: `${data.inputs.deviceId}`,
          SampleTime: new Date(),
          Position: [-122.4194, 37.7749],
        },
      ],
    };
    try {
      const command = new BatchUpdateDevicePositionCommand(
        updateDevicePosParams,
      );
      const response = await locationClient.send(command);
      console.log(
        `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `A validation error occurred while updating the device: ${caught.message}
 \n Exiting program.`,
        );
      }
    }
  },
);
const displayRetrievePosition = new ScenarioOutput(
```

```
  "displayRetrievePosition",
  "7. Retrieve the most recent position update for a specified device.",
);

const sdkRetrievePosition = new ScenarioAction(
  "sdkRetrievePosition",
  async (/** @type {State} */ state) => {
    const devicePositionParams = {
      TrackerName: `${data.inputs.trackerName}`,
      DeviceId: `${data.inputs.deviceId}`,
    };
    try {
      const command = new GetDevicePositionCommand(devicePositionParams);
      const response = await locationClient.send(command);
      state.position = response.Position;
      console.log("Successfully fetched device position: : ", state.position);
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `"The resource was not found: ${caught.message} \n Exiting program.`,
        );
      } else {
        `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
      }
      return;
    }
  },
);
const displayCreateRouteCalc = new ScenarioOutput(
  "displayCreateRouteCalc",
  "8. Create a route calculator.",
);

const sdkCreateRouteCalc = new ScenarioAction(
  "sdkCreateRouteCalc",
  async (/** @type {State} */ state) => {
    const routeCalcParams = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DataSource: "Esri",
    };
    try {
      // Creates a new route calculator with the specified name and data source.
      const command = new CreateRouteCalculatorCommand(routeCalcParams);
      const response = await locationClient.send(command);
```

```
      state.CalculatorName = response.CalculatorName;
      console.log(
        "Route calculator created successfully. Calculator name is: ",
        state.CalculatorName,
      );
    } catch (caught) {
      if (caught instanceof ConflictException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);
const displayDetermineDist = new ScenarioOutput(
  "displayDetermineDist",
  "9. Determine the distance between Seattle and Vancouver using the route
 calculator.",
);

const sdkDetermineDist = new ScenarioAction(
  "sdkDetermineDist",
  async (/** @type {State} */ state) => {
    // Calculates the distance between two locations asynchronously.
    const determineDist = {
      CalculatorName: `${data.inputs.calculatorName}`,
      DeparturePosition: [-122.3321, 47.6062],
      DestinationPosition: [-123.1216, 49.2827],
      TravelMode: "Car",
      DistanceUnit: "Kilometers",
    };
    try {
      const command = new CalculateRouteCommand(determineDist);
      const response = await locationClient.send(command);

      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.Summary.Distance,
      );
    } catch (caught) {
      if (caught instanceof ResourceNotFoundException) {
        console.error(
          `Failed to calculate route: ${caught.message} \n Exiting program.`,
```

```
        );
      }
      return;
    }
  },
);
const displayUseGeoPlacesClient = new ScenarioOutput(
  "displayUseGeoPlacesClient",
  "10. Use the GeoPlacesAsyncClient to perform additional operations. " +
    "This scenario will show use of the GeoPlacesClient that enables" +
    "location search and geocoding capabilities for your applications. " +
    "We are going to use this client to perform these AWS Location tasks: \n" +
    " - Reverse Geocoding (reverseGeocode): Converts geographic coordinates into
 addresses.\n " +
    " - Place Search (searchText): Finds places based on search queries.\n " +
    " - Nearby Search (searchNearby): Finds places near a specific location.\n " +
    "First we will perform a Reverse Geocoding operation",
);

const sdkUseGeoPlacesClient = new ScenarioAction(
  "sdkUseGeoPlacesClient",
  async (/** @type {State} */ state) => {
    const geoPlacesClient = new GeoPlacesClient({ region: region });

    const reverseGeoCodeParams = {
      QueryPosition: [-122.4194, 37.7749],
    };
    const searchTextParams = {
      QueryText: "coffee shop",
      BiasPosition: [-122.4194, 37.7749], //San Fransisco
    };
    const searchNearbyParams = {
      QueryPosition: [-122.4194, 37.7749],
      QueryRadius: Number("1000"),
    };
    try {
      /*   Performs reverse geocoding using the AWS Geo Places API.
     Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
     This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.*/

      console.log("Use latitude 37.7749 and longitude -122.4194.");
      const command = new ReverseGeocodeCommand(reverseGeoCodeParams);
```

```
      const response = await geoPlacesClient.send(command);
      console.log(
        "Successfully calculated route. The distance in kilometers is : ",
        response.ResultItems[0].Distance,
      );
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log(
        "Now we are going to perform a text search using coffee shop",
      );

      /*Searches for a place using the provided search query and prints the detailed
information of the first result.
 @param searchTextParams the search query to be used for the place search (ex,
coffee shop)*/

      const command = new SearchTextCommand(searchTextParams);
      const response = await geoPlacesClient.send(command);
      const placeId = response.ResultItems[0].PlaceId.toString();
      const getPlaceCommand = new GetPlaceCommand({
        PlaceId: placeId,
      });
      const getPlaceResponse = await geoPlacesClient.send(getPlaceCommand);
      console.log(
        `Detailed Place Information: \n Name and address:
${getPlaceResponse.Address.Label}`,
      );

      const foodTypes = getPlaceResponse.FoodTypes;
      if (foodTypes.length) {
        console.log("Food Types: ");
        for (const foodType of foodTypes) {
          console.log("- ", foodType.LocalizedName);
        }
      } else {
        console.log("No food types available.");
      }
```

```
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
    try {
      console.log("\nNow we are going to perform a nearby search.");
      const command = new SearchNearbyCommand(searchNearbyParams);
      const response = await geoPlacesClient.send(command);
      const resultItems = response.ResultItems;
      console.log("\nSuccessfully performed nearby search.");
      for (const resultItem of resultItems) {
        console.log("Name and address: ", resultItem.Address.Label);
        console.log("Distance: ", resultItem.Distance);
      }
    } catch (caught) {
      if (caught instanceof ValidationException) {
        console.error(
          `An conflict occurred: ${caught.message} \n Exiting program.`,
        );
        return;
      }
    }
  },
);

const displayDeleteResources = new ScenarioOutput(
  "displayDeleteResources",
  "11. Delete the AWS Location Services resources. " +
    "Would you like to delete the AWS Location Services resources? (y/n)",
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    const deleteGeofenceCollParams = {
      CollectionName: `${state.CollectionName}`,
    };
    const deleteRouteCalculatorParams = {
      CalculatorName: `${state.CalculatorName}`,
    };
```

```
    const deleteTrackerParams = { TrackerName: `${state.trackerName}` };
    const deleteMapParams = { MapName: `${state.MapName}` };
    try {
      const command = new DeleteMapCommand(deleteMapParams);
      const response = await locationClient.send(command);
      console.log("Map deleted.");
    } catch (error) {
      console.log("Error deleting map: ", error);
    }
    try {
      const command = new DeleteGeofenceCollectionCommand(
        deleteGeofenceCollParams,
      );
      const response = await locationClient.send(command);
      console.log("Geofence collection deleted.");
    } catch (error) {
      console.log("Error deleting geofence collection: ", error);
    }
    try {
      const command = new DeleteRouteCalculatorCommand(
        deleteRouteCalculatorParams,
      );
      const response = await locationClient.send(command);
      console.log("Route calculator deleted.");
    } catch (error) {
      console.log("Error deleting route calculator: ", error);
    }
    try {
      const command = new DeleteTrackerCommand(deleteTrackerParams);
      const response = await locationClient.send(command);
      console.log("Tracker deleted.");
    } catch (error) {
      console.log("Error deleting tracker: ", error);
    }
  },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "Thank you for checking out the Amazon Location Service Use demo. We hope you " +
    "learned something new, or got some inspiration for your own apps today!" +
    " For more Amazon Location Services examples in different programming languages,
 have a look at: " +
```

```
    "https://docs.aws.amazon.com/code-library/latest/ug/
location_code_examples.html",
);

const myScenario = new Scenario("Location Services Scenario", [
  greet,
  pressEnter,
  displayCreateAMap,
  sdkCreateAMap,
  pressEnter,
  displayMapUrl,
  sdkDisplayMapUrl,
  pressEnter,
  displayCreateGeoFenceColl,
  sdkCreateGeoFenceColl,
  pressEnter,
  displayStoreGeometry,
  sdkStoreGeometry,
  pressEnter,
  displayCreateTracker,
  sdkCreateTracker,
  pressEnter,
  displayUpdatePosition,
  sdkUpdatePosition,
  pressEnter,
  displayRetrievePosition,
  sdkRetrievePosition,
  pressEnter,
  displayCreateRouteCalc,
  sdkCreateRouteCalc,
  pressEnter,
  displayDetermineDist,
  sdkDetermineDist,
  pressEnter,
  displayUseGeoPlacesClient,
  sdkUseGeoPlacesClient,
  pressEnter,
  displayDeleteResources,
  pressEnterConfirm,
  sdkDeleteResources,
  goodbye,
]);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
```

```
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Aktionen

`BatchUpdateDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`BatchUpdateDevicePosition`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  BatchUpdateDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });
const updateDevicePosParams = {
  TrackerName: `${data.inputs.trackerName}`,
  Updates: [
    {
      DeviceId: `${data.inputs.deviceId}`,
      SampleTime: new Date(),
      Position: [-122.4194, 37.7749],
    },
  ],
};
export const main = async () => {
  try {
    const command = new BatchUpdateDevicePositionCommand(updateDevicePosParams);
    const response = await locationClient.send(command);
    //console.log("response ", response.Errors[0].Error);

    console.log(
      `Device with id ${data.inputs.deviceId} was successfully updated in the
 location tracking system. `,
      response,
```

```
    );
  } catch (error) {
    console.log("error ", error);
  }
};
```

- API-Details finden Sie BatchUpdateDevicePosition in der AWS SDK für JavaScript API-Referenz.

`CalculateRoute`

Das folgende Codebeispiel zeigt die Verwendung`CalculateRoute`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  CalculateRouteCommand,
  ResourceNotFoundException,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DeparturePosition: [-122.3321, 47.6062],
    DestinationPosition: [-123.1216, 49.2827],
    TravelMode: "Car",
    DistanceUnit: "Kilometers",
```

```
  };
  try {
    const command = new CalculateRouteCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Successfully calculated route. The distance in kilometers is : ",
      response.Summary.Distance,
    );
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- API-Details finden Sie CalculateRoute in der AWS SDK für JavaScript API-Referenz.

`CreateGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateGeofenceCollection`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateGeofenceCollectionCommand,
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };
```

```
const region = "eu-west-1";

export const main = async () => {
  const geoFenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateGeofenceCollectionCommand(geoFenceCollParams);
    const response = await locationClient.send(command);
    console.log(
      "Collection created. Collection name is: ",
      response.CollectionName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error("A conflict occurred. Exiting program.");
      return;
    }
  }
};
```

- API-Details finden Sie CreateGeofenceCollection in der AWS SDK für JavaScript API-Referenz.

`CreateMap`

Das folgende Codebeispiel zeigt die Verwendung`CreateMap`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
```

```
import { CreateMapCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const CreateMapCommandInput = {
    MapName: `${data.inputs.mapName}`,
    Configuration: { style: "VectorEsriNavigation" },
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateMapCommand(CreateMapCommandInput);
    const response = await locationClient.send(command);
    console.log("Map created. Map ARN is : ", response.MapArn);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- API-Details finden Sie CreateMap in der AWS SDK für JavaScript API-Referenz.

`CreateRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteCalculator`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  ConflictException,
  CreateRouteCalculatorCommand,
```

```
  LocationClient,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
const locationClient = new LocationClient({ region: region });

export const main = async () => {
  const routeCalcParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
    DataSource: "Esri",
  };
  try {
    const command = new CreateRouteCalculatorCommand(routeCalcParams);
    const response = await locationClient.send(command);

    console.log(
      "Route calculator created successfully. Calculator name is ",
      response.CalculatorName,
    );
  } catch (caught) {
    if (caught instanceof ConflictException) {
      console.error(
        `An conflict occurred: ${caught.message} \n Exiting program.`,
      );
      return;
    }
  }
};
```

- API-Details finden Sie CreateRouteCalculator in der AWS SDK für JavaScript API-Referenz.

`CreateTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateTracker`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import { CreateTrackerCommand, LocationClient } from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const createTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new CreateTrackerCommand(createTrackerParams);
    const response = await locationClient.send(command);
    //state.trackerName - response.TrackerName;
    console.log("Tracker created. Tracker name is : ", response.TrackerName);
  } catch (error) {
    console.error("Error creating map: ", error);
    throw error;
  }
};
```

- API-Details finden Sie CreateTracker in der AWS SDK für JavaScript API-Referenz.

`DeleteGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGeofenceCollection`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteGeofenceCollectionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteGeofenceCollParams = {
    CollectionName: `${data.inputs.collectionName}`,
  };
  const locationClient = new LocationClient({ region: region });
  try {
    const command = new DeleteGeofenceCollectionCommand(
      deleteGeofenceCollParams,
    );
    const response = await locationClient.send(command);
    console.log("Collection deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.collectionName} Geofence collection not found.`,
      );
      return;
    }
  }
};
```

- API-Details finden Sie DeleteGeofenceCollection in der AWS SDK für JavaScript API-Referenz.

`DeleteMap`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMap`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteMapCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteMapParams = {
    MapName: `${data.inputs.mapName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteMapCommand(deleteMapParams);
    const response = await locationClient.send(command);
    console.log("Map deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(`${data.inputs.mapName} map not found.`);
      return;
    }
  }
};
```

- API-Details finden Sie DeleteMap in der AWS SDK für JavaScript API-Referenz.

`DeleteRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRouteCalculator`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteRouteCalculatorCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteRouteCalculatorParams = {
    CalculatorName: `${data.inputs.calculatorName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteRouteCalculatorCommand(
      deleteRouteCalculatorParams,
    );
    const response = await locationClient.send(command);
    console.log("Route calculator deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
      console.error(
        `${data.inputs.calculatorName} route calculator not found.`,
      );
```

```
      return;
    }
  }
};
```

- API-Details finden Sie DeleteRouteCalculator in der AWS SDK für JavaScript API-Referenz.

`DeleteTracker`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTracker`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  DeleteTrackerCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const deleteTrackerParams = {
    TrackerName: `${data.inputs.trackerName}`,
  };
  try {
    const locationClient = new LocationClient({ region: region });
    const command = new DeleteTrackerCommand(deleteTrackerParams);
    const response = await locationClient.send(command);
    console.log("Tracker deleted.");
  } catch (caught) {
    if (caught instanceof ResourceNotFoundException) {
```

```
      console.error(`${data.inputs.trackerName} tracker not found.`);
      return;
    }
  }
};
```

- API-Details finden Sie DeleteTracker in der AWS SDK für JavaScript API-Referenz.

`GetDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`GetDevicePosition`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  GetDevicePositionCommand,
  LocationClient,
  ResourceNotFoundException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";

export const main = async () => {
  const locationClient = new LocationClient({ region: region });
  const deviceId = `${data.inputs.deviceId}`;
  const trackerName = `${data.inputs.trackerName}`;

  const devicePositionParams = {
    DeviceId: deviceId,
    TrackerName: trackerName,
  };
  try {
```

```
    const command = new GetDevicePositionCommand(devicePositionParams);
    const response = await locationClient.send(command);
    //state.position = response.position;
    console.log("Successfully fetched device position: ", response);
  } catch (error) {
    console.log("Error ", error);
    /*  if (caught instanceof ResourceNotFoundException) {
      console.error(
        `"The resource was not found: ${caught.message} \n Exiting program.`,
      );
    } else {
      `An unexpected error error occurred: ${caught.message} \n Exiting program.`;
    }
    return;*/
  }
};
```

- API-Details finden Sie GetDevicePosition in der AWS SDK für JavaScript API-Referenz.

`PutGeofence`

Das folgende Codebeispiel zeigt die Verwendung`PutGeofence`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { fileURLToPath } from "node:url";
import {
  PutGeofenceCommand,
  LocationClient,
  ValidationException,
} from "@aws-sdk/client-location";
import data from "./inputs.json" with { type: "json" };

const region = "eu-west-1";
```

```
const locationClient = new LocationClient({ region: region });
export const main = async () => {
  const geoFenceGeoParams = {
    CollectionName: `${data.inputs.collectionName}`,
    GeofenceId: `${data.inputs.geoId}`,
    Geometry: {
      Polygon: [
        [
          [-122.3381, 47.6101],
          [-122.3281, 47.6101],
          [-122.3281, 47.6201],
          [-122.3381, 47.6201],
          [-122.3381, 47.6101],
        ],
      ],
    },
  };
  try {
    const command = new PutGeofenceCommand(geoFenceGeoParams);
    const response = await locationClient.send(command);
    console.log("GeoFence created. GeoFence ID is: ", response.GeofenceId);
  } catch (error) {
    console.error(
      `A validation error occurred while creating geofence: ${error} \n Exiting
 program.`,
    );
    return;
  }
};
```

- API-Details finden Sie PutGeofence in der AWS SDK für JavaScript API-Referenz.

AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für JavaScript (v3) with AWS Marketplace Agreement API Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Vereinbarungen

Vereinbarungen

Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung akzeptiert wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function acceptAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
```

```
    console.log("Status: " + response.status);
    console.log("Description: " + response.description);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementCancellationRequest();
```

- API-Details finden Sie AcceptAgreementCancellationRequest in der AWS SDK für JavaScript API-Referenz.

Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung akzeptiert wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    AcceptAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
const PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>";

async function acceptAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new AcceptAgreementPaymentRequestCommand({
```

```
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

acceptAgreementPaymentRequest();
```

- API-Details finden Sie AcceptAgreementPaymentRequest in der AWS SDK für JavaScript API-Referenz.

Ändern Sie die Verlängerungsdauer einer SaaS-Vertragsvereinbarung

Das folgende Codebeispiel zeigt, wie Sie eine SaaS-Vertragsvereinbarung ändern, um ihre Verlängerungsdauer hinzuzufügen oder zu aktualisieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
```

```
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product using a public offer that supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P1M" for 1 month).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the RenewalTerm in your offer.
const RENEWAL_TERM_ID = "<your-renewal-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";
```

```
// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
 * 2. Wait for entitlements to become active.
 * 3. Amend the agreement to enable auto-renewal.
 */
async function amendSaaSContractAgreementRenewalTerm() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                ],
            },
        },
    };

    // Initial agreement: auto-renewal disabled.
    const renewalTerm = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: false,
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };

    // --- Create and accept the initial SaaS agreement request with CONTRACT
 pricing model ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
```

```
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTerm, legalTerm,
supportTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before amending.
    console.log("Waiting for entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("Entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Amend: enable auto-renewal ---
    const renewalTermAmended = {
        id: RENEWAL_TERM_ID,
        configuration: {
            renewalTermConfiguration: {
                enableAutoRenew: true,
            },
        },
    };

    // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "AMEND",
            requestedTerms: [configurableUpfrontPricingTerm, renewalTermAmended,
legalTerm, supportTerm],
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
```

```
    );
    console.log("Amend agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aarResponse.agreementId);
}

amendSaaSContractAgreementRenewalTerm();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Ändern Sie ein AMI ConfigurableUpfrontPricingTerm für das Nutzungspreismodell

Das folgende Codebeispiel zeigt, wie eine AMI-Vereinbarung mit geändert werden kann, ConfigurableUpfrontPricingTerm indem das Modell für die Dimensionsmenge für das Nutzungspreismodell aktualisiert wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");
```

```
/**
 * Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement entitlements are active.
 *
 * Once both agreement entitlements are available, this sample shows how to amend
 the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D" for one year).
 *   - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance
 type).
 *   - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended
 quantity.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
const DIMENSION_1_KEY = "<your-dimension-key>";

// Initial quantity for the dimension.
const DIMENSION_1_VALUE = 1;
```

```
// Term ID for the UsageBasedPricingTerm in your offer.
const USAGE_TERM_ID = "<your-usage-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// New quantity to use when amending the dimension of CUPT.
const NEW_DIMENSION_1_VALUE = 5;

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm.
 * 2. Wait for entitlements to become active, then create and accept an agreement
 request with configurableUpfrontPricingTerm (CUPT).
 * 3. Wait for CUPT entitlements to become active, then amend the dimension
 quantity.
 */
async function amendAmiCUPTAgreement() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
```

```
    );
    console.log("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

    // Wait for entitlements to become active before creating the agreement with
CUPT.
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const cuptAgreementId = aarResponse.agreementId;
```

```
   console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

   // Wait for entitlements to become active before amending.
   console.log("Waiting for CUPT agreement entitlements to become active...");
   const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
cuptAgreementId);
   console.log("CUPT agreement entitlements are now active.");
   formatOutput(cuptEntitlementsResponse);

   // --- Amend Agreement with CUPT ---
   // Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
   const newConfig = {
       id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
       configuration: {
           configurableUpfrontPricingTermConfiguration: {
               selectorValue: SELECTOR_VALUE,
               dimensions: [
                   { dimensionKey: DIMENSION_1_KEY, dimensionValue:
NEW_DIMENSION_1_VALUE }, // Increase quantity for this dimension key
               ],
           },
       },
   };

   const carAmendResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "AMEND",
           requestedTerms: [newConfig, legalTerm, validityTerm],
           sourceAgreementIdentifier: cuptAgreementId,
       })
   );
   console.log("Amendment of CUPT agreement request created. AgreementRequestId: "
+ carAmendResponse.agreementRequestId);

   const aarAmendResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: carAmendResponse.agreementRequestId,
       })
   );
   console.log("Amendment accepted. New AgreementId: " +
aarAmendResponse.agreementId);
```

```
}

amendAmiCUPTAgreement();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion

Das folgende Codebeispiel zeigt, wie Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion erstellen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to an AMI product that offers a free trial period.
 * The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
```

```
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the FreeTrialPricingTerm in your offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the SupportTerm in your offer.
const SUPPORT_TERM_ID = "<your-support-term-id>";

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

/**
 * Creates an AMI Free Trial agreement.
 * The FreeTrialPricingTerm grants access at no cost for the trial period.
 * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
 */
async function createAndAcceptAmiFreeTrialAgreementRequest() {
    const client = new MarketplaceAgreementClient();

    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const supportTerm = { id: SUPPORT_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
```

```
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm accepted. AgreementId:
 " + acceptAgreementRequestResponse.agreementId);
}

createAndAcceptAmiFreeTrialAgreementRequest();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung

Das folgende Codebeispiel zeigt, wie Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung erstellen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
 *
```

```
 * Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P12M" for 12 months).
 *   - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
 *   - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
 */

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P12M" for 12 months).
const SELECTOR_VALUE = "<your-selector-value>";

// First dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-1-key>";
const DIMENSION_1_VALUE = 10;

// Second dimension key and quantity defined in your offer.
const DIMENSION_2_KEY = "<your-dimension-2-key>";
const DIMENSION_2_VALUE = 20;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
const TAX_ESTIMATION = "ENABLED";

/**
 * Creates a SaaS agreement with CONTRACT pricing model with configurable upfront
 pricing
 * for multiple dimensions and tax estimation.
```

```
 */
async function createSaaSContractAgreement() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
 DIMENSION_1_VALUE },
                    { dimensionKey: DIMENSION_2_KEY, dimensionValue:
 DIMENSION_2_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };

    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm],
            taxConfiguration: { taxEstimation: TAX_ESTIMATION },
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId);
}

createSaaSContractAgreement();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Informieren Sie sich über die Einzelheiten einer Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Details zu einer bestimmten Anfrage zur Kündigung einer Vereinbarung abrufen können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    GetAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";

async function getAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
```

```
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementCancellationRequest();
```

- API-Details finden Sie GetAgreementCancellationRequest in der AWS SDK für JavaScript API-Referenz.

Rufen Sie die Einzelheiten einer Zahlungsanforderung für eine Vereinbarung ab

Das folgende Codebeispiel zeigt, wie Sie Details einer bestimmten Zahlungsaufforderung für eine Vereinbarung abrufen können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    GetAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

async function getAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new GetAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
```

```
            paymentRequestId: PAYMENT_REQUEST_ID,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge ID: " + response.chargeId);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

getAgreementPaymentRequest();
```

- API-Details finden Sie GetAgreementPaymentRequest in der AWS SDK für JavaScript API-Referenz.

Listen Sie Anfragen zur Kündigung von Vereinbarungen auf

Das folgende Codebeispiel zeigt, wie Anfragen zur Kündigung von Vereinbarungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    ListAgreementCancellationRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");
```

```
const PARTY_TYPE = "Proposer";

async function listAgreementCancellationRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;

    do {
        const response = await client.send(
            new ListAgreementCancellationRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Cancellation Request ID: " +
 summary.agreementCancellationRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Reason Code: " + summary.reasonCode);
            console.log("Agreement Type: " + summary.agreementType);
            console.log("Catalog: " + summary.catalog);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }

        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementCancellationRequests();
```

- API-Details finden Sie ListAgreementCancellationRequests in der AWS SDK für JavaScript API-Referenz.

Zahlungsanfragen für Vereinbarungen auflisten

Das folgende Codebeispiel zeigt, wie Zahlungsanforderungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    ListAgreementPaymentRequestsCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const PARTY_TYPE = "Proposer";

async function listAgreementPaymentRequests() {
    const client = new MarketplaceAgreementClient();

    let nextToken = null;

    do {
        const response = await client.send(
            new ListAgreementPaymentRequestsCommand({
                partyType: PARTY_TYPE,
                nextToken: nextToken,
            })
        );

        for (const summary of response.items) {
            console.log("Payment Request ID: " + summary.paymentRequestId);
            console.log("Agreement ID: " + summary.agreementId);
            console.log("Status: " + summary.status);
            console.log("Name: " + summary.name);
            console.log("Charge ID: " + summary.chargeId);
            console.log("Charge Amount: " + summary.chargeAmount);
            console.log("Currency Code: " + summary.currencyCode);
            console.log("Created At: " + summary.createdAt);
            console.log("Updated At: " + summary.updatedAt);
            console.log("---");
        }
```

```
        nextToken = response.nextToken;
    } while (nextToken != null);
}

listAgreementPaymentRequests();
```

- API-Details finden Sie ListAgreementPaymentRequests in der AWS SDK für JavaScript API-Referenz.

Lehnen Sie eine Anfrage zur Kündigung einer Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung abgelehnt wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementCancellationRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID
 HERE>";
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementCancellationRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementCancellationRequestCommand({
            agreementId: AGREEMENT_ID,
            agreementCancellationRequestId: AGREEMENT_CANCELLATION_REQUEST_ID,
```

```
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Agreement ID: " + response.agreementId);
    console.log("Cancellation Request ID: " +
 response.agreementCancellationRequestId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Reason Code: " + response.reasonCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementCancellationRequest();
```

- API-Details finden Sie RejectAgreementCancellationRequest in der AWS SDK für JavaScript API-Referenz.

Lehnen Sie eine Zahlungsanforderung für eine Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung abgelehnt wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    RejectAgreementPaymentRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");

const AGREEMENT_ID = "<AGREEMENT ID HERE>";
const PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
```

```
const REJECTION_REASON = "<REJECTION REASON HERE>";

async function rejectAgreementPaymentRequest() {
    const client = new MarketplaceAgreementClient();

    const response = await client.send(
        new RejectAgreementPaymentRequestCommand({
            agreementId: AGREEMENT_ID,
            paymentRequestId: PAYMENT_REQUEST_ID,
            rejectionReason: REJECTION_REASON,
        })
    );

    console.log("Payment Request ID: " + response.paymentRequestId);
    console.log("Agreement ID: " + response.agreementId);
    console.log("Status: " + response.status);
    console.log("Status Message: " + response.statusMessage);
    console.log("Name: " + response.name);
    console.log("Charge Amount: " + response.chargeAmount);
    console.log("Currency Code: " + response.currencyCode);
    console.log("Created At: " + response.createdAt);
    console.log("Updated At: " + response.updatedAt);
}

rejectAgreementPaymentRequest();
```

- API-Details finden Sie RejectAgreementPaymentRequest in der AWS SDK für JavaScript API-Referenz.

Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vertragsbasiertes Angebot

Das folgende Codebeispiel zeigt, wie eine SaaS-Vertragsvereinbarung durch ein neues vertragsbasiertes Angebot ersetzt wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * Note: Unlike other Replace samples, this example starts from an already-active
 * EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with "agmt-").
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
 *   - Term IDs (starting with "term-") — found in the new offer's term list.
 */
```

```
// The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
const EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>";

// The agreementProposalId from the new Agreement-Based Offer.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the LegalTerm in the new offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the FixedUpfrontPricingTerm in the new offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the new offer.
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

/**
 * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
 * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing agreement
 ID.
 */
async function replaceExistingAgreement() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };

    // Replace the agreement with the new offer
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [fixedUpfrontPricingTerm, paymentScheduleTerm,
 legalTerm, validityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: EXISTING_AGREEMENT_ID,
        })
```

```
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId);
}

replaceExistingAgreement();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Vertrag mit Verbrauchspreisen

Das folgende Codebeispiel zeigt, wie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing (CCP) ersetzt wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");
```

```
/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * Flow:
 *   1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 *   2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 *   3. Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the free trial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>";

// Term ID for the FreeTrialPricingTerm in the free trial offer.
const FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>";

// Term ID for the LegalTerm in the free trial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the paid CCP offer to convert to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-
identifier>";

// Term ID for the FixedUpfrontPricingTerm in the CCP offer.
const FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>";

// Term ID for the PaymentScheduleTerm in the CCP offer.
```

```
const PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>";

// Term ID for the UsageBasedPricingTerm in the CCP offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the CCP offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the CCP offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
 * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
 * 3. Replace the free trial agreement with the paid CCP offer.
 */
async function createSaaSFreeTrialAndReplaceWithCCP() {
    const client = new MarketplaceAgreementClient();

    const legalTerm = { id: LEGAL_TERM_ID };
    const freeTrialPricingTerm = { id: FREE_TRIAL_PRICING_TERM_ID };

    // --- Step 1: Agreement with freeTrialPricingTerm ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [freeTrialPricingTerm, legalTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm created.
 AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    console.log("Agreement request with freeTrialPricingTerm accepted. AgreementId:
 " + acceptAgreementRequestResponse.agreementId);
```

```
    // Wait for freeTrialPricingTerm agreement entitlements to become active before
 replacing.
    console.log("Waiting for freeTrialPricingTerm agreement entitlements to become
 active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
 acceptAgreementRequestResponse.agreementId);
    console.log("freeTrialPricingTerm agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP Offer
 ---
    // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
 agreement.
    const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
    const fixedUpfrontPricingTerm = { id: FIXED_UPFRONT_PRICING_TERM_ID };
    const paymentScheduleTerm = { id: PAYMENT_SCHEDULE_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [usageBasedPricingTerm, fixedUpfrontPricingTerm,
 paymentScheduleTerm, validityTerm, newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    console.log("Agreement with freeTrialPricingTerm replaced with paid CCP offer.
 New AgreementId: " + aarResponse.agreementId);
}

createSaaSFreeTrialAndReplaceWithCCP();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Ersetzen Sie eine nutzungsabhängige SaaS-Preisklausel und kündigen Sie die vorherige Vereinbarung

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige SaaS-Preisbedingung ersetzt und die vorherige Vereinbarung gekündigt wird.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    CancelAgreementCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing agreement.
 *
 * Flow:
 *   1. Create and accept the initial agreement request with UBPT.
 *   2. Wait for entitlements to become active.
 *   3. Replace the agreement with a new offer using Intent.REPLACE.
 *   4. Cancel the new agreement using CancelAgreement.
 *
```

```
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept the initial agreement request with UsageBasedPricingTerm.
 * 2. Wait for entitlements to become active.
 * 3. Replace the agreement with a new offer.
 * 4. Cancel the new agreement.
 */
async function replaceSaaSUbptAndCancel() {
    const client = new MarketplaceAgreementClient();
```

```
   const usageBasedPricingTerm = { id: USAGE_BASED_PRICING_TERM_ID };
   const validityTerm = { id: VALIDITY_TERM_ID };
   const legalTerm = { id: LEGAL_TERM_ID };

   // --- Step 1: Create and accept the initial UBPT agreement request ---
   const createAgreementRequestResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "NEW",
           requestedTerms: [usageBasedPricingTerm, validityTerm, legalTerm],
           agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
       })
   );
   console.log("Agreement request with UBPT created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

   const acceptAgreementRequestResponse = await client.send(
       new AcceptAgreementRequestCommand({
           agreementRequestId: createAgreementRequestResponse.agreementRequestId,
       })
   );
   console.log("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId);

   // Wait for entitlements to become active before replacing.
   console.log("Waiting for entitlements to become active...");
   const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
acceptAgreementRequestResponse.agreementId);
   console.log("Entitlements are now active.");
   formatOutput(entitlementsResponse);

   // --- Step 2: Replace the UBPT agreement with a new offer ---
   // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
agreement.
   const newUsageBasedPricingTerm = { id: NEW_USAGE_BASED_PRICING_TERM_ID };
   const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };
   const newLegalTerm = { id: NEW_LEGAL_TERM_ID };

   const carResponse = await client.send(
       new CreateAgreementRequestCommand({
           clientToken: generateClientToken(),
           intent: "REPLACE",
```

```
            requestedTerms: [newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: acceptAgreementRequestResponse.agreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const agreementIdWithTheNewOffer = aarResponse.agreementId;
    console.log("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

    // --- Step 3: Cancel the new agreement ---
    await client.send(
        new CancelAgreementCommand({
            agreementId: agreementIdWithTheNewOffer,
        })
    );
    console.log("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
}

replaceSaaSUbptAndCancel();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Ersetzen Sie eine nutzungsabhängige AMI-Preisklausel, behalten Sie jedoch die ConfigurableUpfrontPricingTerm

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige AMI-Preisklausel ersetzt und gleichzeitig die bestehende beibehalten wird. ConfigurableUpfrontPricingTerm

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput, pollUntilEntitlementsAvailable } =
 require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: An AMI product with USAGE pricing requires two agreements:
 *   1. An agreement with usageBasedPricingTerm — accepted first to establish the
 base agreement.
 *   2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 usageBasedPricingTerm agreement entitlements are active.
 *
 * This sample shows how to replace only the agreement with usageBasedPricingTerm
 with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * Flow:
 *   1. Create and accept the initial agreement request with usageBasedPricingTerm.
 *   2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT) (after usageBasedPricingTerm agreement entitlements are active).
 *   3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.
 *
```

```
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offers:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial
 offer.
 *   - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new
 offer to replace to.
 *   - Term IDs (starting with "term-") — found in each offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement (e.g., "P365D").
 *   - DIMENSION_1_KEY — dimension key defined in the CUPT term.
 */

// The agreementProposalId from the initial offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P365D" for 365 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key defined in the CUPT term.
const DIMENSION_1_KEY = "<your-dimension-key>";

// Quantity for the dimension.
const DIMENSION_1_VALUE = 1;

// Term ID for the UsageBasedPricingTerm in the initial offer.
const USAGE_TERM_ID = "<your-usage-term-id>";

// Term ID for the LegalTerm in the initial offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in the initial offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

// The agreementProposalId from the new offer to replace to.
const NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-
identifier>";

// Term ID for the UsageBasedPricingTerm in the new offer.
const NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";
```

```
// Term ID for the LegalTerm in the new offer.
const NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

// Term ID for the ValidityTerm in the new offer.
const NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create and accept an agreement request with usageBasedPricingTerm, then wait
 for entitlements.
 * 2. Create and accept an agreement request with CUPT, then wait for entitlements.
 * 3. Replace the agreement with usageBasedPricingTerm with a new offer (agreement
 with CUPT is unaffected).
 */
async function replaceAmiUsageWhenCuptExists() {
    const client = new MarketplaceAgreementClient();

    const usageTerm = { id: USAGE_TERM_ID };
    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Step 1: Agreement with UBPT ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [usageTerm, legalTerm, validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with UBPT created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId);

    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
        })
    );
    const usageAgreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request with UBPT accepted. AgreementId: " +
 usageAgreementId);

    // Wait for UBPT agreement entitlements to become active before creating the
 agreement with CUPT.
```

```
    console.log("Waiting for UBPT agreement entitlements to become active...");
    const entitlementsResponse = await pollUntilEntitlementsAvailable(client,
usageAgreementId);
    console.log("UBPT agreement entitlements are now active.");
    formatOutput(entitlementsResponse);

    // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const carResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request with CUPT created. AgreementRequestId: " +
carResponse.agreementRequestId);

    const aarResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carResponse.agreementRequestId,
        })
    );
    const cuptAgreementId = aarResponse.agreementId;
    console.log("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

    // Wait for CUPT agreement entitlements to become active before replacing usage.
    console.log("Waiting for CUPT agreement entitlements to become active...");
```

```
    const cuptEntitlementsResponse = await pollUntilEntitlementsAvailable(client,
 cuptAgreementId);
    console.log("CUPT agreement entitlements are now active.");
    formatOutput(cuptEntitlementsResponse);

    // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer ---
    const newUsageTerm = { id: NEW_USAGE_TERM_ID };
    const newLegalTerm = { id: NEW_LEGAL_TERM_ID };
    const newValidityTerm = { id: NEW_VALIDITY_TERM_ID };

    // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
 usageBasedPricingTerm agreement only.
    // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected by
 this replacement.
    const carReplaceResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "REPLACE",
            requestedTerms: [newUsageTerm, newLegalTerm, newValidityTerm],
            agreementProposalIdentifier: NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier: usageAgreementId,
        })
    );
    console.log("Replace agreement request created. AgreementRequestId: " +
 carReplaceResponse.agreementRequestId);

    const aarReplaceResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: carReplaceResponse.agreementRequestId,
        })
    );
    console.log("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId);
}

replaceAmiUsageWhenCuptExists();
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK für JavaScript API-Referenz.

Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung

Das folgende Codebeispiel zeigt, wie Bestellungen aktualisiert werden, nachdem die Vereinbarung akzeptiert wurde.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
const {
    MarketplaceAgreementClient,
    CreateAgreementRequestCommand,
    AcceptAgreementRequestCommand,
    ListAgreementChargesCommand,
    UpdatePurchaseOrdersCommand,
} = require("@aws-sdk/client-marketplace-agreement");
const { generateClientToken, formatOutput } = require("./utils/AgreementApiUtils");

/**
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference with
 a
 * specific charge via UpdatePurchaseOrders.
 *
 * Before running this sample, replace the placeholder constants below with values
 from
 * your AWS Marketplace offer:
 *   - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
 *   - Term IDs (starting with "term-") — found in the offer's term list.
 *   - SELECTOR_VALUE — duration for the agreement.
```

```
 *   - DIMENSION_1_KEY – dimension key defined in the offer.
 *   - PURCHASE_ORDER_REFERENCE – your internal purchase order number (e.g.,
 "po-123456").
 */

// Your internal purchase order reference number (e.g., "po-123456").
const PURCHASE_ORDER_REFERENCE = "po-123456";

// The agreementProposalId from the offer.
const AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>";

// Term ID for the ConfigurableUpfrontPricingTerm in your offer.
const CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-
term-id>";

// Duration for the agreement (e.g., "P366D" for 366 days).
const SELECTOR_VALUE = "<your-selector-value>";

// Dimension key and quantity defined in your offer.
const DIMENSION_1_KEY = "<your-dimension-key>";
const DIMENSION_1_VALUE = 1;

// Term ID for the LegalTerm in your offer.
const LEGAL_TERM_ID = "<your-legal-term-id>";

// Term ID for the ValidityTerm in your offer.
const VALIDITY_TERM_ID = "<your-validity-term-id>";

/**
 * Full end-to-end flow:
 * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
 reference.
 * 2. List charges to retrieve charge IDs and revisions.
 * 3. Associate the purchase order reference with a specific charge via
 UpdatePurchaseOrders.
 * 4. List charges again to confirm the update.
 */
async function listAgreementChargesAndUpdatePurchaseOrders() {
    const client = new MarketplaceAgreementClient();

    const configurableUpfrontPricingTerm = {
        id: CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
        configuration: {
            configurableUpfrontPricingTermConfiguration: {
```

```
                selectorValue: SELECTOR_VALUE,
                dimensions: [
                    { dimensionKey: DIMENSION_1_KEY, dimensionValue:
DIMENSION_1_VALUE },
                ],
            },
        },
    };

    const legalTerm = { id: LEGAL_TERM_ID };
    const validityTerm = { id: VALIDITY_TERM_ID };

    // --- Create Agreement ---
    const createAgreementRequestResponse = await client.send(
        new CreateAgreementRequestCommand({
            clientToken: generateClientToken(),
            intent: "NEW",
            requestedTerms: [configurableUpfrontPricingTerm, legalTerm,
validityTerm],
            agreementProposalIdentifier: AGREEMENT_PROPOSAL_IDENTIFIER,
        })
    );
    console.log("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId);

    // --- Accept Agreement Request with Purchase Order ---
    // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
    const chargeId =
createAgreementRequestResponse.chargeSummary.expectedCharges[0].id;
    const purchaseOrderAtAcceptance = {
        chargeId: chargeId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
    };
    const acceptAgreementRequestResponse = await client.send(
        new AcceptAgreementRequestCommand({
            agreementRequestId: createAgreementRequestResponse.agreementRequestId,
            purchaseOrders: [purchaseOrderAtAcceptance],
        })
    );
    const agreementId = acceptAgreementRequestResponse.agreementId;
    console.log("Agreement request accepted with purchase order reference '" +
PURCHASE_ORDER_REFERENCE
        + "'. AgreementId: " + agreementId);
```

```
    // --- List Agreement Charges ---
    const listAgreementChargesResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );

    console.log("All charges for agreement " + agreementId + ":");
    formatOutput(listAgreementChargesResponse);

    // --- Update Purchase Order ---
    const firstCharge = listAgreementChargesResponse.items[0];
    const purchaseOrder = {
        agreementId: agreementId,
        purchaseOrderReference: PURCHASE_ORDER_REFERENCE,
        chargeRevision: firstCharge.revision,
        chargeId: firstCharge.id,
    };
    await client.send(
        new UpdatePurchaseOrdersCommand({
            purchaseOrders: [purchaseOrder],
        })
    );
    console.log("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
        + "' updated for ChargeId: " + firstCharge.id);

    // --- Verify Update ---
    const lacResponse = await client.send(
        new ListAgreementChargesCommand({
            agreementId: agreementId,
        })
    );
    console.log("Verified updated charge:");
    formatOutput(lacResponse.items[0]);
}

listAgreementChargesAndUpdatePurchaseOrders();
```

- API-Details finden Sie UpdatePurchaseOrders in der AWS SDK für JavaScript API-Referenz.

Amazon MSK-Beispiele, die das SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon MSK Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines Amazon MSK-Ereignisses unter Verwendung von Lambda. JavaScript

```
exports.handler = async (event) => {
    // Iterate through keys
    for (let key in event.records) {
      console.log('Key: ', key)
      // Iterate through records
      event.records[key].map((record) => {
        console.log('Record: ', record)
        // Decode base64
        const msg = Buffer.from(record.value, 'base64').toString()
```

```
        console.log('Message:', msg)
      })
    }
}
```

Es wird ein Amazon MSK-Ereignis mit Lambda ausgeführt. TypeScript

```
import { MSKEvent, Context } from "aws-lambda";
import { Buffer } from "buffer";
import { Logger } from "@aws-lambda-powertools/logger";

const logger = new Logger({
  logLevel: "INFO",
  serviceName: "msk-handler-sample",
});

export const handler = async (
  event: MSKEvent,
  context: Context
): Promise<void> => {
  for (const [topic, topicRecords] of Object.entries(event.records)) {
    logger.info(`Processing key: ${topic}`);

    // Process each record in the partition
    for (const record of topicRecords) {
      try {
        // Decode the message value from base64
        const decodedMessage = Buffer.from(record.value, 'base64').toString();

        logger.info({
          message: decodedMessage
        });
      }
      catch (error) {
        logger.error('Error processing event', { error });
        throw error;
      }
    };
  }
}
```

Amazon Personalisieren Sie Beispiele mithilfe des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Personalize Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateBatchInferenceJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateBatchInferenceJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: "JOB_NAME",
  jobInput: {
```

```
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchInferenceJobCommand(createBatchInferenceJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateBatchInferenceJob in der AWS SDK für JavaScript API-Referenz.

`CreateBatchSegmentJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateBatchSegmentJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: "NAME",
  jobInput: {
    s3DataSource: {
      path: "INPUT_PATH",
    },
  },
  jobOutput: {
    s3DataDestination: {
      path: "OUTPUT_PATH",
    },
  },
  roleArn: "ROLE_ARN",
  solutionVersionArn: "SOLUTION_VERSION_ARN",
  numResults: 20,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateBatchSegmentJobCommand(createBatchSegmentJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateBatchSegmentJob in der AWS SDK für JavaScript API-Referenz.

`CreateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`CreateCampaign`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: "SOLUTION_VERSION_ARN" /* required */,
  name: "NAME" /* required */,
  minProvisionedTPS: 1 /* optional integer */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateCampaignCommand(createCampaignParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateCampaign in der AWS SDK für JavaScript API-Referenz.

`CreateDataset`

Das folgende Codebeispiel zeigt die Verwendung`CreateDataset`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  datasetType: "DATASET_TYPE" /* required */,
  name: "NAME" /* required */,
  schemaArn: "SCHEMA_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetCommand(createDatasetParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateDataset in der AWS SDK für JavaScript API-Referenz.

`CreateDatasetExportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetExportJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  jobOutput: {
    s3DataDestination: {
      path: "S3_DESTINATION_PATH" /* required */,
      //kmsKeyArn: 'ARN'  /* include if your bucket uses AWS KMS for encryption
    },
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetExportJobCommand(datasetExportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- API-Details finden Sie CreateDatasetExportJob in der AWS SDK für JavaScript API-Referenz.

`CreateDatasetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetGroup`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.

import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset group parameters.
export const createDatasetGroupParam = {
  name: "NAME" /* required */,
};

export const run = async (createDatasetGroupParam) => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(createDatasetGroupParam),
    );
    console.log("Success", response);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run(createDatasetGroupParam);
```

Erstellen Sie eine Domain-Datensatzgruppe.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: "NAME" /* required */,
  domain:
    "DOMAIN" /* required for a domain dsg, specify ECOMMERCE or VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetGroupCommand(domainDatasetGroupParams),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateDatasetGroup in der AWS SDK für JavaScript API-Referenz.

`CreateDatasetImportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetImportJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetImportJobCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: "DATASET_ARN" /* required */,
  dataSource: {
    /* required */
    dataLocation: "S3_PATH",
  },
  jobName: "NAME" /* required */,
  roleArn: "ROLE_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateDatasetImportJobCommand(datasetImportJobParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateDatasetImportJob in der AWS SDK für JavaScript API-Referenz.

`CreateEventTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateEventTracker`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateEventTrackerCommand(createEventTrackerParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateEventTracker in der AWS SDK für JavaScript API-Referenz.

`CreateFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateFilter`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  name: "NAME" /* required */,
  filterExpression: "FILTER_EXPRESSION" /*required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateFilterCommand(createFilterParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateFilter in der AWS SDK für JavaScript API-Referenz.

`CreateRecommender`

Das folgende Codebeispiel zeigt die Verwendung`CreateRecommender`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: "NAME" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateRecommenderCommand(createRecommenderParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateRecommender in der AWS SDK für JavaScript API-Referenz.

`CreateSchema`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchema`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie ein Schema mit einer Domain.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from "node:fs";

const schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = "TEST"; // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: "NAME" /* required */,
  schema: mySchema /* required */,
  domain:
    "DOMAIN" /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSchemaCommand(createDomainSchemaParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateSchema in der AWS SDK für JavaScript API-Referenz.

`CreateSolution`

Das folgende Codebeispiel zeigt die Verwendung`CreateSolution`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: "DATASET_GROUP_ARN" /* required */,
  recipeArn: "RECIPE_ARN" /* required */,
  name: "NAME" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionCommand(createSolutionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateSolution in der AWS SDK für JavaScript API-Referenz.

`CreateSolutionVersion`

Das folgende Codebeispiel zeigt die Verwendung`CreateSolutionVersion`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: "SOLUTION_ARN" /* required */,
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateSolutionVersionCommand(solutionVersionParam),
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

- API-Details finden Sie CreateSolutionVersion in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Personalize Events mithilfe des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Personalize Events Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
```

```
import { PutEventsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000); // 1613443801 is a testing value.
 Replace it with your sentAt timestamp in UNIX format.

// Set put events parameters.
const putEventsParam = {
  eventList: [
    /* required */
    {
      eventType: "EVENT_TYPE" /* required */,
      sentAt: sentAtDate /* required, must be a Date with js */,
      eventId: "EVENT_ID" /* optional */,
      itemId: "ITEM_ID" /* optional */,
    },
  ],
  sessionId: "SESSION_ID" /* required */,
  trackingId: "TRACKING_ID" /* required */,
  userId: "USER_ID" /* required */,
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutEventsCommand(putEventsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie PutEvents in der AWS SDK für JavaScript API-Referenz.

`PutItems`

Das folgende Codebeispiel zeigt die Verwendung`PutItems`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put items parameters. For string properties and values, use the \
 character to escape quotes.
const putItemsParam = {
  datasetArn: "DATASET_ARN" /* required */,
  items: [
    /* required */
    {
      itemId: "ITEM_ID" /*  required */,
      properties:
        '{"PROPERTY1_NAME": "PROPERTY1_VALUE", "PROPERTY2_NAME": "PROPERTY2_VALUE",
 "PROPERTY3_NAME": "PROPERTY3_VALUE"}' /* optional */,
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie PutItems in der AWS SDK für JavaScript API-Referenz.

`PutUsers`

Das folgende Codebeispiel zeigt die Verwendung`PutUsers`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put users parameters. For string properties and values, use the \
 character to escape quotes.
const putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      userId: "USER_ID",
      properties: '{"PROPERTY1_NAME": "PROPERTY1_VALUE"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutUsersCommand(putUsersParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- API-Details finden Sie PutUsers in der AWS SDK für JavaScript API-Referenz.

Amazon Personalize Runtime-Beispiele mithilfe des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Personalize Runtime Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetPersonalizedRanking`

Das folgende Codebeispiel zeigt die Verwendung`GetPersonalizedRanking`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
```

```
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetPersonalizedRankingCommand(getPersonalizedRankingParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie GetPersonalizedRanking in der AWS SDK für JavaScript API-Referenz.

`GetRecommendations`

Das folgende Codebeispiel zeigt die Verwendung`GetRecommendations`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
```

```
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Holen Sie sich eine Empfehlung mit einem Filter (benutzerdefinierte Datensatzgruppe).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
};
```

```
export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Holen Sie sich gefilterte Empfehlungen von einem Empfehler, der in einer Domain-Datensatzgruppe erstellt wurde.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN" /* required */,
  userId: "USER_ID" /* required */,
  numResults: 15 /* optional */,
  filterArn: "FILTER_ARN" /* required to filter recommendations */,
  filterValues: {
    PROPERTY:
      '"VALUE"' /* Only required if your filter has a placeholder parameter */,
  },
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(
      new GetRecommendationsCommand(getRecommendationsParam),
    );
    console.log("Success!", response);
    return response; // For unit tests.
```

```
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie GetRecommendations in der AWS SDK für JavaScript API-Referenz.

Amazon Pinpoint-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Pinpoint Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
export const pinClient = new PinpointClient({ region: REGION });
```

Senden Sie eine E-Mail-Nachricht.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
const subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
const body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----------------------------------------------------
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
 Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
const body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a>
 using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
const charset = "UTF-8";
```

```
const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
  try {
    const { MessageResponse } = await pinClient.send(
      new SendMessagesCommand(params),
    );

    if (!MessageResponse) {
      throw new Error("No message response.");
    }

    if (!MessageResponse.Result) {
      throw new Error("No message result.");
    }
```

```
    const recipientResult = MessageResponse.Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    }
    console.log(recipientResult.MessageId);
  } catch (err) {
    console.log(err.message);
  }
};

run();
```

Senden Sie eine SMS-Nachricht.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

/* The phone number or short code to send the message from. The phone number
 or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number.  For best results, you should specify the phone
 number in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out.";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
```

```
marketing-related content, specify PROMOTIONAL.*/
const messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
const registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html.*/

const senderId = "MySenderID";

// Specify the parameters to pass to the API.
const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    },
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));
    console.log(
      `Message sent!
 ${data.MessageResponse.Result[destinationNumber].StatusMessage}`,
    );
  } catch (err) {
    console.log(err);
  }
};
```

```
run();
```

- API-Details finden Sie SendMessages in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Polly, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.

- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie das innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });

  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
```

```
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });

  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";
```

```
/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
```

```
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Amazon RDS-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien
- Serverless-Beispiele

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für JavaScript (v3)

Zeigt, wie Sie AWS SDK für JavaScript (v3) verwenden, um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte per E-Mail mithilfe von Amazon Simple Email Service (Amazon SES) versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/*
Node.js code here.
*/
// ES6+ example
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

async function createAuthToken() {
  // Define connection authentication parameters
  const dbinfo = {

    hostname: process.env.ProxyHostName,
    port: process.env.Port,
    username: process.env.DBUserName,
    region: process.env.AWS_REGION,

  }

  // Create RDS Signer object
  const signer = new Signer(dbinfo);

  // Request authorization token from RDS, specifying the username
  const token = await signer.getAuthToken();
  return token;
}

async function dbOps() {

  // Obtain auth token
```

```
  const token = await createAuthToken();
  // Define connection configuration
  let connectionConfig = {
    host: process.env.ProxyHostName,
    user: process.env.DBUserName,
    password: token,
    database: process.env.DBName,
    ssl: 'Amazon RDS'
  }
  // Create the connection to the DB
  const conn = await mysql.createConnection(connectionConfig);
  // Obtain the result of the query
  const [res,] = await conn.execute('select ?+? as sum', [3, 2]);
  return res;

}

export const handler = async (event) => {
  // Execute database flow
  const result = await dbOps();
  // Return result
  return {
    statusCode: 200,
    body: JSON.stringify("The selected sum is: " + result[0].sum)
  }
};
```

Herstellen einer Verbindung zu einer Amazon RDS-Datenbank in einer Lambda-Funktion mithilfe von. TypeScript

```
import { Signer } from "@aws-sdk/rds-signer";
import mysql from 'mysql2/promise';

// RDS settings
// Using '!' (non-null assertion operator) to tell the TypeScript compiler that the
 DB settings are not null or undefined,
const proxy_host_name = process.env.PROXY_HOST_NAME!
const port = parseInt(process.env.PORT!)
const db_name = process.env.DB_NAME!
const db_user_name = process.env.DB_USER_NAME!
const aws_region = process.env.AWS_REGION!
```

```
async function createAuthToken(): Promise<string> {

    // Create RDS Signer object
    const signer = new Signer({
        hostname: proxy_host_name,
        port: port,
        region: aws_region,
        username: db_user_name
    });

    // Request authorization token from RDS, specifying the username
    const token = await signer.getAuthToken();
    return token;
}

async function dbOps(): Promise<mysql.QueryResult | undefined> {
    try {
        // Obtain auth token
        const token = await createAuthToken();
        const conn = await mysql.createConnection({
            host: proxy_host_name,
            user: db_user_name,
            password: token,
            database: db_name,
            ssl: 'Amazon RDS' // Ensure you have the CA bundle for SSL connection
        });
        const [rows, fields] = await conn.execute('SELECT ? + ? AS sum', [3, 2]);
        console.log('result:', rows);
        return rows;
    }
    catch (err) {
        console.log(err);
    }
}

export const lambdaHandler = async (event: any): Promise<{ statusCode: number; body:
 string }> => {
    // Execute database flow
    const result = await dbOps();

    // Return error is result is undefined
    if (result == undefined)
```

```
        return {
            statusCode: 500,
            body: JSON.stringify(`Error with connection to DB host`)
        }

    // Return result
    return {
        statusCode: 200,
        body: JSON.stringify(`The selected sum is: ${result[0].sum}`)
    };
};
```

Beispiele für Amazon RDS Data Service mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für JavaScript (v3)

> Zeigt, wie Sie AWS SDK für JavaScript (v3) verwenden, um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte per E-Mail mithilfe von Amazon Simple Email Service (Amazon SES) versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon Redshift-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Erstellen Sie den -Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME", // Required
  NodeType: "NODE_TYPE", //Required
  MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
  MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
 one uppercase letter, and one number
  ClusterType: "CLUSTER_TYPE", // Required
  IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with permissions
 your cluster needs to access other AWS services on your behalf, such as Amazon S3.
  ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
 cluster subnet group to be associated with this cluster. Defaults to 'default' if
 not specified.
  DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
  Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
  try {
```

```
    const data = await redshiftClient.send(new CreateClusterCommand(params));
    console.log(
      `Cluster ${data.Cluster.ClusterIdentifier} successfully created`,
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie CreateCluster in der AWS SDK für JavaScript API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Erstellen Sie den -Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
```

```
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie DeleteCluster in der AWS SDK für JavaScript API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Beschreiben Sie Ihre Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie DescribeClusters in der AWS SDK für JavaScript API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { RedshiftClient } from "@aws-sdk/client-redshift";
```

```
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Ändern Sie einen Cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- API-Details finden Sie ModifyCluster in der AWS SDK für JavaScript API-Referenz.

Beispiele für Amazon Rekognition mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte in Bildern, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden, anhand von Kategorien zu identifizieren. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Amazon S3-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});
```

```
  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
      // Something besides S3 failed.
      throw caught;
    }
  }
};
```

- API-Details finden Sie ListBuckets in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie zunächst alle erforderlichen Module.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

Die vorherigen Importe verweisen auf einige hilfreiche Dienstprogramme. Diese Hilfsprogramme befinden sich lokal in dem GitHub Repository, das am Anfang dieses Abschnitts verlinkt ist. Zu Ihrer Information finden Sie im Folgenden Implementierungen dieser Dienstprogramme.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
```

```
 * @param {{ message: string, choices: { name: string, value: string }[]}} options
 */
select(options) {
  return select(options);
}

/**
 * @param {{ message: string }} options
 */
input(options) {
  return input(options);
}

/**
 * @param {{ message: string }} options
 */
password(options) {
  return password({ ...options, mask: true });
}

/**
 * @param {string} prompt
 */
checkContinue = async (prompt = "") => {
  const prefix = prompt && `${prompt} `;
  const ok = await this.confirm({
    message: `${prefix}Continue?`,
  });
  if (!ok) throw new Error("Exiting...");
};

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[]}} options
 */
checkbox(options) {
  return checkbox(options);
}
```

```
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

Objekte werden in „Buckets“ gespeichert. Definieren wir eine Funktion zum Erstellen eines neuen Buckets.

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

Buckets enthalten „Objekte“. Diese Funktion lädt den Inhalt eines Verzeichnisses als Objekte in Ihren Bucket hoch.

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (const file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
```

```
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

Überprüfen Sie nach dem Hochladen von Objekten, ob sie korrekt hochgeladen wurden. Das kannst du verwenden ListObjects . Sie werden die Eigenschaft „Key“ verwenden, die Antwort enthält jedoch noch weitere nützliche Eigenschaften.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

Gelegentlich möchten Sie vielleicht ein Objekt von einem Bucket in einen anderen kopieren. Benutze den CopyObject Befehl dafür.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });
```

```
      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};
```

Es gibt keine SDK-Methode zum Abrufen mehrerer Objekte aus einem Bucket. Stattdessen erstellen Sie eine Liste von Objekten, die Sie herunterladen und durchlaufen können.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

Es ist Zeit, Ihre Ressourcen zu bereinigen. Ein Bucket muss leer sein, bevor er gelöscht werden kann. Mit diesen beiden Funktionen leeren und löschen Sie den Bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

Die Funktion „main“ fasst alles zusammen. Wenn Sie diese Datei direkt ausführen, wird die Funktion „main“ aufgerufen.

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
```

```
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie das Objekt.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
```

```
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/
${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
      console.error(
        `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the
 active tier.`,
      );
    } else {
      throw caught;
    }
  }
};
```

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
```

```
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket "${sourceBucketName}":
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
```

```
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  CopyObjectCommand,
  NoSuchKey,
```

```
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket "${sourceBucketName}":
 ${caught.name}: ${caught.message}`,
      );
```

```
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
```

```
}
```

Kopieren Sie das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
```

```
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
```

```
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Kopiert das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in 'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string, destinationBucketName:
 string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
```

```
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
 "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
```

```
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie CopyObject in der AWS SDK für JavaScript API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Bucket.

```
import {
  BucketAlreadyExists,
```

```
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
 constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket
 names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
    } else {
      throw caught;
    }
  }
```

```
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateBucket in der AWS SDK für JavaScript API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
```

```
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteBucket in der AWS SDK für JavaScript API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Bucket-Richtlinie.

```
import {
  DeleteBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteBucketPolicy in der AWS SDK für JavaScript API-Referenz.

`DeleteBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketWebsite`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Website-Konfiguration aus dem Bucket.

```
import {
  DeleteBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    // The response code will be successful for both removed configurations and
    // configurations that did not exist in the first place.
    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it never
 was.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}. The
 bucket doesn't exist.`,
```

```
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteBucketWebsite in der AWS SDK für JavaScript API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
```

```
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
 exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteObject in der AWS SDK für JavaScript API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
```

```
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteObjects in der AWS SDK für JavaScript API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die ACL-Berechtigungen ab.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- API-Details finden Sie GetBucketAcl in der AWS SDK für JavaScript API-Referenz.

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketCors`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die CORS-Richtlinie für den Bucket ab.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
```

```
        `\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetBucketCors in der AWS SDK für JavaScript API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Bucket-Richtlinie ab.

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetBucketPolicy in der AWS SDK für JavaScript API-Referenz.

`GetBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketWebsite`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Website-Konfiguration ab.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:\n
${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
```

```
        `Error from S3 while getting website configuration for ${bucketName}. The
 bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie GetBucketWebsite in der AWS SDK für JavaScript API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie das Objekt herunter.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
```

```
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
```

```
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
```

```
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
```

```
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
```

```
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
```

```
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
```

```
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
```

```
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetObject in der AWS SDK für JavaScript API-Referenz.

`GetObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLegalHold`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
```

```
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in ${bucketName}
 from ${bucketName}.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
```

```
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie GetObjectLegalHold in der AWS SDK für JavaScript API-Referenz.

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLockConfiguration`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the bucket>",
      }),
    );
    console.log(
      `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object lock configuration for ${bucketName}.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```

```
const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS SDK für JavaScript API-Referenz.

`GetObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectRetention`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetObjectRetentionCommand,
```

```
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
 ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for "${bucketName}".
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
```

```
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie GetObjectRetention in der AWS SDK für JavaScript API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die Buckets auf.

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing buckets.  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
```

```
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListBuckets in der AWS SDK für JavaScript API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Objekte in Ihrem Bucket auf. Wenn es mehr als ein Objekt gibt, IsTruncated NextContinuationToken wird es verwendet, um über die gesamte Liste zu iterieren.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
```

```
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
      objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
      console.log(
        `Page ${pageNum + 1}\n------\n${objectList.map((o) => `•
 ${o}`).join("\n")}\n`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}". The bucket doesn't
 exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while listing objects for "${bucketName}".  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für JavaScript API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Bucket-ACL fest.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of your
 AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
```

```
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
 doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutBucketAcl in der AWS SDK für JavaScript API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie eine CORS-Regel hinzu.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
              AllowedMethods: ["GET", "PUT"],
              // Allow only requests from the specified origin.
              AllowedOrigins: ["https://www.example.com"],
              // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
```

```
              // The entity tag represents a specific version of the object. The
ETag reflects
              // changes only to the contents of an object, not its metadata.
              ExposeHeaders: ["ETag"],
              // How long the requesting browser should cache the preflight
response. After
              // this time, the preflight request will have to be made again.
              MaxAgeSeconds: 3600,
            },
          ],
        },
      },
    );
    console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}. The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting CORS rules for ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutBucketCors in der AWS SDK für JavaScript API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie die Richtlinie hinzu.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
 policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            AWS: iamRoleArn,
          },
          Action: "s3:GetObject",
          Resource: `arn:aws:s3:::${bucketName}/*`,
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: bucketName,
```

```
  });

  try {
    await client.send(command);
    console.log(
      `GetObject access to the bucket "${bucketName}" was granted to the provided
 IAM role.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "MalformedPolicy"
    ) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". The policy was malformed.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting the bucket policy for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutBucketPolicy in der AWS SDK für JavaScript API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Website-Konfiguration fest.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
```

```
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static
 website. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutBucketWebsite in der AWS SDK für JavaScript API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie das Objekt hoch.

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console
 (160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}.  ${caught.name}:
 ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
```

```
};
```

Laden Sie das Objekt unter der Bedingung hoch, dass sein ETag mit dem angegebenen übereinstimmt.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No such
 key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
 ${caught.message}`,
```

```
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie PutObject in der AWS SDK für JavaScript API-Referenz.

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLegalHold`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" | "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'.",
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
```

```
  });

  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
 "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
      required: true,
```

```
    },
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie PutObjectLegalHold in der AWS SDK für JavaScript API-Referenz.

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLockConfiguration`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the bucket
 "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the bucket
 "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
```

```
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
```

```
     new PutObjectLockConfigurationCommand({
       Bucket: bucketName,
       // The Object Lock configuration that you want to apply to the specified
bucket.
       ObjectLockConfiguration: {
         ObjectLockEnabled: "Enabled",
         Rule: {
           // The default Object Lock retention mode and period that you want to
apply
           // to new objects placed in the specified bucket. Bucket settings
require
           // both a mode and a period. The period can be either Days or Years but
           // you must select one.
           DefaultRetention: {
             // In governance mode, users can't overwrite or delete an object
version
             // or alter its lock settings unless they have special permissions.
With
             // governance mode, you protect objects against being deleted by most
users,
             // but you can still grant some users permission to alter the
retention settings
             // or delete the objects if necessary.
             Mode: "GOVERNANCE",
             Days: Number.parseInt(retentionDays),
           },
         },
       },
     }),
   );
   console.log(
     `Set default retention mode to "GOVERNANCE" with a retention period of
${retentionDays} day(s).`,
   );
 } catch (caught) {
   if (
     caught instanceof S3ServiceException &&
     caught.name === "NoSuchBucket"
   ) {
     console.error(
       `Error from S3 while setting the default object retention for a bucket. The
bucket doesn't exist.`,
     );
   } else if (caught instanceof S3ServiceException) {
```

```
      console.error(
        `Error from S3 while setting the default object retention for a bucket.
 ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    retentionDays: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS SDK für JavaScript API-Referenz.

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectRetention`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most users,
      // but you can still grant some users permission to alter the retention
 settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });
```

```
  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period on
 an object. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the governance mode and retention period on
 an object. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
```

```
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- API-Details finden Sie PutObjectRetention in der AWS SDK für JavaScript API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte URL, um ein Objekt in einen Bucket hochzuladen.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
```

```
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
```

```
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Erstellen Sie eine vorsignierte URL, um ein Objekt aus einem Bucket herunterzuladen.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};
```

```
/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
 configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine Webseite erstellen, die Amazon-S3-Objekte auflistet

Im folgenden Codebeispiel wird veranschaulicht, wie Sie Amazon-S3-Objekte auf einer Webseite auflisten.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Der folgende Code ist die relevante React-Komponente, die das AWS SDK aufruft. Eine lauffähige Version der Anwendung, die diese Komponente enthält, finden Sie unter dem vorherigen GitHub Link.

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach a
 role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all origins.
 Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
```

```
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- API-Details finden Sie ListObjects in der AWS SDK für JavaScript API-Referenz.

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für JavaScript (v3)

Zeigt, wie Sie das verwenden AWS SDK für JavaScript , um eine React-Anwendung zu erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3

- Amazon SNS
- Amazon SQS
- Amazon Textract

Löschen aller Objekte in einem Bucket

Im folgenden Codebeispiel wird gezeigt, wie Sie alle Objekte aus einem Amazon-S3-Bucket löschen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie alle Objekte für einen gegebenen Amazon-S3-Bucket.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );
```

```
    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - DeleteObjects
  - ListObjectsV2

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte in Bildern, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden, anhand von Kategorien zu identifizieren. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Sperren von Amazon-S3-Objekten

Das folgende Codebeispiel zeigt, wie Sie mit S3-Object-Lock-Features arbeiten.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einstiegspunkt für das Szenario (index.js). Dadurch werden alle Schritte orchestriert. Besuchen Sie uns GitHub , um sich die Implementierungsdetails für Szenario ScenarioInput, ScenarioOutput, und anzusehen ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});
```

```
  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
```

```
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Geben Sie Willkommensnachrichten an die Konsole aus (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
```

```
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Stellen Sie Buckets, Objekte und Dateieinstellungen bereit (setup.steps.js).

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
```

```
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-no-lock with object lock False.
         ${state.bucketPrefix}-lock-enabled with object lock True.
         ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
```

```
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: lockEnabledBucketName,
          ObjectLockEnabledForBucket: true,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
      );
      await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
      );
      await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

      state.noLockBucketName = noLockBucketName;
      state.lockEnabledBucketName = lockEnabledBucketName;
      state.retentionBucketName = retentionBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
```

```
    (state) => `The following test files will be created:
         file0.txt in ${state.bucketPrefix}-no-lock.
         file1.txt in ${state.bucketPrefix}-no-lock.
         file0.txt in ${state.bucketPrefix}-lock-enabled.
         file1.txt in ${state.bucketPrefix}-lock-enabled.
         file0.txt in ${state.bucketPrefix}-retention-after-creation.
         file1.txt in ${state.bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
```

```
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

/**
```

```
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
 retention period.
A default retention period will be configured for ${state.bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
```

```
      if (Status !== "Enabled") {
        throw new Error("Bucket versioning is not enabled.");
      }
    });

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
```

```
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
```

```
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be able
 to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
```

```
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
```

```
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be able
 to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
```

```
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  getBucketPrefix,
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

Zeigen Sie Dateien in den Buckets an und löschen Sie sie (repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
```

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
```

```
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
```

```
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
```

```
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
```

```
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
```

```
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

Vernichten Sie alle erstellten Ressourcen (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */
```

```
/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        }
        throw e;
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;
```

```
    try {
      const legalHold = await client.send(
        new GetObjectLegalHoldCommand({
          Bucket: bucket,
          Key,
          VersionId,
        }),
      );

      if (legalHold.LegalHold?.Status === "ON") {
        await client.send(
          new PutObjectLegalHoldCommand({
            Bucket: bucket,
            Key,
            VersionId,
            LegalHold: {
              Status: "OFF",
            },
          }),
        );
      }
    } catch (err) {
      console.log(
        `Unable to fetch legal hold for ${Key} in ${bucket}: '${err.message}'`,
      );
    }

    try {
      const retention = await client.send(
        new GetObjectRetentionCommand({
          Bucket: bucket,
          Key,
          VersionId,
        }),
      );

      if (retention.Retention?.Mode === "GOVERNANCE") {
        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
            BypassGovernanceRetention: true,
```

```
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Senden von bedingten Anforderungen

Das folgende Codebeispiel zeigt, wie Sie Vorbedingungen zu Amazon-S3-Anforderungen hinzufügen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einstiegspunkt für den Workflow (index.js). Dadurch werden alle Schritte orchestriert. Besuchen Sie uns GitHub , um sich die Implementierungsdetails für Szenario ScenarioInput, ScenarioOutput, und anzusehen ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
```

```
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(scenarios), replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Conditional Requests - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
```

```
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
 features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-
v|--verbose]",
  });
}
```

Geben Sie Willkommensnachrichten an die Konsole aus (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3 operations." +
      " You can use conditional requests to add preconditions to S3 read requests to
 return " +
      "or copy an object based on its Entity tag (ETag), or last modified date.You
 can use " +
      "a conditional write requests to prevent overwrites by ensuring there is no
 existing " +
      "object with the same key.\n" +
      "This example will enable you to perform conditional reads and writes that
 will succeed " +
      "or fail based on your selected options.\n" +
      "Sample buckets and a sample object will be created as part of the example.\n"
 +
      "Some steps require a key name prefix to be defined by the user. Before you
 begin, you can " +
      "optionally edit this prefix in ./object_name.json. If you do so, please
 reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
```

```
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Stellen Sie Buckets und Objekte bereit (setup.steps.js).

```
import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );
/**
 * @param {Scenarios} scenarios
 */
```

```
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
         ${state.bucketPrefix}-source-bucket.
         ${state.bucketPrefix}-destination-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

    try {
      await client.send(
        new CreateBucketCommand({
          Bucket: sourceBucketName,
        }),
      );
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: destinationBucketName,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: destinationBucketName },
      );

      state.sourceBucketName = sourceBucketName;
```

```
      state.destinationBucketName = destinationBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
         file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
```

```
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
 ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Rufen Sie Objekte mithilfe von bedingten S3-Anfragen (repl.steps.js) ab, kopieren Sie sie und fügen Sie sie ein.

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
```

```
import data from "./object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following
 choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
        {
          name: "Perform a conditional copy. These examples use the key name prefix
 defined in ./object_name.json.",
```

```
          value: choices.CONDITIONAL_COPY,
        },
        {
          name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
          value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key } = version;
      files.push({ bucket, key: Key });
    }
  }
  return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
  const objectsResponse = await client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    }),
  );
  return objectsResponse.ETag;
```

```
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );
      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
 succeed.",
```

```
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condCopyOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional copy action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
          ],
        },
      );
      const condWriteOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional write action would you like to take?",
        {
          type: "select",
          choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate, the
write will fail.",
          ],
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
            )
```

```
            .join("\n");
          break;
        }
        case choices.CONDITIONAL_READ:
          {
            const selectedCondRead = await condReadOptions.handle(state);
            if (
              selectedCondRead ===
              "If-Match: using the object's ETag. This condition should succeed."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName} read
because ETag provided matches the object's ETag.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-None-Match: using the object's ETag. This condition should fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              const ETag = await getEtag(client, bucket, key);

              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfNoneMatch: ETag,
```

```
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Modified-Since: using yesterday's date. This condition should
succeed."
            ) {
              const date = new Date();
              date.setDate(date.getDate() - 1);

              const bucket = state.sourceBucketName;
              const key = "file01.txt";
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfModifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in bucket ${state.sourceBucketName} read
because it has been created or modified in the last 24 hours.`;
              } catch (err) {
                state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
              }
              break;
            }
            if (
              selectedCondRead ===
              "If-Unmodified-Since: using yesterday's date. This condition should
fail."
            ) {
              const bucket = state.sourceBucketName;
              const key = "file01.txt";
```

```
              const date = new Date();
              date.setDate(date.getDate() - 1);
              try {
                await client.send(
                  new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfUnmodifiedSince: date,
                  }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was read.`;
              } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
              }
              break;
            }
          }
          break;
        case choices.CONDITIONAL_COPY: {
          const selectedCondCopy = await condCopyOptions.handle(state);
          if (
            selectedCondCopy ===
            "If-Match: using the object's ETag. This condition should succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;
            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfMatch: ETag,
                }),
              );
```

```
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's ETag.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-None-Match: using the object's ETag. This condition should fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfNoneMatch: ETag,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
            } catch (err) {
              state.replOutput = `Unable to copy object as ${key} as as ${copiedKey}
to bucket ${state.destinationBucketName}: ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Modified-Since: using yesterday's date. This condition should
succeed."
          ) {
            const bucket = state.sourceBucketName;
```

```
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);

            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfModifiedSince: date,
                }),
              );
              state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the last
24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
            }
            break;
          }
          if (
            selectedCondCopy ===
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;

            const date = new Date();
            date.setDate(date.getDate() - 1);
```

```
            try {
              await client.send(
                new CopyObjectCommand({
                  CopySource: copySource,
                  Bucket: state.destinationBucketName,
                  Key: copiedKey,
                  CopySourceIfUnmodifiedSince: date,
                }),
              );
              state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in the
last 24 hours.`;
            } catch (err) {
              state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
            }
          }
          break;
        }
        case choices.CONDITIONAL_WRITE:
          {
            const selectedCondWrite = await condWriteOptions.handle(state);
            if (
              selectedCondWrite ===
              "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail."
            ) {
              // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
              const key = "text02.txt";
              const __filename = fileURLToPath(import.meta.url);
              const __dirname = dirname(__filename);
              const filePath = path.join(__dirname, "text02.txt");
              try {
                await client.send(
                  new PutObjectCommand({
                    Bucket: `${state.destinationBucketName}`,
                    Key: `${key}`,
                    Body: await readFile(filePath),
                    IfNoneMatch: "*",
                  }),
                );
                state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
```

```
              } catch (err) {
                state.replOutput = `Unable to upload object to bucket
 ${state.destinationBucketName}:${err.message}`;
              }
              break;
            }
          }
          break;

        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
          preformatted: true,
        }),
      },
    },
  );

export { replInput, choices };
```

Vernichten Sie alle erstellten Ressourcen (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */
```

```
/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
            console.log(`An error occurred: ${err.message} `);
          }
        }
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Objects and buckets have already been deleted.");
          continue;
        }
```

```
        throw e;
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CopyObject
  - GetObject
  - PutObject

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine große Datei hoch.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
```

```
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Laden Sie eine große Datei herunter.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };
```

```
  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines S3-Ereignisses mit Lambda. JavaScript

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
```

```
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Verbrauch eines S3-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

SageMaker KI-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit SageMaker KI Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo SageMaker KI

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von SageMaker KI beginnen können.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  SageMakerClient,
  ListNotebookInstancesCommand,
} from "@aws-sdk/client-sagemaker";
```

```
const client = new SageMakerClient({
  region: "us-west-2",
});

export const helloSagemaker = async () => {
  const command = new ListNotebookInstancesCommand({ MaxResults: 5 });

  const response = await client.send(command);
  console.log(
    "Hello Amazon SageMaker! Let's list some of your notebook instances:",
  );

  const instances = response.NotebookInstances || [];

  if (instances.length === 0) {
    console.log(
      "• No notebook instances found. Try creating one in the AWS Management Console
 or with the CreateNotebookInstanceCommand.",
    );
  } else {
    console.log(
      instances
        .map(
          (i) =>
            `• Instance: ${i.NotebookInstanceName}\n  Arn:${
              i.NotebookInstanceArn
            } \n  Creation Date: ${i.CreationTime.toISOString()}`,
        )
        .join("\n"),
    );
  }

  return response;
};
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK für JavaScript API-Referenz.

Aktionen

`CreatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`CreatePipeline`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Funktion, die eine SageMaker AI-Pipeline mithilfe einer lokal bereitgestellten JSON-Definition erstellt.

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
```

```
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}
```

- API-Details finden Sie CreatePipeline in der AWS SDK für JavaScript API-Referenz.

`DeletePipeline`

Das folgende Codebeispiel zeigt die Verwendung`DeletePipeline`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Syntax zum Löschen einer SageMaker AI-Pipeline. Dieser Code ist Teil einer größeren Funktion. Weitere Informationen finden Sie unter „Eine Pipeline erstellen“ oder im GitHub Repository.

```
    await sagemakerClient.send(
      new DeletePipelineCommand({ PipelineName: name }),
    );
```

- API-Details finden Sie DeletePipeline in AWS SDK für JavaScript der API-Referenz.

`DescribePipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribePipelineExecution`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Warten Sie, bis die Ausführung einer SageMaker KI-Pipeline erfolgreich ist, fehlschlägt oder beendet wird.

```
/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
```

```
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before checking
 again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}
```

- API-Details finden Sie DescribePipelineExecution in der AWS SDK für JavaScript API-Referenz.

StartPipelineExecution

Das folgende Codebeispiel zeigt die VerwendungStartPipelineExecution.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie eine SageMaker AI-Pipeline-Ausführung.

```
/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };
```

```
  /**
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse Geocoding
 requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
```

```
          Value: JSON.stringify(jobConfig),
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}
```

- API-Details finden Sie StartPipelineExecution in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erste Schritte mit Geodatenaufträgen und Pipelines

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter Erstellen und Ausführen von SageMaker Pipelines mithilfe von AWS SDKs. Community.aws

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Der folgende Dateiauszug enthält Funktionen, die den SageMaker AI-Client zur Verwaltung einer Pipeline verwenden.

```
import { readFileSync } from "node:fs";

import {
  CreateRoleCommand,
  DeleteRoleCommand,
  CreatePolicyCommand,
  DeletePolicyCommand,
  AttachRolePolicyCommand,
  DetachRolePolicyCommand,
  GetRoleCommand,
  ListPoliciesCommand,
} from "@aws-sdk/client-iam";

import {
  PublishLayerVersionCommand,
  DeleteLayerVersionCommand,
  CreateFunctionCommand,
  Runtime,
  DeleteFunctionCommand,
  CreateEventSourceMappingCommand,
  DeleteEventSourceMappingCommand,
  GetFunctionCommand,
} from "@aws-sdk/client-lambda";

import {
  PutObjectCommand,
  CreateBucketCommand,
  DeleteBucketCommand,
  DeleteObjectCommand,
  GetObjectCommand,
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

import {
  CreatePipelineCommand,
  DeletePipelineCommand,
  DescribePipelineCommand,
  DescribePipelineExecutionCommand,
  PipelineExecutionStatus,
  StartPipelineExecutionCommand,
```

```
} from "@aws-sdk/client-sagemaker";

import { VectorEnrichmentJobDocumentType } from "@aws-sdk/client-sagemaker-
geospatial";

import {
  CreateQueueCommand,
  DeleteQueueCommand,
  GetQueueAttributesCommand,
  GetQueueUrlCommand,
} from "@aws-sdk/client-sqs";

import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * Create the AWS IAM role that will be assumed by AWS Lambda.
 * @param {{ name: string, iamClient: import('@aws-sdk/client-iam').IAMClient }}
 props
 */
export async function createLambdaExecutionRole({ name, iamClient }) {
  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: { Service: ["lambda.amazonaws.com"] },
            },
          ],
        }),
      }),
    );

  let role = null;

  try {
    const { Role } = await createRole();
    role = Role;
  } catch (caught) {
```

```
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create an AWS IAM policy that will be attached to the AWS IAM role assumed by the
 AWS Lambda function.
 * The policy grants permission to work with Amazon SQS, Amazon CloudWatch, and
 Amazon SageMaker.
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient,
 pipelineExecutionRoleArn: string}} props
 */
export async function createLambdaExecutionPolicy({
  name,
  iamClient,
  pipelineExecutionRoleArn,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: [
          "sqs:ReceiveMessage",
          "sqs:DeleteMessage",
          "sqs:GetQueueAttributes",
          "logs:CreateLogGroup",
```

```
          "logs:CreateLogStream",
          "logs:PutLogEvents",
          "sagemaker-geospatial:StartVectorEnrichmentJob",
          "sagemaker-geospatial:GetVectorEnrichmentJob",
          "sagemaker:SendPipelineExecutionStepFailure",
          "sagemaker:SendPipelineExecutionStepSuccess",
          "sagemaker-geospatial:ExportVectorEnrichmentJob",
        ],
        Resource: "*",
      },
      {
        Effect: "Allow",
        // The AWS Lambda function needs permission to pass the pipeline execution
 role to
        // the StartVectorEnrichmentCommand. This restriction prevents an AWS Lambda
 function
        // from elevating privileges. For more information, see:
        // https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_use_passrole.html
        Action: ["iam:PassRole"],
        Resource: `${pipelineExecutionRoleArn}`,
        Condition: {
          StringEquals: {
            "iam:PassedToService": [
              "sagemaker.amazonaws.com",
              "sagemaker-geospatial.amazonaws.com",
            ],
          },
        },
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
```

```
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}

/**
 * Attach an AWS IAM policy to an AWS IAM role.
 * @param {{roleName: string, policyArn: string, iamClient: import('@aws-sdk/client-
iam').IAMClient}} props
 */
export async function attachPolicy({ roleName, policyArn, iamClient }) {
  const attachPolicyCommand = new AttachRolePolicyCommand({
    RoleName: roleName,
    PolicyArn: policyArn,
  });

  await iamClient.send(attachPolicyCommand);
  return {
    cleanUp: async () => {
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: roleName,
```

```
          PolicyArn: policyArn,
        }),
      );
    },
  };
}

/**
 * Create an AWS Lambda layer that contains the Amazon SageMaker and Amazon
 SageMaker Geospatial clients
 * in the runtime. The default runtime supports v3.188.0 of the JavaScript SDK. The
 Amazon SageMaker
 * Geospatial client wasn't introduced until v3.221.0.
 * @param {{ name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient }} props
 */
export async function createLambdaLayer({ name, lambdaClient }) {
  const layerPath = `${dirnameFromMetaUrl(import.meta.url)}lambda/nodejs.zip`;
  const { LayerVersionArn, Version } = await lambdaClient.send(
    new PublishLayerVersionCommand({
      LayerName: name,
      Content: {
        ZipFile: Uint8Array.from(readFileSync(layerPath)),
      },
    }),
  );

  return {
    versionArn: LayerVersionArn,
    version: Version,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteLayerVersionCommand({
          LayerName: name,
          VersionNumber: Version,
        }),
      );
    },
  };
}

/**
 * Deploy the AWS Lambda function that will be used to respond to Amazon SageMaker
 pipeline
```

```
 * execution steps.
 * @param {{roleArn: string, name: string, lambdaClient: import('@aws-sdk/client-
lambda').LambdaClient, layerVersionArn: string}} props
 */
export async function createLambdaFunction({
  name,
  roleArn,
  lambdaClient,
  layerVersionArn,
}) {
  const lambdaPath = `${dirnameFromMetaUrl(
    import.meta.url,
  )}lambda/dist/index.mjs.zip`;

  // If a function of the same name already exists, return that
  // function's ARN instead. By default this is
  // "sagemaker-wkflw-lambda-function", so collisions are
  // unlikely.
  const createFunction = async () => {
    try {
      return await lambdaClient.send(
        new CreateFunctionCommand({
          Code: {
            ZipFile: Uint8Array.from(readFileSync(lambdaPath)),
          },
          Runtime: Runtime.nodejs18x,
          Handler: "index.handler",
          Layers: [layerVersionArn],
          FunctionName: name,
          Role: roleArn,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "ResourceConflictException"
      ) {
        const { Configuration } = await lambdaClient.send(
          new GetFunctionCommand({ FunctionName: name }),
        );
        return Configuration;
      }
      throw caught;
    }
```

```
  };

  // Function creation fails if the Role is not ready. This retries
  // function creation until it succeeds or it times out.
  const { FunctionArn } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    createFunction,
  );

  return {
    arn: FunctionArn,
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteFunctionCommand({ FunctionName: name }),
      );
    },
  };
}

/**
 * This uploads some sample coordinate data to an Amazon S3 bucket.
 * The Amazon SageMaker Geospatial vector enrichment job will take the simple Lat/
Long
 * coordinates in this file and augment them with more detailed location data.
 * @param {{bucketName: string, s3Client: import('@aws-sdk/client-s3').S3Client}}
 props
 */
export async function uploadCSVDataToS3({ bucketName, s3Client }) {
  const s3Path = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../../scenarios/features/sagemaker_pipelines/resources/
latlongtest.csv`;

  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Key: "input/sample_data.csv",
      Body: readFileSync(s3Path),
    }),
  );
}

/**
 * Create the AWS IAM role that will be assumed by the Amazon SageMaker pipeline.
```

```
 * @param {{name: string, iamClient: import('@aws-sdk/client-iam').IAMClient, wait:
 (ms: number) => Promise<void>}} props
 */
export async function createSagemakerRole({ name, iamClient, wait }) {
  let role = null;

  const createRole = () =>
    iamClient.send(
      new CreateRoleCommand({
        RoleName: name,
        AssumeRolePolicyDocument: JSON.stringify({
          Version: "2012-10-17",
          Statement: [
            {
              Effect: "Allow",
              Action: ["sts:AssumeRole"],
              Principal: {
                Service: [
                  "sagemaker.amazonaws.com",
                  "sagemaker-geospatial.amazonaws.com",
                ],
              },
            },
          ],
        }),
      }),
    );

  try {
    const { Role } = await createRole();
    role = Role;
    // Wait for the role to be ready.
    await wait(10);
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Role } = await iamClient.send(
        new GetRoleCommand({ RoleName: name }),
      );
      role = Role;
    } else {
      throw caught;
```

```
    }
  }

  return {
    arn: role.Arn,
    cleanUp: async () => {
      await iamClient.send(new DeleteRoleCommand({ RoleName: name }));
    },
  };
}

/**
 * Create the Amazon SageMaker execution policy. This policy grants permission to
 * invoke the AWS Lambda function, read/write to the Amazon S3 bucket, and send
 messages to
 * the Amazon SQS queue.
 * @param {{ name: string, sqsQueueArn: string, lambdaArn: string, iamClient:
 import('@aws-sdk/client-iam').IAMClient, s3BucketName: string}} props
 */
export async function createSagemakerExecutionPolicy({
  sqsQueueArn,
  lambdaArn,
  iamClient,
  name,
  s3BucketName,
}) {
  const policyConfig = {
    Version: "2012-10-17",
    Statement: [
      {
        Effect: "Allow",
        Action: ["lambda:InvokeFunction"],
        Resource: lambdaArn,
      },
      {
        Effect: "Allow",
        Action: ["s3:*"],
        Resource: [
          `arn:aws:s3:::${s3BucketName}`,
          `arn:aws:s3:::${s3BucketName}/*`,
        ],
      },
      {
        Effect: "Allow",
```

```
        Action: ["sqs:SendMessage"],
        Resource: sqsQueueArn,
      },
    ],
  };

  const createPolicy = () =>
    iamClient.send(
      new CreatePolicyCommand({
        PolicyDocument: JSON.stringify(policyConfig),
        PolicyName: name,
      }),
    );

  let policy = null;

  try {
    const { Policy } = await createPolicy();
    policy = Policy;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "EntityAlreadyExistsException"
    ) {
      const { Policies } = await iamClient.send(new ListPoliciesCommand({}));
      if (Policies) {
        policy = Policies.find((p) => p.PolicyName === name);
      } else {
        throw new Error("No policies found.");
      }
    } else {
      throw caught;
    }
  }

  return {
    arn: policy?.Arn,
    policyConfig,
    cleanUp: async () => {
      await iamClient.send(new DeletePolicyCommand({ PolicyArn: policy?.Arn }));
    },
  };
}
```

```
/**
 * Create the Amazon SageMaker pipeline using a JSON pipeline definition. The
 definition
 * can also be provided as an Amazon S3 object using PipelineDefinitionS3Location.
 * @param {{roleArn: string, name: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient}} props
 */
export async function createSagemakerPipeline({
  // Assumes an AWS IAM role has been created for this pipeline.
  roleArn,
  name,
  // Assumes an AWS Lambda function has been created for this pipeline.
  functionArn,
  sagemakerClient,
}) {
  const pipelineDefinition = readFileSync(
    // dirnameFromMetaUrl is a local utility function. You can find its
 implementation
    // on GitHub.
    `${dirnameFromMetaUrl(
      import.meta.url,
    )}../../../../../scenarios/features/sagemaker_pipelines/resources/
GeoSpatialPipeline.json`,
  )
    .toString()
    .replace(/\*FUNCTION_ARN\*/g, functionArn);

  let arn = null;

  const createPipeline = () =>
    sagemakerClient.send(
      new CreatePipelineCommand({
        PipelineName: name,
        PipelineDefinition: pipelineDefinition,
        RoleArn: roleArn,
      }),
    );

  try {
    const { PipelineArn } = await createPipeline();
    arn = PipelineArn;
  } catch (caught) {
    if (
      caught instanceof Error &&
```

```
      caught.name === "ValidationException" &&
      caught.message.includes(
        "Pipeline names must be unique within an AWS account and region",
      )
    ) {
      const { PipelineArn } = await sagemakerClient.send(
        new DescribePipelineCommand({ PipelineName: name }),
      );
      arn = PipelineArn;
    } else {
      throw caught;
    }
  }

  return {
    arn,
    cleanUp: async () => {
      await sagemakerClient.send(
        new DeletePipelineCommand({ PipelineName: name }),
      );
    },
  };
}

/**
 * Create an Amazon SQS queue. The Amazon SageMaker pipeline will send messages
 * to this queue that are then processed by the AWS Lambda function.
 * @param {{name: string, sqsClient: import('@aws-sdk/client-sqs').SQSClient}} props
 */
export async function createSQSQueue({ name, sqsClient }) {
  const createSqsQueue = () =>
    sqsClient.send(
      new CreateQueueCommand({
        QueueName: name,
        Attributes: {
          DelaySeconds: "5",
          ReceiveMessageWaitTimeSeconds: "5",
          VisibilityTimeout: "300",
        },
      }),
    );

  let queueUrl = null;
  try {
```

```
    const { QueueUrl } = await createSqsQueue();
    queueUrl = QueueUrl;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "QueueNameExists") {
      const { QueueUrl } = await sqsClient.send(
        new GetQueueUrlCommand({ QueueName: name }),
      );
      queueUrl = QueueUrl;
    } else {
      throw caught;
    }
  }

  const { Attributes } = await retry(
    { intervalInMs: 1000, maxRetries: 60 },
    () =>
      sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: queueUrl,
          AttributeNames: ["QueueArn"],
        }),
      ),
  );

  return {
    queueUrl,
    queueArn: Attributes.QueueArn,
    cleanUp: async () => {
      await sqsClient.send(new DeleteQueueCommand({ QueueUrl: queueUrl }));
    },
  };
}

/**
 * Configure the AWS Lambda function to long poll for messages from the Amazon SQS
 * queue.
 * @param {{
 *   paginateListEventSourceMappings: () => Generator<import('@aws-sdk/client-
lambda').ListEventSourceMappingsCommandOutput>,
 *   lambdaName: string,
 *   queueArn: string,
 *   lambdaClient: import('@aws-sdk/client-lambda').LambdaClient}} props
 */
export async function configureLambdaSQSEventSource({
```

```
  lambdaName,
  queueArn,
  lambdaClient,
  paginateListEventSourceMappings,
}) {
  let uuid = null;
  const createEvenSourceMapping = () =>
    lambdaClient.send(
      new CreateEventSourceMappingCommand({
        EventSourceArn: queueArn,
        FunctionName: lambdaName,
      }),
    );

  try {
    const { UUID } = await createEvenSourceMapping();
    uuid = UUID;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "ResourceConflictException"
    ) {
      const paginator = paginateListEventSourceMappings(
        { client: lambdaClient },
        {},
      );
      /**
       * @type {import('@aws-sdk/client-lambda').EventSourceMappingConfiguration[]}
       */
      const eventSourceMappings = [];
      for await (const page of paginator) {
        eventSourceMappings.concat(page.EventSourceMappings || []);
      }

      const { Configuration } = await lambdaClient.send(
        new GetFunctionCommand({ FunctionName: lambdaName }),
      );

      uuid = eventSourceMappings.find(
        (mapping) =>
          mapping.EventSourceArn === queueArn &&
          mapping.FunctionArn === Configuration.FunctionArn,
      ).UUID;
    } else {
```

```
      throw caught;
    }
  }

  return {
    cleanUp: async () => {
      await lambdaClient.send(
        new DeleteEventSourceMappingCommand({
          UUID: uuid,
        }),
      );
    },
  };
}

/**
 * Create an Amazon S3 bucket that will store the simple coordinate file as input
 * and the output of the Amazon SageMaker Geospatial vector enrichment job.
 * @param {{
 *   s3Client: import('@aws-sdk/client-s3').S3Client,
 *   name: string,
 *   paginateListObjectsV2: () => Generator<import('@aws-sdk/client-
s3').ListObjectsCommandOutput>
 * }} props
 */
export async function createS3Bucket({
  name,
  s3Client,
  paginateListObjectsV2,
}) {
  await s3Client.send(new CreateBucketCommand({ Bucket: name }));

  return {
    cleanUp: async () => {
      const paginator = paginateListObjectsV2(
        { client: s3Client },
        { Bucket: name },
      );
      for await (const page of paginator) {
        const objects = page.Contents;
        if (objects) {
          for (const object of objects) {
            await s3Client.send(
              new DeleteObjectCommand({ Bucket: name, Key: object.Key }),
```

```
            );
          }
        }
      }
      await s3Client.send(new DeleteBucketCommand({ Bucket: name }));
    },
  };
}

/**
 * Start the execution of the Amazon SageMaker pipeline. Parameters that are
 * passed in are used in the AWS Lambda function.
 * @param {{
 *   name: string,
 *   sagemakerClient: import('@aws-sdk/client-sagemaker').SageMakerClient,
 *   roleArn: string,
 *   queueUrl: string,
 *   s3InputBucketName: string,
 * }} props
 */
export async function startPipelineExecution({
  sagemakerClient,
  name,
  bucketName,
  roleArn,
  queueUrl,
}) {
  /**
   * The Vector Enrichment Job requests CSV data. This configuration points to a CSV
   * file in an Amazon S3 bucket.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobInputConfig}
   */
  const inputConfig = {
    DataSourceConfig: {
      S3Data: {
        S3Uri: `s3://${bucketName}/input/sample_data.csv`,
      },
    },
    DocumentType: VectorEnrichmentJobDocumentType.CSV,
  };

  /**
```

```
   * The Vector Enrichment Job adds additional data to the source CSV. This
 configuration points
   * to an Amazon S3 prefix where the output will be stored.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").ExportVectorEnrichmentJobOutputConfig}
   */
  const outputConfig = {
    S3Data: {
      S3Uri: `s3://${bucketName}/output/`,
    },
  };

  /**
   * This job will be a Reverse Geocoding Vector Enrichment Job. Reverse Geocoding
 requires
   * latitude and longitude values.
   * @type {import("@aws-sdk/client-sagemaker-
geospatial").VectorEnrichmentJobConfig}
   */
  const jobConfig = {
    ReverseGeocodingConfig: {
      XAttributeName: "Longitude",
      YAttributeName: "Latitude",
    },
  };

  const { PipelineExecutionArn } = await sagemakerClient.send(
    new StartPipelineExecutionCommand({
      PipelineName: name,
      PipelineExecutionDisplayName: `${name}-example-execution`,
      PipelineParameters: [
        { Name: "parameter_execution_role", Value: roleArn },
        { Name: "parameter_queue_url", Value: queueUrl },
        {
          Name: "parameter_vej_input_config",
          Value: JSON.stringify(inputConfig),
        },
        {
          Name: "parameter_vej_export_config",
          Value: JSON.stringify(outputConfig),
        },
        {
          Name: "parameter_step_1_vej_config",
          Value: JSON.stringify(jobConfig),
```

```
        },
      ],
    }),
  );

  return {
    arn: PipelineExecutionArn,
  };
}

/**
 * Poll the executing pipeline until the status is 'SUCCEEDED', 'STOPPED', or
 'FAILED'.
 * @param {{ arn: string, sagemakerClient: import('@aws-sdk/client-
sagemaker').SageMakerClient, wait: (ms: number) => Promise<void>}} props
 */
export async function waitForPipelineComplete({ arn, sagemakerClient, wait }) {
  const command = new DescribePipelineExecutionCommand({
    PipelineExecutionArn: arn,
  });

  let complete = false;
  const intervalInSeconds = 15;
  const COMPLETION_STATUSES = [
    PipelineExecutionStatus.FAILED,
    PipelineExecutionStatus.STOPPED,
    PipelineExecutionStatus.SUCCEEDED,
  ];

  do {
    const { PipelineExecutionStatus: status, FailureReason } =
      await sagemakerClient.send(command);

    complete = COMPLETION_STATUSES.includes(status);

    if (!complete) {
      console.log(
        `Pipeline is ${status}. Waiting ${intervalInSeconds} seconds before checking
 again.`,
      );
      await wait(intervalInSeconds);
    } else if (status === PipelineExecutionStatus.FAILED) {
      throw new Error(`Pipeline failed because: ${FailureReason}`);
    } else if (status === PipelineExecutionStatus.STOPPED) {
```

```
      throw new Error("Pipeline was forcefully stopped.");
    } else {
      console.log(`Pipeline execution ${status}.`);
    }
  } while (!complete);
}

/**
 * Return the string value of an Amazon S3 object.
 * @param {{ bucket: string, key: string, s3Client: import('@aws-sdk/client-
s3').S3Client}} param0
 */
export async function getObject({ bucket, s3Client }) {
  const prefix = "output/";
  const { Contents } = await s3Client.send(
    new ListObjectsV2Command({ MaxKeys: 1, Bucket: bucket, Prefix: prefix }),
  );

  if (!Contents.length) {
    throw new Error("No objects found in bucket.");
  }

  // Find the CSV file.
  const outputObject = Contents.find((obj) => obj.Key.endsWith(".csv"));

  if (!outputObject) {
    throw new Error(`No CSV file found in bucket with the prefix "${prefix}".`);
  }

  const { Body } = await s3Client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: outputObject.Key,
    }),
  );

  return Body.transformToString();
}
```

Diese Funktion ist ein Auszug aus einer Datei, die die vorherigen Bibliotheksfunktionen verwendet, um eine SageMaker AI-Pipeline einzurichten, sie auszuführen und alle erstellten Ressourcen zu löschen.

```
import { retry, wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  attachPolicy,
  configureLambdaSQSEventSource,
  createLambdaExecutionPolicy,
  createLambdaExecutionRole,
  createLambdaFunction,
  createLambdaLayer,
  createS3Bucket,
  createSQSQueue,
  createSagemakerExecutionPolicy,
  createSagemakerPipeline,
  createSagemakerRole,
  getObject,
  startPipelineExecution,
  uploadCSVDataToS3,
  waitForPipelineComplete,
} from "./lib.js";
import { MESSAGES } from "./messages.js";

export class SageMakerPipelinesWkflw {
  names = {
    LAMBDA_EXECUTION_ROLE: "sagemaker-wkflw-lambda-execution-role",
    LAMBDA_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-lambda-execution-role-policy",
    LAMBDA_FUNCTION: "sagemaker-wkflw-lambda-function",
    LAMBDA_LAYER: "sagemaker-wkflw-lambda-layer",
    SAGE_MAKER_EXECUTION_ROLE: "sagemaker-wkflw-pipeline-execution-role",
    SAGE_MAKER_EXECUTION_ROLE_POLICY:
      "sagemaker-wkflw-pipeline-execution-role-policy",
    SAGE_MAKER_PIPELINE: "sagemaker-wkflw-pipeline",
    SQS_QUEUE: "sagemaker-wkflw-sqs-queue",
    S3_BUCKET: `sagemaker-wkflw-s3-bucket-${Date.now()}`,
  };

  cleanUpFunctions = [];

  /**
   * @param {import("@aws-doc-sdk-examples/lib/prompter.js").Prompter} prompter
   * @param {import("@aws-doc-sdk-examples/lib/logger.js").Logger} logger
   * @param {{ IAM: import("@aws-sdk/client-iam").IAMClient, Lambda: import("@aws-
sdk/client-lambda").LambdaClient, SageMaker: import("@aws-sdk/client-
```

```
sagemaker").SageMakerClient, S3: import("@aws-sdk/client-s3").S3Client, SQS:
 import("@aws-sdk/client-sqs").SQSClient }} clients
   */
  constructor(prompter, logger, clients) {
    this.prompter = prompter;
    this.logger = logger;
    this.clients = clients;
  }

  async run() {
    try {
      await this.startWorkflow();
    } catch (err) {
      console.error(err);
      throw err;
    } finally {
      this.logger.logSeparator();
      const doCleanUp = await this.prompter.confirm({
        message: "Clean up resources?",
      });
      if (doCleanUp) {
        await this.cleanUp();
      }
    }
  }

  async cleanUp() {
    // Run all of the clean up functions. If any fail, we log the error and
 continue.
    // This ensures all clean up functions are run.
    for (let i = this.cleanUpFunctions.length - 1; i >= 0; i--) {
      await retry(
        { intervalInMs: 1000, maxRetries: 60, swallowError: true },
        this.cleanUpFunctions[i],
      );
    }
  }

  async startWorkflow() {
    this.logger.logSeparator(MESSAGES.greetingHeader);
    await this.logger.log(MESSAGES.greeting);

    this.logger.logSeparator();
    await this.logger.log(
```

```
      MESSAGES.creatingRole.replace(
        "${ROLE_NAME}",
        this.names.LAMBDA_EXECUTION_ROLE,
      ),
    );

    // Create an IAM role that will be assumed by the AWS Lambda function. This
function
    // is triggered by Amazon SQS messages and calls SageMaker and SageMaker
GeoSpatial actions.
    const { arn: lambdaExecutionRoleArn, cleanUp: lambdaExecutionRoleCleanUp } =
      await createLambdaExecutionRole({
        name: this.names.LAMBDA_EXECUTION_ROLE,
        iamClient: this.clients.IAM,
      });
    // Add a clean up step to a stack for every resource created.
    this.cleanUpFunctions.push(lambdaExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.LAMBDA_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingRole.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    // Create an IAM role that will be assumed by the SageMaker pipeline. The
pipeline
    // sends messages to an Amazon SQS queue and puts/retrieves Amazon S3 objects.
    const {
      arn: pipelineExecutionRoleArn,
      cleanUp: pipelineExecutionRoleCleanUp,
    } = await createSagemakerRole({
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE,
      wait,
```

```
    });
    this.cleanUpFunctions.push(pipelineExecutionRoleCleanUp);

    await this.logger.log(
      MESSAGES.roleCreated.replace(
        "${ROLE_NAME}",
        this.names.SAGE_MAKER_EXECUTION_ROLE,
      ),
    );

    this.logger.logSeparator();

    // Create an IAM policy that allows the AWS Lambda function to invoke SageMaker
APIs.
    const {
      arn: lambdaExecutionPolicyArn,
      policy: lambdaPolicy,
      cleanUp: lambdaExecutionPolicyCleanUp,
    } = await createLambdaExecutionPolicy({
      name: this.names.LAMBDA_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
      iamClient: this.clients.IAM,
      pipelineExecutionRoleArn,
    });
    this.cleanUpFunctions.push(lambdaExecutionPolicyCleanUp);

    console.log(JSON.stringify(lambdaPolicy, null, 2), "\n");

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.LAMBDA_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.LAMBDA_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the Lambda execution policy to the execution role.
    const { cleanUp: lambdaExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.LAMBDA_EXECUTION_ROLE,
      policyArn: lambdaExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(lambdaExecutionRolePolicyCleanUp);
```

```
    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    // Create Lambda layer for SageMaker packages.
    const { versionArn: layerVersionArn, cleanUp: lambdaLayerCleanUp } =
      await createLambdaLayer({
        name: this.names.LAMBDA_LAYER,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaLayerCleanUp);

    await this.logger.log(
      MESSAGES.creatingFunction.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    // Create the Lambda function with the execution role.
    const { arn: lambdaArn, cleanUp: lambdaCleanUp } =
      await createLambdaFunction({
        roleArn: lambdaExecutionRoleArn,
        lambdaClient: this.clients.Lambda,
        name: this.names.LAMBDA_FUNCTION,
        layerVersionArn,
      });
    this.cleanUpFunctions.push(lambdaCleanUp);

    await this.logger.log(
      MESSAGES.functionCreated.replace(
        "${FUNCTION_NAME}",
        this.names.LAMBDA_FUNCTION,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingSQSQueue.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Create an SQS queue for the SageMaker pipeline.
    const {
```

```
      queueUrl,
      queueArn,
      cleanUp: queueCleanUp,
    } = await createSQSQueue({
      name: this.names.SQS_QUEUE,
      sqsClient: this.clients.SQS,
    });
    this.cleanUpFunctions.push(queueCleanUp);

    await this.logger.log(
      MESSAGES.sqsQueueCreated.replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.configuringLambdaSQSEventSource
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    // Configure the SQS queue as an event source for the Lambda.
    const { cleanUp: lambdaSQSEventSourceCleanUp } =
      await configureLambdaSQSEventSource({
        lambdaArn,
        lambdaName: this.names.LAMBDA_FUNCTION,
        queueArn,
        sqsClient: this.clients.SQS,
        lambdaClient: this.clients.Lambda,
      });
    this.cleanUpFunctions.push(lambdaSQSEventSourceCleanUp);

    await this.logger.log(
      MESSAGES.lambdaSQSEventSourceConfigured
        .replace("${LAMBDA_NAME}", this.names.LAMBDA_FUNCTION)
        .replace("${QUEUE_NAME}", this.names.SQS_QUEUE),
    );

    this.logger.logSeparator();

    // Create an IAM policy that allows the SageMaker pipeline to invoke AWS Lambda
    // and send messages to the Amazon SQS queue.
    const {
      arn: pipelineExecutionPolicyArn,
```

```
      policy: sagemakerPolicy,
      cleanUp: pipelineExecutionPolicyCleanUp,
    } = await createSagemakerExecutionPolicy({
      sqsQueueArn: queueArn,
      lambdaArn,
      iamClient: this.clients.IAM,
      name: this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY,
      s3BucketName: this.names.S3_BUCKET,
    });
    this.cleanUpFunctions.push(pipelineExecutionPolicyCleanUp);

    console.log(JSON.stringify(sagemakerPolicy, null, 2));

    await this.logger.log(
      MESSAGES.attachPolicy
        .replace("${POLICY_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE_POLICY)
        .replace("${ROLE_NAME}", this.names.SAGE_MAKER_EXECUTION_ROLE),
    );

    await this.prompter.checkContinue();

    // Attach the SageMaker execution policy to the execution role.
    const { cleanUp: pipelineExecutionRolePolicyCleanUp } = await attachPolicy({
      roleName: this.names.SAGE_MAKER_EXECUTION_ROLE,
      policyArn: pipelineExecutionPolicyArn,
      iamClient: this.clients.IAM,
    });
    this.cleanUpFunctions.push(pipelineExecutionRolePolicyCleanUp);
    // Wait for the role to be ready. If the role is used immediately,
    // the pipeline will fail.
    await wait(5);

    await this.logger.log(MESSAGES.policyAttached);

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingPipeline.replace(
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    // Create the SageMaker pipeline.
```

```
    const { cleanUp: pipelineCleanUp } = await createSagemakerPipeline({
      roleArn: pipelineExecutionRoleArn,
      functionArn: lambdaArn,
      sagemakerClient: this.clients.SageMaker,
      name: this.names.SAGE_MAKER_PIPELINE,
    });
    this.cleanUpFunctions.push(pipelineCleanUp);

    await this.logger.log(
      MESSAGES.pipelineCreated.replace(
        "${PIPELINE_NAME}",
        this.names.SAGE_MAKER_PIPELINE,
      ),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.creatingS3Bucket.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    // Create an S3 bucket for storing inputs and outputs.
    const { cleanUp: s3BucketCleanUp } = await createS3Bucket({
      name: this.names.S3_BUCKET,
      s3Client: this.clients.S3,
    });
    this.cleanUpFunctions.push(s3BucketCleanUp);

    await this.logger.log(
      MESSAGES.s3BucketCreated.replace("${BUCKET_NAME}", this.names.S3_BUCKET),
    );

    this.logger.logSeparator();

    await this.logger.log(
      MESSAGES.uploadingInputData.replace(
        "${BUCKET_NAME}",
        this.names.S3_BUCKET,
      ),
    );

    // Upload CSV Lat/Long data to S3.
    await uploadCSVDataToS3({
      bucketName: this.names.S3_BUCKET,
```

```
      s3Client: this.clients.S3,
    });

    await this.logger.log(MESSAGES.inputDataUploaded);

    this.logger.logSeparator();

    await this.prompter.checkContinue(MESSAGES.executePipeline);

    // Execute the SageMaker pipeline.
    const { arn: pipelineExecutionArn } = await startPipelineExecution({
      name: this.names.SAGE_MAKER_PIPELINE,
      sagemakerClient: this.clients.SageMaker,
      roleArn: pipelineExecutionRoleArn,
      bucketName: this.names.S3_BUCKET,
      queueUrl,
    });

    // Wait for the pipeline execution to finish.
    await waitForPipelineComplete({
      arn: pipelineExecutionArn,
      sagemakerClient: this.clients.SageMaker,
      wait,
    });

    this.logger.logSeparator();

    await this.logger.log(MESSAGES.outputDelay);

    // The getOutput function will throw an error if the output is not
    // found. The retry function will retry a failed function call once
    // ever 10 seconds for 2 minutes.
    const output = await retry({ intervalInMs: 10000, maxRetries: 12 }, () =>
      getObject({
        bucket: this.names.S3_BUCKET,
        s3Client: this.clients.S3,
      }),
    );

    this.logger.logSeparator();
    await this.logger.log(MESSAGES.outputDataRetrieved);
    console.log(output.split("\n").slice(0, 6).join("\n"));
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Secrets Manager-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) zusammen mit Secrets Manager Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  GetSecretValueCommand,
  SecretsManagerClient,
} from "@aws-sdk/client-secrets-manager";

export const getSecretValue = async (secretName = "SECRET_NAME") => {
  const client = new SecretsManagerClient();
  const response = await client.send(
    new GetSecretValueCommand({
      SecretId: secretName,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '584eb612-f8b0-48c9-855e-6d246461b604',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   ARN: 'arn:aws:secretsmanager:us-east-1:xxxxxxxxxxxx:secret:binary-
secret-3873048-xxxxxx',
  //   CreatedDate: 2023-08-08T19:29:51.294Z,
  //   Name: 'binary-secret-3873048',
  //   SecretBinary: Uint8Array(11) [
  //      98, 105, 110, 97, 114,
  //     121,  32, 100, 97, 116,
  //      97
  //   ],
  //   VersionId: '712083f4-0d26-415e-8044-16735142cd6a',
  //   VersionStages: [ 'AWSCURRENT' ]
  // }

  if (response.SecretString) {
    return response.SecretString;
  }

  if (response.SecretBinary) {
    return response.SecretBinary;
  }
};
```

- API-Details finden Sie GetSecretValue in der AWS SDK für JavaScript API-Referenz.

Amazon SES-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptFilter`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  CreateReceiptFilterCommand,
```

```
  ReceiptFilterPolicy,
} from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const createCreateReceiptFilterCommand = ({ policy, ipOrRange, name }) => {
  return new CreateReceiptFilterCommand({
    Filter: {
      IpFilter: {
        Cidr: ipOrRange, // string, either a single IP address (10.0.0.1) or an IP
 address range in CIDR notation (10.0.0.1/24)).
        Policy: policy, // enum ReceiptFilterPolicy, email traffic from the filtered
 addressesOptions.
      },
      /*
        The name of the IP address filter. Only ASCII letters, numbers, underscores,
 or dashes.
        Must be less than 64 characters and start and end with a letter or number.
       */
      Name: name,
    },
  });
};

const FILTER_NAME = getUniqueName("ReceiptFilter");

const run = async () => {
  const createReceiptFilterCommand = createCreateReceiptFilterCommand({
    policy: ReceiptFilterPolicy.Allow,
    ipOrRange: "10.0.0.1",
    name: FILTER_NAME,
  });

  try {
    return await sesClient.send(createReceiptFilterCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- API-Details finden Sie CreateReceiptFilter in der AWS SDK für JavaScript API-Referenz.

`CreateReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRule`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateReceiptRuleCommand, TlsPolicy } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");
const RULE_NAME = getUniqueName("RuleName");
const S3_BUCKET_NAME = getUniqueName("S3BucketName");

const createS3ReceiptRuleCommand = ({
  bucketName,
  emailAddresses,
  name,
  ruleSet,
}) => {
  return new CreateReceiptRuleCommand({
    Rule: {
      Actions: [
        {
          S3Action: {
            BucketName: bucketName,
            ObjectKeyPrefix: "email",
          },
        },
      ],
```

```
      Recipients: emailAddresses,
      Enabled: true,
      Name: name,
      ScanEnabled: false,
      TlsPolicy: TlsPolicy.Optional,
    },
    RuleSetName: ruleSet, // Required
  });
};

const run = async () => {
  const s3ReceiptRuleCommand = createS3ReceiptRuleCommand({
    bucketName: S3_BUCKET_NAME,
    emailAddresses: ["email@example.com"],
    name: RULE_NAME,
    ruleSet: RULE_SET_NAME,
  });

  try {
    return await sesClient.send(s3ReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to create S3 receipt rule.", err);
    throw err;
  }
};
```

- API-Details finden Sie CreateReceiptRule in der AWS SDK für JavaScript API-Referenz.

`CreateReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRuleSet`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateReceiptRuleSetCommand } from "@aws-sdk/client-ses";
```

```
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createCreateReceiptRuleSetCommand = (ruleSetName) => {
  return new CreateReceiptRuleSetCommand({ RuleSetName: ruleSetName });
};

const run = async () => {
  const createReceiptRuleSetCommand =
    createCreateReceiptRuleSetCommand(RULE_SET_NAME);

  try {
    return await sesClient.send(createReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to create receipt rule set", err);
    return err;
  }
};
```

- API-Details finden Sie CreateReceiptRuleSet in der AWS SDK für JavaScript API-Referenz.

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateTemplateCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
```

```
const TEMPLATE_NAME = getUniqueName("TestTemplateName");

const createCreateTemplateCommand = () => {
  return new CreateTemplateCommand({
    /**
     * The template feature in Amazon SES is based on the Handlebars template
 system.
     */
    Template: {
      /**
       * The name of an existing template in Amazon SES.
       */
      TemplateName: TEMPLATE_NAME,
      HtmlPart: `
        <h1>Hello, {{contact.firstName}}!</h1>
        <p>
        Did you know Amazon has a mascot named Peccy?
        </p>
      `,
      SubjectPart: "Amazon Tip",
    },
  });
};

const run = async () => {
  const createTemplateCommand = createCreateTemplateCommand();

  try {
    return await sesClient.send(createTemplateCommand);
  } catch (err) {
    console.log("Failed to create template.", err);
    return err;
  }
};
```

- API-Details finden Sie CreateTemplate in der AWS SDK für JavaScript API-Referenz.

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentity`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const IDENTITY_EMAIL = "fake@example.com";

const createDeleteIdentityCommand = (identityName) => {
  return new DeleteIdentityCommand({
    Identity: identityName,
  });
};

const run = async () => {
  const deleteIdentityCommand = createDeleteIdentityCommand(IDENTITY_EMAIL);

  try {
    return await sesClient.send(deleteIdentityCommand);
  } catch (err) {
    console.log("Failed to delete identity.", err);
    return err;
  }
};
```

- API-Details finden Sie DeleteIdentity in der AWS SDK für JavaScript API-Referenz.

`DeleteReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptFilter`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptFilterCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RECEIPT_FILTER_NAME = getUniqueName("ReceiptFilterName");

const createDeleteReceiptFilterCommand = (filterName) => {
  return new DeleteReceiptFilterCommand({ FilterName: filterName });
};

const run = async () => {
  const deleteReceiptFilterCommand =
    createDeleteReceiptFilterCommand(RECEIPT_FILTER_NAME);

  try {
    return await sesClient.send(deleteReceiptFilterCommand);
  } catch (err) {
    console.log("Error deleting receipt filter.", err);
    return err;
  }
};
```

- API-Details finden Sie DeleteReceiptFilter in der AWS SDK für JavaScript API-Referenz.

`DeleteReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRule`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptRuleCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_NAME = getUniqueName("RuleName");
const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleCommand = () => {
  return new DeleteReceiptRuleCommand({
    RuleName: RULE_NAME,
    RuleSetName: RULE_SET_NAME,
  });
};

const run = async () => {
  const deleteReceiptRuleCommand = createDeleteReceiptRuleCommand();
  try {
    return await sesClient.send(deleteReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule.", err);
    return err;
  }
};
```

- API-Details finden Sie DeleteReceiptRule in der AWS SDK für JavaScript API-Referenz.

`DeleteReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRuleSet`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleSetCommand = () => {
  return new DeleteReceiptRuleSetCommand({ RuleSetName: RULE_SET_NAME });
};

const run = async () => {
  const deleteReceiptRuleSetCommand = createDeleteReceiptRuleSetCommand();

  try {
    return await sesClient.send(deleteReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule set.", err);
    return err;
  }
};
```

- API-Details finden Sie DeleteReceiptRuleSet in der AWS SDK für JavaScript API-Referenz.

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createDeleteTemplateCommand = (templateName) =>
  new DeleteTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const deleteTemplateCommand = createDeleteTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(deleteTemplateCommand);
  } catch (err) {
    console.log("Failed to delete template.", err);
    return err;
  }
};
```

- API-Details finden Sie DeleteTemplate in der AWS SDK für JavaScript API-Referenz.

`GetTemplate`

Das folgende Codebeispiel zeigt die Verwendung`GetTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { GetTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createGetTemplateCommand = (templateName) =>
  new GetTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const getTemplateCommand = createGetTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(getTemplateCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- API-Details finden Sie GetTemplate in der AWS SDK für JavaScript API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- API-Details finden Sie ListIdentities in der AWS SDK für JavaScript API-Referenz.

`ListReceiptFilters`

Das folgende Codebeispiel zeigt die Verwendung`ListReceiptFilters`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListReceiptFiltersCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListReceiptFiltersCommand = () => new ListReceiptFiltersCommand({});

const run = async () => {
  const listReceiptFiltersCommand = createListReceiptFiltersCommand();

  return await sesClient.send(listReceiptFiltersCommand);
};
```

- API-Details finden Sie ListReceiptFilters in der AWS SDK für JavaScript API-Referenz.

`ListTemplates`

Das folgende Codebeispiel zeigt die Verwendung`ListTemplates`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ListTemplatesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListTemplatesCommand = (maxItems) =>
  new ListTemplatesCommand({ MaxItems: maxItems });

const run = async () => {
  const listTemplatesCommand = createListTemplatesCommand(10);

  try {
    return await sesClient.send(listTemplatesCommand);
  } catch (err) {
    console.log("Failed to list templates.", err);
    return err;
  }
```

```
};
```

- API-Details finden Sie ListTemplates in der AWS SDK für JavaScript API-Referenz.

`SendBulkTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendBulkTemplatedEmail`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendBulkTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL_1 = postfix(getUniqueName("Bilbo"), "@example.com");
const VERIFIED_EMAIL_2 = postfix(getUniqueName("Frodo"), "@example.com");

const USERS = [
  { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL_1 },
  { firstName: "Frodo", emailAddress: VERIFIED_EMAIL_2 },
];

/**
 *
```

```
 * @param { { emailAddress: string, firstName: string }[] } users
 * @param { string } templateName the name of an existing template in SES
 * @returns { SendBulkTemplatedEmailCommand }
 */
const createBulkReminderEmailCommand = (users, templateName) => {
  return new SendBulkTemplatedEmailCommand({
    /**
     * Each 'Destination' uses a corresponding set of replacement data. We can map
 each user
     * to a 'Destination' and provide user specific replacement data to create
 personalized emails.
     *
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{name}},</h1><p>Don't forget about the party gifts!</p>
     * Destination 1: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</
p>
     * Destination 2: <h1>Hello Frodo,</h1><p>Don't forget about the party gifts!</
p>
     */
    Destinations: users.map((user) => ({
      Destination: { ToAddresses: [user.emailAddress] },
      ReplacementTemplateData: JSON.stringify({ name: user.firstName }),
    })),
    DefaultTemplateData: JSON.stringify({ name: "Shireling" }),
    Source: VERIFIED_EMAIL_1,
    Template: templateName,
  });
};

const run = async () => {
  const sendBulkTemplateEmailCommand = createBulkReminderEmailCommand(
    USERS,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendBulkTemplateEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
```

```
};
```

- API-Details finden Sie SendBulkTemplatedEmail in der AWS SDK für JavaScript API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendEmailCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createSendEmailCommand = (toAddress, fromAddress) => {
  return new SendEmailCommand({
    Destination: {
      /* required */
      CcAddresses: [
        /* more items */
      ],
      ToAddresses: [
        toAddress,
        /* more To-email addresses */
      ],
    },
    Message: {
      /* required */
      Body: {
        /* required */
        Html: {
          Charset: "UTF-8",
          Data: "HTML_FORMAT_BODY",
        },
```

```
        Text: {
          Charset: "UTF-8",
          Data: "TEXT_FORMAT_BODY",
        },
      },
      Subject: {
        Charset: "UTF-8",
        Data: "EMAIL_SUBJECT",
      },
    },
    Source: fromAddress,
    ReplyToAddresses: [
      /* more items */
    ],
  });
};

const run = async () => {
  const sendEmailCommand = createSendEmailCommand(
    "recipient@example.com",
    "sender@example.com",
  );

  try {
    return await sesClient.send(sendEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- API-Details finden Sie SendEmail in der AWS SDK für JavaScript API-Referenz.

`SendRawEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendRawEmail`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie nodemailer, um eine E-Mail mit einem Anhang zu senden.

```
import sesClientModule from "@aws-sdk/client-ses";
/**
 * nodemailer wraps the SES SDK and calls SendRawEmail. Use this for more advanced
 * functionality like adding attachments to your email.
 *
 * https://nodemailer.com/transports/ses
 */
import nodemailer from "nodemailer";

/**
 * @param {string} from An Amazon SES verified email address.
 * @param {*} to An Amazon SES verified email address.
 */
export const sendEmailWithAttachments = (
  from = "from@example.com",
  to = "to@example.com",
) => {
  const ses = new sesClientModule.SESClient({});
  const transporter = nodemailer.createTransport({
    SES: { ses, aws: sesClientModule },
  });

  return new Promise((resolve, reject) => {
    transporter.sendMail(
      {
        from,
        to,
        subject: "Hello World",
        text: "Greetings from Amazon SES!",
        attachments: [{ content: "Hello World!", filename: "hello.txt" }],
      },
      (err, info) => {
        if (err) {
```

```
          reject(err);
        } else {
          resolve(info);
        }
      },
    );
  });
};
```

- API-Details finden Sie SendRawEmail in der AWS SDK für JavaScript API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL = postfix(getUniqueName("Bilbo"), "@example.com");
```

```
const USER = { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL };

/**
 *
 * @param { { emailAddress: string, firstName: string } } user
 * @param { string } templateName - The name of an existing template in Amazon SES.
 * @returns { SendTemplatedEmailCommand }
 */
const createReminderEmailCommand = (user, templateName) => {
  return new SendTemplatedEmailCommand({
    /**
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{contact.firstName}},</h1><p>Don't forget about the
 party gifts!</p>
     * Destination: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</p>
     */
    Destination: { ToAddresses: [user.emailAddress] },
    TemplateData: JSON.stringify({ contact: { firstName: user.firstName } }),
    Source: VERIFIED_EMAIL,
    Template: templateName,
  });
};

const run = async () => {
  const sendReminderEmailCommand = createReminderEmailCommand(
    USER,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendReminderEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

- API-Details finden Sie SendTemplatedEmail in der AWS SDK für JavaScript API-Referenz.

`UpdateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTemplate`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
const HTML_PART = "<h1>Hello, World!</h1>";

const createUpdateTemplateCommand = () => {
  return new UpdateTemplateCommand({
    Template: {
      TemplateName: TEMPLATE_NAME,
      HtmlPart: HTML_PART,
      SubjectPart: "Example",
      TextPart: "Updated template text.",
    },
  });
};

const run = async () => {
  const updateTemplateCommand = createUpdateTemplateCommand();

  try {
    return await sesClient.send(updateTemplateCommand);
  } catch (err) {
    console.log("Failed to update template.", err);
    return err;
  }
};
```

- API-Details finden Sie UpdateTemplate in der AWS SDK für JavaScript API-Referenz.

`VerifyDomainIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyDomainIdentity`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { VerifyDomainIdentityCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * You must have access to the domain's DNS settings to complete the
 * domain verification process.
 */
const DOMAIN_NAME = postfix(getUniqueName("Domain"), ".example.com");

const createVerifyDomainIdentityCommand = () => {
  return new VerifyDomainIdentityCommand({ Domain: DOMAIN_NAME });
};

const run = async () => {
  const VerifyDomainIdentityCommand = createVerifyDomainIdentityCommand();

  try {
    return await sesClient.send(VerifyDomainIdentityCommand);
  } catch (err) {
    console.log("Failed to verify domain.", err);
    return err;
  }
};
```

- API-Details finden Sie VerifyDomainIdentity in der AWS SDK für JavaScript API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Import required AWS SDK clients and commands for Node.js
import { VerifyEmailIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const EMAIL_ADDRESS = "name@example.com";

const createVerifyEmailIdentityCommand = (emailAddress) => {
  return new VerifyEmailIdentityCommand({ EmailAddress: emailAddress });
};

const run = async () => {
  const verifyEmailIdentityCommand =
    createVerifyEmailIdentityCommand(EMAIL_ADDRESS);
  try {
    return await sesClient.send(verifyEmailIdentityCommand);
  } catch (err) {
    console.log("Failed to verify email identity.", err);
    return err;
  }
};
```

- API-Details finden Sie VerifyEmailIdentity in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für JavaScript (v3)

Zeigt, wie Sie AWS SDK für JavaScript (v3) verwenden, um eine Webanwendung zu erstellen, die Arbeitselemente in einer Amazon Aurora-Datenbank verfolgt und Berichte per E-Mail mithilfe von Amazon Simple Email Service (Amazon SES) versendet. In diesem Beispiel wird ein Frontend verwendet, das React.js für die Interaktion mit einem Node.js Express-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung mit AWS-Services.
- Auflisten, hinzufügen und aktualisieren von Elementen in einer Aurora-Tabelle.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.

- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte in Bildern, die sich in einem Amazon Simple Storage Service (Amazon S3) -Bucket befinden, anhand von Kategorien zu identifizieren. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Rekognition

- Amazon S3
- Amazon SES

Amazon SNS-Beispiele, die das SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon SNS Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SNS.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen SNS-Client und listen Sie Themen in Ihrem Konto auf.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});

  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- API-Details finden Sie ListTopics in der AWS SDK für JavaScript API-Referenz.

Aktionen

`CheckIfPhoneNumberIsOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`CheckIfPhoneNumberIsOptedOut`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   isOptedOut: false
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CheckIfPhoneNumberIsOptedOut in der AWS SDK für JavaScript API-Referenz.

`ConfirmSubscription`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSubscription`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
```

```
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm a
 subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ConfirmSubscription in der AWS SDK für JavaScript API-Referenz.

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
```

```
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateTopic in der AWS SDK für JavaScript API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteTopic in der AWS SDK für JavaScript API-Referenz.

`GetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetSMSAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für JavaScript -API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDelayRe
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie GetTopicAttributes in der AWS SDK für JavaScript API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListSubscriptions in der AWS SDK für JavaScript API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
```

```
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListTopics in der AWS SDK für JavaScript API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
```

```
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Veröffentlichen Sie eine Nachricht zu einem Thema mit Gruppen-, Duplizierungs- und Attributoptionen.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
```

```
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
```

```
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für JavaScript -API-Referenz.

`SetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSMSAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
```

```
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für JavaScript - API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- API-Details finden Sie SetTopicAttributes in der AWS SDK für JavaScript API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die VerwendungSubscribe.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm a
 subscription.
 * @param {string} emailAddress - The email address that is subscribed to the topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
```

```
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Abonnieren Sie eine mobile Anwendung für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint is
 created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
```

```
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Abonnieren Sie ein Thema mit einem Filter.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute set
 to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für JavaScript -API-Referenz.

Abonnement beenden

Das folgende Codebeispiel zeigt die VerwendungUnsubscribe.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client in einem separaten Modul und exportieren Sie ihn.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave it
 blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importieren Sie das SDK- und Client-Module und rufen Sie die API auf.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für JavaScript -API-Referenz.

Szenarien

Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Codebeispiele zeigen, wie eine Anwendung erstellt wird, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

SDK für JavaScript (v3)

Das Beispiel zeigt, wie man eine App erstellt, die es Benutzern ermöglicht, Daten an eine Amazon-DynamoDB-Tabelle zu übermitteln und eine Textnachricht an den Administrator mit Amazon Simple Notification Service (Amazon SNS) zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für JavaScript (v3)

Zeigt, wie Sie das verwenden AWS SDK für JavaScript , um eine React-Anwendung zu erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt für dieses Szenario.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
```

```
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Der obige Code stellt die erforderlichen Abhängigkeiten bereit und startet das Szenario. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
```

```
 * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
 * @param {import('../../libs/prompter.js').Prompter} prompter
 * @param {import('../../libs/logger.js').Logger} logger
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }

  const response = await this.snsClient.send(
    new CreateTopicCommand({
```

```
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
```

```
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );
```

```
      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
```

```
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });
```

```
    let groupId;
    let deduplicationId;
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
```

```
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
```

```
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
```

```
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie eine AWS Lambda Funktion mithilfe der JavaScript Lambda-Laufzeit-API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für JavaScript (v3)

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Laufzeit-API. JavaScript In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall

auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Dieses Beispiel ist auch verfügbar im AWS SDK für JavaScript Entwicklerhandbuch für v3.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines SNS-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
```

```
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Verarbeiten eines SNS-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Amazon SQS-Beispiele, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Initialisieren Sie einen Amazon-SQS-Client und listen Sie Warteschlangen auf.

```
import { SQSClient, paginateListQueues } from "@aws-sdk/client-sqs";
```

```
export const helloSqs = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SQSClient({});

  // You can also use `ListQueuesCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListQueuesCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedQueues = paginateListQueues({ client }, {});
  const queues = [];

  for await (const page of paginatedQueues) {
    if (page.QueueUrls?.length) {
      queues.push(...page.QueueUrls);
    }
  }

  const suffix = queues.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SQS! You have ${queues.length} queue${suffix} in your account.`,
  );
  console.log(queues.map((t) => `  * ${t}`).join("\n"));
};
```

- API-Details finden Sie ListQueues in der AWS SDK für JavaScript API-Referenz.

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung`ChangeMessageVisibility`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Amazon-SQS-Nachricht und ändern Sie deren Timeout-Sichtbarkeit.

```
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
  SQSClient,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 1,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 1,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  const response = await client.send(
    new ChangeMessageVisibilityCommand({
      QueueUrl: queueUrl,
      ReceiptHandle: Messages[0].ReceiptHandle,
      VisibilityTimeout: 20,
    }),
  );
  console.log(response);
  return response;
};
```

- API-Details finden Sie ChangeMessageVisibility in der AWS SDK für JavaScript API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (sqsQueueName = SQS_QUEUE_NAME) => {
  const command = new CreateQueueCommand({
    QueueName: sqsQueueName,
    Attributes: {
      DelaySeconds: "60",
      MessageRetentionPeriod: "86400",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Erstellen einer Amazon-SQS-Warteschlange mit Langabfrage

```
import { CreateQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "queue_name";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const response = await client.send(
    new CreateQueueCommand({
      QueueName: queueName,
      Attributes: {
```

```
        // When the wait time for the ReceiveMessage API action is greater than 0,
        // long polling is in effect. The maximum long polling wait time is 20
        // seconds. Long polling helps reduce the cost of using Amazon SQS by,
        // eliminating the number of empty responses and false empty responses.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        ReceiveMessageWaitTimeSeconds: "20",
      },
    }),
  );
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie CreateQueue in der AWS SDK für JavaScript API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Amazon-SQS-Nachrichten ab und löschen Sie sie.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
```

```
const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- API-Details finden Sie DeleteMessage in der AWS SDK für JavaScript API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }

  if (Messages.length === 1) {
    console.log(Messages[0].Body);
```

```
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK für JavaScript API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-SQS-Warteschlange.

```
import { DeleteQueueCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "test-queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
```

```
  const command = new DeleteQueueCommand({ QueueUrl: queueUrl });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteQueue in der AWS SDK für JavaScript API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { GetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const getQueueAttributes = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new GetQueueAttributesCommand({
    QueueUrl: queueUrl,
    AttributeNames: ["DelaySeconds"],
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '747a1192-c334-5682-a508-4cd5e8dc4e79',
  //     extendedRequestId: undefined,
```

```
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: { DelaySeconds: '1' }
  // }
  return response;
};
```

- API-Details finden Sie GetQueueAttributes in der AWS SDK für JavaScript API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die URL für eine Amazon-SQS-Warteschlange ab.

```
import { GetQueueUrlCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_NAME = "test-queue";

export const main = async (queueName = SQS_QUEUE_NAME) => {
  const command = new GetQueueUrlCommand({ QueueName: queueName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.

- API-Details finden Sie GetQueueUrl in der AWS SDK für JavaScript API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Ihre Amazon-SQS-Warteschlangen auf.

```
import { paginateListQueues, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});

export const main = async () => {
  const paginatedListQueues = paginateListQueues({ client }, {});

  /** @type {string[]} */
  const urls = [];
  for await (const page of paginatedListQueues) {
    const nextUrls = page.QueueUrls?.filter((qurl) => !!qurl) || [];
    urls.push(...nextUrls);
    for (const url of urls) {
      console.log(url);
    }
  }

  return urls;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListQueues in der AWS SDK für JavaScript API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht von einer Amazon-SQS-Warteschlange.

```
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
  SQSClient,
  DeleteMessageBatchCommand,
} from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

const receiveMessage = (queueUrl) =>
  client.send(
    new ReceiveMessageCommand({
      AttributeNames: ["SentTimestamp"],
      MaxNumberOfMessages: 10,
      MessageAttributeNames: ["All"],
      QueueUrl: queueUrl,
      WaitTimeSeconds: 20,
      VisibilityTimeout: 20,
    }),
  );

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const { Messages } = await receiveMessage(queueUrl);

  if (!Messages) {
    return;
  }
```

```
  if (Messages.length === 1) {
    console.log(Messages[0].Body);
    await client.send(
      new DeleteMessageCommand({
        QueueUrl: queueUrl,
        ReceiptHandle: Messages[0].ReceiptHandle,
      }),
    );
  } else {
    await client.send(
      new DeleteMessageBatchCommand({
        QueueUrl: queueUrl,
        Entries: Messages.map((message) => ({
          Id: message.MessageId,
          ReceiptHandle: message.ReceiptHandle,
        })),
      }),
    );
  }
};
```

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
import { ReceiveMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new ReceiveMessageCommand({
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueUrl,
    // The duration (in seconds) for which the call waits for a message
    // to arrive in the queue before returning. If a message is available,
    // the call returns sooner than WaitTimeSeconds. If no messages are
    // available and the wait time expires, the call returns successfully
    // with an empty list of messages.
    // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/
API_ReceiveMessage.html#API_ReceiveMessage_RequestSyntax
```

```
    WaitTimeSeconds: 20,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für JavaScript API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Nachricht an eine Amazon-SQS-Warteschlange.

```
import { SendMessageCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (sqsQueueUrl = SQS_QUEUE_URL) => {
  const command = new SendMessageCommand({
    QueueUrl: sqsQueueUrl,
    DelaySeconds: 10,
    MessageAttributes: {
      Title: {
        DataType: "String",
        StringValue: "The Whistler",
      },
      Author: {
        DataType: "String",
        StringValue: "John Grisham",
```

```
      },
      WeeksOn: {
        DataType: "Number",
        StringValue: "6",
      },
    },
    MessageBody:
      "Information about current NY Times fiction bestseller for week of
 12/11/2016.",
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie SendMessage in der AWS SDK für JavaScript API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue-url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    QueueUrl: queueUrl,
    Attributes: {
```

```
      DelaySeconds: "1",
    },
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Konfigurieren Sie eine Amazon-SQS-Warteschlange für die Verwendung einer Langabfrage.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";

export const main = async (queueUrl = SQS_QUEUE_URL) => {
  const command = new SetQueueAttributesCommand({
    Attributes: {
      ReceiveMessageWaitTimeSeconds: "20",
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

Konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

```
import { SetQueueAttributesCommand, SQSClient } from "@aws-sdk/client-sqs";

const client = new SQSClient({});
const SQS_QUEUE_URL = "queue_url";
const DEAD_LETTER_QUEUE_ARN = "dead_letter_queue_arn";

export const main = async (
  queueUrl = SQS_QUEUE_URL,
  deadLetterQueueArn = DEAD_LETTER_QUEUE_ARN,
) => {
```

```
  const command = new SetQueueAttributesCommand({
    Attributes: {
      RedrivePolicy: JSON.stringify({
        // Amazon SQS supports dead-letter queues (DLQ), which other
        // queues (source queues) can target for messages that can't
        // be processed (consumed) successfully.
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-dead-letter-queues.html
        deadLetterTargetArn: deadLetterQueueArn,
        maxReceiveCount: "10",
      }),
    },
    QueueUrl: queueUrl,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für JavaScript (v3)

Zeigt, wie Sie das verwenden AWS SDK für JavaScript , um eine React-Anwendung zu erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dies ist der Einstiegspunkt für dieses Szenario.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;
```

```
  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Der obige Code stellt die erforderlichen Abhängigkeiten bereit und startet das Szenario. Der nächste Abschnitt enthält den Großteil des Beispiels.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
```

```
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }

  const response = await this.snsClient.send(
    new CreateTopicCommand({
      Name: this.topicName,
      Attributes: {
        FifoTopic: this.isFifo ? "true" : "false",
```

```
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      },
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    const maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
```

```
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
```

```
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];
```

```
      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId;
    let deduplicationId;
```

```
    let choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
```

```
        : {}),
    }),
  );

  const publishAnother = await this.prompter.confirm({
    message: MESSAGES.publishAnother,
  });

  if (publishAnother) {
    await this.publishMessages();
  }
}

async receiveAndDeleteMessages() {
  for (const queue of this.queues) {
    const { Messages } = await this.sqsClient.send(
      new ReceiveMessageCommand({
        QueueUrl: queue.queueUrl,
      }),
    );

    if (Messages) {
      await this.logger.log(
        MESSAGES.messagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
      console.log(Messages);

      await this.sqsClient.send(
        new DeleteMessageBatchCommand({
          QueueUrl: queue.queueUrl,
          Entries: Messages.map((message) => ({
            Id: message.MessageId,
            ReceiptHandle: message.ReceiptHandle,
          })),
        }),
      );
    } else {
      await this.logger.log(
        MESSAGES.noMessagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
```

```
        ),
      );
    }
  }

  const deleteAndPoll = await this.prompter.confirm({
    message: MESSAGES.deleteAndPollConfirmation,
  });

  if (deleteAndPoll) {
    await this.receiveAndDeleteMessages();
  }
}

async destroyResources() {
  for (const subscriptionArn of this.subscriptionArns) {
    await this.snsClient.send(
      new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
    );
  }

  for (const queue of this.queues) {
    await this.sqsClient.send(
      new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
    );
  }

  if (this.topicArn) {
    await this.snsClient.send(
      new DeleteTopicCommand({ TopicArn: this.topicArn }),
    );
  }
}

async start() {
  console.clear();

  try {
    this.logger.logSeparator(MESSAGES.headerWelcome);
    await this.welcome();
    this.logger.logSeparator(MESSAGES.headerFifo);
    await this.confirmFifo();
    this.logger.logSeparator(MESSAGES.headerCreateTopic);
    await this.createTopic();
```

```
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Verarbeiten eines SQS-Ereignisses mit Lambda. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message) {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Verarbeiten eines SQS-Ereignisses mit Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, Context, SQSHandler, SQSRecord } from "aws-lambda";

export const functionHandler: SQSHandler = async (
  event: SQSEvent,
  context: Context
): Promise<void> => {
  for (const message of event.Records) {
    await processMessageAsync(message);
  }
  console.info("done");
};

async function processMessageAsync(message: SQSRecord): Promise<any> {
  try {
    console.log(`Processed message ${message.body}`);
    // TODO: Do interesting work based on the new message
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Meldung von SQS-Batch-Elementfehlern mithilfe von Lambda. JavaScript

```
// Node.js 20.x Lambda runtime, AWS SDK for Javascript V3
export const handler = async (event, context) => {
    const batchItemFailures = [];
    for (const record of event.Records) {
        try {
            await processMessageAsync(record, context);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }
    return { batchItemFailures };
};

async function processMessageAsync(record, context) {
    if (record.body && record.body.includes("error")) {
        throw new Error("There is an error in the SQS Message.");
    }
    console.log(`Processed message: ${record.body}`);
}
```

Meldung von Fehlern bei SQS-Batch-Elementen unter Verwendung von Lambda. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SQSEvent, SQSBatchResponse, Context, SQSBatchItemFailure, SQSRecord } from
 'aws-lambda';

export const handler = async (event: SQSEvent, context: Context):
 Promise<SQSBatchResponse> => {
    const batchItemFailures: SQSBatchItemFailure[] = [];

    for (const record of event.Records) {
        try {
            await processMessageAsync(record);
        } catch (error) {
            batchItemFailures.push({ itemIdentifier: record.messageId });
        }
    }

    return {batchItemFailures: batchItemFailures};
};
```

```
async function processMessageAsync(record: SQSRecord): Promise<void> {
    if (record.body && record.body.includes("error")) {
        throw new Error('There is an error in the SQS Message.');
    }
    console.log(`Processed message ${record.body}`);
}
```

Beispiele für Step Functions unter Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SFNClient, StartExecutionCommand } from "@aws-sdk/client-sfn";

/**
 * @param {{ sfnClient: SFNClient, stateMachineArn: string }} config
 */
export async function startExecution({ sfnClient, stateMachineArn }) {
  const response = await sfnClient.send(
    new StartExecutionCommand({
      stateMachineArn,
    }),
  );
  console.log(response);
  // Example response:
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '202a9309-c16a-454b-adeb-c4d19afe3bf2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   executionArn: 'arn:aws:states:us-
east-1:000000000000:execution:MyStateMachine:aaaaaaaa-f787-49fb-a20c-1b61c64eafe6',
  //   startDate: 2024-01-04T15:54:08.362Z
  // }
  return response;
}

// Call function if run directly
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  startExecution({ sfnClient: new SFNClient({}), stateMachineArn: "ARN" });
}
```

- API-Details finden Sie StartExecution in der AWS SDK für JavaScript API-Referenz.

AWS STS Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

Übernehmen Sie die IAM-Rolle.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
```

```
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie AssumeRole in der AWS SDK für JavaScript API-Referenz.

Support Beispiele zur Verwendung des SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) with Aktionen ausführen und gängige Szenarien implementieren Support.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie „main()“ auf, um das Beispiel auszuführen.

```
import {
  DescribeServicesCommand,
  SupportClient,
} from "@aws-sdk/client-support";

// Change the value of 'region' to your preferred AWS Region.
const client = new SupportClient({ region: "us-east-1" });

const getServiceCount = async () => {
  try {
    const { services } = await client.send(new DescribeServicesCommand({}));
    return services.length;
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

export const main = async () => {
  try {
```

```
    const count = await getServiceCount();
    console.log(`Hello, AWS Support! There are ${count} services available.`);
  } catch (err) {
    console.error("Failed to get service count: ", err.message);
  }
};
```

- API-Details finden Sie DescribeServices in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario im Terminal aus.

```
import {
  AddAttachmentsToSetCommand,
  AddCommunicationToCaseCommand,
```

```
  CreateCaseCommand,
  DescribeAttachmentCommand,
  DescribeCasesCommand,
  DescribeCommunicationsCommand,
  DescribeServicesCommand,
  DescribeSeverityLevelsCommand,
  ResolveCaseCommand,
  SupportClient,
} from "@aws-sdk/client-support";
import * as inquirer from "@inquirer/prompts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};

const client = new SupportClient({ region: "us-east-1" });

// Verify that the account has a Support plan.
export const verifyAccount = async () => {
  const command = new DescribeServicesCommand({});

  try {
    await client.send(command);
  } catch (err) {
    if (err.name === "SubscriptionRequiredException") {
      throw new Error(
        "You must be subscribed to the AWS Support plan to use this feature.",
      );
    }
    throw err;
  }
};

/**
 * Select a service from the list returned from DescribeServices.
 */
export const getService = async () => {
  const { services } = await client.send(new DescribeServicesCommand({}));
  const selectedService = await inquirer.select({
    message:
      "Select a service. Your support case will be created for this service. The
 list of services is truncated for readability.",
```

```
    choices: services.slice(0, 10).map((s) => ({ name: s.name, value: s })),
  });
  return selectedService;
};

/**
 * @param {{ categories: import('@aws-sdk/client-support').Category[]}} service
 */
export const getCategory = async (service) => {
  const selectedCategory = await inquirer.select({
    message: "Select a category.",
    choices: service.categories.map((c) => ({ name: c.name, value: c })),
  });
  return selectedCategory;
};

// Get the available severity levels for the account.
export const getSeverityLevel = async () => {
  const command = new DescribeSeverityLevelsCommand({});
  const { severityLevels } = await client.send(command);
  const selectedSeverityLevel = await inquirer.select({
    message: "Select a severity level.",
    choices: severityLevels.map((s) => ({ name: s.name, value: s })),
  });
  return selectedSeverityLevel;
};

/**
 * Create a new support case
 * @param {{
 *  selectedService: import('@aws-sdk/client-support').Service
 *  selectedCategory: import('@aws-sdk/client-support').Category
 *  selectedSeverityLevel: import('@aws-sdk/client-support').SeverityLevel
 * }} selections
 * @returns
 */
export const createCase = async ({
  selectedService,
  selectedCategory,
  selectedSeverityLevel,
}) => {
  const command = new CreateCaseCommand({
    subject: "IGNORE: Test case",
    communicationBody: "This is a test. Please ignore.",
```

```
    serviceCode: selectedService.code,
    categoryCode: selectedCategory.code,
    severityCode: selectedSeverityLevel.code,
  });
  const { caseId } = await client.send(command);
  return caseId;
};

// Get a list of open support cases created today.
export const getTodaysOpenCases = async () => {
  const d = new Date();
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
  });

  const { cases } = await client.send(command);

  if (cases.length === 0) {
    throw new Error(
      "Unexpected number of cases. Expected more than 0 open cases.",
    );
  }
  return cases;
};

// Create an attachment set.
export const createAttachmentSet = async () => {
  const command = new AddAttachmentsToSetCommand({
    attachments: [
      {
        fileName: "example.txt",
        data: new TextEncoder().encode("some example text"),
      },
    ],
  });
  const { attachmentSetId } = await client.send(command);
  return attachmentSetId;
};

export const linkAttachmentSetToCase = async (attachmentSetId, caseId) => {
  const command = new AddCommunicationToCaseCommand({
    attachmentSetId,
```

```
    caseId,
    communicationBody: "Adding attachment set to case.",
  });
  await client.send(command);
};

// Get all communications for a support case.
export const getCommunications = async (caseId) => {
  const command = new DescribeCommunicationsCommand({
    caseId,
  });
  const { communications } = await client.send(command);
  return communications;
};

/**
 * @param {import('@aws-sdk/client-support').Communication[]} communications
 */
export const getFirstAttachment = (communications) => {
  const firstCommWithAttachment = communications.find(
    (c) => c.attachmentSet.length > 0,
  );
  return firstCommWithAttachment?.attachmentSet[0].attachmentId;
};

// Get an attachment.
export const getAttachment = async (attachmentId) => {
  const command = new DescribeAttachmentCommand({
    attachmentId,
  });
  const { attachment } = await client.send(command);
  return attachment;
};

// Resolve the case matching the given case ID.
export const resolveCase = async (caseId) => {
  const shouldResolve = await inquirer.confirm({
    message: `Do you want to resolve ${caseId}?`,
  });

  if (shouldResolve) {
    const command = new ResolveCaseCommand({
      caseId: caseId,
    });
```

```
    await client.send(command);
    return true;
  }
  return false;
};

/**
 * Find a specific case in the list of provided cases by case ID.
 * If the case is not found, and the results are paginated, continue
 * paging through the results.
 * @param {{
 *   caseId: string,
 *   cases: import('@aws-sdk/client-support').CaseDetails[]
 *   nextToken: string
 * }} options
 * @returns
 */
export const findCase = async ({ caseId, cases, nextToken }) => {
  const foundCase = cases.find((c) => c.caseId === caseId);

  if (foundCase) {
    return foundCase;
  }

  if (nextToken) {
    const response = await client.send(
      new DescribeCasesCommand({
        nextToken,
        includeResolvedCases: true,
      }),
    );
    return findCase({
      caseId,
      cases: response.cases,
      nextToken: response.nextToken,
    });
  }

  throw new Error(`${caseId} not found.`);
};

// Get all cases created today.
export const getTodaysResolvedCases = async (caseIdToWaitFor) => {
```

```
  const d = new Date("2023-01-18");
  const startOfToday = new Date(d.getFullYear(), d.getMonth(), d.getDate());
  const command = new DescribeCasesCommand({
    includeCommunications: false,
    afterTime: startOfToday.toISOString(),
    includeResolvedCases: true,
  });
  const { cases, nextToken } = await client.send(command);
  await findCase({ cases, caseId: caseIdToWaitFor, nextToken });
  return cases.filter((c) => c.status === "resolved");
};

const main = async () => {
  let caseId;
  try {
    console.log(wrapText("Welcome to the AWS Support basic usage scenario."));

    // Verify that the account is subscribed to support.
    await verifyAccount();

    // Provided a truncated list of services and prompt the user to select one.
    const selectedService = await getService();

    // Provided the categories for the selected service and prompt the user to
 select one.
    const selectedCategory = await getCategory(selectedService);

    // Provide the severity available severity levels for the account and prompt the
 user to select one.
    const selectedSeverityLevel = await getSeverityLevel();

    // Create a support case.
    console.log("\nCreating a support case.");
    caseId = await createCase({
      selectedService,
      selectedCategory,
      selectedSeverityLevel,
    });
    console.log(`Support case created: ${caseId}`);

    // Display a list of open support cases created today.
    const todaysOpenCases = await retry(
      { intervalInMs: 1000, maxRetries: 15 },
      getTodaysOpenCases,
```

```
    );
    console.log(
      `\nOpen support cases created today: ${todaysOpenCases.length}`,
    );
    console.log(todaysOpenCases.map((c) => `${c.caseId}`).join("\n"));

    // Create an attachment set.
    console.log("\nCreating an attachment set.");
    const attachmentSetId = await createAttachmentSet();
    console.log(`Attachment set created: ${attachmentSetId}`);

    // Add the attachment set to the support case.
    console.log(`\nAdding attachment set to ${caseId}`);
    await linkAttachmentSetToCase(attachmentSetId, caseId);
    console.log(`Attachment set added to ${caseId}`);

    // List the communications for a support case.
    console.log(`\nListing communications for ${caseId}`);
    const communications = await getCommunications(caseId);
    console.log(
      communications
        .map(
          (c) =>
            `Communication created on ${c.timeCreated}. Has
${c.attachmentSet.length} attachments.`,
        )
        .join("\n"),
    );

    // Describe the first attachment.
    console.log(`\nDescribing attachment ${attachmentSetId}`);
    const attachmentId = getFirstAttachment(communications);
    const attachment = await getAttachment(attachmentId);
    console.log(
      `Attachment is the file '${
        attachment.fileName
      }' with data: \n${new TextDecoder().decode(attachment.data)}`,
    );

    // Confirm that the support case should be resolved.
    const isResolved = await resolveCase(caseId);
    if (isResolved) {
      // List the resolved cases and include the one previously created.
      // Resolved cases can take a while to appear.
```

```
      console.log(
        "\nWaiting for case status to be marked as resolved. This can take some
 time.",
      );
      const resolvedCases = await retry(
        { intervalInMs: 20000, maxRetries: 15 },
        () => getTodaysResolvedCases(caseId),
      );
      console.log("Resolved cases:");
      console.log(resolvedCases.map((c) => c.caseId).join("\n"));
    }
  } catch (err) {
    console.error(err);
  }
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt die Verwendung`AddAttachmentsToSet`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AddAttachmentsToSetCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new attachment set or add attachments to an existing set.
    // Provide an 'attachmentSetId' value to add attachments to an existing set.
    // Use AddCommunicationToCase or CreateCase to associate an attachment set with
 a support case.
    const response = await client.send(
      new AddAttachmentsToSetCommand({
        // You can add up to three attachments per set. The size limit is 5 MB per
 attachment.
        attachments: [
          {
            fileName: "example.txt",
            data: new TextEncoder().encode("some example text"),
          },
        ],
      }),
    );
    // Use this ID in AddCommunicationToCase or CreateCase.
    console.log(response.attachmentSetId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie AddAttachmentsToSet in der AWS SDK für JavaScript API-Referenz.

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`AddCommunicationToCase`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { AddCommunicationToCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  let attachmentSetId;

  try {
    // Add a communication to a case.
    const response = await client.send(
      new AddCommunicationToCaseCommand({
        communicationBody: "Adding an attachment.",
        // Set value to an existing support case id.
        caseId: "CASE_ID",
        // Optional. Set value to an existing attachment set id to add attachments
 to the case.
        attachmentSetId,
      }),
    );
    console.log(response);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie AddCommunicationToCase in der AWS SDK für JavaScript API-Referenz.

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung`CreateCase`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a new case and log the case id.
    // Important: This creates a real support case in your account.
    const response = await client.send(
      new CreateCaseCommand({
        // The subject line of the case.
        subject: "IGNORE: Test case",
        // Use DescribeServices to find available service codes for each service.
        serviceCode: "service-quicksight-end-user",
        // Use DescribeSecurityLevels to find available severity codes for your
 support plan.
        severityCode: "low",
        // Use DescribeServices to find available category codes for each service.
        categoryCode: "end-user-support",
        // The main description of the support case.
        communicationBody: "This is a test. Please ignore.",
      }),
    );
    console.log(response.caseId);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie CreateCase in der AWS SDK für JavaScript API-Referenz.

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAttachment`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeAttachmentCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the metadata and content of an attachment.
    const response = await client.send(
      new DescribeAttachmentCommand({
        // Set value to an existing attachment id.
        // Use DescribeCommunications or DescribeCases to find an attachment id.
        attachmentId: "ATTACHMENT_ID",
      }),
    );
    console.log(response.attachment?.fileName);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie DescribeAttachment in der AWS SDK für JavaScript API-Referenz.

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCases`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeCasesCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all of the unresolved cases in your account.
    // Filter or expand results by providing parameters to the DescribeCasesCommand.
 Refer
    // to the TypeScript definition and the API doc for more information on possible
 parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecasescommandinput.html
    const response = await client.send(new DescribeCasesCommand({}));
    const caseIds = response.cases.map((supportCase) => supportCase.caseId);
    console.log(caseIds);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie DescribeCases in der AWS SDK für JavaScript API-Referenz.

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCommunications`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeCommunicationsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get all communications for the support case.
    // Filter results by providing parameters to the DescribeCommunicationsCommand.
 Refer
    // to the TypeScript definition and the API doc for more information on possible
 parameters.
    // https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/client-
support/interfaces/describecommunicationscommandinput.html
    const response = await client.send(
      new DescribeCommunicationsCommand({
        // Set value to an existing case id.
        caseId: "CASE_ID",
      }),
    );
    const text = response.communications.map((item) => item.body).join("\n");
    console.log(text);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie DescribeCommunications in der AWS SDK für JavaScript API-Referenz.

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSeverityLevels`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DescribeSeverityLevelsCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Get the list of severity levels.
    // The available values depend on the support plan for the account.
    const response = await client.send(new DescribeSeverityLevelsCommand({}));
    console.log(response.severityLevels);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie DescribeSeverityLevels in der AWS SDK für JavaScript API-Referenz.

`ResolveCase`

Das folgende Codebeispiel zeigt die Verwendung`ResolveCase`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { ResolveCaseCommand } from "@aws-sdk/client-support";

import { client } from "../libs/client.js";

const main = async () => {
  try {
    const response = await client.send(
      new ResolveCaseCommand({
        caseId: "CASE_ID",
      }),
    );

    console.log(response.finalCaseStatus);
    return response;
  } catch (err) {
    console.error(err);
  }
};
```

- API-Details finden Sie ResolveCase in der AWS SDK für JavaScript API-Referenz.

Beispiele für Systems Manager, die SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Systems Manager

Das folgende Codebeispiel zeigt die ersten Schritte mit Systems Manager.

SDK für JavaScript (v3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { paginateListDocuments, SSMClient } from "@aws-sdk/client-ssm";

// Call ListDocuments and display the result.
export const main = async () => {
  const client = new SSMClient();
  const listDocumentsPaginated = [];
  console.log(
    "Hello, AWS Systems Manager! Let's list some of your documents:\n",
  );
  try {
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListDocuments({ client }, { MaxResults: 5 });
    for await (const page of paginator) {
      listDocumentsPaginated.push(...page.DocumentIdentifiers);
    }
  } catch (caught) {
    console.error(`There was a problem saying hello: ${caught.message}`);
```

```
    throw caught;
  }

  for (const { Name, DocumentFormat, CreatedDate } of listDocumentsPaginated) {
    console.log(`${Name} - ${DocumentFormat} - ${CreatedDate}`);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Details finden Sie ListDocuments in der AWS SDK für JavaScript API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Wartungsfenster.
- Ändern Sie den Zeitplan des Wartungsfensters.
- Erstellen Sie ein Dokument.
- Senden Sie einen Befehl an eine bestimmte EC2-Instance.
- Erstellen Sie eine OpsItem.
- Aktualisieren und lösen Sie das OpsItem.
- Löschen Sie das Wartungsfenster OpsItem und das Dokument.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { fileURLToPath } from "node:url";
import {
  CreateDocumentCommand,
  CreateMaintenanceWindowCommand,
  CreateOpsItemCommand,
  DeleteDocumentCommand,
  DeleteMaintenanceWindowCommand,
  DeleteOpsItemCommand,
  DescribeOpsItemsCommand,
  DocumentAlreadyExists,
  OpsItemStatus,
  waitUntilCommandExecuted,
  CancelCommandCommand,
  paginateListCommandInvocations,
  SendCommandCommand,
  UpdateMaintenanceWindowCommand,
  UpdateOpsItemCommand,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * @typedef {{
 *   ssmClient: import('@aws-sdk/client-ssm').SSMClient,
 *   documentName?: string
 *   maintenanceWindow?: string
 *   winId?: int
 *   ec2InstanceId?: string
 *   requestedDateTime?: Date
 *   opsItemId?: string
 *   askToDeleteResources?: boolean
 * }} State
 */

const defaultMaintenanceWindow = "ssm-maintenance-window";
const defaultDocumentName = "ssmdocument";
```

```
// The timeout duration is highly dependent on the specific setup and environment
 necessary. This example handles only the most common error cases, and uses a much
 shorter duration than most productions systems would use.
const COMMAND_TIMEOUT_DURATION_SECONDS = 30; // 30 seconds

const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `Welcome to the AWS Systems Manager SDK Getting Started scenario.
    This program demonstrates how to interact with Systems Manager using the AWS SDK
 for JavaScript V3.
    Systems Manager is the operations hub for your AWS applications and resources
 and a secure end-to-end management solution.
    The program's primary functions include creating a maintenance window, creating
 a document, sending a command to a document,
    listing documents, listing commands, creating an OpsItem, modifying an OpsItem,
 and deleting Systems Manager resources.
    Upon completion of the program, all AWS resources are cleaned up.
    Let's get started...`,
  { header: true },
);

const createMaintenanceWindow = new ScenarioOutput(
  "createMaintenanceWindow",
  "Step 1: Create a Systems Manager maintenance window.",
);

const getMaintenanceWindow = new ScenarioInput(
  "maintenanceWindow",
  "Please enter the maintenance window name:",
  { type: "input", default: defaultMaintenanceWindow },
);

export const sdkCreateMaintenanceWindow = new ScenarioAction(
  "sdkCreateMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
```

```
          Schedule: "cron(0 10 ? * MON-FRI *)", //The schedule of the maintenance
 window in the form of a cron or rate expression.
          Duration: 2, //The duration of the maintenance window in hours.
          Cutoff: 1, //The number of hours before the end of the maintenance window
 that Amazon Web Services Systems Manager stops scheduling new tasks for execution.
          AllowUnassociatedTargets: true, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
        }),
      );
      state.winId = response.WindowId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const modifyMaintenanceWindow = new ScenarioOutput(
  "modifyMaintenanceWindow",
  "Modify the maintenance window by changing the schedule.",
);

const sdkModifyMaintenanceWindow = new ScenarioAction(
  "sdkModifyMaintenanceWindow",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new UpdateMaintenanceWindowCommand({
          WindowId: state.winId,
          Schedule: "cron(0 0 ? * MON *)",
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while modifying the maintenance window. Please fix the
 error and try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
```

```
  },
);

const createSystemsManagerActions = new ScenarioOutput(
  "createSystemsManagerActions",
  "Create a document that defines the actions that Systems Manager performs on your
 EC2 instance.",
);

const getDocumentName = new ScenarioInput(
  "documentName",
  "Please enter the document: ",
  { type: "input", default: defaultDocumentName },
);

const sdkCreateSSMDoc = new ScenarioAction(
  "sdkCreateSSMDoc",
  async (/** @type {State} */ state) => {
    const contentData = `{
                "schemaVersion": "2.2",
                "description": "Run a simple shell command",
                "mainSteps": [
                    {
                        "action": "aws:runShellScript",
                        "name": "runEchoCommand",
                        "inputs": {
                          "runCommand": [
                            "echo 'Hello, world!'"
                          ]
                        }
                    }
                ]
            }`;
    try {
      await state.ssmClient.send(
        new CreateDocumentCommand({
          Content: contentData,
          Name: state.documentName,
          DocumentType: "Command",
        }),
      );
    } catch (caught) {
      console.log(`Exception type: (${typeof caught})`);
      if (caught instanceof DocumentAlreadyExists) {
```

```
        console.log("Document already exists. Continuing...\n");
      } else {
        console.error(caught.message);
        console.log(
          `An error occurred while creating the document. Please fix the error and
 try again. Error message: ${caught.message}`,
        );
        throw caught;
      }
    }
  },
);

const ec2HelloWorld = new ScenarioOutput(
  "ec2HelloWorld",
  `Now you have the option of running a command on an EC2 instance that echoes
 'Hello, world!'. In order to run this command, you must provide the instance ID
 of a Linux EC2 instance. If you do not already have a running Linux EC2 instance
 in your account, you can create one using the AWS console. For information about
 creating an EC2 instance, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
ec2-launch-instance-wizard.html.`,
);

const enterIdOrSkipEC2HelloWorld = new ScenarioInput(
  "enterIdOrSkipEC2HelloWorld",
  "Enter your EC2 InstanceId or press enter to skip this step: ",
  { type: "input", default: "" },
);

const sdkEC2HelloWorld = new ScenarioAction(
  "sdkEC2HelloWorld",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new SendCommandCommand({
          DocumentName: state.documentName,
          InstanceIds: [state.ec2InstanceId],
          TimeoutSeconds: COMMAND_TIMEOUT_DURATION_SECONDS,
        }),
      );
      state.CommandId = response.Command.CommandId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
```

```
        `An error occurred while sending the command. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const sdkGetCommandTime = new ScenarioAction(
  "sdkGetCommandTime",
  async (/** @type {State} */ state) => {
    const listInvocationsPaginated = [];
    console.log(
      "Let's get the time when the specific command was sent to the specific managed
 node.",
    );

    console.log(
      `First, we'll wait for the command to finish executing. This may take up to
 ${COMMAND_TIMEOUT_DURATION_SECONDS} seconds.`,
    );
    const commandExecutedResult = waitUntilCommandExecuted(
      { client: state.ssmClient },
      {
        CommandId: state.CommandId,
        InstanceId: state.ec2InstanceId,
      },
    );
    // This is necessary because the TimeoutSeconds of SendCommandCommand is only
 for the delivery, not execution.
    try {
      await new Promise((_, reject) =>
        setTimeout(
          reject,
          COMMAND_TIMEOUT_DURATION_SECONDS * 1000,
          new Error("Command Timed Out"),
        ),
      );
    } catch (caught) {
      if (caught.message === "Command Timed Out") {
```

```
        commandExecutedResult.state = "TIMED_OUT";
      } else {
        throw caught;
      }
    }

    if (commandExecutedResult.state !== "SUCCESS") {
      console.log(
        `The command with id: ${state.CommandId} did not execute in the allotted
 time. Canceling command.`,
      );
      state.ssmClient.send(
        new CancelCommandCommand({
          CommandId: state.CommandId,
        }),
      );
      state.enterIdOrSkipEC2HelloWorld === "";
      return;
    }

    for await (const page of paginateListCommandInvocations(
      { client: state.ssmClient },
      { CommandId: state.CommandId },
    )) {
      listInvocationsPaginated.push(...page.CommandInvocations);
    }
    /**
     * @type {import('@aws-sdk/client-ssm').CommandInvocation}
     */
    const commandInvocation = listInvocationsPaginated.shift(); // Because the call
 was made with CommandId, there's only one result, so shift it off.
    state.requestedDateTime = commandInvocation.RequestedDateTime;

    console.log(
      `The command invocation happened at: ${state.requestedDateTime}.`,
    );
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      state.enterIdOrSkipEC2HelloWorld === "",
  },
);

const createSSMOpsItem = new ScenarioOutput(
```

```
  "createSSMOpsItem",
  `Now we will create a Systems Manager OpsItem. An OpsItem is a feature provided by
 the Systems Manager service. It is a type of operational data item that allows you
 to manage and track various operational issues, events, or tasks within your AWS
 environment.
You can create OpsItems to track and manage operational issues as they arise. For
 example, you could create an OpsItem whenever your application detects a critical
 error or an anomaly in your infrastructure.`,
);

const sdkCreateSSMOpsItem = new ScenarioAction(
  "sdkCreateSSMOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new CreateOpsItemCommand({
          Description: "Created by the System Manager Javascript API",
          Title: "Disk Space Alert",
          Source: "EC2",
          Category: "Performance",
          Severity: "2",
        }),
      );
      state.opsItemId = response.OpsItemId;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while creating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const updateOpsItem = new ScenarioOutput(
  "updateOpsItem",
  (/** @type {State} */ state) =>
    `Now we will update the OpsItem: ${state.opsItemId}`,
);

const sdkUpdateOpsItem = new ScenarioAction(
  "sdkUpdateOpsItem",
  async (/** @type {State} */ state) => {
```

```
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Description: `An update to ${state.opsItemId}`,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const getOpsItemStatus = new ScenarioOutput(
  "getOpsItemStatus",
  (/** @type {State} */ state) =>
    `Now we will get the status of the OpsItem: ${state.opsItemId}`,
);

const sdkOpsItemStatus = new ScenarioAction(
  "sdkGetOpsItemStatus",
  async (/** @type {State} */ state) => {
    try {
      const response = await state.ssmClient.send(
        new DescribeOpsItemsCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      state.opsItemStatus = response.OpsItemStatus;
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while describing the ops item. Please fix the error and
 try again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);
```

```
const resolveOpsItem = new ScenarioOutput(
  "resolveOpsItem",
  (/** @type {State} */ state) =>
    `Now we will resolve the OpsItem: ${state.opsItemId}`,
);

const sdkResolveOpsItem = new ScenarioAction(
  "sdkResolveOpsItem",
  async (/** @type {State} */ state) => {
    try {
      const _response = await state.ssmClient.send(
        new UpdateOpsItemCommand({
          OpsItemId: state.opsItemId,
          Status: OpsItemStatus.RESOLVED,
        }),
      );
    } catch (caught) {
      console.error(caught.message);
      console.log(
        `An error occurred while updating the ops item. Please fix the error and try
 again. Error message: ${caught.message}`,
      );
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  "Would you like to delete the Systems Manager resources created during this
 example run?",
  { type: "confirm" },
);

const confirmDeleteChoice = new ScenarioOutput(
  "confirmDeleteChoice",
  (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You chose to delete the resources.";
    }
    return "The Systems Manager resources will not be deleted. Please delete them
 manually to avoid charges.";
  },
```

```
);

export const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    try {
      await state.ssmClient.send(
        new DeleteOpsItemCommand({
          OpsItemId: state.opsItemId,
        }),
      );
      console.log(`The ops item: ${state.opsItemId} was successfully deleted.`);
    } catch (caught) {
      console.log(
        `There was a problem deleting the ops item: ${state.opsItemId}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteMaintenanceWindowCommand({
          Name: state.maintenanceWindow,
          WindowId: state.winId,
        }),
      );
      console.log(
        `The maintenance window: ${state.maintenanceWindow} was successfully
 deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the maintenance window: ${state.opsItemId}.
 Please delete it manually. Error: ${caught.message}`,
      );
    }

    try {
      await state.ssmClient.send(
        new DeleteDocumentCommand({
          Name: state.documentName,
        }),
      );
      console.log(
```

```
        `The document: ${state.documentName} was successfully deleted.`,
      );
    } catch (caught) {
      console.log(
        `There was a problem deleting the document: ${state.documentName}. Please
 delete it manually. Error: ${caught.message}`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the Systems Manager Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "SSM Basics",
  [
    greet,
    pressEnter,
    createMaintenanceWindow,
    getMaintenanceWindow,
    sdkCreateMaintenanceWindow,
    modifyMaintenanceWindow,
    pressEnter,
    sdkModifyMaintenanceWindow,
    createSystemsManagerActions,
    getDocumentName,
    sdkCreateSSMDoc,
    ec2HelloWorld,
    enterIdOrSkipEC2HelloWorld,
    sdkEC2HelloWorld,
    sdkGetCommandTime,
    pressEnter,
    createSSMOpsItem,
    pressEnter,
    sdkCreateSSMOpsItem,
    updateOpsItem,
    pressEnter,
    sdkUpdateOpsItem,
    getOpsItemStatus,
```

```
    pressEnter,
    sdkOpsItemStatus,
    resolveOpsItem,
    pressEnter,
    sdkResolveOpsItem,
    askToDeleteResources,
    confirmDeleteChoice,
    sdkDeleteResources,
    goodbye,
  ],
  { ssmClient: new SSMClient({}) },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript - API-Referenz.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand

- UpdateOpsItem

Aktionen

`CreateDocument`

Das folgende Codebeispiel zeigt die Verwendung`CreateDocument`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ content, name, documentType }) => {
  const client = new SSMClient({});
  try {
    const { documentDescription } = await client.send(
      new CreateDocumentCommand({
        Content: content, // The content for the new SSM document. The content must
 not exceed 64KB.
        Name: name,
        DocumentType: documentType, // Document format type can be JSON, YAML, or
 TEXT. The default format is JSON.
      }),
    );
    console.log("Document created successfully.");
    return { DocumentDescription: documentDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DocumentAlreadyExists") {
      console.warn(`${caught.message}. Did you provide a new document name?`);
    } else {
```

```
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateDocument in der AWS SDK für JavaScript API-Referenz.

`CreateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`CreateMaintenanceWindow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM maintenance window.
 * @param {{ name: string, allowUnassociatedTargets: boolean, duration: number,
 cutoff: number, schedule: string, description?: string }}
 */
export const main = async ({
  name,
  allowUnassociatedTargets, // Allow the maintenance window to run on managed nodes,
 even if you haven't registered those nodes as targets.
  duration, // The duration of the maintenance window in hours.
  cutoff, // The number of hours before the end of the maintenance window that
 Amazon Web Services Systems Manager stops scheduling new tasks for execution.
  schedule, // The schedule of the maintenance window in the form of a cron or rate
 expression.
  description = undefined,
}) => {
  const client = new SSMClient({});
```

```
  try {
    const { windowId } = await client.send(
      new CreateMaintenanceWindowCommand({
        Name: name,
        Description: description,
        AllowUnassociatedTargets: allowUnassociatedTargets, // Allow the maintenance
window to run on managed nodes, even if you haven't registered those nodes as
targets.
        Duration: duration, // The duration of the maintenance window in hours.
        Cutoff: cutoff, // The number of hours before the end of the maintenance
window that Amazon Web Services Systems Manager stops scheduling new tasks for
execution.
        Schedule: schedule, // The schedule of the maintenance window in the form of
a cron or rate expression.
      }),
    );
    console.log(`Maintenance window created with Id: ${windowId}`);
    return { WindowId: windowId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateMaintenanceWindow in der AWS SDK für JavaScript API-Referenz.

`CreateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`CreateOpsItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { CreateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Create an SSM OpsItem.
 * @param {{ title: string, source: string, category?: string, severity?: string }}
 */
export const main = async ({
  title,
  source,
  category = undefined,
  severity = undefined,
}) => {
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new CreateOpsItemCommand({
        Title: title,
        Source: source, // The origin of the OpsItem, such as Amazon EC2 or Systems
 Manager.
        Category: category,
        Severity: severity,
      }),
    );
    console.log(`Ops item created with id: ${opsItemId}`);
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie CreateOpsItem in der AWS SDK für JavaScript API-Referenz.

`DeleteDocument`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDocument`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteDocumentCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM document.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(new DeleteDocumentCommand({ Name: documentName }));
    console.log(`Document '${documentName}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteDocument in der AWS SDK für JavaScript API-Referenz.

`DeleteMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMaintenanceWindow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { DeleteMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Delete an SSM maintenance window.
 * @param {{ windowId: string }}
 */
export const main = async ({ windowId }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new DeleteMaintenanceWindowCommand({ WindowId: windowId }),
    );
    console.log(`Maintenance window '${windowId}' deleted.`);
    return { Deleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie DeleteMaintenanceWindow in der AWS SDK für JavaScript API-Referenz.

`DescribeOpsItems`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOpsItems`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import {
  OpsItemFilterOperator,
  OpsItemFilterKey,
  paginateDescribeOpsItems,
  SSMClient,
} from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Describe SSM OpsItems.
 * @param {{ opsItemId: string }}
 */
export const main = async ({ opsItemId }) => {
  const client = new SSMClient({});
  try {
    const describeOpsItemsPaginated = [];
    for await (const page of paginateDescribeOpsItems(
      { client },
      {
        OpsItemFilters: {
          Key: OpsItemFilterKey.OPSITEM_ID,
          Operator: OpsItemFilterOperator.EQUAL,
          Values: opsItemId,
        },
      },
    )) {
      describeOpsItemsPaginated.push(...page.OpsItemSummaries);
    }
    console.log("Here are the ops items:");
    console.log(describeOpsItemsPaginated);
    return { OpsItemSummaries: describeOpsItemsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide this value?`);
```

```
    }
    throw caught;
  }
};
```

- API-Details finden Sie DescribeOpsItems in der AWS SDK für JavaScript API-Referenz.

`ListCommandInvocations`

Das folgende Codebeispiel zeigt die Verwendung`ListCommandInvocations`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { paginateListCommandInvocations, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * List SSM command invocations on an instance.
 * @param {{ instanceId: string }}
 */
export const main = async ({ instanceId }) => {
  const client = new SSMClient({});
  try {
    const listCommandInvocationsPaginated = [];
    // The paginate function is a wrapper around the base command.
    const paginator = paginateListCommandInvocations(
      { client },
      {
        InstanceId: instanceId,
      },
    );
    for await (const page of paginator) {
      listCommandInvocationsPaginated.push(...page.CommandInvocations);
    }
    console.log("Here is the list of command invocations:");
```

```
    console.log(listCommandInvocationsPaginated);
    return { CommandInvocations: listCommandInvocationsPaginated };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid instance ID?`);
    }
    throw caught;
  }
};
```

- API-Details finden Sie ListCommandInvocations in der AWS SDK für JavaScript API-Referenz.

`SendCommand`

Das folgende Codebeispiel zeigt die Verwendung`SendCommand`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { SendCommandCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Send an SSM command to a managed node.
 * @param {{ documentName: string }}
 */
export const main = async ({ documentName }) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new SendCommandCommand({
        DocumentName: documentName,
      }),
    );
    console.log("Command sent successfully.");
    return { Success: true };
```

```
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Did you provide a valid document name?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie SendCommand in der AWS SDK für JavaScript API-Referenz.

`UpdateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateMaintenanceWindow`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateMaintenanceWindowCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";

/**
 * Update an SSM maintenance window.
 * @param {{ windowId: string, allowUnassociatedTargets?: boolean, duration?:
 number, enabled?: boolean, name?: string, schedule?: string }}
 */
export const main = async ({
  windowId,
  allowUnassociatedTargets = undefined, //Allow the maintenance window to run on
 managed nodes, even if you haven't registered those nodes as targets.
  duration = undefined, //The duration of the maintenance window in hours.
  enabled = undefined,
  name = undefined,
  schedule = undefined, //The schedule of the maintenance window in the form of a
 cron or rate expression.
}) => {
```

```
  const client = new SSMClient({});
  try {
    const { opsItemArn, opsItemId } = await client.send(
      new UpdateMaintenanceWindowCommand({
        WindowId: windowId,
        AllowUnassociatedTargets: allowUnassociatedTargets,
        Duration: duration,
        Enabled: enabled,
        Name: name,
        Schedule: schedule,
      }),
    );
    console.log("Maintenance window updated.");
    return { OpsItemArn: opsItemArn, OpsItemId: opsItemId };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ValidationError") {
      console.warn(`${caught.message}. Are these values correct?`);
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie UpdateMaintenanceWindow in der AWS SDK für JavaScript API-Referenz.

`UpdateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateOpsItem`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import { UpdateOpsItemCommand, SSMClient } from "@aws-sdk/client-ssm";
import { parseArgs } from "node:util";
```

```
/**
 * Update an SSM OpsItem.
 * @param {{ opsItemId: string, status?: OpsItemStatus }}
 */
export const main = async ({
  opsItemId,
  status = undefined, // The OpsItem status. Status can be Open, In Progress, or
 Resolved
}) => {
  const client = new SSMClient({});
  try {
    await client.send(
      new UpdateOpsItemCommand({
        OpsItemId: opsItemId,
        Status: status,
      }),
    );
    console.log("Ops item updated.");
    return { Success: true };
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "OpsItemLimitExceededException"
    ) {
      console.warn(
        `Couldn't create ops item because you have exceeded your open OpsItem limit.
 ${caught.message}.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- API-Details finden Sie UpdateOpsItem in der AWS SDK für JavaScript API-Referenz.

Amazon Textract-Beispiele mit SDK für JavaScript (v3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Textract Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für JavaScript (v3)

Zeigt, wie Sie das verwenden AWS SDK für JavaScript , um eine React-Anwendung zu erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie das innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });
```

```
  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });
```

```
  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
```

```
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly

- Amazon Textract
- Amazon Translate

Amazon Transcribe-Beispiele mit SDK für (v3) JavaScript

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Transcribe Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DeleteMedicalTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteMedicalTranscriptionJob`

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
```

```
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Einen Auftrag für medizinische Transkription löschen.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
 'medical_transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteMedicalTranscriptionJob in der AWS SDK für JavaScript API-Referenz.

`DeleteTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTranscriptionJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Transkriptionsauftrags.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME", // Required. For example, 'transciption_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteTranscriptionJobCommand(params),
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie DeleteTranscriptionJob in der AWS SDK für JavaScript API-Referenz.

`ListMedicalTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListMedicalTranscriptionJobs`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Auflisten medizinischer Transkriptionsjobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
```

```
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the same
 region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListMedicalTranscriptionJobs in der AWS SDK für JavaScript API-Referenz.

`ListTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTranscriptionJobs`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten von Transkriptionsaufträgen.

```
// Import the required AWS SDK clients and commands for Node.js

import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params),
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie ListTranscriptionJobs in der AWS SDK für JavaScript API-Referenz.

`StartMedicalTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`StartMedicalTranscriptionJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Einen Auftrag für medizinische Transkription starten.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the same
 region
    // as the API endpoint that you are calling.For example,
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
};

export const run = async () => {
```

```
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie StartMedicalTranscriptionJob in der AWS SDK für JavaScript API-Referenz.

`StartTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`StartTranscriptionJob`.

SDK für JavaScript (v3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einen Transkriptionsauftrag starten.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
```

```
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME",
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartTranscriptionJobCommand(params),
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Erstellen Sie den Client.

```
import { TranscribeClient } from "@aws-sdk/client-transcribe";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

- Weitere Informationen finden Sie im AWS SDK für JavaScript -Entwicklerhandbuch.
- API-Details finden Sie StartTranscriptionJob in der AWS SDK für JavaScript API-Referenz.

Szenarien

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Beispiele für Amazon Translate, die das SDK für JavaScript (v3) verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für JavaScript (v3) mit Amazon Translate Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Amazon-Transcribe-Streaming-App

Das folgende Code-Beispiel zeigt, wie Sie eine App erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse per E-Mail sendet.

SDK für JavaScript (v3)

Zeigt, wie Amazon Transcribe verwendet wird, um eine App zu erstellen, die Live-Audio in Echtzeit aufzeichnet, transkribiert und übersetzt und die Ergebnisse mit Amazon Simple Email Service (Amazon SES) per E-Mail sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für JavaScript (v3)

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel Building an Amazon Lex-Chatbot im AWS SDK für JavaScript Entwicklerhandbuch.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für JavaScript (v3)

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub. Die folgenden Auszüge zeigen, wie das innerhalb von Lambda-Funktionen verwendet AWS SDK für JavaScript wird.

```
import {
  ComprehendClient,
  DetectDominantLanguageCommand,
  DetectSentimentCommand,
} from "@aws-sdk/client-comprehend";

/**
 * Determine the language and sentiment of the extracted text.
 *
 * @param {{ source_text: string}} extractTextOutput
 */
export const handler = async (extractTextOutput) => {
  const comprehendClient = new ComprehendClient({});

  const detectDominantLanguageCommand = new DetectDominantLanguageCommand({
    Text: extractTextOutput.source_text,
  });
```

```
  // The source language is required for sentiment analysis and
  // translation in the next step.
  const { Languages } = await comprehendClient.send(
    detectDominantLanguageCommand,
  );

  const languageCode = Languages[0].LanguageCode;

  const detectSentimentCommand = new DetectSentimentCommand({
    Text: extractTextOutput.source_text,
    LanguageCode: languageCode,
  });

  const { Sentiment } = await comprehendClient.send(detectSentimentCommand);

  return {
    sentiment: Sentiment,
    language_code: languageCode,
  };
};
```

```
import {
  DetectDocumentTextCommand,
  TextractClient,
} from "@aws-sdk/client-textract";

/**
 * Fetch the S3 object from the event and analyze it using Amazon Textract.
 *
 * @param {import("@types/aws-lambda").EventBridgeEvent<"Object Created">}
 eventBridgeS3Event
 */
export const handler = async (eventBridgeS3Event) => {
  const textractClient = new TextractClient();

  const detectDocumentTextCommand = new DetectDocumentTextCommand({
    Document: {
      S3Object: {
        Bucket: eventBridgeS3Event.bucket,
        Name: eventBridgeS3Event.object,
      },
    },
  });
```

```
  // Textract returns a list of blocks. A block can be a line, a page, word, etc.
  // Each block also contains geometry of the detected text.
  // For more information on the Block type, see https://docs.aws.amazon.com/
textract/latest/dg/API_Block.html.
  const { Blocks } = await textractClient.send(detectDocumentTextCommand);

  // For the purpose of this example, we are only interested in words.
  const extractedWords = Blocks.filter((b) => b.BlockType === "WORD").map(
    (b) => b.Text,
  );

  return extractedWords.join(" ");
};
```

```
import { PollyClient, SynthesizeSpeechCommand } from "@aws-sdk/client-polly";
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

/**
 * Synthesize an audio file from text.
 *
 * @param {{ bucket: string, translated_text: string, object: string}}
 sourceDestinationConfig
 */
export const handler = async (sourceDestinationConfig) => {
  const pollyClient = new PollyClient({});

  const synthesizeSpeechCommand = new SynthesizeSpeechCommand({
    Engine: "neural",
    Text: sourceDestinationConfig.translated_text,
    VoiceId: "Ruth",
    OutputFormat: "mp3",
  });

  const { AudioStream } = await pollyClient.send(synthesizeSpeechCommand);

  const audioKey = `${sourceDestinationConfig.object}.mp3`;

  // Store the audio file in S3.
  const s3Client = new S3Client();
  const upload = new Upload({
    client: s3Client,
```

```
    params: {
      Bucket: sourceDestinationConfig.bucket,
      Key: audioKey,
      Body: AudioStream,
      ContentType: "audio/mp3",
    },
  });

  await upload.done();
  return audioKey;
};
```

```
import {
  TranslateClient,
  TranslateTextCommand,
} from "@aws-sdk/client-translate";

/**
 * Translate the extracted text to English.
 *
 * @param {{ extracted_text: string, source_language_code: string}}
 textAndSourceLanguage
 */
export const handler = async (textAndSourceLanguage) => {
  const translateClient = new TranslateClient({});

  const translateCommand = new TranslateTextCommand({
    SourceLanguageCode: textAndSourceLanguage.source_language_code,
    TargetLanguageCode: "en",
    Text: textAndSourceLanguage.extracted_text,
  });

  const { TranslatedText } = await translateClient.send(translateCommand);

  return { translated_text: TranslatedText };
};
```

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly

- Amazon Textract
- Amazon Translate

Codebeispiele für SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK for Java 2.x with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for Java 2.x Developer Guide — Weitere Informationen zur Verwendung von Java mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für API Gateway unter Verwendung von SDK für Java 2.x
- Beispiele für Application Auto Scaling unter Verwendung von SDK für Java 2.x
- Beispiele für Application Recovery Controller unter Verwendung von SDK für Java 2.x
- Aurora-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für Auto Scaling unter Verwendung von SDK für Java 2.x
- AWS Batch Beispiele mit SDK für Java 2.x

- Beispiele für Amazon Bedrock unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Java 2.x
- CloudFront Beispiele zur Verwendung des SDK für Java 2.x
- CloudWatch Beispiele zur Verwendung des SDK für Java 2.x
- CloudWatch Beispiele für Ereignisse, die das SDK für Java 2.x verwenden
- CloudWatch Protokolliert Beispiele mithilfe des SDK für Java 2.x
- Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Java 2.x
- AWS Control Tower Beispiele zur Verwendung des SDK für Java 2.x
- Firehose-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Java 2.x
- DynamoDB-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon EC2 unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon ECR unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon ECS unter Verwendung von SDK für Java 2.x
- Elastic Load Balancing – Beispiele für Version 2 unter Verwendung von SDK für Java 2.x
- MediaStore Beispiele mit SDK für Java 2.x
- AWS Entity Resolution Beispiele zur Verwendung des SDK für Java 2.x
- OpenSearch Servicebeispiele, die SDK für Java 2.x verwenden
- EventBridge Beispiele zur Verwendung des SDK für Java 2.x
- EventBridge Scheduler-Beispiele, die SDK für Java 2.x verwenden
- Forecast-Beispiele unter Verwendung von SDK für Java 2.x
- Amazon Glacier-Beispiele mit SDK für Java 2.x
- AWS Glue Beispiele zur Verwendung des SDK für Java 2.x
- HealthImaging Beispiele zur Verwendung des SDK für Java 2.x
- IAM-Beispiele unter Verwendung von SDK für Java 2.x
- Amazon Inspector-Beispiele, die SDK für Java 2.x verwenden

- AWS IoT Beispiele zur Verwendung des SDK für Java 2.x
- AWS IoT data Beispiele zur Verwendung des SDK für Java 2.x
- AWS IoT SiteWise Beispiele zur Verwendung des SDK für Java 2.x
- Beispiele für Amazon Keyspaces unter Verwendung von SDK für Java 2.x
- Kinesis-Beispiele unter Verwendung von SDK für Java 2.x
- AWS KMS Beispiele mit SDK für Java 2.x
- Lambda-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Lex unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Location unter Verwendung von SDK für Java 2.x
- Beispiele für Location Service Places unter Verwendung von SDK für Java 2.x
- AWS Marketplace Katalogisieren Sie API-Beispiele mithilfe des SDK für Java 2.x
- AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für Java 2.x
- MediaConvert Beispiele zur Verwendung des SDK für Java 2.x
- Beispiele für Migration Hub unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon MSK unter Verwendung von SDK für Java 2.x
- Neptune-Beispiele unter Verwendung von SDK für Java 2.x
- Beispiele für Partner Central unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Personalize unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Personalize Events unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Personalize Runtime unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Pinpoint unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Pinpoint SMS und Voice API unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Polly unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon RDS unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Redshift unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Rekognition unter Verwendung von SDK für Java 2.x
- Codebeispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon S3 unter Verwendung von SDK für Java 2.x

- Beispiele für Amazon S3 Control unter Verwendung von SDK für Java 2.x
- Beispiele für S3 Directory Buckets unter Verwendung von SDK für Java 2.x
- SageMaker KI-Beispiele, die das SDK für Java 2.x verwenden
- Beispiele für Secrets Manager unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon SES unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon SES API v2 unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon SNS unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon SQS unter Verwendung von SDK für Java 2.x
- Beispiele für Step Functions unter Verwendung von SDK für Java 2.x
- AWS STS Beispiele mit SDK für Java 2.x
- Support Beispiele zur Verwendung des SDK für Java 2.x
- Beispiele für Systems Manager unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Textract unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Transcribe unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon Translate unter Verwendung von SDK für Java 2.x
- X-Ray Beispiele zur Verwendung des SDK für Java 2.x

ACM-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von ACM Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AddTagsToCertificate`

Das folgende Codebeispiel zeigt die Verwendung`AddTagsToCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTagsToCertificate {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
```

```
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add tags
 to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie AddTagsToCertificate in der AWS SDK for Java 2.x API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        deleteCertificate(certArn);
    }

    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
```

```
}
```

- API-Details finden Sie DeleteCertificate in der AWS SDK for Java 2.x API-Referenz.

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to describe
     * @throws AcmException if an error occurs while describing the certificate
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " + response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie DescribeCertificate in der AWS SDK for Java 2.x API-Referenz.

`ExportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ExportCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and certificate
 chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
 want to export.
```

```
    * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
    */
   public static void exportCert(String certArn) throws IOException {
       AcmClient acmClient = AcmClient.create();

       // Initialize a file descriptor for the passphrase file.
       RandomAccessFile filePassphrase = null;
       ByteBuffer bufPassphrase = null;

       // Create a file stream for reading the private key passphrase.
       try {
           filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
       } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
           throw ex;
       }

       // Create a channel to map the file.
       FileChannel channelPassphrase = filePassphrase.getChannel();

       // Map the file to the buffer.
       try {
           bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY, 0,
channelPassphrase.size());
           channelPassphrase.close();
           filePassphrase.close();
       } catch (IOException ex) {
           throw ex;
       }

       // Create a request object.
       ExportCertificateRequest req = ExportCertificateRequest.builder()
           .certificateArn(certArn)
           .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
           .build();

       // Export the certificate.
       ExportCertificateResponse result = null;
       try {
           result = acmClient.exportCertificate(req);
       } catch (InvalidArnException | InvalidTagException |
ResourceNotFoundException ex) {
           throw ex;
```

```
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);

        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- API-Details finden Sie ExportCertificate in der AWS SDK for Java 2.x API-Referenz.

`ImportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ImportCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportCert {
```

```
   public static void main(String[] args) {
       final String usage = """
           Usage: <bucketName> <certificateKey> <privateKeyKey>

           Where:
               bucketName - The name of the S3 bucket containing the certificate
and private key.
               certificateKey - The object key for the SSL/TLS certificate file in
S3.
               privateKeyKey - The object key for the private key file in S3.
           """;

       if (args.length != 3) {
           System.out.println(usage);
           return;
       }

       String bucketName = args[0];
       String certificateKey = args[1];
       String privateKeyKey = args[2];

       String certificateArn = importCertificate(bucketName, certificateKey,
privateKeyKey);
       System.out.println("Certificate imported with ARN: " + certificateArn);
   }

   /**
    * Imports an SSL/TLS certificate and private key from S3 into AWS Certificate
Manager (ACM).
    *
    * @param bucketName     The name of the S3 bucket.
    * @param certificateKey The key for the SSL/TLS certificate file in S3.
    * @param privateKeyKey  The key for the private key file in S3.
    * @return The ARN of the imported certificate.
    */
   public static String importCertificate(String bucketName, String certificateKey,
String privateKeyKey) {
       AcmClient acmClient = AcmClient.create();
       S3Client s3Client = S3Client.create();

       try {
           byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
```

```
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))

.privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
acmClient.importCertificate(request);
            return response.certificateArn();

        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " + e.awsErrorDetails().errorMessage());
        }
        return "";
    }

    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String bucketName,
String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
```

```
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- API-Details finden Sie ImportCertificate in der AWS SDK for Java 2.x API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that have
 a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest = ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
```

```
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie ListCertificates in der AWS SDK for Java 2.x API-Referenz.

`ListTagsForCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
```

```
        });
    }
}
```

- API-Details finden Sie ListTagsForCertificate in der AWS SDK for Java 2.x API-Referenz.

`RemoveTagsFromCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTagsFromCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
```

```
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from which
 to remove tags
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
 RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie RemoveTagsFromCertificate in der AWS SDK for Java 2.x API-Referenz.

`RenewCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RenewCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
 renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
```

```
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
 RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie RenewCertificate in der AWS SDK for Java 2.x API-Referenz.

`RequestCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RequestCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RequestCert {
```

```
    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");

        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response = acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- API-Details finden Sie RequestCertificate in der AWS SDK for Java 2.x API-Referenz.

Beispiele für API Gateway unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von API Gateway Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden AWS. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- AWS Beiträge der Gemeinschaft

Aktionen

`CreateDeployment`

Das folgende Codebeispiel zeigt die Verwendung`CreateDeployment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

       try {
           CreateDeploymentRequest request = CreateDeploymentRequest.builder()
                   .restApiId(restApiId)
                   .description("Created using the AWS API Gateway Java API")
                   .stageName(stageName)
                   .build();

           CreateDeploymentResponse response =
apiGateway.createDeployment(request);
           System.out.println("The id of the deployment is " + response.id());
           return response.id();
```

```
        } catch (ApiGatewayException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateDeployment in der AWS SDK for Java 2.x API-Referenz.

`CreateRestApi`

Das folgende Codebeispiel zeigt die Verwendung`CreateRestApi`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createAPI(ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

        try {
            CreateRestApiRequest request = CreateRestApiRequest.builder()
                    .cloneFrom(restApiId)
                    .description("Created using the Gateway Java API")
                    .name(restApiName)
                    .build();

            CreateRestApiResponse response = apiGateway.createRestApi(request);
            System.out.println("The id of the new api is " + response.id());
            return response.id();

        } catch (ApiGatewayException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
```

```
    }
```

- API-Details finden Sie CreateRestApi in der AWS SDK for Java 2.x API-Referenz.

`DeleteDeployment`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDeployment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

       try {
           DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
                   .restApiId(restApiId)
                   .deploymentId(deploymentId)
                   .build();

           apiGateway.deleteDeployment(request);
           System.out.println("Deployment was deleted");

       } catch (ApiGatewayException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DeleteDeployment in der AWS SDK for Java 2.x API-Referenz.

`DeleteRestApi`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRestApi`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static void deleteAPI(ApiGatewayClient apiGateway, String restApiId) {

    try {
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()
                .restApiId(restApiId)
                .build();

        apiGateway.deleteRestApi(request);
        System.out.println("The API was successfully deleted");

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API-Details finden Sie DeleteRestApi in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für Application Auto Scaling unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Application Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteScalingPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
 appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
 ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
 DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
```

```
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");
```

```
        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- API-Details finden Sie DeleteScalingPolicy in der AWS SDK for Java 2.x API-Referenz.

`RegisterScalableTarget`

Das folgende Codebeispiel zeigt die Verwendung`RegisterScalableTarget`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyCon
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
```

```
               policyName - The name of the policy to create.

           """;

       if (args.length != 3) {
           System.out.println(usage);
           System.exit(1);
       }

       System.out.println("This example registers an Amazon DynamoDB table, which
is the resource to scale.");
       String tableId = args[0];
       String roleARN = args[1];
       String policyName = args[2];
       ServiceNamespace ns = ServiceNamespace.DYNAMODB;
       ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
       ApplicationAutoScalingClient appAutoScalingClient =
ApplicationAutoScalingClient.builder()
           .region(Region.US_EAST_1)
           .build();

       registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
tableWCUs);
       verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
       configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
   }

   public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
       try {
           RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
               .serviceNamespace(ns)
               .scalableDimension(tableWCUs)
               .resourceId(tableId)
               .roleARN(roleARN)
               .minCapacity(5)
               .maxCapacity(10)
               .build();

           appAutoScalingClient.registerScalableTarget(targetRequest);
```

```
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
    public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
        // Check if the policy exists before creating a new one.
        DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
            .serviceNamespace(ns)
            .resourceId(tableId)
            .scalableDimension(tableWCUs)
            .build());

        if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
            // If policies exist, consider updating an existing policy instead of
creating a new one.
            System.out.println("Policy already exists. Consider updating it
instead.");
            List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
            for (ScalingPolicy pol : polList) {
```

```
                System.out.println("Policy name:" +pol.policyName());
            }
        } else {
            // If no policies exist, proceed with creating a new policy.
            PredefinedMetricSpecification specification =
 PredefinedMetricSpecification.builder()

 .predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
                .build();

            TargetTrackingScalingPolicyConfiguration policyConfiguration =
 TargetTrackingScalingPolicyConfiguration.builder()
                .predefinedMetricSpecification(specification)
                .targetValue(50.0)
                .scaleInCooldown(60)
                .scaleOutCooldown(60)
                .build();

            PutScalingPolicyRequest putScalingPolicyRequest =
 PutScalingPolicyRequest.builder()
                .targetTrackingScalingPolicyConfiguration(policyConfiguration)
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .policyName(policyName)
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling policy
 for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- API-Details finden Sie RegisterScalableTarget in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Application Recovery Controller unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK for Java 2.x Application Recovery Controllers Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung`GetRoutingControlState`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
```

```
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- API-Details finden Sie GetRoutingControlState in der AWS SDK for Java 2.x API-Referenz.

`UpdateRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung`UpdateRoutingControlState`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
```

```
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

.routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- API-Details finden Sie UpdateRoutingControlState in der AWS SDK for Java 2.x API-Referenz.

Aurora-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
```

```
                        .println("Database name: " + cluster.databaseName() + " Arn
 = " + cluster.dbClusterArn()));
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK for Java 2.x - API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
```

```
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
```

```
                "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group family
name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster identifier
value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n" +
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
```

```
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();

            rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .skipFinalSnapshot(true)
                    .build();
```

```
            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier + " was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();
```

```
            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .build();

           String endpoint = "";
           while (!instanceReady) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
               List<DBInstance> instanceList = response.dbInstances();
               for (DBInstance instance : instanceList) {
                   instanceReadyStr = instance.dbInstanceStatus();
                   if (instanceReadyStr.contains("available")) {
                       endpoint = instance.endpoint().address();
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
```

```
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
```

```
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   // Modify the auto_increment_offset parameter.
   public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
       try {
           Parameter parameter1 = Parameter.builder()
                   .parameterName("auto_increment_offset")
                   .applyMethod("immediate")
                   .parameterValue("5")
                   .build();

           List<Parameter> paraList = new ArrayList<>();
           paraList.add(parameter1);
           ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dClusterGroupName)
                   .parameters(paraList)
                   .build();

           ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
           System.out.println(
                   "The parameter group " + response.dbClusterParameterGroupName()
+ " was successfully modified");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
```

```
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
```

```
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
            String dbParameterGroupFamily) {
        try {
            CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();
```

```
            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances

- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK for Java 2.x - API-Referenz.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK for Java 2.x API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK for Java 2.x API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDBInstanceCluster(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbInstanceClusterIdentifier,
           String instanceClass) {
       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .engine("aurora-mysql")
                   .dbInstanceClass(instanceClass)
                   .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
       return "";
   }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK for Java 2.x -API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(clusterDBARN) == 0) {
                       System.out.println(clusterDBARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }

           DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                   .builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .build();

           rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
           System.out.println(dbClusterGroupName + " was deleted.");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") == 0)) {
```

```
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
            String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");
```

```
            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
                       .build();
           }

           DescribeDbClusterParametersResponse response = rdsClient
                   .describeDBClusterParameters(dbParameterGroupsRequest);
           List<Parameter> dbParameters = response.parameters();
           String paraName;
           for (Parameter para : dbParameters) {
               // Only print out information about either auto_increment_offset or
               // auto_increment_increment.
               paraName = para.parameterName();
               if ((paraName.compareTo("auto_increment_offset") == 0)
                       || (paraName.compareTo("auto_increment_increment ") == 0)) {
                   System.out.println("*** The parameter name is  " + paraName);
                   System.out.println("*** The parameter value is  " +
para.parameterValue());
                   System.out.println("*** The parameter data type is " +
para.dataType());
                   System.out.println("*** The parameter description is " +
para.description());
```

```
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK for Java 2.x - API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
```

```
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
engineOb.engine());
                System.out.println("The version number of the database engine " +
engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
```

```
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK for Java 2.x API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Auto Scaling unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
        });
    }
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
```

```
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_WEST_2)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will create a launch template using a
CloudFormation script");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
```

```
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " + e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
        }
    }


    public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
        // Step 1: Create new launch template version
        String newAmiId = "ami-0025f0db847eb6254";
        RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
                .imageId(newAmiId)
                .build();

        CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
                .launchTemplateName(launchTemplateName)
                .versionDescription("Updated with valid AMI")
                .sourceVersion("1")
                .launchTemplateData(launchTemplateData)
                .build();

        CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
        int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

        // Step 2: Modify default version
        ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
```

```
                .launchTemplateName(launchTemplateName)
                .defaultVersion(String.valueOf(newVersionNumber))
                .build();

        ec2.modifyLaunchTemplate(modifyRequest);
        System.out.println("Updated launch template to version " + newVersionNumber
+ " with AMI " + newAmiId);
    }


    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
```

```
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                              Ec2Client ec2Client,
                                              String groupName,
                                              String launchTemplateName,
                                              String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest = DescribeSubnetsRequest.builder()
                    .filters(Filter.builder().name("vpc-id").values(vpcId).build())
                    .build();

            DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC ID
```

```
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
                    waiter.waitUntilGroupExists(groupsRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (Ec2Exception | AutoScalingException e) {
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
```

```
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
```

```
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
       try {
           DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
           System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
           System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
```

```
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group  " +
groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
```

```
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
```

```
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
```

```
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
 String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK for Java 2.x API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAutoScalingInstances {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient, String
groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
```

```
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK for Java 2.x API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
                   .build();

           DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
           List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
```

```
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK for Java 2.x API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
```

```
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK for Java 2.x API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK for Java 2.x API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK for Java 2.x API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK for Java 2.x API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
       try {
           TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                   .instanceId(instanceId)
                   .shouldDecrementDesiredCapacity(false)
                   .build();

           autoScalingClient.terminateInstanceInAutoScalingGroup(request);
           System.out.println("You have terminated instance " + instanceId);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK for Java 2.x API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
                   .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .build();

           WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                   .waitUntilGroupInService(groupsRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("You successfully updated the auto scaling group  " +
groupName);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
```

```
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

```
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
```

```
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
```

```
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();

       // Create a new instance profile based on badCredsProfileName.
       templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
       String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
       System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

       String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
       System.out.println("The association Id value is " + profileAssociationId);
```

```
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
```

```
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
```

```
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();
```

```
                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                        target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
```

```
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
```

```
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();
```

```
        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
```

```
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
```

```
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
```

```
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
```

```
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }
```

```
    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }
```

```
   // Gets data about the instances in the EC2 Auto Scaling group.
   public String getBadInstance(String groupName) {
       DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
               .autoScalingGroupNames(groupName)
               .build();

       DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
       AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
       List<String> instanceIds = autoScalingGroup.instances().stream()
               .map(instance -> instance.instanceId())
               .collect(Collectors.toList());

       String[] instanceIdArray = instanceIds.toArray(new String[0]);
       for (String instanceId : instanceIdArray) {
           System.out.println("Instance ID: " + instanceId);
           return instanceId;
       }
       return "";
   }

   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
```

```
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
```

```
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
```

```
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
```

```
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;
```

```
    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
```

```
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }
```

```
    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {
```

```
    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup

- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

AWS Batch Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS Batch.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Batch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Batch beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
```

```
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
```

```
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
}
```

- API-Details finden Sie JobsPaginator in der Liste in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine AWS Batch Rechenumgebung.
- Überprüfen des Status der Datenverarbeitungsumgebung
- Richten Sie eine AWS Batch Job-Warteschlange und eine Jobdefinition ein.
- Registrieren einer Auftragsdefinition
- Reichen Sie einen AWS Batch Job ein.
- Abrufen einer Liste der für die Auftragswarteschlange relevanten Aufträge
- Überprüfen des Auftragsstatus
- AWS Batch Ressourcen löschen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Batch Funktionen demonstriert.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from Amazon
 ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr
 *
 */
```

```
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that dynamically
 provisions the required compute
            resources for batch computing workloads. The Java V2 `BatchAsyncClient`
 allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute environment,
 job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr
```

```
            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
```

```
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch jobs.
            After creating a compute environment, you can define job queues and job
definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to easily
configure and manage the compute
```

```
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" + computeEnvironmentName
+ "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +" Compute
Environment.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the execution
of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
```

```
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and so
on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build the
Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64
```

```
                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
```

```
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}", job.jobId(),
job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future = batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);
```

```
        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
```

```
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: " +
ex.getMessage());
```

```
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }


batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");

            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
```

```
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
                            .orElseThrow(() -> new RuntimeException("Default VPC not
found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
```

```
                                                        .build(),
                                                Filter.builder()
                                                        .name("default-for-az")
                                                        .values("true")
                                                        .build())
                                        .build())
                                .thenApply(DescribeSubnetsResponse::subnets)
                                .thenApply(subnets -> subnets.stream()
                                        .findFirst()
                                        .map(Subnet::subnetId)
                                        .orElseThrow(() -> new RuntimeException("No
 default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture = defaultVpcFuture
                    .thenCompose(vpc ->
 ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                        .filters(Filter.builder()
                                                        .name("group-name")
                                                        .values("default")
                                                        .build(),
                                                Filter.builder()
                                                        .name("vpc-id")
                                                        .values(vpc.vpcId())
                                                        .build())
                                        .build())

 .thenApply(DescribeSecurityGroupsResponse::securityGroups)
                                .thenApply(securityGroups -> securityGroups.stream()
                                        .findFirst()
                                        .map(SecurityGroup::groupId)
                                        .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

Eine Wrapper-Klasse für AWS Batch SDK-Methoden.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
```

```
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchActions.class);

    private static BatchAsyncClient getAsyncClient() {
        if (batchClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
```

```
                    .build())
                .build();

            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }

    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute environment
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
     * @throws BatchException if there is an error creating the compute environment
     * @throws RuntimeException if there is an unexpected error during the operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();
```

```
        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
               throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }

    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
     * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
     * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
     */
    public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
        if (jobQueueName == null || jobQueueName.isEmpty()) {
            throw new IllegalArgumentException("Job queue name cannot be null or
empty");
        }
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }
```

```
        CreateJobQueueRequest request = CreateJobQueueRequest.builder()
            .jobQueueName(jobQueueName)
            .priority(1)
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }

    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
```

```
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
     *                         that provides permissions for the containers in the
job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes exceptionally
with
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
```

```
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
```

```
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
```

```
    * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
    *         The future completes when the job queue has been successfully deleted
or if an error occurs.
    *         If successful, the future will be completed with a {@code Void}
value.
    *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
    */
   public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
       DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .build();

       CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
       return responseFuture.whenComplete((deleteResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(deleteResponse -> null);
   }

   /**
    * Asynchronously describes the job queue associated with the specified compute
environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
```

```
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
```

```
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }

   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
```

```
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
     * @return a {@link CompletableFuture} that completes when the job queue is
disabled
     */
    public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
        AtomicBoolean isDisabled = new AtomicBoolean(false);
        return CompletableFuture.runAsync(() -> {
            while (!isDisabled.get()) {
                DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                    .jobQueues(jobQueueArn)
                    .build();
```

```
                CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
                responseFuture.whenComplete((describeResponse, ex) -> {
                    if (describeResponse != null) {
                        for (JobQueueDetail jobQueue : describeResponse.jobQueues())
{
                            if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                                isDisabled.set(true);
                                break;
                            }
                        }
                    } else {
                        throw new RuntimeException("Error describing job queues",
ex);
                    }
                }).join();

                if (!isDisabled.get()) {
                    try {
                        logger.info("Waiting for job queue to be disabled...");
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                        throw new RuntimeException("Thread interrupted while waiting
for job queue to be disabled", e);
                    }
                }
            }
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to be
disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
```

```
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new RuntimeException("Failed
to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn = response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

Aktionen

`CreateComputeEnvironment`

Das folgende Codebeispiel zeigt die Verwendung`CreateComputeEnvironment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute environment
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
           .serviceRole(batchIAMRole)
           .build();

       CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
              throw new RuntimeException(errorMessage, ex);
           }
       });

       return response;
```

```
    }
```

- API-Details finden Sie CreateComputeEnvironment in der AWS SDK for Java 2.x API-Referenz.

`CreateJobQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateJobQueue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
```

```
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }
```

- API-Details finden Sie CreateJobQueue in der AWS SDK for Java 2.x API-Referenz.

`DeleteComputeEnvironment`

Das folgende Codebeispiel zeigt die Verwendung`DeleteComputeEnvironment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
```

```
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }
```

- API-Details finden Sie DeleteComputeEnvironment in der AWS SDK for Java 2.x API-Referenz.

`DeleteJobQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJobQueue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
```

```
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }
```

- API-Details finden Sie DeleteJobQueue in der AWS SDK for Java 2.x API-Referenz.

`DeregisterJobDefinition`

Das folgende Codebeispiel zeigt die Verwendung`DeregisterJobDefinition`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
```

```
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }
```

- API-Details finden Sie DeregisterJobDefinition in der AWS SDK for Java 2.x API-Referenz.

`DescribeComputeEnvironments`

Das folgende Codebeispiel zeigt die Verwendung`DescribeComputeEnvironments`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
```

```
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- API-Details finden Sie DescribeComputeEnvironments in der AWS SDK for Java 2.x API-Referenz.

`DescribeJobQueues`

Das folgende Codebeispiel zeigt die Verwendung`DescribeJobQueues`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the job queue associated with the specified compute
environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name = getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
                           logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                       }
                   }
               }
           } else {
               throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(describeJobQueuesResponse -> {
           String jobQueueARN = "";
           for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
```

```
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- API-Details finden Sie DescribeJobQueues in der AWS SDK for Java 2.x API-Referenz.

`DescribeJobs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeJobs`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
```

```
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- API-Details finden Sie DescribeJobs in der AWS SDK for Java 2.x API-Referenz.

`ListJobsPaginator`

Das folgende Codebeispiel zeigt die Verwendung`ListJobsPaginator`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
```

```
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }
```

- API-Details finden Sie ListJobsPaginator in der AWS SDK for Java 2.x API-Referenz.

`RegisterJobDefinition`

Das folgende Codebeispiel zeigt die Verwendung`RegisterJobDefinition`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
     *                         that provides permissions for the containers in the
job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes exceptionally
with
```

```
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
```

```
                        future.complete(result);
                    }
                });

        return future;
    }
```

- API-Details finden Sie RegisterJobDefinition in der AWS SDK for Java 2.x API-Referenz.

`SubmitJob`

Das folgende Codebeispiel zeigt die Verwendung`SubmitJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
     */
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
```

```
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }
```

- API-Details finden Sie SubmitJob in der AWS SDK for Java 2.x API-Referenz.

`UpdateComputeEnvironment`

Das folgende Codebeispiel zeigt die Verwendung`UpdateComputeEnvironment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
```

```
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }
```

- API-Details finden Sie UpdateComputeEnvironment in der AWS SDK for Java 2.x API-Referenz.

`UpdateJobQueue`

Das folgende Codebeispiel zeigt die Verwendung`UpdateJobQueue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();
```

```
        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }
```

- API-Details finden Sie UpdateJobQueue in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Bedrock unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetFoundationModel`

Das folgende Codebeispiel zeigt die Verwendung`GetFoundationModel`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erfahren Sie mehr über ein Basismodell mithilfe des synchronen Amazon-Bedrock-Clients.

```
   /**
    * Get details about an Amazon Bedrock foundation model.
    *
    * @param bedrockClient   The service client for accessing Amazon Bedrock.
    * @param modelIdentifier The model identifier.
    * @return An object containing the foundation model's details.
    */
   public static FoundationModelDetails getFoundationModel(BedrockClient
bedrockClient, String modelIdentifier) {
       try {
           GetFoundationModelResponse response = bedrockClient.getFoundationModel(
                   r -> r.modelIdentifier(modelIdentifier)
           );

           FoundationModelDetails model = response.modelDetails();

           System.out.println(" Model ID:                     " + model.modelId());
           System.out.println(" Model ARN:                    " +
model.modelArn());
           System.out.println(" Model Name:                   " +
model.modelName());
           System.out.println(" Provider Name:                " +
model.providerName());
           System.out.println(" Lifecycle status:             " +
model.modelLifecycle().statusAsString());
           System.out.println(" Input modalities:             " +
model.inputModalities());
           System.out.println(" Output modalities:            " +
model.outputModalities());
           System.out.println(" Supported customizations:     " +
model.customizationsSupported());
           System.out.println(" Supported inference types:    " +
model.inferenceTypesSupported());
           System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());

           return model;

       } catch (ValidationException e) {
           throw new IllegalArgumentException(e.getMessage());
       } catch (SdkException e) {
           System.err.println(e.getMessage());
```

```
            throw new RuntimeException(e);
        }
    }
```

Erfahren Sie mehr über ein Basismodell mithilfe des asynchronen Amazon-Bedrock-Clients.

```
    /**
     * Get details about an Amazon Bedrock foundation model.
     *
     * @param bedrockClient   The async service client for accessing Amazon Bedrock.
     * @param modelIdentifier The model identifier.
     * @return An object containing the foundation model's details.
     */
    public static FoundationModelDetails getFoundationModel(BedrockAsyncClient
bedrockClient, String modelIdentifier) {
        try {
            CompletableFuture<GetFoundationModelResponse> future =
bedrockClient.getFoundationModel(
                    r -> r.modelIdentifier(modelIdentifier)
            );

            FoundationModelDetails model = future.get().modelDetails();

            System.out.println(" Model ID:                      " + model.modelId());
            System.out.println(" Model ARN:                     " +
model.modelArn());
            System.out.println(" Model Name:                    " +
model.modelName());
            System.out.println(" Provider Name:                 " +
model.providerName());
            System.out.println(" Lifecycle status:              " +
model.modelLifecycle().statusAsString());
            System.out.println(" Input modalities:              " +
model.inputModalities());
            System.out.println(" Output modalities:             " +
model.outputModalities());
            System.out.println(" Supported customizations:      " +
model.customizationsSupported());
            System.out.println(" Supported inference types:     " +
model.inferenceTypesSupported());
            System.out.println(" Response streaming supported: " +
model.responseStreamingSupported());
```

```
            return model;

        } catch (ExecutionException e) {
            if (e.getMessage().contains("ValidationException")) {
                throw new IllegalArgumentException(e.getMessage());
            } else {
                System.err.println(e.getMessage());
                throw new RuntimeException(e);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- API-Details finden Sie GetFoundationModel in der AWS SDK for Java 2.x API-Referenz.

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie die verfügbaren Amazon-Bedrock-Basismodelle mithilfe des synchronen Amazon-Bedrock-Clients auf.

```
    /**
     * Lists Amazon Bedrock foundation models that you can use.
     * You can filter the results with the request parameters.
     *
     * @param bedrockClient The service client for accessing Amazon Bedrock.
     * @return A list of objects containing the foundation models' details
     */
```

```
   public static List<FoundationModelSummary> listFoundationModels(BedrockClient
bedrockClient) {

       try {
           ListFoundationModelsResponse response =
bedrockClient.listFoundationModels(r -> {});

           List<FoundationModelSummary> models = response.modelSummaries();

           if (models.isEmpty()) {
               System.out.println("No available foundation models in " +
region.toString());
           } else {
               for (FoundationModelSummary model : models) {
                   System.out.println("Model ID: " + model.modelId());
                   System.out.println("Provider: " + model.providerName());
                   System.out.println("Name:     " + model.modelName());
                   System.out.println();
               }
           }

           return models;

       } catch (SdkClientException e) {
           System.err.println(e.getMessage());
           throw new RuntimeException(e);
       }
   }
```

Listen Sie die verfügbaren Amazon-Bedrock-Basismodelle mithilfe des Amazon-Bedrock-Clients auf.

```
   /**
    * Lists Amazon Bedrock foundation models that you can use.
    * You can filter the results with the request parameters.
    *
    * @param bedrockClient The async service client for accessing Amazon Bedrock.
    * @return A list of objects containing the foundation models' details
    */
   public static List<FoundationModelSummary>
listFoundationModels(BedrockAsyncClient bedrockClient) {
       try {
```

```
            CompletableFuture<ListFoundationModelsResponse> future =
bedrockClient.listFoundationModels(r -> {});

            List<FoundationModelSummary> models = future.get().modelSummaries();

            if (models.isEmpty()) {
                System.out.println("No available foundation models in " +
region.toString());
            } else {
                for (FoundationModelSummary model : models) {
                    System.out.println("Model ID: " + model.modelId());
                    System.out.println("Provider: " + model.providerName());
                    System.out.println("Name:     " + model.modelName());
                    System.out.println();
                }
            }

            return models;

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        } catch (ExecutionException e) {
            System.err.println(e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

- API-Details finden Sie ListFoundationModels in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien
- Amazon Nova
- Amazon Nova Canvas
- Amazon Titan Image Generator
- Amazon Titan Text Embeddings
- Anthropic Claude
- Cohere Command
- Meta Llama
- Mistral AI
- Stable Image Core

Szenarien

Erstellen einer Playground-Anwendung für die Interaktion mit Amazon-Bedrock-Basismodellen

Das folgende Codebeispiel zeigt, wie Sie Playgrounds erstellen, um mit Amazon-Bedrock-Basismodellen über verschiedene Modalitäten zu interagieren.

SDK für Java 2.x

Der Java Foundation Model (FM) Playground ist eine Spring-Boot-Beispielanwendung, die zeigt, wie Amazon Bedrock mit Java verwendet wird. Dieses Beispiel zeigt, wie Java-Entwickler Amazon Bedrock verwenden können, um generative AI-enabled Anwendungen zu erstellen. Sie können die Amazon-Bedrock-Basismodelle in den folgenden drei Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHub Projekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Generieren von Videos aus Text-Prompts mit Amazon Bedrock

Das folgende Codebeispiel zeigt, wie eine Spring Boot-App mithilfe von Amazon Bedrock und dem Modell Videos aus Textanfragen generiert. Nova-Reel

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie Videos aus Textanfragen mit Amazon Bedrock und. Nova-Reel

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
```

```
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

.s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
                        .build())
                .build();

       try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
           CompletableFuture<StartAsyncInvokeResponse>
startAsyncInvokeResponseCompletableFuture =
bedrockClient.startAsyncInvoke(request);

           //blocking operation to wait for the AWS API response
           StartAsyncInvokeResponse startAsyncInvokeResponse =
startAsyncInvokeResponseCompletableFuture.get();
           System.out.println("invocation ARN: " +
startAsyncInvokeResponse.invocationArn());

           GenerateVideoResponse response = new GenerateVideoResponse();
           response.setStatus("inProgress");
           response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

           return response;
       } catch (Exception e) {
           System.out.println(e);
           throw new RuntimeException(e);
       }

   }

   public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
       GenerateVideoResponse response = new GenerateVideoResponse();

       try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
           //creating async request to fetch status by invocation Arn
           GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

           CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);
```

```
            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
 asyncInvokeResponse.statusAsString());

            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
 BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetAsyncInvoke
  - StartAsyncInvoke

Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();
```

```
    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.
```

```
                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
```

```
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }
```

```
        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());
```

```
                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
```

```
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
```

```
            System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");

        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
```

```
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
```

```
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
 longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
 WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
```

```
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
```

```
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
    /**
```

```
    * Sends an asynchronous converse request to the AI model.
    *
    * @param modelId      the unique identifier of the AI model to be used for the
converse request
    * @param systemPrompt the system prompt to be included in the converse request
    * @param conversation a list of messages representing the conversation history
    * @param toolSpec     the specification of the tool to be used in the converse
request
    * @return the converse response received from the AI model
    */
   public ConverseResponse sendConverseRequestAsync(String modelId, String
systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
       List<Tool> toolList = new ArrayList<>();
       Tool tool = Tool.builder()
           .toolSpec(toolSpec)
           .build();

       toolList.add(tool);

       ToolConfiguration configuration = ToolConfiguration.builder()
           .tools(toolList)
           .build();

       SystemContentBlock block = SystemContentBlock.builder()
           .text(systemPrompt)
           .build();

       ConverseRequest request = ConverseRequest.builder()
           .modelId(modelId)
           .system(block)
           .messages(conversation)
           .toolConfig(configuration)
           .build();

       try {
           ConverseResponse response = getClient().converse(request).join();
           return response;

       } catch (ModelNotReadyException ex) {
           throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
       } catch (BedrockRuntimeException ex) {
           throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
```

```
        }
    }
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock und dem asynchronen Java-Client eine Textnachricht an Amazon Nova.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
```

```
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
```

```
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request);
            return asyncResponse.thenApply(
                    response -> response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;
```

```
/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
```

```
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).text();

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converse();
        System.out.println(response);
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;

/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
```

```
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
```

```
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of each
 chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Java 2.x API-Referenz.

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen

zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die primäre Ausführung des Szenarioablaufs. Dieses Szenario orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
```

```
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);
```

```
        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
```

```
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process
```

```
PrintModelResponse(modelResponse.output().message().content().get(0).text());
           if (!interactive) {
               defaultPrompt = "x";
           }
       }
   }

   /**
    * Handles the use of a tool by the model in a conversation.
    *
    * @param modelResponse the response from the model, which may include a tool
use request
    * @param conversation  the current conversation, which will be updated with the
tool use results
    * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
    */
   private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
       List<ContentBlock> toolResults = new ArrayList<>();

       // The model's response can consist of multiple content blocks
       for (ContentBlock contentBlock : modelResponse.message().content()) {
           if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
               // If the content block contains text, print it to the console
               PrintModelResponse(contentBlock.text());
           }

           if (contentBlock.toolUse() != null) {
               ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

               // Add the tool use ID and the tool's response to the list of
results
               List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
               ToolResultContentBlock block = ToolResultContentBlock.builder()
                       .json(toolResponse.getContent())
                       .build();
               contentBlockList.add(block);

               ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                       .toolUseId(toolResponse.getToolUseId())
                       .content(contentBlockList)
                       .build();
```

```
                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
```

```
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }
```

```
   private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
       System.out.println("Calling Bedrock...");

       // Send the conversation, system prompt, and tool configuration, and return
the response
       return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
   }

   public static String getUserInput(String prompt, Scanner scanner) {
       String userInput = defaultPrompt;
       if (interactive) {
           System.out.println("*".repeat(80));
           System.out.println(prompt + " (x to exit): \n\t");
           userInput = scanner.nextLine();
       }

       if (userInput == null || userInput.trim().isEmpty()) {
           return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
       }

       if (userInput.equalsIgnoreCase("x")) {
           return null;
       }

       return userInput;
   }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
```

```
        }
    }

    public static void printFooter() {
        System.out.println("""
                ==================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ==================================================
                """);
    }
}
```

Das in der Demo verwendete Wettertool. Diese Datei definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));
```

```
        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
```

```
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }
```

```
    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Die Converse-API-Aktion mit einer Toolkonfiguration.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
```

```
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Amazon Nova Canvas

InvokeModel

Das folgende Codebeispiel zeigt, wie Amazon Nova Canvas auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {
```

```
        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));
```

```
        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- API-Details finden Sie InvokeModel in der AWS SDK for Java 2.x API-Referenz.

Amazon Titan Image Generator

InvokeModel

Die folgenden Codebeispiele zeigen, wie Sie Amazon Titan Image in Amazon Bedrock aufrufen, um ein Bild zu generieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G2.
        var modelId = "amazon.titan-image-generator-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
```

```
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max. 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

```
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- API-Details finden Sie InvokeModel in der AWS SDK for Java 2.x API-Referenz.

Amazon Titan Text Embeddings

InvokeModel

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Beginnen Sie mit der Erstellung Ihrer ersten Einbettung.
- Erstellen Sie Einbettungen, indem Sie die Anzahl der Dimensionen und die Normalisierung konfigurieren (nur V2).

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Ihre erste Einbettung mit Titan Text Embeddings V2.

```
// Generate and print an embedding with Amazon Titan Text Embeddings.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
```

```
public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Text Embeddings V2.
        var modelId = "amazon.titan-embed-text-v2:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-embed-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{inputText}}\" }";

        // The text to convert into an embedding.
        var inputText = "Please recommend books with a theme similar to the movie
 'Inception'.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{inputText}}",
 inputText);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
embedding").queryFrom(responseBody).toString();
```

```
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

Rufen Sie Titan Text Embeddings V2 auf, indem Sie die Anzahl der Dimensionen und die Normalisierung konfigurieren.

```
    /**
     * Invoke Amazon Titan Text Embeddings V2 with additional inference parameters.
     *
     * @param inputText  - The text to convert to an embedding.
     * @param dimensions - The number of dimensions the output embeddings should
 have.
     *                   Values accepted by the model: 256, 512, 1024.
     * @param normalize  - A flag indicating whether or not to normalize the output
 embeddings.
     * @return The {@link JSONObject} representing the model's response.
     */
    public static JSONObject invokeModel(String inputText, int dimensions, boolean
 normalize) {

        // Create a Bedrock Runtime client in the AWS Region of your choice.
        var client = BedrockRuntimeClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Titan Embed Text v2.0.
        var modelId = "amazon.titan-embed-text-v2:0";
```

```
        // Create the request for the model.
        var nativeRequest = """
                {
                    "inputText": "%s",
                    "dimensions": %d,
                    "normalize": %b
                }
                """.formatted(inputText, dimensions, normalize);

        // Encode and send the request.
        var response = client.invokeModel(request -> {
            request.body(SdkBytes.fromUtf8String(nativeRequest));
            request.modelId(modelId);
        });

        // Decode the model's response.
        var modelResponse = new JSONObject(response.body().asUtf8String());

        // Extract and print the generated embedding and the input text token count.
        var embedding = modelResponse.getJSONArray("embedding");
        var inputTokenCount = modelResponse.getBigInteger("inputTextTokenCount");
        System.out.println("Embedding: " + embedding);
        System.out.println("\nInput token count: " + inputTokenCount);

        // Return the model's native response.
        return modelResponse;
    }
```

- API-Details finden Sie InvokeModel in der AWS SDK for Java 2.x API-Referenz.

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
```

```
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().getFirst().text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client.

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
```

```
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
 response.output().message().content().getFirst().text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
 text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// Use the Converse API to send a text message to Anthropic Claude
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
```

```
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
 handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Java 2.x API-Referenz.

Reasoning

Das folgende Codebeispiel zeigt, wie Sie die Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock verwenden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Argumentationsfunktion von Anthropic Claude 3.7 Sonnet mit dem asynchronen Bedrock-Runtime-Client.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and response
 text
                        for (ContentBlock block : content) {
```

```
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

Verwenden Sie die Argumentationsfunktion von Anthropic Claude 3.7 Sonnet mit dem synchronen Bedrock-Runtime-Client.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
```

```
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
```

```
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, text);

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Reasoning mit einer Streamingantwort

Das folgende Codebeispiel zeigt, wie Sie die Reasoning-Fähigkeit von Anthropic Claude 3.7 Sonnet in Amazon Bedrock verwenden.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Argumentationsfunktion Fähigkeit von Anthropic Claude 3.7 Sonnet, um Streaming-Textantworten zu generieren.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
```

```
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-sonnet-4-20250514-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new AtomicReference<>();

        // Set up the stream handler to processes chunks of the response as they
 arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
```

```
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of each
chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

Cohere Command

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client eine Textnachricht an Cohere Command.

```
// Use the Converse API to send a text message to Cohere Command
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
```

```
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
```

```
        converseAsync();
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Cohere Command
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Java 2.x API-Referenz.

Meta Llama

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie eine Textnachricht an Meta Llama, indem Sie die Converse-API von Bedrock mit dem asynchronen Java-Client verwenden.

```
// Use the Converse API to send a text message to Meta Llama
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
```

```
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
```

```
    }
}
```

- Details zur API finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Meta Llama
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Java 2.x API-Referenz.

Mistral AI

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }

    }
```

```
    public static void main(String[] args) {
        converse();
    }
}
```

Senden Sie mithilfe der Converse-API von Bedrock mit dem asynchronen Java-Client eine Textnachricht an Mistral.

```
// Use the Converse API to send a text message to Mistral
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
```

```
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().get(0).text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
```

```
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der AWS SDK for Java 2.x -API-Referenz.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
// Use the Converse API to send a text message to Mistral
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
 software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Create a handler to extract and print the response text in real-time.
        var responseStreamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            String responseText = chunk.delta().text();
                            System.out.print(responseText);
                        }).build()
                ).onError(err ->
                        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
                ).build();

        try {
            // Send the message with a basic inference configuration and attach the
handler.
            client.converseStream(request -> request.modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)
                    ), responseStreamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Java 2.x API-Referenz.

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
// Create an image with Stability AI Stable Image Core.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {
```

```
    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Stable Image Core.
        var modelId = "stability.stable-image-core-v1:1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
diffusion-stable-image-core-text-image-request-response.html
        var nativeRequestTemplate = """
                {
                    "prompt": "{{prompt}}",
                    "aspect_ratio": "1:1",
                    "seed": {{seed}},
                    "output_format": "png"
                }""";

        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random seed for the image generation (max. 4,294,967,294).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        String nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );
```

```
            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/images/0")
                    .queryFrom(responseBody)
                    .toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }

}
```

- API-Details finden Sie InvokeModel in der AWS SDK for Java 2.x API-Referenz.

CloudFront Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren CloudFront.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateDistribution`

Das folgende Codebeispiel zeigt die Verwendung`CreateDistribution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im folgenden Beispiel wird ein Bucket von Amazon Simple Storage Service(Amazon S3) als Inhaltsquelle verwendet.

Nach der Erstellung der Distribution erstellt der Code eine, CloudFrontWaiter um zu warten, bis die Distribution bereitgestellt ist, bevor die Distribution zurückgegeben wird.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)

 .id(originId)

 .s3OriginConfig(builder4 -> builder4

            .originAccessIdentity(

                            ""))

 .originAccessControlId(

            originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2
```

```
  .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

  .targetOriginId(originId)
                                                                    .minTTL(200L)
                                                                    .forwardedValues(b5
  -> b5

  .cookies(cp -> cp

            .forward(ItemSelection.NONE))

  .queryString(true))
                                                                    .trustedKeyGroups(b3
  -> b3

  .quantity(1)

  .items(keyGroupId)

  .enabled(true))
                                                                    .allowedMethods(b4 -
> b4

  .quantity(2)

  .items(Method.HEAD, Method.GET)

  .cachedMethods(b5 -> b5

            .quantity(2)

            .items(Method.HEAD,

                           Method.GET))))
                                                        .cacheBehaviors(b -> b
                                                                    .quantity(1)
                                                                    .items(b2 -> b2

  .pathPattern("/index.html")

  .viewerProtocolPolicy(

            ViewerProtocolPolicy.ALLOW_ALL)
```

```
.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

           .quantity(1)

           .items(keyGroupId)

           .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

           .cookies(cp -> cp

                           .forward(ItemSelection.NONE))

           .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

           .items(Method.HEAD,

                           Method.GET)

           .cachedMethods(b6 -> b6

                           .quantity(2)

                           .items(Method.HEAD,

                                           Method.GET)))))
                                               .enabled(true)
                                               .comment("Distribution built with
java")

.callerReference(Instant.now().toString())));

               final Distribution distribution = createDistResponse.distribution();
               logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
distribution.domainName(),
                               distribution.id());
```

```
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- API-Details finden Sie CreateDistribution in der AWS SDK for Java 2.x API-Referenz.

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
```

```
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName, filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }

    public static String createNewFunction(CloudFrontClient cloudFrontClient, String
 functionName, String filePath) {
        try {
```

```
            InputStream fileIs =
 CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
 cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateFunction in der AWS SDK for Java 2.x API-Referenz.

`CreateKeyGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Schlüsselgruppe erfordert mindestens einen öffentlichen Schlüssel, der zur Überprüfung signierter URLs oder Cookies verwendet wird.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b -> b.keyGroupConfig(c
 -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- API-Details finden Sie CreateKeyGroup in der AWS SDK for Java 2.x API-Referenz.

`CreatePublicKey`

Das folgende Codebeispiel zeigt die Verwendung`CreatePublicKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Codebeispiel liest einen öffentlichen Schlüssel ein und lädt ihn auf Amazon CloudFront hoch.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
    private static final Logger logger =
 LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
 publicKeyFileName) {
        try (InputStream is =
 CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- API-Details finden Sie CreatePublicKey in der AWS SDK for Java 2.x API-Referenz.

`DeleteDistribution`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDistribution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Codebeispiel aktualisiert eine Distribution auf deaktiviert, verwendet einen Waiter, der auf die Bereitstellung der Änderung wartet, und löscht anschließend die Distribution.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
 response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

 .cacheBehaviors(distConfig.cacheBehaviors())
```

```
.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .enabled(false)
                                                .origins(distConfig.origins())
                                                .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                .priceClass(distConfig.priceClass())
                                                .aliases(distConfig.aliases())
                                                .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

               logger.info("Distribution [{}] is DISABLED, waiting for deployment
before deleting ...",
                                distributionId);
               GetDistributionResponse distributionResponse;
               try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                       ResponseOrException<GetDistributionResponse>
responseOrException = cfWaiter
                                       .waitUntilDistributionDeployed(builder ->
builder.id(distributionId)).matched();
                       distributionResponse = responseOrException.response()
                                       .orElseThrow(() -> new
RuntimeException("Could not disable distribution"));
               }
```

```
                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful()) {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- API-Details finden Sie DeleteDistribution in der AWS SDK for Java 2.x API-Referenz.

`UpdateDistribution`

Das folgende Codebeispiel zeigt die Verwendung`UpdateDistribution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
 idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();

            GetDistributionResponse response =
 cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to comment
 and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
```

```
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

        } catch (CloudFrontException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie UpdateDistribution in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Multi-Tenant-Distribution und eines Distributionsmandanten

Das folgende Codebeispiel zeigt, wie Sie eine mandantenfähige Distribution und einen Distributionsmandanten mit verschiedenen Konfigurationen erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Beispiel zeigt, wie Sie eine Multi-Tenant-Distribution mit Parametern und einem Platzhalterzertifikat erstellen.

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
                                                                    S3Client
 s3Client,
                                                                    final String
 bucketName,
                                                                    final String
 certificateArn) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
```

```
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .viewerCertificate(certBuilder -> certBuilder
                                .acmCertificateArn(certificateArn)
                                .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
```

```
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
cfWaiter
                    .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
created"));
        }
        return distribution;
    }

    public static Distribution CreateMultiTenantDistributionNoCert(CloudFrontClient
cloudFrontClient,
                                                                   S3Client s3Client,
                                                                   final String
bucketName) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
originId.

        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))))
```

```
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }
}
```

Das folgende Beispiel zeigt, wie Sie einen dieser Vorlage zugeordneten Distributionsmandanten erstellen und dabei den oben definierten Parameter verwenden. Beachten Sie, dass Sie hier keine Zertifikatsinformationen hinzufügen müssen, da Ihre Domain bereits durch die übergeordnete Vorlage abgedeckt ist.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
```

```
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant createDistributionTenantNoCert(CloudFrontClient
 cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("no-cert-tenant")
        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
```

```
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

Wenn das Viewer-Zertifikat in der übergeordneten Vorlage weggelassen wurde, müssen Sie stattdessen Zertifikatsinformationen für die damit verknüpften Mandanten hinzufügen. Das folgende Beispiel zeigt, wie Sie dies mithilfe eines ACM-Zertifikats-ARN tun, der die erforderliche Domain für den Mandanten abdeckt.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {
```

```
    public static DistributionTenant
 createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId,
                                                                    String
 certificateArn) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .enabled(false)
                .name("tenant-with-cert")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .customizations(b3 -> b3
                        .certificate(b4 -> b4
                                .arn(certificateArn))) // NOTE: Cert must be in Us-
East-1 and cover the domain provided in this request

        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
```

```
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
               ));
        return distributionTenant;
    }

}
```

Das folgende Beispiel zeigt, wie Sie dies mit einer CloudFront-hosted verwalteten Zertifikatsanforderung tun können. Diese Methode eignet sich ideal, wenn noch kein Datenverkehr zu Ihrer Domain besteht. In diesem Fall erstellen wir eine, ConnectionGroup um eine zu generieren RoutingEndpoint. Dann verwenden wir das, RoutingEndpoint um DNS-Einträge zu erstellen, die den Domainbesitz bestätigen und auf die verwiesen wird CloudFront. CloudFront stellt dann automatisch ein Token bereit, um den Domainbesitz zu bestätigen und ein verwaltetes Zertifikat zu erstellen.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,
```

```
                                                                        Route53Client
route53Client,
                                                                        String
distributionId,
                                                                        String domain,
                                                                        String
hostedZoneId) throws InterruptedException {
       CreateConnectionGroupResponse createConnectionGroupResponse =
cloudFrontClient.createConnectionGroup(builder -> builder
               .ipv6Enabled(true)
               .name("cf-hosted-connection-group")
               .enabled(true));

       route53Client.changeResourceRecordSets(builder -> builder
               .hostedZoneId(hostedZoneId)
               .changeBatch(b1 -> b1
                       .comment("cf-hosted domain validation record")
                       .changes(b2 -> b2
                               .resourceRecordSet(b3 -> b3
                                       .name(domain)
                                       .type(RRType.CNAME)
                                       .ttl(300L)
                                       .resourceRecords(b4 -> b4

.value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                               .action("CREATE"))
               ));

       // Give the R53 record time to propagate, if it isn't being returned by
servers yet, the following call will fail
       Thread.sleep(60000);

       CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
               .distributionId(distributionId)
               .domains(b1 -> b1
                       .domain(domain))

.connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
               .enabled(false)
               .name("cf-hosted-tenant")
               .parameters(b2 -> b2
                       .name("tenantName")
                       .value("myTenant"))
```

```
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

Das folgende Beispiel zeigt, wie Sie dies mit einer selbst gehosteten verwalteten Zertifikatsanforderung durchführen können. Dies ist ideal, wenn bereits Datenverkehr zu Ihrer Domain besteht und Sie während einer Migration keine Ausfallzeiten in Kauf nehmen können. Am Ende dieses Beispiels wird der Mandant in einem Zustand erstellt, in dem die Domainvalidierung und die DNS-Einrichtung noch ausstehen. Folgen Sie den Schritten [hier](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certificates.html#complete -domain-ownership), um die Einrichtung abzuschließen, wenn Sie bereit sind, den Traffic zu migrieren.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                        String
 distributionId,
                                                                        String
 domain) {
```

```
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDistribution
  - CreateDistributionTenant

Löschen von Signaturressourcen

Das folgende Codebeispiel zeigt, wie Ressourcen gelöscht werden, die für den Zugriff auf eingeschränkte Inhalte in einem Bucket von Amazon Simple Storage Service(Amazon S3) verwendet werden.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;

public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient, final
 String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
```

```
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));

        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

Signieren von URLs und Cookies

Das folgende Codebeispiel zeigt, wie signierte URLs und Cookies erstellt werden, die den Zugriff auf eingeschränkte Ressourcen ermöglichen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die CannedSignerRequest Klasse, um URLs oder Cookies mit einer vordefinierten Richtlinie zu signieren.

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {

    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

Verwenden Sie die CustomSignerRequest Klasse, um URLs oder Cookies mit einer benutzerdefinierten Richtlinie zu signieren. Die Methoden `activeDate` und `ipRange` sind optional.

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
```

```
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

Das folgende Beispiel zeigt die Verwendung der CloudFrontUtilities Klasse zum Erstellen signierter Cookies und URLs. Sehen Sie sich dieses Codebeispiel unter an GitHub.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;
```

```
public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
```

```
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- API-Details finden Sie CloudFrontUtilities in der AWS SDK for Java 2.x API-Referenz.

CloudWatch Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hallo CloudWatch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudWatch beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
```

```
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is: "
 + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListMetrics in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listet CloudWatch Namespaces und Metriken auf.
- Rufen Sie Statistiken für eine Metrik und die geschätzte Fakturierung ab.
- Erstellen und aktualisieren Sie ein Dashboard.
- Erstellen Sie eine Metrik und fügen Sie ihr Daten hinzu.
- Erstellen und lösen Sie einen Alarm aus und zeigen Sie dann den Alarmverlauf an.
- Fügen Sie einen Anomaliedetektor hinzu.

- Ermitteln Sie ein Metrik-Image, dann bereinigen Sie die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die CloudWatch Funktionen demonstriert.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
```

```
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    static CloudWatchActions cwActions = new CloudWatchActions();

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchScenario.class);
    static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) throws Throwable {

        final String usage = """

            Usage:
              <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage> \s

            Where:
              myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)\s
```

```
             costDateWeek - The start date to use to get AWS/Billing statistics.
(For example, 2023-01-11T18:35:24.00Z.)\s
             dashboardName - The name of the dashboard to create.\s
             dashboardJson - The location of a JSON file to use to create a
dashboard. (See jsonWidgets.json in javav2/example_code/cloudwatch.)\s
             dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
             settings - The location of a JSON file from which various values are
read. (See settings.json in javav2/example_code/cloudwatch.)\s
             metricImage - The location of a BMP file that is used to create a
graph.\s
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }
       String myDate = args[0];
       String costDateWeek = args[1];
       String dashboardName = args[2];
       String dashboardJson = args[3];
       String dashboardAdd = args[4];
       String settings = args[5];
       String metricImage = args[6];

       logger.info(DASHES);
       logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
       logger.info("""
           Amazon CloudWatch is a comprehensive monitoring and observability
service
           provided by Amazon Web Services (AWS). It is designed to help you
monitor your
           AWS resources, applications, and services, as well as on-premises
resources,
           in real-time.

           CloudWatch collects and tracks various types of data, including
metrics,
           logs, and events, from your AWS and on-premises resources. It allows you
to set
           alarms and automatically respond to changes in your environment,
           enabling you to quickly identify and address issues before they impact
your
           applications or services.
```

```
            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch tasks
by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings, String
metricImage ) throws Throwable {
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
        1. List at least five available unique namespaces from Amazon CloudWatch.
        Select one from the list.
        """);
        String selectedNamespace;
        String selectedMetrics;
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
```

```
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building blocks
of CloudWatch
            and provide data points that represent a specific aspect of your system
or application over time.

            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
```

```
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension = cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
```

```
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified time
period.
            """);
        waitForInputToContinue(scanner);
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is the
ability to
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future = cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been reached: Error message:
{}, Error code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("12. Push data into the custom metric to trigger the alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.

            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential issues
or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
```

```
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
 cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
 code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für CloudWatch SDK-Methoden.

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
```

```
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
```

```
    * @param dashboardName the name of the dashboard to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
    * @throws RuntimeException if the dashboard deletion fails
    */
   public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
       DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
           .dashboardNames(dashboardName)
           .build();

       return getAsyncClient().deleteDashboards(dashboardsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
               } else {
                   logger.info("{} was successfully deleted.", dashboardName);
               }
           });
   }


   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
```

```
                      "AWS/Billing",
                      "EstimatedCharges",
                      "Currency",
                      "USD"
                     ]
                  ]
             }
           """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
           .metricWidget(myJSON)
           .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
           .thenCompose(response -> {
               SdkBytes sdkBytes = response.metricWidgetImage();
               byte[] bytes = sdkBytes.asByteArray();
               return CompletableFuture.runAsync(() -> {
                   try {
                       File outputFile = new File(fileName);
                       try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                           outputStream.write(bytes);
                       }
                   } catch (IOException e) {
                       throw new RuntimeException("Failed to write image to file",
e);
                   }
               });
           })
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error getting and saving metric
image", exception);
               } else {
                   logger.info("Image data saved successfully to {}", fileName);
               }
           });
   }


   /**
    * Describes the anomaly detectors based on the specified JSON file.
```

```
    *
    * @param fileName the name of the JSON file containing the custom metric
namespace and name
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                   List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                   for (AnomalyDetector detector : anomalyDetectorList) {
                       logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                       logger.info("State: {} ", detector.stateValue());
```

```
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
```

```
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
```

```
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
```

```
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
     *
     * @param metricRequest    the request to describe the alarms for the specified
metric
     * @param customMetricName the name of the custom metric to check for an alarm
     * @param retries          the number of retries to perform if no alarm is found
     * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }

        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
            if (response.hasMetricAlarms()) {
```

```
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }


    /**
     * Adds metric data for an alarm asynchronously.
     *
     * @param fileName the name of the JSON file containing the metric data
     * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
     */
    public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
```

```
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
```

```
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                // Read values from the JSON file.
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
```

```
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
```

```
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
```

```
    *
    * @param fileName the name of the JSON file containing the alarm configuration
    * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
    * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
    */
   public CompletableFuture<String> createAlarmAsync(String fileName) {
       com.fasterxml.jackson.databind.JsonNode rootNode;
       try {
           JsonParser parser = new JsonFactory().createParser(new File(fileName));
           rootNode = new ObjectMapper().readTree(parser);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the alarm configuration
file", e);
       }

       // Extract values from the JSON node.
       String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
       String customMetricName = rootNode.findValue("customMetricName").asText();
       String alarmName = rootNode.findValue("exampleAlarmName").asText();
       String emailTopic = rootNode.findValue("emailTopic").asText();
       String accountId = rootNode.findValue("accountId").asText();
       String region = rootNode.findValue("region").asText();

       // Create a List for alarm actions.
       List<String> alarmActions = new ArrayList<>();
       alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

       PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
           .alarmActions(alarmActions)
           .alarmDescription("Example metric alarm")
           .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
           .threshold(100.00)
           .metricName(customMetricName)
           .namespace(customMetricNamespace)
           .evaluationPeriods(1)
           .period(10)
           .statistic("Maximum")
           .datapointsToAlarm(1)
```

```
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with a
     * {@link PutDashboardResponse} when the dashboard is successfully updated
     */
    public CompletableFuture<PutDashboardResponse> addMetricToDashboardAsync(String
fileName, String dashboardName) {
        String dashboardBody;
        try {
            dashboardBody = readFileAsString(fileName);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the dashboard file", e);
        }

        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to update dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Error updating dashboard", ex);
```

```
                } else {
                    logger.info("{} was successfully updated.", dashboardName);
                    return response;
                }
            });
    }

    /**
     * Creates a new custom metric.
     *
     * @param dataPoint the data point to be added to the custom metric
     * @return a {@link CompletableFuture} representing the asynchronous operation
of adding the custom metric
     */
    public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
```

```
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
     * @param dashboardName the name of the dashboard to be created
     * @param fileName      the name of the file containing the dashboard body
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
     * @throws IOException if there is an error reading the dashboard body from the
file
     */
```

```
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
                       for (DashboardValidationMessage message : messages) {
                           logger.info("Message: {}", message.message());
                       }
                   }
                   return response; // Return the response for further use
               }
           });
   }


   /**
    * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
    *
    * @param costDateWeek the start date for the metric statistics, in the format
of an ISO-8601 date string (e.g., "2023-04-05")
    * @return a {@link CompletableFuture} that, when completed, contains the {@link
GetMetricStatisticsResponse} with the retrieved metric statistics
    * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
    */
```

```
   public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
       Instant start = Instant.parse(costDateWeek);
       Instant endDate = Instant.now();

       // Define dimension
       Dimension dimension = Dimension.builder()
           .name("Currency")
           .value("USD")
           .build();

       List<Dimension> dimensionList = new ArrayList<>();
       dimensionList.add(dimension);

       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .metricName("EstimatedCharges")
           .namespace("AWS/Billing")
           .dimensions(dimensionList)
           .statistics(Statistic.MAXIMUM)
           .startTime(start)
           .endTime(endDate)
           .period(86400) // One day period
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
                   }
               } else {
                   throw new RuntimeException("Failed to get metric statistics: " +
exception.getMessage(), exception);
               }
           });
   }
```

```
   /**
    * Retrieves and displays metric statistics for the specified parameters.
    *
    * @param nameSpace    the namespace for the metric
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

        String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
```

```
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
```

```
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or dimensions
are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();
```

```
        return getAsyncClient().listMetrics(request).handle((response, exception) ->
 {
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
 exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
 dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
 dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics

- GetMetricWidgetImage
- ListMetrics
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- PutMetricData

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a CloudWatch alarm.
    *
    * @param alarmName the name of the alarm to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
    * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
    * or a {@link RuntimeException} is thrown if the operation fails
    */
   public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
       DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
           .alarmNames(alarmName)
           .build();

       return getAsyncClient().deleteAlarms(request)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }
```

- API-Details finden Sie DeleteAlarms in der AWS SDK for Java 2.x API-Referenz.

`DeleteAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAnomalyDetector`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
```

```
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- API-Details finden Sie DeleteAnomalyDetector in der AWS SDK for Java 2.x API-Referenz.

`DeleteDashboards`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDashboards`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes the specified dashboard.
    *
    * @param dashboardName the name of the dashboard to be deleted
    * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
    * @throws RuntimeException if the dashboard deletion fails
    */
   public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
       DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
           .dashboardNames(dashboardName)
           .build();

       return getAsyncClient().deleteDashboards(dashboardsRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
               } else {
                   logger.info("{} was successfully deleted.", dashboardName);
               }
           });
   }
```

- API-Details finden Sie DeleteDashboards in der AWS SDK for Java 2.x API-Referenz.

`DescribeAlarmHistory`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmHistory`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the alarm history for a given alarm name and date range.
    *
    * @param fileName the path to the JSON file containing the alarm name
    * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
    * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
    */
   public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       // Use the alarm name to describe alarm history with a paginator.
       return readFileFuture.thenCompose(alarmName -> {
           try {
               Instant start = Instant.parse(date);
               Instant endDate = Instant.now();
               DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                   .startDate(start)
```

```
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }
```

- API-Details finden Sie DescribeAlarmHistory in der AWS SDK for Java 2.x API-Referenz.

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
```

```
            });
    }
```

- API-Details finden Sie DescribeAlarms in der AWS SDK for Java 2.x API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });
```

```
        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
     *
     * @param metricRequest    the request to describe the alarms for the specified
metric
     * @param customMetricName the name of the custom metric to check for an alarm
     * @param retries          the number of retries to perform if no alarm is found
     * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
     */
    private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
        if (retries == 0) {
```

```
            return CompletableFuture.completedFuture(null).thenRun(() ->
                logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
            );
        }

        return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
            if (response.hasMetricAlarms()) {
                logger.info("Alarm state found for {}", customMetricName);
                return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
            } else {
                return CompletableFuture.runAsync(() -> {
                    try {
                        Thread.sleep(20000);
                        logger.info(".");
                    } catch (InterruptedException e) {
                        throw new RuntimeException("Interrupted while waiting to
retry", e);
                    }
                }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
            }
        }));
    }
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK for Java 2.x API-Referenz.

`DescribeAnomalyDetectors`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAnomalyDetectors`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Describes the anomaly detectors based on the specified JSON file.
    *
    * @param fileName the name of the JSON file containing the custom metric
namespace and name
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                   .maxResults(10)
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

               return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                   List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
```

```
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
 detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
 e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
 exception);
            }
        });
    }
```

- API-Details finden Sie DescribeAnomalyDetectors in der AWS SDK for Java 2.x API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK for Java 2.x API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie EnableAlarmActions in der AWS SDK for Java 2.x API-Referenz.

`GetMetricData`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricData`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric information
    * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               // Parse the JSON string to extract relevant values.
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
```

```
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- API-Details finden Sie GetMetricData in der AWS SDK for Java 2.x API-Referenz.

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricStatistics`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

         String metricOption, String date, Dimension myDimension) {
```

```
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400) // 1 day period
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }
```

- API-Details finden Sie GetMetricStatistics in der AWS SDK for Java 2.x API-Referenz.

`GetMetricWidgetImage`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricWidgetImage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
           """;
```

```
        GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

        return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }
```

- API-Details finden Sie GetMetricWidgetImage in der AWS SDK for Java 2.x API-Referenz.

`ListDashboards`

Das folgende Codebeispiel zeigt die Verwendung`ListDashboards`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists the available dashboards.
    *
    * @return a {@link CompletableFuture} that completes when the operation is
finished.
    * The future will complete exceptionally if an error occurs while listing the
dashboards.
    */
   public CompletableFuture<Void> listDashboardsAsync() {
       ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
       ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
       return paginator.subscribe(response -> {
           response.dashboardEntries().forEach(entry -> {
               logger.info("Dashboard name is: {} ", entry.dashboardName());
               logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
           });
       }).exceptionally(ex -> {
           logger.info("Failed to list dashboards: {} ", ex.getMessage());
           throw new RuntimeException("Error occurred while listing dashboards",
ex);
       });
   }
```

- API-Details finden Sie ListDashboards in der AWS SDK for Java 2.x API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves a list of metric names for the specified namespace.
    *
    * @param namespace the namespace for which to retrieve the metric names
    * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
    * the metric names in the specified namespace
    * @throws RuntimeException if an error occurs while listing the metrics
    */
   public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
       ListMetricsRequest request = ListMetricsRequest.builder()
           .namespace(namespace)
           .build();

       ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
       Set<String> metSet = new HashSet<>();
       CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
           response.metrics().forEach(metric -> {
               String metricName = metric.metricName();
               metSet.add(metricName);
           });
       });

       return future
           .thenApply(ignored -> new ArrayList<>(metSet))
           .exceptionally(exception -> {
               throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
           });
   }
```

- API-Details finden Sie ListMetrics in der AWS SDK for Java 2.x API-Referenz.

`PutAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`PutAnomalyDetector`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Adds an anomaly detector for the given file.
    *
    * @param fileName the name of the file containing the anomaly detector
configuration
    * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
    */
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
```

```
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- API-Details finden Sie PutAnomalyDetector in der AWS SDK for Java 2.x API-Referenz.

`PutDashboard`

Das folgende Codebeispiel zeigt die Verwendung`PutDashboard`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
                       for (DashboardValidationMessage message : messages) {
                           logger.info("Message: {}", message.message());
                       }
                   }
                   return response; // Return the response for further use
               }
           });
   }
```

- API-Details finden Sie PutDashboard in der AWS SDK for Java 2.x API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName = rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
```

```
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK for Java 2.x API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
               Instant instant = Instant.now();
```

```
                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- API-Details finden Sie PutMetricData in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Überwachen der Leistung von Amazon DynamoDB

Im folgenden Codebeispiel wird gezeigt, wie die Verwendung von DynamoDB durch eine Anwendung zur Leistungsüberwachung konfiguriert wird.

SDK für Java 2.x

Im folgenden Codebeispiel wird gezeigt, wie eine Java-Anwendung zur Leistungsüberwachung von DynamoDB konfiguriert wird. Die Anwendung sendet Metrikdaten an eine CloudWatch Stelle, an die Sie die Leistung überwachen können.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch
- DynamoDB

CloudWatch Beispiele für Ereignisse, die das SDK für Java 2.x verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with CloudWatch Events Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutEventsRequestEntry;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutEvents {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <resourceArn>

                Where:
                   resourceArn - An Amazon Resource Name (ARN) related to the
 events.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String resourceArn = args[0];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWEvents(cwe, resourceArn);
        cwe.close();
    }

    public static void putCWEvents(CloudWatchEventsClient cwe, String resourceArn) {
        try {
            final String EVENT_DETAILS = "{ \"key1\": \"value1\", \"key2\":
 \"value2\" }";

            PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
                    .detail(EVENT_DETAILS)
                    .detailType("sampleSubmitted")
                    .resources(resourceArn)
                    .source("aws-sdk-java-cloudwatch-example")
                    .build();

            PutEventsRequest request = PutEventsRequest.builder()
                    .entries(requestEntry)
                    .build();

            cwe.putEvents(request);
            System.out.println("Successfully put CloudWatch event");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutEvents in der AWS SDK for Java 2.x API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.PutRuleResponse;
import software.amazon.awssdk.services.cloudwatchevents.model.RuleState;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> roleArn>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  roleArn - A role ARN value (for example,
 arn:aws:iam::xxxxxx047983:user/MyUser).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String roleArn = args[1];
```

```
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWRule(cwe, ruleName, roleArn);
        cwe.close();
    }

    public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String
 roleArn) {
        try {
            PutRuleRequest request = PutRuleRequest.builder()
                    .name(ruleName)
                    .roleArn(roleArn)
                    .scheduleExpression("rate(5 minutes)")
                    .state(RuleState.ENABLED)
                    .build();

            PutRuleResponse response = cwe.putRule(request);
            System.out.printf(
                    "Successfully created CloudWatch events rule %s with arn %s",
                    roleArn, response.ruleArn());

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutRule in der AWS SDK for Java 2.x API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient;
import software.amazon.awssdk.services.cloudwatchevents.model.PutTargetsRequest;
import software.amazon.awssdk.services.cloudwatchevents.model.Target;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutTargets {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <ruleName> <functionArn> <targetId>\s

                Where:
                  ruleName - A rule name (for example, myrule).
                  functionArn - An AWS Lambda function ARN (for example,
 arn:aws:lambda:us-west-2:xxxxxx047983:function:lamda1).
                  targetId - A target id value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        String functionArn = args[1];
        String targetId = args[2];
        CloudWatchEventsClient cwe = CloudWatchEventsClient.builder()
                .build();

        putCWTargets(cwe, ruleName, functionArn, targetId);
        cwe.close();
    }
```

```
    public static void putCWTargets(CloudWatchEventsClient cwe, String ruleName,
 String functionArn, String targetId) {
        try {
            Target target = Target.builder()
                    .arn(functionArn)
                    .id(targetId)
                    .build();

            PutTargetsRequest request = PutTargetsRequest.builder()
                    .targets(target)
                    .rule(ruleName)
                    .build();

            cwe.putTargets(request);
            System.out.printf(
                    "Successfully created CloudWatch events target for rule %s",
                    ruleName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutTargets in der AWS SDK for Java 2.x API-Referenz.

CloudWatch Protokolliert Beispiele mithilfe des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DeleteSubscriptionFilterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <filter> <logGroup>

                Where:
                  filter - The name of the subscription filter (for example,
 MyFilter).
```

```
                  logGroup - The name of the log group. (for example, testgroup).
                """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String logGroup = args[1];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .build();

        deleteSubFilter(logs, filter, logGroup);
        logs.close();
    }

    public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
 String logGroup) {
        try {
            DeleteSubscriptionFilterRequest request =
 DeleteSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .logGroupName(logGroup)
                    .build();

            logs.deleteSubscriptionFilter(request);
            System.out.printf("Successfully deleted CloudWatch logs subscription
 filter %s", filter);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteSubscriptionFilter in der AWS SDK for Java 2.x API-Referenz.

`DescribeLogStreams`

Das folgende Codebeispiel zeigt die Verwendung`DescribeLogStreams`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sucht nach Log-Streams innerhalb einer angegebenen Loggruppe, die einem bestimmten Präfix entsprechen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }
```

```
        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();
```

```
            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("--------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

Druckt Metadaten über den neuesten Log-Stream in einer angegebenen Loggruppe.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }
```

```
        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient logsClient,
String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
logGroupName);
                return;
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
stream.firstEventTimestamp());
```

```
            System.out.println("  Last Event Time: " + stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- API-Details finden Sie DescribeLogStreams in der AWS SDK for Java 2.x API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeSubscriptionFiltersResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.SubscriptionFilter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeSubscriptionFilters {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroup>

                Where:
                  logGroup - A log group name (for example, myloggroup).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String logGroup = args[0];
        CloudWatchLogsClient logs = CloudWatchLogsClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        describeFilters(logs, logGroup);
        logs.close();
    }

    public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
        try {
            boolean done = false;
            String newToken = null;

            while (!done) {
                DescribeSubscriptionFiltersResponse response;
                if (newToken == null) {
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .logGroupName(logGroup)
                            .limit(1).build();

                    response = logs.describeSubscriptionFilters(request);
                } else {
```

```
                    DescribeSubscriptionFiltersRequest request =
 DescribeSubscriptionFiltersRequest.builder()
                            .nextToken(newToken)
                            .logGroupName(logGroup)
                            .limit(1).build();
                    response = logs.describeSubscriptionFilters(request);
                }

                for (SubscriptionFilter filter : response.subscriptionFilters()) {
                    System.out.printf("Retrieved filter with name %s, " + "pattern
 %s " + "and destination arn %s",
                            filter.filterName(),
                            filter.filterPattern(),
                            filter.destinationArn());
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
}
```

- API-Details finden Sie DescribeSubscriptionFilters in der AWS SDK for Java 2.x API-Referenz.

`GetLogEvents`

Das folgende Codebeispiel zeigt die Verwendung`GetLogEvents`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 myloggroup).
```

```
                  logStreamName - The name of the log stream (for example,
 mystream).

                """;

       // if (args.length != 2) {
       //     System.out.print(usage);
       //     System.exit(1);
//        }

        String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
        String logStreamName = "weathertop-java-stream" ; //args[1];

        Region region = Region.US_EAST_1 ;
        CloudWatchLogsClient cloudWatchLogsClient = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
        cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();

            DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
                return;
            }
```

```
            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7, ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest = GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetLogEvents in der AWS SDK for Java 2.x API-Referenz.

`PutSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`PutSubscriptionFilter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.PutSubscriptionFilterRequest;

/**
 * Before running this code example, you need to grant permission to CloudWatch
 * Logs the right to execute your Lambda function.
 * To perform this task, you can use this CLI command:
 *
 * aws lambda add-permission --function-name "lamda1" --statement-id "lamda1"
 * --principal "logs.us-west-2.amazonaws.com" --action "lambda:InvokeFunction"
 * --source-arn "arn:aws:logs:us-west-2:111111111111:log-group:testgroup:*"
 * --source-account "111111111111"
 *
 * Make sure you replace the function name with your function name and replace
 * '111111111111' with your account details.
 * For more information, see "Subscription Filters with AWS Lambda" in the
 * Amazon CloudWatch Logs Guide.
 *
 *
 * Also, before running this Java V2 code example,set up your development
 * environment,including your credentials.
 *
 * For more information,see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */

public class PutSubscriptionFilter {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                  <filter> <pattern> <logGroup> <functionArn>\s

                Where:
                  filter - A filter name (for example, myfilter).
                  pattern - A filter pattern (for example, ERROR).
                  logGroup - A log group name (testgroup).
                  functionArn - An AWS Lambda function ARN (for example,
arn:aws:lambda:us-west-2:111111111111:function:lambda1) .
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String filter = args[0];
        String pattern = args[1];
        String logGroup = args[2];
        String functionArn = args[3];
        Region region = Region.US_WEST_2;
        CloudWatchLogsClient cwl = CloudWatchLogsClient.builder()
                .region(region)
                .build();

        putSubFilters(cwl, filter, pattern, logGroup, functionArn);
        cwl.close();
    }

    public static void putSubFilters(CloudWatchLogsClient cwl,
            String filter,
            String pattern,
            String logGroup,
            String functionArn) {

        try {
            PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
                    .filterName(filter)
                    .filterPattern(pattern)
                    .logGroupName(logGroup)
                    .destinationArn(functionArn)
                    .build();
```

```
            cwl.putSubscriptionFilter(request);
            System.out.printf(
                    "Successfully created CloudWatch logs subscription filter %s",
                    filter);

        } catch (CloudWatchLogsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutSubscriptionFilter in der AWS SDK for Java 2.x API-Referenz.

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für Java 2.x

Binden Sie die erforderlichen Dateien ein.

```
import io.reactivex.FlowableSubscriber;
import io.reactivex.annotations.NonNull;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsAsyncClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionStart;
import software.amazon.awssdk.services.cloudwatchlogs.model.LiveTailSessionUpdate;
import software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseHandler;
import software.amazon.awssdk.services.cloudwatchlogs.model.CloudWatchLogsException;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.StartLiveTailResponseStream;

import java.util.Date;
import java.util.List;
import java.util.concurrent.atomic.AtomicReference;
```

Bearbeiten Sie die Ereignisse aus der Live-Tail-Sitzung.

```
   private static StartLiveTailResponseHandler
getStartLiveTailResponseStreamHandler(
           AtomicReference<Subscription> subscriptionAtomicReference) {
       return StartLiveTailResponseHandler.builder()
           .onResponse(r -> System.out.println("Received initial response"))
           .onError(throwable -> {
               CloudWatchLogsException e = (CloudWatchLogsException)
throwable.getCause();
               System.err.println(e.awsErrorDetails().errorMessage());
               System.exit(1);
           })
           .subscriber(() -> new FlowableSubscriber<>() {
               @Override
               public void onSubscribe(@NonNull Subscription s) {
                   subscriptionAtomicReference.set(s);
                   s.request(Long.MAX_VALUE);
               }

               @Override
               public void onNext(StartLiveTailResponseStream event) {
                   if (event instanceof LiveTailSessionStart) {
                       LiveTailSessionStart sessionStart = (LiveTailSessionStart)
event;
                       System.out.println(sessionStart);
                   } else if (event instanceof LiveTailSessionUpdate) {
                       LiveTailSessionUpdate sessionUpdate =
(LiveTailSessionUpdate) event;
                       List<LiveTailSessionLogEvent> logEvents =
sessionUpdate.sessionResults();
                       logEvents.forEach(e -> {
                           long timestamp = e.timestamp();
                           Date date = new Date(timestamp);
                           System.out.println("[" + date + "] " + e.message());
                       });
                   } else {
                       throw CloudWatchLogsException.builder().message("Unknown
event type").build();
                   }
               }
```

```
                @Override
                public void onError(Throwable throwable) {
                    System.out.println(throwable.getMessage());
                    System.exit(1);
                }

                @Override
                public void onComplete() {
                    System.out.println("Completed Streaming Session");
                }
            })
            .build();
    }
```

Starten Sie die Live-Tail-Sitzung.

```
        CloudWatchLogsAsyncClient cloudWatchLogsAsyncClient =
                CloudWatchLogsAsyncClient.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();

        StartLiveTailRequest request =
                StartLiveTailRequest.builder()
                    .logGroupIdentifiers(logGroupIdentifiers)
                    .logStreamNames(logStreamNames)
                    .logEventFilterPattern(logEventFilterPattern)
                    .build();

        /* Create a reference to store the subscription */
        final AtomicReference<Subscription> subscriptionAtomicReference = new
 AtomicReference<>(null);

        cloudWatchLogsAsyncClient.startLiveTail(request,
 getStartLiveTailResponseStreamHandler(subscriptionAtomicReference));
```

Beenden Sie die Live-Tail-Sitzung nach Ablauf einer gewissen Zeit.

```
        /* Set a timeout for the session and cancel the subscription. This will:
         * 1). Close the stream
         * 2). Stop the Live Tail session
```

```
         */
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        if (subscriptionAtomicReference.get() != null) {
            subscriptionAtomicReference.get().cancel();
            System.out.println("Subscription to stream closed");
        }
```

- API-Details finden Sie StartLiveTail in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda

- Amazon SNS

Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung`CreateIdentityPool`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();
```

```
            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateIdentityPool in der AWS SDK for Java 2.x API-Referenz.

`DeleteIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentityPool`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- API-Details finden Sie DeleteIdentityPool in der AWS SDK for Java 2.x API-Referenz.

`GetCredentialsForIdentity`

Das folgende Codebeispiel zeigt die Verwendung`GetCredentialsForIdentity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s
```

```
            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetCredentialsForIdentity in der AWS SDK for Java 2.x API-Referenz.

`ListIdentityPools`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentityPools`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
```

```
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListIdentityPools in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }
```

```
    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListUserPools in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
```

```
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie AdminGetUser in der AWS SDK for Java 2.x API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
```

```
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK for Java 2.x API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK for Java 2.x API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
```

```
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK for Java 2.x API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK for Java 2.x API-Referenz.

`CreateUserPool`

Das folgende Codebeispiel zeigt die Verwendung`CreateUserPool`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateUserPool in der AWS SDK for Java 2.x API-Referenz.

`CreateUserPoolClient`

Das folgende Codebeispiel zeigt die Verwendung`CreateUserPoolClient`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
```

```
                    userPoolId - The ID for the user pool.
                """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
 String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " + response.userPoolClient().clientName()
 + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateUserPoolClient in der AWS SDK for Java 2.x API-Referenz.

`ListUserPools`

Das folgende Codebeispiel zeigt die Verwendung`ListUserPools`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient cognitoClient)
 {
```

```
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response = cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID " +
 userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListUserPools in der AWS SDK for Java 2.x API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();
```

```
            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
 user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username() + "
 Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListUsers in der AWS SDK for Java 2.x API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK for Java 2.x API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
           String password, String email) {
       AttributeType userAttrs = AttributeType.builder()
               .name("email")
               .value(email)
               .build();

       List<AttributeType> userAttrsList = new ArrayList<>();
       userAttrsList.add(userAttrs);
       try {
           SignUpRequest signUpRequest = SignUpRequest.builder()
                   .userAttributes(userAttrsList)
                   .username(userName)
                   .clientId(clientId)
                   .password(password)
                   .build();

           identityProviderClient.signUp(signUpRequest);
           System.out.println("User has been signed up ");

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie SignUp in der AWS SDK for Java 2.x API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.

- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
```

```
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the AWS
 CDK script.
                    poolId - The pool Id that you can get from the AWS CDK script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();
```

```
        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);
```

```
            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
```

```
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
```

```
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge

- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Beispiele für Amazon Comprehend unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDocumentClassifier`

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierRequest;
import
 software.amazon.awssdk.services.comprehend.model.CreateDocumentClassifierResponse;
import
 software.amazon.awssdk.services.comprehend.model.DocumentClassifierInputDataConfig;

/**
 * Before running this code example, you can setup the necessary resources, such
 * as the CSV file and IAM Roles, by following this document:
 * https://aws.amazon.com/blogs/machine-learning/building-a-custom-classifier-using-
amazon-comprehend/
 *
 * Also, set up your development environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DocumentClassifierDemo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <dataAccessRoleArn> <s3Uri> <documentClassifierName>

                Where:
                  dataAccessRoleArn - The ARN value of the role used for this
 operation.
                  s3Uri - The Amazon S3 bucket that contains the CSV file.
                  documentClassifierName - The name of the document classifier.
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dataAccessRoleArn = args[0];
        String s3Uri = args[1];
        String documentClassifierName = args[2];

        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        createDocumentClassifier(comClient, dataAccessRoleArn, s3Uri,
documentClassifierName);
        comClient.close();
    }

    public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri,
            String documentClassifierName) {
        try {
            DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
                    .s3Uri(s3Uri)
                    .build();

            CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
                    .documentClassifierName(documentClassifierName)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .languageCode("en")
                    .inputDataConfig(config)
                    .build();

            CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient
                    .createDocumentClassifier(createDocumentClassifierRequest);
            String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
            System.out.println("Document Classifier ARN: " + documentClassifierArn);
```

```
        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateDocumentClassifier in der AWS SDK for Java 2.x API-Referenz.

`DetectDominantLanguage`

Das folgende Codebeispiel zeigt die Verwendung`DetectDominantLanguage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageRequest;
import
 software.amazon.awssdk.services.comprehend.model.DetectDominantLanguageResponse;
import software.amazon.awssdk.services.comprehend.model.DominantLanguage;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLanguage {
```

```
    public static void main(String[] args) {
        // Specify French text - "It is raining today in Seattle".
        String text = "Il pleut aujourd'hui à Seattle";
        Region region = Region.US_EAST_1;

        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectDominantLanguage");
        detectTheDominantLanguage(comClient, text);
        comClient.close();
    }

    public static void detectTheDominantLanguage(ComprehendClient comClient, String
 text) {
        try {
            DetectDominantLanguageRequest request =
 DetectDominantLanguageRequest.builder()
                    .text(text)
                    .build();

            DetectDominantLanguageResponse resp =
 comClient.detectDominantLanguage(request);
            List<DominantLanguage> allLanList = resp.languages();
            for (DominantLanguage lang : allLanList) {
                System.out.println("Language is " + lang.languageCode());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectDominantLanguage in der AWS SDK for Java 2.x API-Referenz.

`DetectEntities`

Das folgende Codebeispiel zeigt die Verwendung`DetectEntities`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectEntitiesResponse;
import software.amazon.awssdk.services.comprehend.model.Entity;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectEntities {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectEntities");
        detectAllEntities(comClient, text);
        comClient.close();
    }

    public static void detectAllEntities(ComprehendClient comClient, String text) {
        try {
```

```
            DetectEntitiesRequest detectEntitiesRequest =
 DetectEntitiesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectEntitiesResponse detectEntitiesResult =
 comClient.detectEntities(detectEntitiesRequest);
            List<Entity> entList = detectEntitiesResult.entities();
            for (Entity entity : entList) {
                System.out.println("Entity text is " + entity.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectEntities in der AWS SDK for Java 2.x API-Referenz.

`DetectKeyPhrases`

Das folgende Codebeispiel zeigt die Verwendung`DetectKeyPhrases`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesRequest;
import software.amazon.awssdk.services.comprehend.model.DetectKeyPhrasesResponse;
import software.amazon.awssdk.services.comprehend.model.KeyPhrase;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectKeyPhrases {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectKeyPhrases");
        detectAllKeyPhrases(comClient, text);
        comClient.close();
    }

    public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
 {
        try {
            DetectKeyPhrasesRequest detectKeyPhrasesRequest =
 DetectKeyPhrasesRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectKeyPhrasesResponse detectKeyPhrasesResult =
 comClient.detectKeyPhrases(detectKeyPhrasesRequest);
            List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
            for (KeyPhrase keyPhrase : phraseList) {
                System.out.println("Key phrase text is " + keyPhrase.text());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API-Details finden Sie DetectKeyPhrases in der AWS SDK for Java 2.x API-Referenz.

`DetectSentiment`

Das folgende Codebeispiel zeigt die Verwendung`DetectSentiment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSentimentResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSentiment {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
```

```
                .region(region)
                .build();

        System.out.println("Calling DetectSentiment");
        detectSentiments(comClient, text);
        comClient.close();
    }

    public static void detectSentiments(ComprehendClient comClient, String text) {
        try {
            DetectSentimentRequest detectSentimentRequest =
 DetectSentimentRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSentimentResponse detectSentimentResult =
 comClient.detectSentiment(detectSentimentRequest);
            System.out.println("The Neutral value is " +
 detectSentimentResult.sentimentScore().neutral());

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectSentiment in der AWS SDK for Java 2.x API-Referenz.

`DetectSyntax`

Das folgende Codebeispiel zeigt die Verwendung`DetectSyntax`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.comprehend.ComprehendClient;
import software.amazon.awssdk.services.comprehend.model.ComprehendException;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxRequest;
import software.amazon.awssdk.services.comprehend.model.DetectSyntaxResponse;
import software.amazon.awssdk.services.comprehend.model.SyntaxToken;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectSyntax {
    public static void main(String[] args) {
        String text = "Amazon.com, Inc. is located in Seattle, WA and was founded
 July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to
 blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable
 Seattle - based companies are Starbucks and Boeing.";
        Region region = Region.US_EAST_1;
        ComprehendClient comClient = ComprehendClient.builder()
                .region(region)
                .build();

        System.out.println("Calling DetectSyntax");
        detectAllSyntax(comClient, text);
        comClient.close();
    }

    public static void detectAllSyntax(ComprehendClient comClient, String text) {
        try {
            DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
                    .text(text)
                    .languageCode("en")
                    .build();

            DetectSyntaxResponse detectSyntaxResult =
 comClient.detectSyntax(detectSyntaxRequest);
            List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
            for (SyntaxToken token : syntaxTokens) {
```

```
                System.out.println("Language is " + token.text());
                System.out.println("Part of speech is " +
 token.partOfSpeech().tagAsString());
            }

        } catch (ComprehendException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectSyntax in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Java 2.x

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract

- Amazon Translate

AWS Control Tower Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS Control Tower.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Control Tower

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Control Tower beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloControlTower {
```

```
    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
     * and list all available baselines.
     * This example uses the default settings specified in your shared credentials
     * and config files.
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
baselines:\n");

        ListBaselinesIterable paginator = controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));

            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode())) {
                System.out.println("Access denied. Please ensure you have the
necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
```

```
            }
        }
    }
}
```

- API-Details finden Sie ListBaselines in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listen Sie Landing Zones auf.
- Listen Sie Baselines auf, aktivieren Sie sie, rufen Sie sie ab, setzen Sie sie zurück und deaktivieren Sie sie.
- Listen Sie Steuerelemente auf, aktivieren Sie sie, rufen Sie sie ab und deaktivieren Sie sie.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Control Tower Funktionen demonstriert.

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;
```

```
    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);

        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ----------------------------
    // Utilities
    // ----------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ----------------------------
            // Step 1: Landing Zones
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in the
 demo.
        For instructions on how to set up a landing zone,
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);
```

```
            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-dependent
steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1, landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the list
(" +
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing Landing
Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }

            // ----------------------------
            // Setup Organization + Sandbox OU
            // ----------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);

                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();
```

```
                System.out.println("Organization ID: " + result.orgId());
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // -----------------------------
            // Step 2: Baselines
            // -----------------------------
            System.out.println(DASHES);
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");



            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();

            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
```

```
            }

            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if (eb.baselineIdentifier().equals(controlTowerBaseline.arn()))
 {
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
                if (askYesNo("Do you want to enable the Control Tower Baseline? (y/
n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");

                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
 the OU or account ARN
                                    baselineId,             // baselineIdentifier #
 the Control Tower baseline ARN
```

```
                                    "5.0"                   // baselineVersion #
version string
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
enabledBaselineId);
                    if (enabledBaselineId == null) {
                       enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower Baseline?
(y/n): ")) {
                        String operationId =

actions.resetEnabledBaselineAsync(enabledBaselineId).join();
                        System.out.println("Reset baseline operation ID: " +
operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower Baseline?
(y/n): ")) {
                        String operationId =

actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
operationId);

                        System.out.println("Now we will re-enable the baseline and
wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();
```

```
                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }

                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
```

```
                }

                waitForInputToContinue(scanner);
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
                        askYesNo("Do you want to disable the control? (y/n): ")) {

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
            System.out.println("Unexpected error running scenario: " +
e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
```

```
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
                System.out.println("Continuing...");
                break;
            }
        }
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return orgAsyncClient;
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
```

```
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }

    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();

            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }
```

```
    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }

    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();

        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&

"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode())) {
                        System.out.println("No organization found. Creating one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}" +
createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
```

```
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
                                    new RuntimeException("No root found in
organization")
                            );
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);

                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU: "
+ ou.id());
                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
                                        System.out.println("Sandbox OU not found.");
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
       }).exceptionally(ex -> {
           Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
           System.out.println("Failed to setup organization: {}" +
cause.getMessage());
           throw new CompletionException(cause);
       });
```

```
    }

    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
```

```
                                            "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }


    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
```

```
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }

    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
```

```
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
```

```
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }


    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
     */
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
```

```
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();
```

```
                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
```

```
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }


    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
```

```
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();
```

```
                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }



    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();
```

```
                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }


    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
```

```
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
 targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
```

```
                                if (e.getMessage() != null &&
e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
```

```
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
```

```
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default and
cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }
```

```
                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);
```

```
                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
```

```
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }



    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
```

```
                                    "Error getting control operation status:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
 %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
 operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }


    /**
     * Lists all controls in the Control Tower control catalog.
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
 getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.controls() != null && !
response.controls().isEmpty()) {
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
```

```
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
sdkEx.getMessage().contains("AccessDeniedException")) {
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
%s".formatted(sdkEx.getMessage()),
                                    sdkEx
                            );
                        }
                    }

                    throw new CompletionException("Failed to list controls", cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");
```

```
        ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);

                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
```

```
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
 errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
 cause.getMessage());
                    return null;
                });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl

- EnableBaseline
- EnableControl
- GetControlOperation
- GetLandingZoneOperation
- ListBaselines
- ListEnabledBaselines
- ListEnabledControls
- ListLandingZones
- ResetEnabledBaseline

Aktionen

`DisableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`DisableBaseline`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Disables a baseline for a specified target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting disable of enabled baseline…");
```

```
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
```

```
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- API-Details finden Sie DisableBaseline in der AWS SDK for Java 2.x API-Referenz.

`DisableControl`

Das folgende Codebeispiel zeigt die Verwendung`DisableControl`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
```

```
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
```

```
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }
```

- API-Details finden Sie DisableControl in der AWS SDK for Java 2.x API-Referenz.

`EnableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`EnableBaseline`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
     */
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
```

```
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
```

```
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }
```

```
                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }
```

- API-Details finden Sie EnableBaseline in der AWS SDK for Java 2.x API-Referenz.

`EnableControl`

Das folgende Codebeispiel zeigt die Verwendung`EnableControl`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Enables a control for a specified target.
     *
```

```
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
```

```
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ValidationException".equals(errorCode)
                                && message.contains("enforced by default")) {
                            System.out.println("Control is enforced by default and
cannot be enabled manually.");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
```

```
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- API-Details finden Sie EnableControl in der AWS SDK for Java 2.x API-Referenz.

`GetBaselineOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetBaselineOperation`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
```

```
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
                                                .formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
```

```
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- API-Details finden Sie GetBaselineOperation in der AWS SDK for Java 2.x API-Referenz.

`GetControlOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetControlOperation`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
```

```
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- API-Details finden Sie GetControlOperation in der AWS SDK for Java 2.x API-Referenz.

`ListBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListBaselines`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- API-Details finden Sie ListBaselines in der AWS SDK for Java 2.x API-Referenz.

`ListEnabledBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledBaselines`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists all enabled baselines using pagination to retrieve complete results.
    *
    * @return a list of all enabled baselines
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
       System.out.println("Starting list enabled baselines paginator…");

       ListEnabledBaselinesRequest request =
               ListEnabledBaselinesRequest.builder().build();

       ListEnabledBaselinesPublisher paginator =
               getAsyncClient().listEnabledBaselinesPaginator(request);

       List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
       return paginator.subscribe(response -> {
                   if (response.enabledBaselines() != null
                           && !response.enabledBaselines().isEmpty()) {

                       response.enabledBaselines().forEach(baseline -> {
                           enabledBaselines.add(baseline);
                       });
                   } else {
                       System.out.println("Page contained no enabled baselines.");
                   }
               })
               .thenRun(() ->
                       System.out.println(
                               "Successfully listed enabled baselines. Total: " +
                                       enabledBaselines.size()
                       )
```

```
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

- API-Details finden Sie ListEnabledBaselines in der AWS SDK for Java 2.x API-Referenz.

`ListEnabledControls`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledControls`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
 targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
```

```
                        "Successfully retrieved "+enabledControls.size() +" enabled
controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }
```

- API-Details finden Sie ListEnabledControls in der AWS SDK for Java 2.x API-Referenz.

`ListLandingZones`

Das folgende Codebeispiel zeigt die Verwendung`ListLandingZones`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
```

```
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
+ e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
    }
```

- API-Details finden Sie ListLandingZones in der AWS SDK for Java 2.x API-Referenz.

`ResetEnabledBaseline`

Das folgende Codebeispiel zeigt die Verwendung`ResetEnabledBaseline`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Resets an enabled baseline for a specific target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   Runnable poller = new Runnable() {
                       @Override
                       public void run() {
                           getBaselineOperationAsync(operationId)
```

```
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
errorCode + ", Message=" + errorMessage);
                        return null;
```

```
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
cause.getMessage());
                    return null;
                });
    }
```

- API-Details finden Sie ResetEnabledBaseline in der AWS SDK for Java 2.x API-Referenz.

Firehose-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Firehose Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung`PutRecord`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
    *
    * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();

           PutRecordRequest putRecordRequest = PutRecordRequest.builder()
               .deliveryStreamName(deliveryStreamName)
               .record(firehoseRecord)
```

```
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- API-Details finden Sie PutRecord in der AWS SDK for Java 2.x API-Referenz.

`PutRecordBatch`

Das folgende Codebeispiel zeigt die Verwendung`PutRecordBatch`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- API-Details finden Sie PutRecordBatch in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Einfügen von Datensätzen in Firehose

Das folgende Codebeispiel zeigt, wie Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze verwendet werden kann.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
```

```
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
     *
     * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();
```

```
            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
                    } catch (Exception e) {
```

```
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime, Instant
endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)
```

```
 .dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).build(
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8)) {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutRecord

- PutRecordBatch

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

DynamoDB-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und von ihnen verwaltet werden. AWS Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
```

```
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
```

```
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }
```

Rufen Sie ein Element aus einer Tabelle ab.

```
    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
```

```
            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Vollständiges Beispiel.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */
```

```
public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named year
 and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
```

```
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
```

```
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                    .partitionValue(2013)
                    .build());
```

```
            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " + rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
```

```
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\"]")
                .build())
            .action(AttributeAction.PUT)
```

```
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
```

```
            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
```

```
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt, wie Batch-Elemente mithilfe des Service-Clients abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchReadItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItems(dynamoDbClient, tableName);
    }

    public static void getBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
```

```
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
            .build();

        // Make the batchGetItem request.
        BatchGetItemResponse batchGetItemResponse =
 dynamoDbClient.batchGetItem(batchGetItemRequest);

        // Extract and print the retrieved items.
        Map<String, List<Map<String, AttributeValue>>> responses =
 batchGetItemResponse.responses();
        if (responses.containsKey(tableName)) {
            List<Map<String, AttributeValue>> musicItems = responses.get(tableName);
            for (Map<String, AttributeValue> item : musicItems) {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            }
        } else {
            System.out.println("No items retrieved.");
        }
    }
}
```

Zeigt, wie Batch-Elemente mithilfe des Service-Clients und eines Paginators abgerufen werden.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchGetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.KeysAndAttributes;
```

```
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BatchGetItemsPaginator {

    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        getBatchItemsPaginator(dynamoDbClient, tableName) ;
    }

    public static void getBatchItemsPaginator(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Define the primary key values for the items you want to retrieve.
        Map<String, AttributeValue> key1 = new HashMap<>();
        key1.put("Artist", AttributeValue.builder().s("Artist1").build());

        Map<String, AttributeValue> key2 = new HashMap<>();
        key2.put("Artist", AttributeValue.builder().s("Artist2").build());

        // Construct the batchGetItem request.
        Map<String, KeysAndAttributes> requestItems = new HashMap<>();
        requestItems.put(tableName, KeysAndAttributes.builder()
            .keys(List.of(key1, key2))
            .projectionExpression("Artist, SongTitle")
            .build());

        BatchGetItemRequest batchGetItemRequest = BatchGetItemRequest.builder()
            .requestItems(requestItems)
```

```
            .build();

        // Use batchGetItemPaginator for paginated requests.
        dynamoDbClient.batchGetItemPaginator(batchGetItemRequest).stream()
            .flatMap(response -> response.responses().getOrDefault(tableName,
 Collections.emptyList()).stream())
            .forEach(item -> {
                System.out.println("Artist: " + item.get("Artist").s() +
                    ", SongTitle: " + item.get("SongTitle").s());
            });
    }
}
```

- API-Details finden Sie BatchGetItem in der AWS SDK for Java 2.x API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügt mithilfe des Service-Clients viele Elemente in eine Tabelle ein.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.BatchWriteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutRequest;
import software.amazon.awssdk.services.dynamodb.model.WriteRequest;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class BatchWriteItems {
    public static void main(String[] args){
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music).\s
                """;

        String tableName = "Music";
        Region region = Region.US_EAST_1;
        DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
            .region(region)
            .build();

        addBatchItems(dynamoDbClient, tableName);
    }

    public static void addBatchItems(DynamoDbClient dynamoDbClient, String
 tableName) {
        // Specify the updates you want to perform.
        List<WriteRequest> writeRequests = new ArrayList<>();

        // Set item 1.
        Map<String, AttributeValue> item1Attributes = new HashMap<>();
        item1Attributes.put("Artist",
 AttributeValue.builder().s("Artist1").build());
        item1Attributes.put("Rating", AttributeValue.builder().s("5").build());
        item1Attributes.put("Comments", AttributeValue.builder().s("Great
 song!").build());
        item1Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle1").build());
```

```
 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item1Attribut

        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
 AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
 melody.").build());
        item2Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle2").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
 BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
 dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " + batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Fügt mithilfe des erweiterten Clients viele Elemente in eine Tabelle ein.

```
import com.example.dynamodb.Customer;
import com.example.dynamodb.Music;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
```

```
import software.amazon.awssdk.enhanced.dynamodb.Key;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.enhanced.dynamodb.model.BatchWriteItemEnhancedRequest;
import software.amazon.awssdk.enhanced.dynamodb.model.WriteBatch;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/*
 * Before running this code example, create an Amazon DynamoDB table named Customer
 with these columns:
 *   - id - the id of the record that is the key
 *   - custName - the customer name
 *   - email - the email value
 *   - registrationDate - an instant value when the item was added to the table
 *
 * Also, ensure that you have set up your development environment, including your
 credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnhancedBatchWriteItems {
        public static void main(String[] args) {
                Region region = Region.US_EAST_1;
                DynamoDbClient ddb = DynamoDbClient.builder()
                                .region(region)
                                .build();
                DynamoDbEnhancedClient enhancedClient =
 DynamoDbEnhancedClient.builder()
                                .dynamoDbClient(ddb)
                                .build();
                putBatchRecords(enhancedClient);
                ddb.close();
        }

        public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {
                try {
```

```
                        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer",
                                        TableSchema.fromBean(Customer.class));
                        DynamoDbTable<Music> musicMappedTable =
enhancedClient.table("Music",
                                        TableSchema.fromBean(Music.class));
                        LocalDate localDate = LocalDate.parse("2020-04-07");
                        LocalDateTime localDateTime = localDate.atStartOfDay();
                        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

                        Customer record2 = new Customer();
                        record2.setCustName("Fred Pink");
                        record2.setId("id110");
                        record2.setEmail("fredp@noserver.com");
                        record2.setRegistrationDate(instant);

                        Customer record3 = new Customer();
                        record3.setCustName("Susan Pink");
                        record3.setId("id120");
                        record3.setEmail("spink@noserver.com");
                        record3.setRegistrationDate(instant);

                        Customer record4 = new Customer();
                        record4.setCustName("Jerry orange");
                        record4.setId("id101");
                        record4.setEmail("jorange@noserver.com");
                        record4.setRegistrationDate(instant);

                        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest
= BatchWriteItemEnhancedRequest
                                        .builder()
                                        .writeBatches(

WriteBatch.builder(Customer.class) // add items to the Customer

      // table

.mappedTableResource(customerMappedTable)

.addPutItem(builder -> builder.item(record2))

.addPutItem(builder -> builder.item(record3))

.addPutItem(builder -> builder.item(record4))
```

```
                                                                                .build(),

WriteBatch.builder(Music.class) // delete an item from the Music

   // table

.mappedTableResource(musicMappedTable)

.addDeleteItem(builder -> builder.key(

   Key.builder().partitionValue(

                  "Famous Band")

                  .build()))
                                                                                .build())
                                        .build();

                      // Add three items to the Customer table and delete one item
from the Music
                      // table.

enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);
                      System.out.println("done");

               } catch (DynamoDbException e) {
                      System.err.println(e.getMessage());
                      System.exit(1);
               }
       }
}
```

- API-Details finden Sie BatchWriteItem in der AWS SDK for Java 2.x API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.OnDemandThroughput;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <tableName> <key>

            Where:
```

```
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example, Artist).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .tableName(tableName)
            .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
```

```
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
 dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateTable in der AWS SDK for Java 2.x API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import java.util.HashMap;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyval>

                Where:
                    tableName - The Amazon DynamoDB table to delete the item from
 (for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to delete (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Deleting item \"%s\" from %s\n", keyVal, tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
 String key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
```

```
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        DeleteItemRequest deleteReq = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(keyToGet)
                .build();

        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteItem in der AWS SDK for Java 2.x API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to delete (for example,
 Music3).

                **Warning** This program will delete the table that you specify!
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Deleting the Amazon DynamoDB table %s...\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
}
```

- API-Details finden Sie DeleteTable in der AWS SDK for Java 2.x API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information about
(for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
                System.out.println("Throughput");
```

```
                System.out.format("  Read Capacity : %d\n",
 throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
 throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
 tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
 a.attributeType());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- API-Details finden Sie DescribeTable in der AWS SDK for Java 2.x API-Referenz.

`DescribeTimeToLive`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTimeToLive`.

SDK für Java 2.x

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API-Details finden Sie DescribeTimeToLive in der AWS SDK for Java 2.x API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ruft ein Element aus einer Tabelle ab, indem das verwendet wird DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
```

```
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal>

                Where:
                    tableName - The Amazon DynamoDB table from which an item is
 retrieved (for example, Music3).\s
                    key - The key used in the Amazon DynamoDB table (for example,
 Artist).\s
                    keyval - The key value that represents the item to get (for
 example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
 tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
```

```
                .build();

        getDynamoDBItem(ddb, tableName, key, keyVal);
        ddb.close();
    }

    public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
 key, String keyVal) {
        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        GetItemRequest request = GetItemRequest.builder()
                .key(keyToGet)
                .tableName(tableName)
                .build();

        try {
            // If there is no matching item, GetItem does not return any data.
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
            if (returnedItem.isEmpty())
                System.out.format("No item found with the key %s!\n", key);
            else {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");
                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
 returnedItem.get(key1).toString());
                }
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetItem in der AWS SDK for Java 2.x API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ListTablesRequest;
import software.amazon.awssdk.services.dynamodb.model.ListTablesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTables {
    public static void main(String[] args) {
        System.out.println("Listing your Amazon DynamoDB tables:\n");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        listAllTables(ddb);
        ddb.close();
    }

    public static void listAllTables(DynamoDbClient ddb) {
        boolean moreTables = true;
        String lastName = null;

        while (moreTables) {
```

```
            try {
                ListTablesResponse response = null;
                if (lastName == null) {
                    ListTablesRequest request = ListTablesRequest.builder().build();
                    response = ddb.listTables(request);
                } else {
                    ListTablesRequest request = ListTablesRequest.builder()
                            .exclusiveStartTableName(lastName).build();
                    response = ddb.listTables(request);
                }

                List<String> tableNames = response.tableNames();
                if (tableNames.size() > 0) {
                    for (String curName : tableNames) {
                        System.out.format("* %s\n", curName);
                    }
                } else {
                    System.out.println("No tables found!");
                    System.exit(0);
                }

                lastName = response.lastEvaluatedTableName();
                if (lastName == null) {
                    moreTables = false;
                }

            } catch (DynamoDbException e) {
                System.err.println(e.getMessage());
                System.exit(1);
            }
        }
        System.out.println("\nDone!");
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK for Java 2.x API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügt ein Objekt in eine Tabelle ein mit DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedPutItem example.
 */
public class PutItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
 <awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
 (for example, Music3).
```

```
                    key - The key used in the Amazon DynamoDB table (for example,
Artist).
                    keyval - The key value that represents the item to get (for
example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
Day).
                **Warning** This program will  place an item that you specify into a
table!
                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];
        String awards = args[5];
        String awardVal = args[6];
        String songTitle = args[7];
        String songTitleVal = args[8];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
                songTitleVal);
        System.out.println("Done!");
        ddb.close();
    }

    public static void putItemInTable(DynamoDbClient ddb,
```

```
            String tableName,
            String key,
            String keyVal,
            String albumTitle,
            String albumTitleValue,
            String awards,
            String awardVal,
            String songTitle,
            String songTitleVal) {

        HashMap<String, AttributeValue> itemValues = new HashMap<>();
        itemValues.put(key, AttributeValue.builder().s(keyVal).build());
        itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
        itemValues.put(albumTitle,
 AttributeValue.builder().s(albumTitleValue).build());
        itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

        PutItemRequest request = PutItemRequest.builder()
                .tableName(tableName)
                .item(itemValues)
                .build();

        try {
            PutItemResponse response = ddb.putItem(request);
            System.out.println(tableName + " was successfully updated. The request
 id is "
                    + response.responseMetadata().requestId());

        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            System.err.println("Be sure that it exists and that you've typed its
 name correctly!");
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutItem in der AWS SDK for Java 2.x API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fragt eine Tabelle mit DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To query items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client. See the EnhancedQueryRecords example.
 */
public class Query {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <partitionKeyName> <partitionKeyVal>

                Where:
```

```
                    tableName - The Amazon DynamoDB table to put the item in (for
 example, Music3).
                    partitionKeyName - The partition key name of the Amazon DynamoDB
 table (for example, Artist).
                    partitionKeyVal - The value of the partition key that should
 match (for example, Famous Band).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String partitionKeyName = args[1];
        String partitionKeyVal = args[2];

        // For more information about an alias, see:
        // https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
Expressions.ExpressionAttributeNames.html
        String partitionAlias = "#a";

        System.out.format("Querying %s", tableName);
        System.out.println("");
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        int count = queryTable(ddb, tableName, partitionKeyName, partitionKeyVal,
 partitionAlias);
        System.out.println("There were " + count + "  record(s) returned");
        ddb.close();
    }

    public static int queryTable(DynamoDbClient ddb, String tableName, String
 partitionKeyName, String partitionKeyVal,
            String partitionAlias) {
        // Set up an alias for the partition key name in case it's a reserved word.
        HashMap<String, String> attrNameAlias = new HashMap<String, String>();
        attrNameAlias.put(partitionAlias, partitionKeyName);

        // Set up mapping of the partition name with the value.
        HashMap<String, AttributeValue> attrValues = new HashMap<>();
```

```
        attrValues.put(":" + partitionKeyName, AttributeValue.builder()
                .s(partitionKeyVal)
                .build());

        QueryRequest queryReq = QueryRequest.builder()
                .tableName(tableName)
                .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
                .expressionAttributeNames(attrNameAlias)
                .expressionAttributeValues(attrValues)
                .build();

        try {
            QueryResponse response = ddb.query(queryReq);
            return response.count();

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return -1;
    }
}
```

Fragt eine Tabelle mithilfe von `DynamoDbClient` und eines sekundären Index ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
```

```
 * Create the Movies table by running the Scenario example and loading the Movie
 * data from the JSON file. Next create a secondary
 * index for the Movies table that uses only the year column. Name the index
 * **year-index**. For more information, see:
 *
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html
 */
public class QueryItemsUsingIndex {
    public static void main(String[] args) {
        String tableName = "Movies";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        queryIndex(ddb, tableName);
        ddb.close();
    }

    public static void queryIndex(DynamoDbClient ddb, String tableName) {
        try {
            Map<String, String> expressionAttributesNames = new HashMap<>();
            expressionAttributesNames.put("#year", "year");
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(":yearValue",
 AttributeValue.builder().n("2013").build());

            QueryRequest request = QueryRequest.builder()
                    .tableName(tableName)
                    .indexName("year-index")
                    .keyConditionExpression("#year = :yearValue")
                    .expressionAttributeNames(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Scannt eine Amazon DynamoDB-Tabelle mit. DynamoDbClient

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */
```

```
public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information from
 (for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        scanItems(ddb, tableName);
        ddb.close();
    }

    public static void scanItems(DynamoDbClient ddb, String tableName) {
        try {
            ScanRequest scanRequest = ScanRequest.builder()
                    .tableName(tableName)
                    .build();

            ScanResponse response = ddb.scan(scanRequest);
            for (Map<String, AttributeValue> item : response.items()) {
                Set<String> keys = item.keySet();
                for (String key : keys) {
                    System.out.println("The key name is " + key + "\n");
                    System.out.println("The value is " + item.get(key).s());
                }
            }
```

```
        } catch (DynamoDbException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK for Java 2.x -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateItem`

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisiert ein Element in einer Tabelle mit DynamoDbClient.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.AttributeAction;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.AttributeValueUpdate;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To update an Amazon DynamoDB table using the AWS SDK for Java V2, its better
```

```
 * practice to use the
 * Enhanced Client, See the EnhancedModifyItem example.
 */
public class UpdateItem {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName> <key> <keyVal> <name> <updateVal>

                Where:
                    tableName - The Amazon DynamoDB table (for example, Music3).
                    key - The name of the key in the table (for example, Artist).
                    keyVal - The value of the key (for example, Famous Band).
                    name - The name of the column where the value is updated (for
 example, Awards).
                    updateVal - The value used to update an item (for example, 14).
                 Example:
                    UpdateItem Music3 Artist Famous Band Awards 14
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String name = args[3];
        String updateVal = args[4];

        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();
        updateTableItem(ddb, tableName, key, keyVal, name, updateVal);
        ddb.close();
    }

    public static void updateTableItem(DynamoDbClient ddb,
            String tableName,
            String key,
            String keyVal,
```

```
            String name,
            String updateVal) {

        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put(key, AttributeValue.builder()
                .s(keyVal)
                .build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put(name, AttributeValueUpdate.builder()
                .value(AttributeValue.builder().s(updateVal).build())
                .action(AttributeAction.PUT)
                .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(itemKey)
                .attributeUpdates(updatedValues)
                .build();

        try {
            ddb.updateItem(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("The Amazon DynamoDB table was updated!");
    }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

`UpdateTimeToLive`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTimeToLive`.

SDK für Java 2.x

Aktivieren Sie mit AWS SDK for Java 2.x TTL für eine vorhandene DynamoDB-Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
```

```
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final String
 attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

Deaktivieren Sie mit AWS SDK for Java 2.x TTL für eine vorhandene DynamoDB-Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API-Details finden Sie UpdateTimeToLive in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Codebeispiele zeigen, wie eine Anwendung erstellt wird, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Vergleichen mehrerer Werte mit einem einzigen Attribut

Im folgenden Codebeispiel wird gezeigt, wie Sie mehrere Werte mit einem einzigen Attribut in DynamoDB vergleichen.

- Verwenden des IN-Operators, um mehrere Werte mit einem einzigen Attribut zu vergleichen
- Vergleichen des IN-Operators mit mehreren OR-Bedingungen
- Verständnis der Vorteilen von IN hinsichtlich Leistung und Ausdruckskomplexität

SDK für Java 2.x

Vergleichen Sie mehrere Werte mit einem einzigen Attribut in DynamoDB mithilfe von. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
```

```
    /**
     * Queries a table using the IN operator to compare multiple values with a
single attribute.
     *
     * <p>This method demonstrates how to use the IN operator in a filter expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
```

```
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values with
a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the IN
operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);
```

```
        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number of
values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
```

```
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or boolean)
     * @param count The number of values to generate
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value = AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: " +
valueType);
```

```
            }

            values.add(value);
        }

        return values;
    }
```

Anwendungsbeispiel für den Vergleich mehrerer Werte mit. AWS SDK for Java 2.x

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
```

```
                "Category",
                categories);

            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);
```

```
            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator limit
for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
condition expressions");
            System.out.println("7. The IN operator supports up to 100 comparison
values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - Query
  - Scan

Bedingtes Aktualisieren der TTL eines Elements

Im folgenden Codebeispiel wird gezeigt, wie die TTL eines Elements bedingungsabhängig aktualisiert wird.

SDK für Java 2.x

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash or
 partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
 located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
```

```
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR + "=:c,
" + EXPIRE_AT_ATTR + "=:e";
    private static final String CONDITION_EXPRESSION = EXPIRE_AT_ATTR + " > :c";
    private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with TTL
successful.";
    private static final String CONDITION_FAILED_MESSAGE = "Condition check failed.
The item may have already expired.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs an UpdateTTLConditional with a default DynamoDB client.
     */
    public UpdateTTLConditional() {
        this.dynamoDbClient = null;
    }

    /**
     * Constructs an UpdateTTLConditional with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Main method to demonstrate conditionally updating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new UpdateTTLConditional().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Zählen von Ausdrucksoperatoren

Im folgenden Codebeispiel wird gezeigt, wie Ausdrucksoperatoren in DynamoDB gezählt werden.

- Machen Sie sich mit der Obergrenze von 300 Operatoren in DynamoDB vertraut.
- Zählen Sie Operatoren in komplexen Ausdrücken.
- Optimieren Sie Ausdrücke, um innerhalb der Grenzwerte zu bleiben.

SDK für Java 2.x

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param conditionsCount Number of conditions to include
     * @param useAnd Whether to use AND (true) or OR (false) between conditions
     * @return Map containing the filter expression, attribute values, and operator
 count
     */
    public static Map<String, Object> createComplexFilterExpression(int
 conditionsCount, boolean useAnd) {
        // Initialize the expression parts and attribute values
        StringBuilder filterExpression = new StringBuilder();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of conditions
        for (int i = 0; i < conditionsCount; i++) {
            // Add the operator between conditions (except for the first one)
            if (i > 0) {
                filterExpression.append(useAnd ? " AND " : " OR ");
            }

            // Alternate between different comparison operators for variety
            String valueKey = ":val" + i;

            switch (i % 5) {
                case 0:
                    filterExpression.append("attribute").append(i).append(" =
 ").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    filterExpression.append("attribute").append(i).append(" >
 ").append(valueKey);
```

```
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
- 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
```

```
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Creates a complex update expression with a specified number of operations.
     *
     * <p>This method demonstrates how to generate a complex update expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param operationsCount Number of operations to include
     * @return Map containing the update expression, attribute values, and operator
count
     */
    public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
        // Initialize the expression parts and attribute values
        StringBuilder updateExpression = new StringBuilder("SET ");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of SET operations
        for (int i = 0; i < operationsCount; i++) {
            // Add comma between operations (except for the first one)
            if (i > 0) {
                updateExpression.append(", ");
            }

            // Alternate between different types of SET operations
            String valueKey = ":val" + i;

            switch (i % 3) {
                case 0:
                    // Simple assignment (1 operator: =)
                    updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    // Addition (2 operators: = and +)
                    updateExpression
                        .append("attribute")
```

```
                        .append(i)
                        .append(" = attribute")
                        .append(i)
                        .append(" + ")
                        .append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);
```

```
        return result;
    }

    /**
     * Test the operator limit by attempting an operation with a complex expression.
     *
     * <p>This method demonstrates what happens when an expression approaches or
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
            createComplexUpdateExpression((int) Math.ceil(operatorCount / 1.5)); //
Adjust to get close to target count

        String updateExpression = (String) expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
```

```
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int totalOperations) {
```

```
        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into "
+ batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
```

```
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();

            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
```

```
    public static Map<String, Integer> countOperatorsInExpression(String expression)
 {
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|begins_with|
contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\\-]\
\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
```

```
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

Beispiel für die Verwendung des Ausdrucksoperators „Zählen mit AWS SDK for Java 2.x“.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating DynamoDB expression operator counting and
the 300 operator limit");

       try {
           // Example 1: Analyze a simple expression
           System.out.println("\nExample 1: Analyzing a simple expression");
           String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
           Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

           System.out.println("Expression: " + simpleExpression);
           System.out.println("Operator count breakdown:");
           System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
           System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
           System.out.println("- Functions: " + simpleCount.get("functions"));
           System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
           System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
           System.out.println("- Total operators: " + simpleCount.get("total"));

           // Example 2: Analyze a complex expression
           System.out.println("\nExample 2: Analyzing a complex expression");
           String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
               + "(Price > :price AND contains(Description, :keyword) AND "
               + "(Rating >= :minRating OR Reviews > :minReviews))";
           Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

           System.out.println("Expression: " + complexExpression);
           System.out.println("Operator count breakdown:");
           System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
```

```
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " + complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching the
operator limit");
            Map<String, Object> approachingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression into
multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>) breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any expression
is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as 1
operator");
```

```
            System.out.println("4. Each function call (attribute_exists, contains,
etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
operator");
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself and
the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the number
of values");
            System.out.println("8. Parentheses for grouping and attribute paths
don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
reports '301 operators'");
            System.out.println("10. For complex operations, break them into multiple
smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Tabelle mit einem globalen sekundären Index

Das folgende Codebeispiel zeigt, wie Sie eine Tabelle mit einem globalen sekundären Index erstellen.

SDK für Java 2.x

Erstellen Sie eine DynamoDB-Tabelle mit Global Secondary Index mithilfe von. AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(ISSUE_ID_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(TITLE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(CREATE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(DUE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());

            // Key schema for table
            final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.HASH)
                .build()); // Partition key
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.RANGE)
                .build()); // Sort key

            // Initial provisioned throughput settings for the indexes
            final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
                .readCapacityUnits(1L)
                .writeCapacityUnits(1L)
                .build();

            // CreateDateIndex
            final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
```

```
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(CREATE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex = GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();
```

```
            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex = GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));
```

```
        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie CreateTable in der AWS SDK for Java 2.x API-Referenz.

Erstellen einer Tabelle mit aktiviertem Warmdurchsatz

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit aktiviertem Warmdurchsatz erstellt wird.

SDK für Java 2.x

Erstellen Sie mithilfe von AWS SDK for Java 2.x eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
```

```
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }

    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
 attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
 throughput settings.
```

```
    *
    * @param ddb The DynamoDB client
    * @param tableName The name of the table to create
    * @param partitionKey The partition key attribute name
    * @param sortKey The sort key attribute name
    * @param miscellaneousKeyAttribute Additional key attribute name for GSI
    * @param nonKeyAttribute Non-key attribute to include in GSI projection
    * @param tableReadCapacityUnits Read capacity units for the table
    * @param tableWriteCapacityUnits Write capacity units for the table
    * @param tableWarmReadUnitsPerSecond Warm read units per second for the table
    * @param tableWarmWriteUnitsPerSecond Warm write units per second for the table
    * @param globalSecondaryIndexName The name of the GSI to create
    * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the GSI
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per second
for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
       final Long tableWarmWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadCapacityUnits,
       final Long globalSecondaryIndexWriteCapacityUnits,
       final Long globalSecondaryIndexWarmReadUnitsPerSecond,
       final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

       // Define the table attributes
       final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
       final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
       final AttributeDefinition miscellaneousKeyAttributeDefinition =
```

```
            buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
        final AttributeDefinition[] attributeDefinitions = {
            partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
        };

        // Define the table key schema
        final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
        final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
        final KeySchemaElement[] keySchema = {partitionKeyElement, sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);
```

```
        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- API-Details finden Sie CreateTable in der AWS SDK for Java 2.x API-Referenz.

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen eines Elements mit einer TTL

Das folgende Codebeispiel zeigt, wie Sie ein Element mit TTL erstellen.

SDK für Java 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB items.
 */
public class CreateTTL {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
```

```
                primaryKey - The name of the primary key. Also known as the hash or
partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
successful.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs a CreateTTL instance with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public CreateTTL(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Constructs a CreateTTL with a default DynamoDB client.
     */
    public CreateTTL() {
        this.dynamoDbClient = null;
    }

    /**
     * Main method to demonstrate creating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
```

```
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
```

```
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
 AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- API-Details finden Sie PutItem in der AWS SDK for Java 2.x API-Referenz.

Erstellen und Verwalten von globalen MRSC-Tabellen

Das folgende Codebeispiel zeigt, wie globale DynamoDB-Tabellen mit Multi-Region Strong Consistency (MRSC) erstellt und verwaltet werden.

- Erstellen Sie eine Tabelle mit Multi-Region starker Konsistenz.
- Überprüfen Sie die MRSC-Konfiguration und den Replikatstatus.
- Testen Sie die starke Konsistenz in allen Regionen mit sofortigen Lesevorgängen.
- Führen Sie bedingte Schreibvorgänge mit MRSC-Garantien durch.
- Bereinigen Sie die globalen MRSC-Tabellenressourcen.

SDK für Java 2.x

Erstellen Sie eine regionale Tabelle, die für die MRSC-Konvertierung bereit ist, mit AWS SDK for Java 2.x.

```
   public static CreateTableResponse createRegionalTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating regional table: " + tableName + " (must be empty
for MRSC)");

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
```

```
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Konvertiert eine regionale Tabelle in MRSC mit Replikaten und Witness. AWS SDK for Java 2.x

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());

            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;
```

```
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Beschreiben Sie eine globale MRSC-Tabellenkonfiguration mit. AWS SDK for Java 2.x

```
    public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing MRSC global table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());

            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
```

```
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
            }

            if (response.table().globalTableWitnesses() != null
                && !response.table().globalTableWitnesses().isEmpty()) {
                LOGGER.info("Number of witnesses: "
                    + response.table().globalTableWitnesses().size());
                response.table()
                    .globalTableWitnesses()
                    .forEach(witness -> LOGGER.info(
                        "Witness region: " + witness.regionName() + ", Status: " +
witness.witnessStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Fügen Sie mit AWS SDK for Java 2.x Testelemente hinzu, um die starke Konsistenz von MRSC zu überprüfen.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
```

```
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
```

```
                .cause(e)
                .build();
        }
    }
```

Lesen Sie Elemente mit konsistenten Lesevorgängen aus MRSC-Replikaten mit. AWS SDK for Java 2.x

```
    public static GetItemResponse getItemWithConsistentRead(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Reading item from MRSC global table with consistent read: "
+ tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            GetItemRequest getItemRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .consistentRead(true)
                .build();

            GetItemResponse response = dynamoDbClient.getItem(getItemRequest);
```

```
            if (response.hasItem()) {
                LOGGER.info("Item found with strong consistency - no wait time
needed");
            } else {
                LOGGER.info("Item not found");
            }

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Führen Sie bedingte Aktualisierungen mit MRSC-Garantien mithilfe von. AWS SDK for Java 2.x

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
```

```
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());

            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
```

```
                .cause(e)
                .build();
        }
    }
```

Warten Sie, bis die MRSC-Replikate und Zeugen aktiv werden. AWS SDK for Java 2.x

```
    public static void waitForMRSCReplicasActive(
        final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
        throws InterruptedException {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (maxWaitTimeSeconds <= 0) {
            throw new IllegalArgumentException("Max wait time must be positive");
        }

        try {
            LOGGER.info("Waiting for MRSC replicas and witnesses to become active: "
+ tableName);

            final long startTime = System.currentTimeMillis();
            final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
            int backoffSeconds = 5; // Start with 5 second intervals
            final int maxBackoffSeconds = 30; // Cap at 30 seconds

            while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
                DescribeTableResponse response = describeMRSCTable(dynamoDbClient,
tableName);

                boolean allActive = true;
                StringBuilder statusReport = new StringBuilder();

                if (response.table().multiRegionConsistency() == null
                    || !MultiRegionConsistency.STRONG
                        .toString()
```

```
.equals(response.table().multiRegionConsistency().toString())) {
                    allActive = false;
                    statusReport
                        .append("MultiRegionConsistency: ")
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
                if (response.table().globalTableWitnesses() == null
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }

                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString())) {
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
                                .append(" ");
                        }
                    }
                }

                // Check witness status
```

```
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
                        if (!"ACTIVE".equals(witness.witnessStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }

                if (allActive) {
                    LOGGER.info("All MRSC replicas and witnesses are now active: " +
tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active. Status: "
+ statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active after "
+ maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

Bereinigen Sie MRSC-Replikate und Zeugen mithilfe von. AWS SDK for Java 2.x

```
   public static UpdateTableResponse cleanupMRSCReplicas(
       final DynamoDbClient dynamoDbClient,
       final String tableName,
       final Region replicaRegion,
       final Region witnessRegion) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (replicaRegion == null) {
           throw new IllegalArgumentException("Replica region cannot be null");
       }
       if (witnessRegion == null) {
           throw new IllegalArgumentException("Witness region cannot be null");
       }

       try {
           LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

           // Remove replica using ReplicationGroupUpdate
           ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
               .delete(DeleteReplicationGroupMemberAction.builder()
                   .regionName(replicaRegion.id())
                   .build())
               .build();

           // Remove witness
           GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
               .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                   .regionName(witnessRegion.id())
                   .build())
               .build();

           UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
               .tableName(tableName)
               .replicaUpdates(List.of(replicaUpdate))
```

```
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

Vollständige Demonstration des MRSC-Workflows mit. AWS SDK for Java 2.x

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
```

```
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);

            // Use the existing GlobalTableOperations method for basic table waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
            LOGGER.info("Step 4: Testing strong consistency with data operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");

            // Immediately read from replica region (no wait needed with MRSC)
```

```
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The Beatles",
"Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
            }

            // Test conditional update from replica region
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles", "Hey
Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null) {
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
                cleanupVerification = describeMRSCTable(primaryClient, tableName);
            }

            // Delete the primary table
```

```
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency (MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle eventual
consistency)");
            LOGGER.info("- Support for conditional writes and transactions across
regions");
            LOGGER.info("- Consistent read operations from any region without
waiting");

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

Erstellen und Verwalten globaler Tabellen zur Demonstration von MREC

Im folgenden Codebeispiel wird gezeigt, wie globale DynamoDB-Tabellen mit Replikaten in mehreren Regionen erstellt und verwaltet werden.

- Erstellen Sie eine Tabelle mit einem globalen sekundären Index und DynamoDB Streams.
- Fügen Sie Replikate in verschiedenen Regionen hinzu, um eine globale Tabelle zu erstellen.
- Entfernen Sie Replikate aus einer globalen Tabelle.

- Fügen Sie Testelemente hinzu, um die regionsübergreifende Replikation zu überprüfen.
- Beschreiben Sie die globale Tabellenkonfiguration und den Replikatstatus.

SDK für Java 2.x

Erstellen Sie eine Tabelle mit Global Secondary Index und DynamoDB-Streams mithilfe von. AWS SDK for Java 2.x

```
   public static CreateTableResponse createTableWithGSI(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (indexName == null || indexName.trim().isEmpty()) {
           throw new IllegalArgumentException("Index name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating table: " + tableName + " with GSI: " + indexName);

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
                   KeySchemaElement.builder()
                       .attributeName("Artist")
                       .keyType(KeyType.HASH)
```

```
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Warten Sie, bis eine Tabelle aktiv wird, indem Sie sie verwenden AWS SDK for Java 2.x.

```
    public static void waitForTableActive(final DynamoDbClient dynamoDbClient, final
String tableName) {

        if (dynamoDbClient == null) {
```

```
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Waiting for table to become active: " + tableName);

            try (DynamoDbWaiter waiter =
                DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
                DescribeTableRequest request =
                    DescribeTableRequest.builder().tableName(tableName).build();

                waiter.waitUntilTableExists(request);
                LOGGER.info("Table is now active: " + tableName);
            }

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to wait for table to become active: " + tableName
+ " - " + e.getMessage());
            throw e;
        }
    }
```

Fügen Sie ein Replikat hinzu, mit dem Sie eine globale Tabelle erstellen AWS SDK for Java 2.x oder erweitern möchten.

```
    public static UpdateTableResponse addReplica(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final String indexName,
        final Long readCapacity) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
```

```
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }

        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
```

```
            LOGGER.severe("Failed to add replica in region: " + replicaRegion.id() +
" - " + e.getMessage());
            throw e;
        }
    }
```

Entfernen Sie ein Replikat aus einer globalen Tabelle mit. AWS SDK for Java 2.x

```
    public static UpdateTableResponse removeReplica(
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .delete(builder -> builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());
```

```
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

Fügen Sie Testelemente hinzu, mit AWS SDK for Java 2.x denen die Replikation überprüft werden soll.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to table: " + tableName);

            Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
            item.put(
                "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
```

```
                    .s(artist)
                    .build());
            item.put(
                "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(songTitle)
                    .build());

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Beschreiben Sie die globale Tabellenkonfiguration und die Replikate mit AWS SDK for Java 2.x.

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();
```

```
            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

Vollständiges Beispiel für globale Tabellenoperationen unter Verwendung von AWS SDK for Java 2.x.

```
    public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

        String tableName = "Music";
        String indexName = "SongTitleIndex";
        Long readCapacity = 15L;

        // Create DynamoDB client for the source region
        try (DynamoDbClient dynamoDbClient =
            DynamoDbClient.builder().region(sourceRegion).build()) {

            try {
```

```
                // Step 1: Create the initial table with GSI and streams
                LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
                createTableWithGSI(dynamoDbClient, tableName, indexName);

                // Step 2: Wait for table to become active
                LOGGER.info("Step 2: Waiting for table to become active");
                waitForTableActive(dynamoDbClient, tableName);

                // Step 3: Add replica in destination region
                LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
                addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);

                // Step 4: Wait a moment for replica creation to start
                Thread.sleep(5000);

                // Step 5: Describe table to view replica information
                LOGGER.info("Step 5: Describing table to view replicas");
                describeTable(dynamoDbClient, tableName);

                // Step 6: Add a test item to verify replication
                LOGGER.info("Step 6: Adding test item to verify replication");
                putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

                LOGGER.info("Global table setup completed successfully!");
                LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

                // Step 7: Remove replica and clean up table
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
```

```
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTable
  - DescribeTable
  - PutItem
  - UpdateTable

Erkennen von PSA in Bildern

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Überwachen der Leistung von Amazon DynamoDB

Im folgenden Codebeispiel wird gezeigt, wie die Verwendung von DynamoDB durch eine Anwendung zur Leistungsüberwachung konfiguriert wird.

SDK für Java 2.x

Im folgenden Codebeispiel wird gezeigt, wie eine Java-Anwendung zur Leistungsüberwachung von DynamoDB konfiguriert wird. Die Anwendung sendet Metrikdaten an eine CloudWatch Stelle, an die Sie die Leistung überwachen können.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch
- DynamoDB

Ausführen erweiterter Abfrageoperationen

Im folgenden Codebeispiel wird gezeigt, wie Sie erweiterte Abfrageoperationen in DynamoDB ausführen.

- Abfragen von Tabellen mithilfe verschiedener Filter- und Bedingungstechniken
- Implementieren der Paginierung für große Ergebnismengen
- Verwenden globaler sekundärer Indizes für alternative Zugriffsmuster
- Anwenden von Konsistenzkontrollen auf Grundlage der Anwendungsanforderungen

SDK für Java 2.x

Abfrage mit stark konsistenten Lesevorgängen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

Abfrage unter Verwendung eines globalen sekundären Index mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
 response.count() + " items");
```

```
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
```

```
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
 + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
 found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
 tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Abfrage mit Paginierung unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
```

```
        final String tableName, final String partitionKeyName, final String
partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());
```

```
                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
 response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
 pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
 " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
```

```
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
```

```
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
 partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
```

```
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();
```

```
            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
 DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
 multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

Abfrage mit einem Filterausdruck und Verwendungslimit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
 filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());
```

```
        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Ausführen von Listenoperationen

Das folgende Codebeispiel zeigt, wie Listenoperationen für DynamoDB-Tabellen ausgeführt werden.

- Hinzufügen von Elementen zu einem Listenattribut
- Entfernen von Elementen aus einem Listenattribut

- Aktualisieren von bestimmten Elementen in einer Liste nach Index
- Verwenden von Funktionen zum Anfügen und zum Auflisten von Indizes

SDK für Java 2.x

Demonstrieren Sie Listenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
```

```
        AttributeValue listValue =
AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {
```

```
        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
     *
     * <p>This method demonstrates how to update a specific element in a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
```

```
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
```

```
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
```

```
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating list operations in DynamoDB");

        try {
            // Example 1: Append items to a list
            System.out.println("\nExample 1: Appending items to a list");
            List<AttributeValue> tagsToAppend = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Gadget").build());

            UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);

            System.out.println("Updated list attribute: " +
appendResponse.attributes());

            // Example 2: Prepend items to a list
            System.out.println("\nExample 2: Prepending items to a list");
            List<AttributeValue> tagsToPrepend = List.of(
                AttributeValue.builder().s("Featured").build(),
                AttributeValue.builder().s("New").build());

            UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

            System.out.println("Updated list attribute: " +
prependResponse.attributes());

            // Example 3: Update a specific element in a list
```

```
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " + currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
newItems)");
```

```
            System.out.println("4. To prepend items, use list_append(newItems,
 existingList)");
            System.out.println("5. Use index notation (list[0]) to access or update
 specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
 list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Ausführen von Map-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Map-Operationen in DynamoDB ausgeführt werden.

- Fügen Sie verschachtelte Attribute in Map-Strukturen hinzu und aktualisieren Sie sie.
- Entfernen bestimmter Felder aus Maps
- Arbeiten mit tief verschachtelten Map-Attributen

SDK für Java 2.x

Demonstrieren Sie Kartenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // A single UpdateExpression can't reference both a map and a path inside
        // that same map (for example "#mapName" and "#mapName.#mapKey"): DynamoDB
        // rejects overlapping document paths with a ValidationException. Instead,
        // perform the update in two steps.

        // Step 1: create the map with an empty value if it doesn't already exist.
        dynamoDbClient.updateItem(UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = if_not_exists(#mapName, :emptyMap)")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(
                ":emptyMap", AttributeValue.builder().m(new HashMap<>()).build()))
            .build());

        // Step 2: set the key within the now-guaranteed-to-exist map.
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName.#mapKey = :value")
```

```
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
```

```
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
```

```
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
```

```
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName)) {
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Kartenoperationen mit AWS SDK for Java 2.x.

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating map operations in DynamoDB");

       try {
           // Example 1: Create a map with multiple attributes
           System.out.println("\nExample 1: Creating a map with multiple
attributes");
           Map<String, AttributeValue> productDetails = new HashMap<>();
           productDetails.put("Color", AttributeValue.builder().s("Red").build());
           productDetails.put("Weight", AttributeValue.builder().n("2.5").build());
           productDetails.put(
               "Dimensions", AttributeValue.builder().s("10x20x5").build());

           UpdateItemResponse createResponse =
               createMapWithAttributes(dynamoDbClient, tableName, key, "Details",
productDetails);

           System.out.println("Created map attribute: " +
createResponse.attributes());

           // Example 2: Update a specific attribute in a map
           System.out.println("\nExample 2: Updating a specific attribute in a
map");
           UpdateItemResponse updateResponse = updateMapAttributeSafe(
               dynamoDbClient,
               tableName,
               key,
               "Details",
               "Color",
               AttributeValue.builder().s("Blue").build());

           System.out.println("Updated map attribute: " +
updateResponse.attributes());
```

```
            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap = getMapAttribute(dynamoDbClient,
tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
```

```
            System.out.println("3. You can update individual attributes without
 overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
 structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes in
 a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Ausführen von Set-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Set-Operationen in DynamoDB ausgeführt werden.

- Hinzufügen von Elementen zu einem Set-Attribut
- Entfernen von Elementen aus einem Set-Attribut
- Verwenden der Operationen ADD und DELETE mit Sätzen

SDK für Java 2.x

Demonstrieren Sie Set-Operationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
```

```
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
```

```
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
```

```
     * <p>This method demonstrates how to use the DELETE operation to remove values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
```

```
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
```

```
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);

            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not exist
in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }
```

```
    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
```

```
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
```

```
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Set-Operationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating set operations in DynamoDB");

        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a number
set
                );
```

```
            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);

            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );

            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());
```

```
            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false // Not a
number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());

            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
```

```
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number sets
(NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
            System.out.println("8. You can create a set with multiple values in a
single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("Creating an Amazon DynamoDB table named " + tableName
            + " with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Adding multiple records into the " + tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        System.out.println("Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
```

```
    }

    public static boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND year
= ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }
```

```
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();
```

```
            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse = dbWaiter
                .waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
 BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
```

```
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie2)
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);

            BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
```

```
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);

            BatchExecuteStatementRequest batchRequest =
 BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
 ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
 response.toString());
            System.out.println("Added new movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
```

```
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);
```

```
        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
```

```
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
```

```
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK for Java 2.x API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a key
 named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("Updating a record.");
        updateTableItem(ddb);

        System.out.println("Querying the movies released in 2013.");
```

```
        queryTable(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();
```

```
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {

            // Add 200 movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            AttributeValue att1 = AttributeValue.builder()
```

```
                .n(String.valueOf(year))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s(title)
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s(info)
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            // Insert the movie into the Amazon DynamoDB table.
            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added Movie " + title);

            parameters.remove(att1);
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
```

```
            System.out.println("ExecuteStatement successful: " +
 response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
 'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {
```

```
        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK for Java 2.x API-Referenz.

Abfragen einer Tabelle mit einem globalen sekundären Index

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem globalen sekundären Index abgefragt wird.

- Abfragen einer DynamoDB-Tabelle mit ihrem Primärschlüssel
- Abfragen eines globalen sekundären Index (GSI) für alternative Zugriffsmuster
- Vergleichen von Tabellenabfragen und GSI-Abfragen

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit ihrem Primärschlüssel und einem Global Secondary Index (GSI) mit ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
```

```
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

Vergleichen Sie das direkte Abfragen einer Tabelle mit dem Abfragen eines GSI mit. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition key
attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        try (DynamoDbClient ddb = DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " + basePartitionKeyName +
" = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
```

```
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName + "
= " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the base
table (projection)");
            System.out.println("3. GSIs consume read capacity units from the GSI's
capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data (cannot
use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mithilfe einer „begins_with“-Bedingung

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einer „begins_with“-Bedingung abgefragt wird.

- Verwenden der Funktion „begins_with“ in einem Schlüsselbedingungsausdruck
- Filtern der Elemente auf Grundlage eines Präfixmusters für den Sortierschlüssel

SDK für Java 2.x

Fragen Sie mit AWS SDK for Java 2.x eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung für den Sortierschlüssel ab.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix", sortKeyPrefix);
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

Demonstrieren Sie mit AWS SDK for Java 2.x die Verwendung von „begins_with“ mit unterschiedlichen Präfixlängen.

```
    public static void main(String[] args) {
        try {
            CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
            LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
                config.getPartitionKeyName(),
                config.getPartitionKeyValue(),
                config.getSortKeyName(),
                config.getSortKeyPrefix()
            });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new BeginsWithQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withSortKeyName(config.getSortKeyName())
                .withSortKeyPrefix(config.getSortKeyPrefix())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> LOGGER.info(item.toString()));

            // Demonstrate with a different prefix
            if (!config.getSortKeyPrefix().isEmpty()) {
                String shorterPrefix = config.getSortKeyPrefix()
                    .substring(0, Math.max(1, config.getSortKeyPrefix().length() /
2));
                LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

                final QueryResponse response2 = new BeginsWithQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withSortKeyName(config.getSortKeyName())
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();
```

```
                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
 response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit einem Datumsbereich

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mithilfe eines Datumsbereichs im Sortierschlüssel abgefragt wird.

- Abfragen von Elementen innerhalb eines bestimmten Datumsbereichs
- Verwenden von Vergleichsoperatoren für Sortierschlüssel im Datumsformat

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

Zeigt, wie eine DynamoDB-Tabelle mit Datumsbereichsfilterung abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <dateKeyName>
<startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });
```

```
            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date range:
{0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date range:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with time
components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
```

```
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit einem komplexen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem komplexen Filterausdruck abgefragt wird.

- Anwenden von komplexen Filterausdrücken auf Abfrageergebnisse
- Kombinieren von mehreren Bedingungen über logische Operatoren
- Filtern von Elementen auf der Grundlage von Nicht-Schlüsselattributen.

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck ab mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
```

```
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit einem dynamischen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem dynamischen Filterausdruck abgefragt wird.

- Erstellen von Filterausdrücken dynamisch zur Laufzeit
- Konstruieren von Filterbedingungen auf der Grundlage von Benutzereingaben oder Anwendungsstatus
- Hinzufügen oder Entfernen von Filterkriterien unter bestimmten Bedingungen

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
```

```
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
 partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
```

```
            }
        }
    }
```

Veranschaulicht, wie dynamische Filterausdrücke mit verwendet werden. AWS SDK for Java 2.x

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
```

```
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
 DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
 multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit einem Filterausdruck und einem Limit

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem Filterausdruck und Limit abgefragt wird.

- Anwenden von Filterausdrücken auf Abfrageergebnisse mit einem Limit für die ausgewerteten Elemente
- Verstehen, wie sich Limits auf gefilterte Abfrageergebnisse auswirken
- Steuern der maximalen Anzahl der in einer Abfrage verarbeiteten Elemente

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit einem Filterausdruck ab und beschränken Sie die Verwendung. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
```

```
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit verschachtelten Attributen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit verschachtelten Attributen abgefragt wird.

- Zugreifen und Filtern nach verschachtelten Attributen in DynamoDB-Elementen
- Verwenden von Dokumentpfadausdrücken, um auf verschachtelte Elemente zu verweisen

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit verschachtelten Attributen mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
```

```
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
```

```
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful. Found
" + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
    }
```

Demonstriert, wie eine DynamoDB-Tabelle mit verschachtelten Attributen abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <nestedPath>
<nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
```

```
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM()) {
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
```

```
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath + "."
+ nestedAttr + ": "
                            + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to access
nested attributes");
            System.out.println("2. Use expression attribute names for each component
of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit Paginierung

Das folgende Codebeispiel zeigt, wie eine Tabelle mit Paginierung abgefragt wird.

- Implementieren der Paginierung für DynamoDB-Abfrageergebnisse
- Verwenden Sie den LastEvaluatedKey , um nachfolgende Seiten abzurufen.

- Steuern der Anzahl der Elemente pro Seite mit dem Parameter „Limit“

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit Paginierung unter Verwendung von ab. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
```

```
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
```

```
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

Demonstriert, wie eine DynamoDB-Tabelle mit Paginierung abgefragt wird.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    pageSize (optional) - The maximum number of items to return per
page. (Default: 10)
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;
```

```
        System.out.println("Querying items with pagination (page size: " + pageSize
+ ")");

        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " + allItems.size()
+ " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() - sampleSize) +
" more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
```

```
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen einer Tabelle mit strikt konsistenten Lesevorgängen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit strikt konsistenten Lesevorgängen abgefragt wird.

- Konfigurieren des Konsistenzniveaus für DynamoDB-Abfragen
- Verwenden der strikt konsistenten Lesevorgänge, um die aktuellen Daten zu erhalten
- Erkennen der Kompromisse zwischen letztendlicher Konsistenz und starker Konsistenz

SDK für Java 2.x

Fragen Sie eine DynamoDB-Tabelle mit konfigurierbarer Lesekonsistenz mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen nach TTL-Elementen

Im folgenden Codebeispiel wird gezeigt, wie TTL-Elemente abgefragt werden.

SDK für Java 2.x

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Abfragen von Tabellen mithilfe von Datums- und Uhrzeitmustern

Im folgenden Codebeispiel wird gezeigt, wie Datums- und Uhrzeitmuster zum Abfragen von Tabellen verwendet werden.

- Speichern und Abfragen von date/time Werten in DynamoDB.
- Implementieren von Datumsbereichsabfragen mithilfe von Sortierschlüsseln
- Formatieren von Datumszeichenfolgen für effektive Abfragen

SDK für Java 2.x

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
```

```
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();
```

```
        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
 items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
 e.getMessage());
            throw e;
        }
    }
```

Abfrage mit Datums-/Uhrzeitvariablen mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
```

```
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " + e.getMessage());
            throw e;
        }
    }
```

Führen Sie mit AWS SDK for Java 2.x Abfragen innerhalb von Datumsbereichen in Unix-Epochenzeitstempeln aus.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
```

```
            ":startDate",
AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
AttributeValue.builder().n(String.valueOf(endEpoch)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
e.getMessage());
            throw e;
        }
    }
```

Abfrage innerhalb von Datumsbereichen mithilfe von LocalDateTime Objekten mit. AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
```

```
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
            throw new IllegalArgumentException("Start and end LocalDateTime must not
 be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct format
        final String startDate =
 startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
 endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
 dateKeyName, startDate, endDate);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK for Java 2.x -API-Referenz.

Verstehen der Reihenfolge der Aktualisierungsausdrücke

Im folgenden Codebeispiel wird die Reihenfolge der Aktualisierungsausdrücke erklärt.

- Erfahren, wie DynamoDB Aktualisierungsausdrücke verarbeitet
- Verstehen der Reihenfolge der Operationen in Aktualisierungsausdrücken
- Vermeiden von unerwarteten Ergebnissen durch Verständnis der Ausdrucksauswertung

SDK für Java 2.x

Demonstrieren Sie das Aktualisieren der Ausdrucksreihenfolge mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();
```

```
            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
```

```
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter =
Counter")
                .expressionAttributeValues(
                    Map.of(":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " + e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
```

```
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
```

```
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
```

```
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating update expression order in DynamoDB");

        try {
            // Example 1: Demonstrate update order effects
            System.out.println("\nExample 1: Demonstrating update order effects");
            Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

            if ((boolean) orderResults.get("success")) {
                System.out.println("SET first, then ADD:");
                System.out.println("  " + orderResults.get("setFirstResponse"));

                System.out.println("ADD first, then SET:");
                System.out.println("  " + orderResults.get("addFirstResponse"));

                System.out.println("SET with multiple attributes:");
                System.out.println("  " + orderResults.get("multiSetResponse"));

                System.out.println("SET with self-reference:");
                System.out.println("  " +
orderResults.get("selfReferenceResponse"));
            } else {
                System.out.println("Error: " + orderResults.get("error"));
            }

            // Example 2: Update with SET first, then REMOVE
            System.out.println("\nExample 2: Update with SET first, then REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());
```

```
            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
            System.out.println("1. Update expressions are processed in this order:
SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed from
left to right");
            System.out.println("3. SET operations use the item state before any
updates in the expression");
            System.out.println("4. When an attribute is referenced multiple times,
the first operation wins");
```

```
            System.out.println("5. To reference a new value, split the update into
 multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
 change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
 separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der Einstellung für den Warmdurchsatz einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie die Warmdurchsatzeinstellung einer Tabelle aktualisiert wird.

SDK für Java 2.x

Aktualisieren Sie mit AWS SDK for Java 2.x die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
```

```
    * Updates a DynamoDB table with warm throughput settings for both the table and
a global secondary index.
    *
    * @param ddb The DynamoDB client
    * @param tableName The name of the table to update
    * @param tableReadUnitsPerSecond Read units per second for the table
    * @param tableWriteUnitsPerSecond Write units per second for the table
    * @param globalSecondaryIndexName The name of the global secondary index to
update
    * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for the
GSI
    * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for the
GSI
    */
   public static void updateDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final Long tableReadUnitsPerSecond,
       final Long tableWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadUnitsPerSecond,
       final Long globalSecondaryIndexWriteUnitsPerSecond) {

       final WarmThroughput tableWarmThroughput =
           buildWarmThroughput(tableReadUnitsPerSecond, tableWriteUnitsPerSecond);
       final WarmThroughput gsiWarmThroughput =
           buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

       final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
           .update(UpdateGlobalSecondaryIndexAction.builder()
               .indexName(globalSecondaryIndexName)
               .warmThroughput(gsiWarmThroughput)
               .build())
           .build();

       final UpdateTableRequest request = UpdateTableRequest.builder()
           .tableName(tableName)
           .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
           .warmThroughput(tableWarmThroughput)
           .build();

       try {
```

```
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- API-Details finden Sie UpdateTable in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der TTL eines Elements

Das folgende Codebeispiel zeigt, wie Sie die TTL eines Elements aktualisieren.

SDK für Java 2.x

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
```

```
        keyMap.put(PRIMARY_KEY_ATTR,
AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR, AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response = dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Zustandsmaschine erstellt wird, die AWS Lambda Funktionen nacheinander aufruft.

SDK für Java 2.x

Zeigt, wie ein AWS serverloser Workflow mithilfe von AWS Step Functions und erstellt wird. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step-Funktionen

Verwenden von Operationen mit unteilbaren Zählern

Im folgenden Codebeispiel wird gezeigt, wie Sie Operationen mit unteilbaren Zählern in DynamoDB verwenden.

- Inkrementieren von unteilbaren Zählern mithilfe von ADD- und SET-Operationen
- Sicheres Erhöhen von Zählern, die möglicherweise nicht existieren
- Implementieren einer optimistischen Sperre für Zähleroperationen

SDK für Java 2.x

Demonstrieren Sie atomare Gegenoperationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
```

```
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
```

```
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = if_not_exists(#counterName, :zero)
+ :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
```

```
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if the
counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",
                AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
```

```
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
```

```
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "SetCounter", 1);
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;

            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
```

```
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName)) {
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());

            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse = incrementCounterSafely(dynamoDbClient,
tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());
```

```
            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " + comparison.get("addDuration")
+ " ns");
                System.out.println("SET duration: " + comparison.get("setDuration")
+ " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " + comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
```

```
            System.out.println("4. Use if_not_exists to handle the case where the
 counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
 from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be isolated
 from other writes");
            System.out.println("7. ADD can only be used with number and set data
 types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Java 2.x API-Referenz.

Verwenden von bedingten Operationen

Im folgenden Codebeispiel wird gezeigt, wie bedingte Operationen in DynamoDB verwendet werden.

- Implementieren von bedingten Schreibvorgängen, um das Überschreiben von Daten zu verhindern
- Verwenden von Bedingungsausdrücken, um Geschäftsregeln durchzusetzen
- Ordnungsgemäßer Umgang mit Fehlern bei bedingten Prüfungen

SDK für Java 2.x

Demonstrieren Sie bedingte Operationen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
```

```
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
```

```
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr", conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

```
    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();

            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr = :data")
                    .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
```

```
                    .build();

                UpdateItemResponse initResponse =
dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
"#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
```

```
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
     * <p>This method demonstrates how to use multiple conditions in a condition
expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditions Map of attribute names to values for conditions
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdateWithMultipleConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        Map<String, AttributeValue> conditions,
```

```
        String updateAttribute,
        AttributeValue updateValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Build the condition expression and attribute names/values
            StringBuilder conditionExpression = new StringBuilder();
            Map<String, String> expressionAttributeNames = new HashMap<>();
            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

            // Add update attribute
            expressionAttributeNames.put("#updateAttr", updateAttribute);
            expressionAttributeValues.put(":updateVal", updateValue);

            // Add conditions
            int i = 0;
            for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
                String attrName = condition.getKey();
                AttributeValue attrValue = condition.getValue();

                String nameKey = "#cond" + i;
                String valueKey = ":val" + i;

                expressionAttributeNames.put(nameKey, attrName);
                expressionAttributeValues.put(valueKey, attrValue);

                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" = ").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
```

```
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update was
not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

Beispiel für die Verwendung bedingter Operationen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
```

```
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
```

```
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " + lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price", AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());

            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }

            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the condition
fails");
            System.out.println("4. Conditions can be used with update, delete, and
put operations");
            System.out.println("5. Multiple conditions can be combined with AND and
OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
```

```
                "7. Conditional operations consume the same amount of write capacity
whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteItem
  - PutItem
  - UpdateItem

Verwenden von Ausdrucksattributnamen

Im folgenden Codebeispiel wird gezeigt, wie Sie Ausdrucksattributnamen in DynamoDB verwenden.

- Arbeiten mit reservierten Wörtern in DynamoDB-Ausdrücken
- Verwenden von Platzhaltern für Ausdrucksattributnamen
- Behandeln von Sonderzeichen in Attributnamen

SDK für Java 2.x

Demonstrieren Sie Ausdrucksattributnamen mit AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that contain special characters.
     *
```

```
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Queries items using an attribute that is a reserved word.
     *
     * <p>This method demonstrates how to use expression attribute names in a query
     * when the attribute is a reserved word.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key
     * @param reservedWordAttribute The reserved word attribute to filter on
     * @param value The value to compare against
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static QueryResponse queryWithReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the query parameters using expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attr", reservedWordAttribute);

        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);
        expressionAttributeValues.put(":value", value);

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attr = :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
```

```
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;

            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
```

```
    }
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating expression attribute names in DynamoDB");

        try {
            // Example 1: Update an attribute that is a reserved word
            System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
            UpdateItemResponse response1 = updateReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
```

```
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());

            System.out.println("Updated nested attribute: " +
response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute name
placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
            filters.put(
                "Product-Type", AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 = scanWithMultipleAttributeNames(dynamoDbClient,
tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
```

```
            System.out.println("1. Use expression attribute names (#name) for
reserved words");
            System.out.println("2. Use expression attribute names for attributes
with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute names
in a single expression");
            System.out.println("6. Expression attribute names are case-sensitive");
            System.out.println("7. Expression attribute names are only used in
expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - Query
  - UpdateItem

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();
```

```
    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
```

```
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für Amazon EC2 unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon EC2 beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
```

```
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets additional information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI’s
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
```

```
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            Where:
               keyName -  A key pair name (for example, TestKeyPair).\s
               fileName -  A file name where the key information is written to.\s
               groupName - The name of the security group.\s
               groupDesc - The description of the security group.\s
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to easily
 launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.

            EC2 provides a wide range of instance types, each with different
 compute, memory,
```

```
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2 instances
 as part of their
            application deployment pipelines. With EC2, teams can focus on building
 and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for this
 service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material as
 a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a private
 key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
            logger.info("Key Pair successfully created. Key Fingerprint: " +
 response.keyFingerprint());
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2
in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on an
EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
```

```
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM) or
server image.
           It uniquely identifies a particular version of an EC2 instance, including
its operating system,
           pre-installed software, and any custom configurations. This allows you to
consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future = ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
```

```
        logger.info("Once the EC2 instance is created, it is placed into a running
state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
```

```
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains the
same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to be
accessed consistently from a
            known IP address.
```

```
            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you have
purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique Allocation
ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: " +allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: " +associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definieren Sie eine Klasse, die EC2-Aktionen umschließt.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
```

```
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger = LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .build();
```

```
            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
```

```
    * @return a CompletableFuture that completes when the security group is deleted
    */
   public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
       DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
           .groupId(groupId)
           .build();

       CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
           } else if (resp == null) {
               throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
           }
       }).thenApply(resp -> null);
   }

   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has been
terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
```

```
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
```

```
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();

       // Disassociate the address asynchronously.
       CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              throw new RuntimeException("Failed to disassociate address", ex);
           }
       });

       return response;
   }

   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
```

```
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after associating
address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
```

```
   public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
       DescribeInstancesRequest request = DescribeInstancesRequest.builder()
           .instanceIds(newInstanceId)
           .build();

       DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
       AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.reservations().stream()
               .flatMap(reservation -> reservation.instances().stream())
               .filter(instance -> instance.instanceId().equals(newInstanceId))
               .findFirst()
               .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
       }).thenApply(v -> {
           String publicIpAddress = publicIpAddressRef.get();
           if (publicIpAddress == null) {
               throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
           }
           return publicIpAddress;
       }).exceptionally(ex -> {
           logger.info("Failed to describe instances: " + ex.getMessage());
           throw new RuntimeException("Failed to describe instances", ex);
       });
   }

   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
```

```
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
```

```
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
               }
           }
           return "";
       });
   }

   /**
    * Asynchronously describes an AWS EC2 image with the specified image ID.
    *
    * @param imageId the ID of the image to be described
```

```
    * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
    * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
    */
   public CompletableFuture<String> describeImageAsync(String imageId) {
       DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
           .imageIds(imageId)
           .build();

       AtomicReference<String> imageIdRef = new AtomicReference<>();
       DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
       return paginator.subscribe(response -> {
           response.images().stream()
               .filter(image -> image.imageId().equals(imageId))
               .findFirst()
               .ifPresent(image -> {
                   logger.info("The description of the image is " +
image.description());
                   logger.info("The name of the image is " + image.name());
                   imageIdRef.set(image.imageId());
               });
       }).thenApply(v -> {
           String id = imageIdRef.get();
           if (id == null) {
               throw new RuntimeException("No images found with the provided image
ID.");
           }
           return id;
       }).exceptionally(ex -> {
           logger.info("Failed to describe image: " + ex.getMessage());
           throw new RuntimeException("Failed to describe image", ex);
       });
   }

   /**
    * Retrieves the parameter values asynchronously using the AWS Systems Manager
(SSM) API.
    *
    * @return a {@link CompletableFuture} that holds the response from the SSM API
call to get parameters by path
    */
   public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
```

```
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });

        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();
```

```
        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }

    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
```

```
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
```

```
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
```

```
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
     *
     * @return a {@link CompletableFuture} that, when completed, contains the first
default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
```

```
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
 start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Allocates an Elastic IP address asynchronously in the VPC domain.
    *
    * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
    */
   public CompletableFuture<String> allocateAddressAsync() {
       AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
           .domain(DomainType.VPC)
           .build();

       CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
       return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to allocate address", ex);
           }
       });
   }
```

- API-Details finden Sie AllocateAddress in der AWS SDK for Java 2.x API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after associating
address.");
           }
       }).whenComplete((result, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to associate address", ex);
           }
       });
   }
```

- API-Details finden Sie AssociateAddress in der AWS SDK for Java 2.x API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security group
    * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
```

```
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK for Java 2.x API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
               }
           } else {
               throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
```

```
            }
        });

        return responseFuture;
    }
```

- API-Details finden Sie CreateKeyPair in der AWS SDK for Java 2.x API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
```

```
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
```

```
            });
    }
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
```

```
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK for Java 2.x API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
```

```
            }
        }).thenApply(resp -> null);
    }
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK for Java 2.x API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type string
for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
```

```
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK for Java 2.x API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
```

```
        });
    }
```

- API-Details finden Sie DescribeInstances in der AWS SDK for Java 2.x API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK for Java 2.x API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
       }).thenApply(v -> {
           String groupId = groupIdRef.get();
           if (groupId == null) {
```

```
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Java 2.x API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
```

```
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
 ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

- API-Details finden Sie DisassociateAddress in der AWS SDK for Java 2.x API-Referenz.

`GetPasswordData`

Das folgende Codebeispiel zeigt die Verwendung`GetPasswordData`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
 instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want to
 fetch the password data
     * @return a {@link CompletableFuture} that completes when the password data has
 been fetched
     * @throws RuntimeException if there was a failure in fetching the password data
     */
    public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
 ec2AsyncClient, String instanceId) {
```

```
        GetPasswordDataRequest getPasswordDataRequest =
 GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance: " +
 instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- API-Details finden Sie GetPasswordData in der AWS SDK for Java 2.x API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }
```

- API-Details finden Sie ReleaseAddress in der AWS SDK for Java 2.x API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
               .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
               .thenCompose(waitResponse -> getAsyncClient().waiter()
                   .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                   .thenApply(runningResponse -> instanceIdVal));
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }
```

- API-Details finden Sie RunInstances in der AWS SDK for Java 2.x API-Referenz.

StartInstances

Das folgende Codebeispiel zeigt die VerwendungStartInstances.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
    *
    * @param instanceId the ID of the instance to start
    * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> startInstanceAsync(String instanceId) {
       StartInstancesRequest startRequest = StartInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
       CompletableFuture<Void> resultFuture = new CompletableFuture<>();
       return getAsyncClient().startInstances(startRequest)
           .thenCompose(response ->
               ec2Waiter.waitUntilInstanceRunning(describeRequest)
           )
           .thenAccept(waiterResponse -> {
               logger.info("Successfully started instance " + instanceId);
```

```
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- API-Details finden Sie StartInstances in der AWS SDK for Java 2.x API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();
```

```
        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }
```

- API-Details finden Sie StopInstances in der AWS SDK for Java 2.x API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has been
terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }
```

- API-Details finden Sie TerminateInstances in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
```

```
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
```

```
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
```

```
   private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
           throws IOException, InterruptedException {
       loadBalancer.deleteLoadBalancer(lbName);
       System.out.println("*** Wait 30 secs for resource to be deleted");
       TimeUnit.SECONDS.sleep(30);
       loadBalancer.deleteTargetGroup(targetGroupName);
       autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
       autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
       autoScaler.deleteTemplate(templateName);
       database.deleteTable(tableName);
   }

   private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
       Scanner in = new Scanner(System.in);
       System.out.println(
               """
                       For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                       to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                       against various kinds of failures.

                       Some of the resources create by this demo are:
                       \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                       \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                       \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                       \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                       """);

       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("Creating and populating a DynamoDB table named " +
tableName);
       Database database = new Database();
       database.createTable(tableName, fileName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);
```

```
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
```

```
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
```

```
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
```

```
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
```

```
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
```

```
   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode = response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
                               new
InputStreamReader(response.getEntity().getContent()));
                       StringBuilder jsonResponse = new StringBuilder();
```

```
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                        target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
```

```
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
```

```
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
```

```
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
```

```
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
```

```
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }
```

```
                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
```

```
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
```

```
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
```

```
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
```

```
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
```

```
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();
```

```
        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();
```

```
        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();
```

```
            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
```

```
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");
```

```
            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
 {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener

- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Beispiele für Amazon ECR unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon ECR

Das folgende Codebeispiel zeigt, wie Sie mit Amazon ECR beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

    public static void main(String[] args) {
        final String usage = """
            Usage:    <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.
            """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String repoName = args[0];
        EcrClient ecrClient = EcrClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listImageTags(ecrClient, repoName);
    }
    public static void listImageTags(EcrClient ecrClient, String repoName){
        ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
            .repositoryName(repoName)
            .build();

        ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
        imagesIterable.stream()
            .flatMap(r -> r.imageIds().stream())
            .forEach(image -> System.out.println("The docker image tag is: "
 +image.imageTag()));
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der AWS SDK for Java 2.x -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-ECR-Repository.
- Legen Sie Repository-Richtlinien fest.
- Rufen Sie Repository-URIs ab.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.

- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.
- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
```

```
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = """
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
             service provided by AWS. It allows developers and organizations to
 securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
 throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
             programmatically interact with the Amazon ECR service. This allows
 developers to
             automate the storage, retrieval, and management of container images as
 part of their application
             deployment pipelines. With ECR, teams can focus on building and
 deploying their
```

```
            applications without having to worry about the underlying
 infrastructure required to
            host and manage a container registry.

           This scenario walks you through how to perform key operations for this
 service.
           Let's get started...

           You have two choices:
           1 - Run the entire program.
           2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
           this program that did not complete).
           """);

       while (true) {
           String input = scanner.nextLine();
           if (input.trim().equalsIgnoreCase("1")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else if (input.trim().equalsIgnoreCase("2")) {
               String repoName = "echo-text";
               ecrActions.deleteECRRepository(repoName);
               return;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }

       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println("""
          1. Create an ECR repository.

          The first task is to ensure we have a local Docker image named echo-
text.
          If this image exists, then an Amazon ECR repository is created.

          An ECR repository is a private Docker container repository provided
          by Amazon Web Services (AWS). It is a managed service that makes it easy
          to store, manage, and deploy Docker container images.\s
```

```
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);
```

```
        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
```

```
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
```

```
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images, ensuring
that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
```

```
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
```

```
        System.out.println("9. As an optional step, you can interact with the image
 in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
 CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");
```

```
            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
          }
       }
   }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
```

```
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
    private static EcrAsyncClient ecrClient;

    private static DockerClient dockerClient;

    private static Logger logger = LoggerFactory.getLogger(ECRActions.class);
```

```
   /**
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
               if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                   System.out.println("The Amazon ECR repository already exists,
moving on...");
                   DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                       .repositoryNames(repoName)
                       .build();
```

```
                    DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                    return describeResponse.repositories().get(0).repositoryArn();
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }

    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
" repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
```

```
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }



    private static DockerClient getDockerClient() {
        String osName = System.getProperty("os.name");
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
            DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
            dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
        } else {
            dockerClient = DockerClientBuilder.getInstance().build();
        }
        return dockerClient;
    }

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static EcrAsyncClient getAsyncClient() {

        /*
         The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
         and it is designed to provide a high-performance, asynchronous HTTP client
for interacting with AWS services.
         It uses the Netty framework to handle the underlying network communication
and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
responses.
```

```
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50)  // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
 call attempt timeout.
            .build();

        if (ecrClient == null) {
            ecrClient = EcrAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ecrClient;
    }

    /**
     * Sets the lifecycle policy for the specified repository.
     *
     * @param repoName the name of the repository for which to set the lifecycle
 policy.
     */
    public void setLifeCyclePolicy(String repoName) {
        /*
           This policy helps to maintain the size and efficiency of the container
 registry
           by automatically removing older and potentially unused images,
           ensuring that the storage is optimized and the registry remains up-to-
date.
         */
        String polText = """
             {
             "rules": [
```

```
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
            ]
            }
            """;

        StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
StartLifecyclePolicyPreviewRequest.builder()
            .lifecyclePolicyText(polText)
            .repositoryName(repoName)
            .build();

        CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
        response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
            if (lifecyclePolicyPreviewResponse != null) {
                System.out.println("Lifecycle policy preview started
successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });
        // Wait for the CompletableFuture to complete.
        response.join();
    }

    /**
```

```
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
           } else {
               if (describeRepositoriesResponse != null) {
                   if (!describeRepositoriesResponse.repositories().isEmpty()) {
                       String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                       System.out.println("Repository URI found: " +
repositoryUri);
                   } else {
                       System.out.println("No repositories found for the given
name.");
                   }
               } else {
```

```
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }

    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the console.
     * If an exception occurs, the method handles the exception and prints the error
message.
     *
     * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
     * @throws RuntimeException if there is an unexpected error during the
operation.
     */
    public void getAuthToken() {
        CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
        response.whenComplete((authorizationTokenResponse, ex) -> {
            if (authorizationTokenResponse != null) {
                AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
                String token = authorizationData.authorizationToken();
                if (!token.isEmpty()) {
                    System.out.println("The token was successfully retrieved.");
                }
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
                }
            }
        });
```

```
        response.join();
    }

    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     * @throws EcrException if an AWS error occurs while getting the repository
policy.
     */
    public String getRepoPolicy(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .build();

        CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy retrieved successfully.");
            } else {
                if (ex.getCause() instanceof EcrException) {
                    throw (EcrException) ex.getCause();
                } else {
                    String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                    throw new RuntimeException(errorMessage, ex);
                }
            }
        });

        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }

    /**
     * Sets the repository policy for the specified ECR repository.
     *
```

```
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
    * @throws EcrException                     if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
       SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .policyText(policyDocument)
           .build();

       CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy set successfully.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof RepositoryPolicyNotFoundException) {
```

```
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }

    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
```

```
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }

    // Make sure local image echo-text exists.
    public boolean isEchoTextImagePresent() {
        try {
            List<Image> images = getDockerClient().listImagesCmd().exec();
            boolean helloWorldFound = false;
            for (Image image : images) {
                String[] repoTags = image.getRepoTags();
                if (repoTags != null) {
                    for (String tag : repoTags) {
                        if (tag.startsWith("echo-text")) {
                            System.out.println(tag);
                            helloWorldFound = true;
                        }
                    }
                }
            }
            if (helloWorldFound) {
                System.out.println("The local image named echo-text exists.");
                return true;
            } else {
```

```
                System.out.println("The local image named echo-text does not
 exist.");
                return false;
            }
        } catch (DockerClientException ex) {
            logger.error("ERROR: " + ex.getMessage());
            return false;
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Aktionen

`CreateRepository`

Das folgende Codebeispiel zeigt die Verwendung`CreateRepository`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
    *
    * @param repoName the name of the repository to create.
    * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
    * @throws IllegalArgumentException     If repository name is invalid.
    * @throws RuntimeException             if an error occurs while creating the
repository.
    */
   public String createECRRepository(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       CreateRepositoryRequest request = CreateRepositoryRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
       try {
           CreateRepositoryResponse result = response.join();
           if (result != null) {
               System.out.println("The " + repoName + " repository was created
successfully.");
               return result.repository().repositoryArn();
           } else {
               throw new RuntimeException("Unexpected response type");
           }
       } catch (CompletionException e) {
           Throwable cause = e.getCause();
           if (cause instanceof EcrException ex) {
               if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
                   System.out.println("The Amazon ECR repository already exists,
moving on...");
                   DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
                       .repositoryNames(repoName)
                       .build();
                   DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
                   return describeResponse.repositories().get(0).repositoryArn();
```

```
                } else {
                    throw new RuntimeException(ex);
                }
            } else {
                throw new RuntimeException(e);
            }
        }
    }
```

- API-Details finden Sie CreateRepository in der AWS SDK for Java 2.x API-Referenz.

`DeleteRepository`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRepository`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     * @throws IllegalArgumentException if the repository name is null or empty.
     * @throws EcrException if there is an error deleting the repository.
     * @throws RuntimeException if an unexpected error occurs during the deletion
process.
     */
    public void deleteECRRepository(String repoName) {
        if (repoName == null || repoName.isEmpty()) {
            throw new IllegalArgumentException("Repository name cannot be null or
empty");
        }

        DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
            .force(true)
```

```
            .repositoryName(repoName)
            .build();

        CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
        response.whenComplete((deleteRepositoryResponse, ex) -> {
            if (deleteRepositoryResponse != null) {
                System.out.println("You have successfully deleted the " + repoName +
" repository");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                }
            }
        });

        // Wait for the CompletableFuture to complete
        response.join();
    }
```

- API-Details finden Sie DeleteRepository in der AWS SDK for Java 2.x API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
```

```
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
     * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
     */
    public void verifyImage(String repositoryName, String imageTag) {
        DescribeImagesRequest request = DescribeImagesRequest.builder()
            .repositoryName(repositoryName)
            .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
            .build();

        CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
        response.whenComplete((describeImagesResponse, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof EcrException) {
                        throw (EcrException) cause;
                    } else {
                        throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                    }
                } else {
                    throw new RuntimeException("Unexpected error: " +
 ex.getCause());
                }
            } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
                System.out.println("Image is present in the repository.");
            } else {
                System.out.println("Image is not present in the repository.");
            }
        });

        // Wait for the CompletableFuture to complete.
        response.join();
    }
```

- API-Details finden Sie DescribeImages in der AWS SDK for Java 2.x API-Referenz.

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRepositories`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the repository URI for the specified repository name.
    *
    * @param repoName the name of the repository to retrieve the URI for.
    * @return the repository URI for the specified repository name.
    * @throws EcrException        if there is an error retrieving the repository
information.
    * @throws CompletionException if the asynchronous operation completes
exceptionally.
    */
   public void getRepositoryURI(String repoName) {
       DescribeRepositoriesRequest request = DescribeRepositoriesRequest.builder()
           .repositoryNames(repoName)
           .build();

       CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
       response.whenComplete((describeRepositoriesResponse, ex) -> {
           if (ex != null) {
               Throwable cause = ex.getCause();
               if (cause instanceof InterruptedException) {
                   Thread.currentThread().interrupt();
                   String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               } else if (cause instanceof EcrException) {
                   throw (EcrException) cause;
               } else {
                   String errorMessage = "Unexpected error: " + cause.getMessage();
                   throw new RuntimeException(errorMessage, cause);
               }
```

```
            } else {
                if (describeRepositoriesResponse != null) {
                    if (!describeRepositoriesResponse.repositories().isEmpty()) {
                        String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
                        System.out.println("Repository URI found: " +
repositoryUri);
                    } else {
                        System.out.println("No repositories found for the given
name.");
                    }
                } else {
                    System.err.println("No response received from
describeRepositories.");
                }
            }
        });
        response.join();
    }
```

- API-Details finden Sie DescribeRepositories in der AWS SDK for Java 2.x API-Referenz.

`GetAuthorizationToken`

Das folgende Codebeispiel zeigt die Verwendung`GetAuthorizationToken`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
     * If the operation is successful, the method prints the token to the console.
```

```
    * If an exception occurs, the method handles the exception and prints the error
message.
    *
    * @throws EcrException     if there is an error retrieving the authorization
token from ECR.
    * @throws RuntimeException if there is an unexpected error during the
operation.
    */
   public void getAuthToken() {
       CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
       response.whenComplete((authorizationTokenResponse, ex) -> {
           if (authorizationTokenResponse != null) {
               AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
               String token = authorizationData.authorizationToken();
               if (!token.isEmpty()) {
                   System.out.println("The token was successfully retrieved.");
               }
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
               }
           }
       });
       response.join();
   }
```

- API-Details finden Sie GetAuthorizationToken in der AWS SDK for Java 2.x API-Referenz.

`GetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetRepositoryPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Gets the repository policy for the specified repository.
    *
    * @param repoName the name of the repository.
    * @throws EcrException if an AWS error occurs while getting the repository
policy.
    */
   public String getRepoPolicy(String repoName) {
       if (repoName == null || repoName.isEmpty()) {
           throw new IllegalArgumentException("Repository name cannot be null or
empty");
       }

       GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
           .repositoryName(repoName)
           .build();

       CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               System.out.println("Repository policy retrieved successfully.");
           } else {
               if (ex.getCause() instanceof EcrException) {
                   throw (EcrException) ex.getCause();
               } else {
                   String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                   throw new RuntimeException(errorMessage, ex);
               }
           }
       });
```

```
        GetRepositoryPolicyResponse result = response.join();
        return result != null ? result.policyText() : null;
    }
```

- API-Details finden Sie GetRepositoryPolicy in der AWS SDK for Java 2.x API-Referenz.

`PushImageCmd`

Das folgende Codebeispiel zeigt die Verwendung`PushImageCmd`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName  the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    public void pushDockerImage(String repoName, String imageName) {
        System.out.println("Pushing " + imageName + " to Amazon ECR will take a few
seconds.");
        CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
            .thenApply(response -> {
                String token =
response.authorizationData().get(0).authorizationToken();
                String decodedToken = new String(Base64.getDecoder().decode(token));
                String password = decodedToken.substring(4);

                DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
```

```
                assert repoData != null;
                String registryURL = repoData.repositoryUri().split("/")[0];

                AuthConfig authConfig = new AuthConfig()
                    .withUsername("AWS")
                    .withPassword(password)
                    .withRegistryAddress(registryURL);
                return authConfig;
            })
            .thenCompose(authConfig -> {
                DescribeRepositoriesResponse descrRepoResponse =
 getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
                Repository repoData =
 descrRepoResponse.repositories().stream().filter(r ->
 r.repositoryName().equals(repoName)).findFirst().orElse(null);
                getDockerClient().tagImageCmd(imageName + ":latest",
 repoData.repositoryUri() + ":latest", imageName).exec();
                try {

 getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
                    System.out.println("The " + imageName + " was pushed to ECR");

                } catch (InterruptedException e) {
                    throw (RuntimeException) e.getCause();
                }
                return CompletableFuture.completedFuture(authConfig);
            });

        authResponseFuture.join();
    }
```

- API-Details finden Sie PushImageCmd in der AWS SDK for Java 2.x API-Referenz.

`SetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`SetRepositoryPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Sets the repository policy for the specified ECR repository.
    *
    * @param repoName the name of the ECR repository.
    * @param iamRole  the IAM role to be granted access to the repository.
    * @throws RepositoryPolicyNotFoundException if the repository policy does not
exist.
    * @throws EcrException                      if there is an unexpected error
setting the repository policy.
    */
   public void setRepoPolicy(String repoName, String iamRole) {
       /*
         This example policy document grants the specified AWS principal the
permission to perform the
         `ecr:BatchGetImage` action. This policy is designed to allow the specified
principal
         to retrieve Docker images from the ECR repository.
        */
       String policyDocumentTemplate = """
            {
             "Version":"2012-10-17",
             "Statement" : [ {
               "Sid" : "new statement",
               "Effect" : "Allow",
               "Principal" : {
                 "AWS" : "%s"
               },
               "Action" : "ecr:BatchGetImage"
             } ]
           }
            """;

       String policyDocument = String.format(policyDocumentTemplate, iamRole);
```

```
        SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
            .repositoryName(repoName)
            .policyText(policyDocument)
            .build();

        CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                System.out.println("Repository policy set successfully.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof RepositoryPolicyNotFoundException) {
                    throw (RepositoryPolicyNotFoundException) cause;
                } else if (cause instanceof EcrException) {
                    throw (EcrException) cause;
                } else {
                    String errorMessage = "Unexpected error: " + cause.getMessage();
                    throw new RuntimeException(errorMessage, cause);
                }
            }
        });
        response.join();
    }
```

- API-Details finden Sie SetRepositoryPolicy in der AWS SDK for Java 2.x API-Referenz.

`StartLifecyclePolicyPreview`

Das folgende Codebeispiel zeigt die Verwendung`StartLifecyclePolicyPreview`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
    *
    * @param repositoryName The name of the Amazon ECR repository.
    * @param imageTag       The tag of the image to verify.
    * @throws EcrException             if there is an error retrieving the image
 information from Amazon ECR.
    * @throws CompletionException      if the asynchronous operation completes
 exceptionally.
    */
   public void verifyImage(String repositoryName, String imageTag) {
       DescribeImagesRequest request = DescribeImagesRequest.builder()
           .repositoryName(repositoryName)
           .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
           .build();

       CompletableFuture<DescribeImagesResponse> response =
 getAsyncClient().describeImages(request);
       response.whenComplete((describeImagesResponse, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException) {
                   Throwable cause = ex.getCause();
                   if (cause instanceof EcrException) {
                       throw (EcrException) cause;
                   } else {
                       throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
                   }
               } else {
                   throw new RuntimeException("Unexpected error: " +
 ex.getCause());
               }
           } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
               System.out.println("Image is present in the repository.");
           } else {
               System.out.println("Image is not present in the repository.");
           }
       });

       // Wait for the CompletableFuture to complete.
       response.join();
   }
```

- API-Details finden Sie StartLifecyclePolicyPreview in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon ECS unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandConfiguration;
import software.amazon.awssdk.services.ecs.model.ExecuteCommandLogging;
import software.amazon.awssdk.services.ecs.model.ClusterConfiguration;
import software.amazon.awssdk.services.ecs.model.CreateClusterResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
```

```
import software.amazon.awssdk.services.ecs.model.CreateClusterRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCluster {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <clusterName>\s

                Where:
                   clusterName - The name of the ECS cluster to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        String clusterArn = createGivenCluster(ecsClient, clusterName);
        System.out.println("The cluster ARN is " + clusterArn);
        ecsClient.close();
    }

    public static String createGivenCluster(EcsClient ecsClient, String clusterName)
 {
        try {
            ExecuteCommandConfiguration commandConfiguration =
 ExecuteCommandConfiguration.builder()
                    .logging(ExecuteCommandLogging.DEFAULT)
                    .build();
```

```
            ClusterConfiguration clusterConfiguration =
 ClusterConfiguration.builder()
                    .executeCommandConfiguration(commandConfiguration)
                    .build();

            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                    .clusterName(clusterName)
                    .configuration(clusterConfiguration)
                    .build();

            CreateClusterResponse response =
 ecsClient.createCluster(clusterRequest);
            return response.cluster().clusterArn();

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateCluster in der AWS SDK for Java 2.x API-Referenz.

`CreateService`

Das folgende Codebeispiel zeigt die Verwendung`CreateService`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.AwsVpcConfiguration;
import software.amazon.awssdk.services.ecs.model.NetworkConfiguration;
```

```
import software.amazon.awssdk.services.ecs.model.CreateServiceRequest;
import software.amazon.awssdk.services.ecs.model.LaunchType;
import software.amazon.awssdk.services.ecs.model.CreateServiceResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateService {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <clusterName> <serviceName> <securityGroups>
 <subnets> <taskDefinition>

                                Where:
                                  clusterName - The name of the ECS cluster.
                                  serviceName - The name of the ECS service to
 create.
                                  securityGroups - The name of the security group.
                                  subnets - The name of the subnet.
                                  taskDefinition - The name of the task definition.
                                """;

                if (args.length != 5) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String clusterName = args[0];
                String serviceName = args[1];
                String securityGroups = args[2];
                String subnets = args[3];
                String taskDefinition = args[4];
                Region region = Region.US_EAST_1;
                EcsClient ecsClient = EcsClient.builder()
                                .region(region)
                                .build();
```

```
                String serviceArn = createNewService(ecsClient, clusterName,
serviceName, securityGroups, subnets,
                                     taskDefinition);
                System.out.println("The ARN of the service is " + serviceArn);
                ecsClient.close();
        }

        public static String createNewService(EcsClient ecsClient,
                        String clusterName,
                        String serviceName,
                        String securityGroups,
                        String subnets,
                        String taskDefinition) {

                try {
                        AwsVpcConfiguration vpcConfiguration =
AwsVpcConfiguration.builder()
                                        .securityGroups(securityGroups)
                                        .subnets(subnets)
                                        .build();

                        NetworkConfiguration configuration =
NetworkConfiguration.builder()
                                        .awsvpcConfiguration(vpcConfiguration)
                                        .build();

                        CreateServiceRequest serviceRequest =
CreateServiceRequest.builder()
                                        .cluster(clusterName)
                                        .networkConfiguration(configuration)
                                        .desiredCount(1)
                                        .launchType(LaunchType.FARGATE)
                                        .serviceName(serviceName)
                                        .taskDefinition(taskDefinition)
                                        .build();

                        CreateServiceResponse response =
ecsClient.createService(serviceRequest);
                        return response.service().serviceArn();

                } catch (EcsException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
```

```
                }
                return "";
        }
}
```

- API-Details finden Sie CreateService in der AWS SDK for Java 2.x API-Referenz.

`DeleteService`

Das folgende Codebeispiel zeigt die Verwendung`DeleteService`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DeleteServiceRequest;
import software.amazon.awssdk.services.ecs.model.EcsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterName> <serviceArn>\s
```

```
                Where:
                  clusterName - The name of the ECS cluster.
                  serviceArn - The ARN of the ECS service.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        deleteSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void deleteSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            DeleteServiceRequest serviceRequest = DeleteServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .build();

            ecsClient.deleteService(serviceRequest);
            System.out.println("The Service was successfully deleted");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteService in der AWS SDK for Java 2.x API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeClustersRequest;
import software.amazon.awssdk.services.ecs.model.DescribeClustersResponse;
import software.amazon.awssdk.services.ecs.model.Cluster;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeClusters {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> \s

                Where:
                  clusterArn - The ARN of the ECS cluster to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String clusterArn = args[0];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        descCluster(ecsClient, clusterArn);
    }

    public static void descCluster(EcsClient ecsClient, String clusterArn) {
        try {
            DescribeClustersRequest clustersRequest =
 DescribeClustersRequest.builder()
                    .clusters(clusterArn)
                    .build();

            DescribeClustersResponse response =
 ecsClient.describeClusters(clustersRequest);
            List<Cluster> clusters = response.clusters();
            for (Cluster cluster : clusters) {
                System.out.println("The cluster name is " + cluster.clusterName());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK for Java 2.x API-Referenz.

`DescribeTasks`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTasks`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.DescribeTasksRequest;
import software.amazon.awssdk.services.ecs.model.DescribeTasksResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.Task;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTaskDefinitions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <clusterArn> <taskId>\s

                Where:
                  clusterArn - The ARN of an ECS cluster.
                  taskId - The task Id value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterArn = args[0];
```

```
        String taskId = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        getAllTasks(ecsClient, clusterArn, taskId);
        ecsClient.close();
    }

    public static void getAllTasks(EcsClient ecsClient, String clusterArn, String
 taskId) {
        try {
            DescribeTasksRequest tasksRequest = DescribeTasksRequest.builder()
                    .cluster(clusterArn)
                    .tasks(taskId)
                    .build();

            DescribeTasksResponse response = ecsClient.describeTasks(tasksRequest);
            List<Task> tasks = response.tasks();
            for (Task task : tasks) {
                System.out.println("The task ARN is " + task.taskDefinitionArn());
            }

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeTasks in der AWS SDK for Java 2.x API-Referenz.

`ListClusters`

Das folgende Codebeispiel zeigt die Verwendung`ListClusters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.ListClustersResponse;
import software.amazon.awssdk.services.ecs.model.EcsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        listAllClusters(ecsClient);
        ecsClient.close();
    }

    public static void listAllClusters(EcsClient ecsClient) {
        try {
            ListClustersResponse response = ecsClient.listClusters();
            List<String> clusters = response.clusterArns();
            for (String cluster : clusters) {
                System.out.println("The cluster arn is " + cluster);
            }
```

```
        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListClusters in der AWS SDK for Java 2.x API-Referenz.

`UpdateService`

Das folgende Codebeispiel zeigt die Verwendung`UpdateService`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecs.EcsClient;
import software.amazon.awssdk.services.ecs.model.EcsException;
import software.amazon.awssdk.services.ecs.model.UpdateServiceRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class UpdateService {

    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:
                   <clusterName> <serviceArn>\s

                Where:
                  clusterName - The cluster name.
                  serviceArn - The service ARN value.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clusterName = args[0];
        String serviceArn = args[1];
        Region region = Region.US_EAST_1;
        EcsClient ecsClient = EcsClient.builder()
                .region(region)
                .build();

        updateSpecificService(ecsClient, clusterName, serviceArn);
        ecsClient.close();
    }

    public static void updateSpecificService(EcsClient ecsClient, String
 clusterName, String serviceArn) {
        try {
            UpdateServiceRequest serviceRequest = UpdateServiceRequest.builder()
                    .cluster(clusterName)
                    .service(serviceArn)
                    .desiredCount(0)
                    .build();

            ecsClient.updateService(serviceRequest);
            System.out.println("The service was modified");

        } catch (EcsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie UpdateService in der AWS SDK for Java 2.x API-Referenz.

Elastic Load Balancing – Beispiele für Version 2 unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Elastic Load Balancing

Das folgende Codebeispiel zeigt die ersten Schritte mit Elastic Load Balancing.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloLoadBalancer {
```

```
        public static void main(String[] args) {
                ElasticLoadBalancingV2Client loadBalancingV2Client =
 ElasticLoadBalancingV2Client.builder()
                                .region(Region.US_EAST_1)
                                .build();

                DescribeLoadBalancersResponse loadBalancersResponse =
 loadBalancingV2Client
                                .describeLoadBalancers(r -> r.pageSize(10));
                List<LoadBalancer> loadBalancerList =
 loadBalancersResponse.loadBalancers();
                for (LoadBalancer lb : loadBalancerList)
                        System.out.println("Load Balancer DNS name = " +
 lb.dnsName());
        }
}
```

- API-Details finden Sie DescribeLoadBalancers in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`CreateListener`

Das folgende Codebeispiel zeigt die Verwendung`CreateListener`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
 String lbName, int port,
```

```
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
```

```
.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- API-Details finden Sie CreateListener in der AWS SDK for Java 2.x API-Referenz.

`CreateLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`CreateLoadBalancer`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
```

```
   public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
           Action action = Action.builder()
                   .targetGroupArn(targetGroupARN)
                   .type("forward")
                   .build();
```

```
            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- API-Details finden Sie CreateLoadBalancer in der AWS SDK for Java 2.x API-Referenz.

`CreateTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateTargetGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
```

```
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }
```

- API-Details finden Sie CreateTargetGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`DeleteLoadBalancer`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Deletes a load balancer.
```

```
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }
```

- API-Details finden Sie DeleteLoadBalancer in der AWS SDK for Java 2.x API-Referenz.

`DeleteTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTargetGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Deletes the target group.
```

```
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }
```

- API-Details finden Sie DeleteTargetGroup in der AWS SDK for Java 2.x API-Referenz.

`DescribeTargetHealth`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTargetHealth`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
```

```
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
```

- API-Details finden Sie DescribeTargetHealth in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
```

```
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
```

```
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
```

```
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
```

```
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       System.out.println(
               """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();
```

```
        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
```

```
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
```

```
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
```

```
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
```

```
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
```

```
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
```

```
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();
```

```
        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
```

```
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
```

```
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();
```

```
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
{
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

MediaStore Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren MediaStore.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateContainer`

Das folgende Codebeispiel zeigt die Verwendung`CreateContainer`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>
```

```
                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateContainer in der AWS SDK for Java 2.x API-Referenz.

`DeleteContainer`

Das folgende Codebeispiel zeigt die Verwendung`DeleteContainer`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>
```

```
                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteContainer in der AWS SDK for Java 2.x API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """
```

```
                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item to
delete.
                   containerName - The name of the container.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData, String
completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
```

```
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- API-Details finden Sie DeleteObject in der AWS SDK for Java 2.x API-Referenz.

`DescribeContainer`

Das folgende Codebeispiel zeigt die Verwendung`DescribeContainer`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
 containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
 DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
 mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
 containerResponse.container().name());
```

```
            System.out.println("The container ARN is " +
 containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie DescribeContainer in der AWS SDK for Java 2.x API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is saved,
 including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }
```

```
    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();

            // Write out the data to a file.
            ResponseInputStream<GetObjectResponse> data =
 mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- API-Details finden Sie GetObject in der AWS SDK for Java 2.x API-Referenz.

`ListContainers`

Das folgende Codebeispiel zeigt die Verwendung`ListContainers`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
```

```
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListContainers in der AWS SDK for Java 2.x API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file location
 to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);
```

```
            PutObjectRequest objectRequest = PutObjectRequest.builder()
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- API-Details finden Sie PutObject in der AWS SDK for Java 2.x API-Referenz.

AWS Entity Resolution Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS Entity Resolution.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Entity Resolution

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Entity Resolution beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEntityResoultion {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(HelloEntityResoultion.class);

   private static EntityResolutionAsyncClient entityResolutionAsyncClient;
   public static void main(String[] args) {
       listMatchingWorkflows();
   }

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
```

```
    }

    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved workflows.
     * It uses an asynchronous approach with a paginator and waits for the operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- API-Details finden Sie ListMatchingWorkflows in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Schemazuordnung.
- Erstellen Sie einen AWS Entity Resolution Arbeitsablauf.
- Starten Sie den passenden Auftrag für den Workflow.

- Rufen Sie Details für den passenden Auftrag ab.
- Rufen Sie die Schemazuordnung ab.
- Listen Sie alle Schemazuordnungen auf.
- Markieren Sie die Schemazuordnungsressource.
- Löschen Sie die AWS Entity Resolution Assets.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Entity Resolution Funktionen demonstriert.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" + UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();

    public static void main(String[] args) throws InterruptedException {
```

```
        logger.info("Welcome to the AWS Entity Resolution Scenario.");
        logger.info("""
            AWS Entity Resolution is a fully-managed machine learning service
provided by
            Amazon Web Services (AWS) that helps organizations extract, link, and
            organize information from multiple data sources. It leverages natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer 360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application, the
next step uploads
```

```
            a CloudFormation template whose resulting stack creates the following
resources:
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
```

```
            """;
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
```

```
            consolidated, accurate view for improved data quality and decision-
making.

            In this example, the schema mapping lines up with the fields in the JSON
 and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " + jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
 ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
 cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping: {}",
 cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
 actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " + csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
 ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
```

```
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
```

```
                logger.error("Workflow conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other API
methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}", cause.getMessage(),
cause);
            } else {
```

```
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
```

```
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged.
For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running state.
            In order to view the results, you need to wait for the workflow that we
started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can perform

            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName, jobId).join();
                logger.info("Number of input records: {}", metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
```

```
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
                logger.info("The following represents the output data generated by
the Entity Resolution workflow based on the JSON and CSV input data. The output
data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match even
though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the workflow?
(y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.
```

```
            You can delete the workflow manually in the AWS Management Console at a
later time.

            If you already waited for the workflow to complete in the previous
step,
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}", ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
```

```
                return;
            }

        } else {
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
```

```
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;

            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
 actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
 "SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
 state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: " +
 e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

Eine Wrapper-Klasse für AWS Entity Resolution SDK-Methoden.

```
public class EntityResActions {
```

```
   private static final String PREFIX = "eroutput/";
   private static final Logger logger =
LoggerFactory.getLogger(EntityResActions.class);

   private static EntityResolutionAsyncClient entityResolutionAsyncClient;

   private static S3AsyncClient s3AsyncClient;

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
```

```
        return entityResolutionAsyncClient;
    }

    public static S3AsyncClient getS3AsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
```

```
    * @param schemaName the name of the schema to delete
    * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
               }
           });
   }

   /**
    * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
    * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
    */
```

```
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }
```

```
        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();
```

```
        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
```

```
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                 the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName            the name of the workflow to be created
     * @param outputBucket            the S3 bucket path where the workflow output
will be stored
     * @param jsonGlueTableArn        the ARN of the Glue Data Catalog table to be
used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {
```

```
        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
```

```
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id was
not found.", cause);
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
```

```
     * Uploads data to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the data to
     * @param jsonData   the JSON data to be uploaded
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous operation
of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

    public void uploadInputData(String bucketName, String jsonData, String csvData)
{
        // Upload JSON data.
        String jsonKey = "jsonData/data.json";
        PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(jsonKey)
            .contentType("application/json")
            .build();

        CompletableFuture<PutObjectResponse> jsonUploadResponse =
getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

        // Upload CSV data.
        String csvKey = "csvData/data.csv";
        PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(csvKey)
            .contentType("text/csv")
            .build();
        CompletableFuture<PutObjectResponse> csvUploadResponse =
getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

        CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    // Wrap an AWS exception.
                    throw new CompletionException("Failed to upload files", ex);
                }
            }).join();

    }
```

```
    /**
     * Finds the latest file in the S3 bucket that starts with "run-" in any depth
of subfolders
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with "run-"
in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }
```

```
            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String bucketName,
String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from S3.");
                }
            });
    }
```

```
    /**
     * Parses CSV content from a String into a list of records.
     */
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }

    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ? row[columnIndex].length() :
0)
                .max(Integer::compareTo)
                .orElse(0));
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for better
readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3 ===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
```

```
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
    }

    private static void printRow(String[] row, int[] columnWidths, boolean isHeader)
 {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateMatchingWorkflow
  - CreateSchemaMapping
  - DeleteMatchingWorkflow
  - DeleteSchemaMapping
  - GetMatchingJob
  - GetSchemaMapping
  - ListMatchingWorkflows

- ListSchemaMappings
- StartMatchingJob
- TagResource

Aktionen

`CreateMatchingWorkflow`

Das folgende Codebeispiel zeigt die Verwendung`CreateMatchingWorkflow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                 the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName            the name of the workflow to be created
    * @param outputBucket            the S3 bucket path where the workflow output
will be stored
    * @param jsonGlueTableArn        the ARN of the Glue Data Catalog table to be
used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
```

```
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
```

```
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- API-Details finden Sie CreateMatchingWorkflow in der AWS SDK for Java 2.x API-Referenz.

`CreateSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchemaMapping`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to create
    * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
    */
   public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
       List<SchemaInputAttribute> schemaAttributes = null;
       if (schemaName.startsWith("json")) {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
           );
       } else {
           schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
           );
       }

       CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .mappedInputFields(schemaAttributes)
           .build();

       return getResolutionAsyncClient().createSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("[{}] schema mapping Created Successfully!",
schemaName);
               } else {
```

```
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie CreateSchemaMapping in der AWS SDK for Java 2.x API-Referenz.

`DeleteMatchingWorkflow`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMatchingWorkflow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
```

```
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The workflow to delete was
not found.", cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
               }
           });
   }
```

- API-Details finden Sie DeleteMatchingWorkflow in der AWS SDK for Java 2.x API-Referenz.

`DeleteSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSchemaMapping`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to delete
    * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }
```

```
                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }
```

- API-Details finden Sie DeleteSchemaMapping in der AWS SDK for Java 2.x API-Referenz.

GetMatchingJob

Das folgende Codebeispiel zeigt die VerwendungGetMatchingJob.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    // Successfully fetched the matching job details, log the job
status.
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie GetMatchingJob in der AWS SDK for Java 2.x API-Referenz.

`GetSchemaMapping`

Das folgende Codebeispiel zeigt die Verwendung`GetSchemaMapping`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the schema mapping asynchronously.
```

```
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
       GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   response.mappedInputFields().forEach(attribute ->
                       logger.info("Attribute Name: " + attribute.fieldName() +
                           ", Attribute Type: " + attribute.type().toString()));
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested schema mapping
was not found.", cause);
                   }

                   // Wrap other exceptions in a CompletionException with the
message.
                   throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
               }
           });
   }
```

- API-Details finden Sie GetSchemaMapping in der AWS SDK for Java 2.x API-Referenz.

`ListSchemaMappings`

Das folgende Codebeispiel zeigt die Verwendung`ListSchemaMappings`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
    * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
    */
   public void ListSchemaMappings() {
       ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
           .build();

       ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

       // Iterate through the pages of results
       CompletableFuture<Void> future = paginator.subscribe(response -> {
           response.schemaList().forEach(schemaMapping ->
               logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
           );
       });

       // Wait for the asynchronous operation to complete
       future.join();
   }
```

- API-Details finden Sie ListSchemaMappings in der AWS SDK for Java 2.x API-Referenz.

StartMatchingJob

Das folgende Codebeispiel zeigt die VerwendungStartMatchingJob.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts a matching job asynchronously for the specified workflow name.
    *
    * @param workflowName the name of the workflow for which to start the matching
job
    * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
    * string if the operation fails
    */
   public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
       StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().startMatchingJob(jobRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   String jobId = response.jobId();
                   logger.info("Job ID: " + jobId);
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while starting the job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                   }
```

```
                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- API-Details finden Sie StartMatchingJob in der AWS SDK for Java 2.x API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
```

```
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie TagResource in der AWS SDK for Java 2.x API-Referenz.

OpenSearch Servicebeispiele, die SDK für Java 2.x verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with OpenSearch Service Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo OpenSearch Service

Das folgende Codebeispiel zeigt, wie Sie mit OpenSearch Service beginnen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
```

```
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- API-Details finden Sie ListVersions in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Lernen Sie die OpenSearch Kernfunktionen des Service kennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine OpenSearch Service-Domain.
- Stellt detaillierte Informationen zu einer bestimmten OpenSearch Servicedomäne bereit.
- Listet alle OpenSearch Servicedomänen auf, die dem Konto gehören.
- Wartet, bis der Änderungsstatus der OpenSearch Servicedomäne den Status „Abgeschlossen“ erreicht.
- Ändert die Konfiguration einer vorhandenen OpenSearch Servicedomäne.
- Fügt der OpenSearch Service-Domäne ein Tag hinzu.
- Listet die Tags auf, die einer OpenSearch Service-Domäne zugeordnet sind.
- Entfernt Tags aus einer OpenSearch Service-Domain.
- Löscht die OpenSearch Servicedomäne.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die OpenSearch Servicefunktionen demonstriert.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.

            Use the Amazon OpenSearch Service API to create, configure, and manage
 OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are focused
 on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/latest/
clients/java/).
```

```
            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources and
storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.
```

```
            In this step, we'll initiate the creation of the domain. We'll check on
the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
```

```
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
```

```
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is creating
a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling the
number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource was not found: Error message:
{}, Error code {}", resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change, allowing
you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: " +ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
            associate with an OpenSearch Service domain. You can use these tags to
track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
```

```
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: " +
cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we created
in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
```

```
    }
 }
```

Eine Wrapper-Klasse für OpenSearch Service SDK-Methoden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
```

```
    private static final Logger logger =
LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();
```

```
        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
```

```
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }

    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }
```

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
```

```
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
```

```
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }


    /**
     * Asynchronously lists the tags associated with the specified Amazon Resource
Name (ARN).
     * @param arn the Amazon Resource Name (ARN) of the resource for which to list
the tags
     * @return a {@link CompletableFuture} that, when completed, will contain a list
of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list domain tags",
exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

Aktionen

`AddTags`

Das folgende Codebeispiel zeigt die Verwendung`AddTags`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
```

```
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- API-Details finden Sie AddTags in der AWS SDK for Java 2.x API-Referenz.

`ChangeProgress`

Das folgende Codebeispiel zeigt die Verwendung`ChangeProgress`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
    * @param domainName the name of the OpenSearch domain to check the progress for
    * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
    */
   public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
       DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
           .domainName(domainName)
           .build();

       return CompletableFuture.runAsync(() -> {
           boolean isCompleted = false;
           long startTime = System.currentTimeMillis();

           while (!isCompleted) {
               try {
                   // Handle the async client call using `join` to block
synchronously for the result
                   DescribeDomainChangeProgressResponse response = getAsyncClient()
                       .describeDomainChangeProgress(request)
                       .handle((resp, ex) -> {
                           if (ex != null) {
                               throw new RuntimeException("Failed to check domain
progress", ex);
                           }
                           return resp;
                       }).join();

                   String state = response.changeProgressStatus().statusAsString();
 // Get the status as string
```

```
                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- API-Details finden Sie ChangeProgress in der AWS SDK for Java 2.x API-Referenz.

`CreateDomain`

Das folgende Codebeispiel zeigt die Verwendung`CreateDomain`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
```

```
     * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }
```

- API-Details finden Sie CreateDomain in der AWS SDK for Java 2.x API-Referenz.

`DeleteDomain`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDomain`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a specific domain asynchronously.
    * @param domainName the name of the domain to be deleted
    * @return a {@link CompletableFuture} that completes when the domain has been
deleted
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
       DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
           .domainName(domainName)
           .build();

       // Delete domain asynchronously
       return getAsyncClient().deleteDomain(domainRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
               }
           });
   }
```

- API-Details finden Sie DeleteDomain in der AWS SDK for Java 2.x API-Referenz.

`DescribeDomain`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDomain`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }
```

- API-Details finden Sie DescribeDomain in der AWS SDK for Java 2.x API-Referenz.

`ListDomainNames`

Das folgende Codebeispiel zeigt die Verwendung`ListDomainNames`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously lists all the domains in the current AWS account.
    * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
    *         the domains in the account.
    * @throws RuntimeException if there was a failure while listing the domains.
    */
   public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
       ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
           .engineType("OpenSearch")
           .build();

       return getAsyncClient().listDomainNames(namesRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to list all domains",
exception);
               }
               return response.domainNames();  // Return the list of domain names
on success
           });
   }
```

- API-Details finden Sie ListDomainNames in der AWS SDK for Java 2.x API-Referenz.

`ListTags`

Das folgende Codebeispiel zeigt die Verwendung`ListTags`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- API-Details finden Sie ListTags in der AWS SDK for Java 2.x API-Referenz.

`UpdateDomainConfig`

Das folgende Codebeispiel zeigt die Verwendung`UpdateDomainConfig`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();
```

```
        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- API-Details finden Sie UpdateDomainConfig in der AWS SDK for Java 2.x API-Referenz.

EventBridge Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hallo EventBridge

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von EventBridge beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class HelloEventBridge {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        listBuses(eventBrClient);
        eventBrClient.close();
    }

    public static void listBuses(EventBridgeClient eventBrClient) {
        try {
            ListEventBusesRequest busesRequest = ListEventBusesRequest.builder()
                    .limit(10)
                    .build();

            ListEventBusesResponse response =
 eventBrClient.listEventBuses(busesRequest);
            List<EventBus> buses = response.eventBuses();
```

```
            for (EventBus bus : buses) {
                System.out.println("The name of the event bus is: " + bus.name());
                System.out.println("The ARN of the event bus is: " + bus.arn());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListEventBuses in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Regel und fügen Sie ihr ein Ziel hinzu.
- Aktivieren und deaktivieren Sie Regeln.
- Listen Sie Regeln und Ziele auf und aktualisieren Sie sie.
- Senden Sie Ereignisse und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
 {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
 bucket name to create.
```

```
                    topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
                .region(region)
                .build();
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
 the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to the
 S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
 pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
 trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
```

```
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
```

```
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
```

```
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

```
    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }

    public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
            String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
```

```
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
```

```
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();
```

```
        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
```

```
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";
```

```
        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();
```

```
            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents

- PutRule
- PutTargets

Aktionen

`DeleteRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
       DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
               .name(ruleName)
               .build();

       eventBrClient.deleteRule(ruleRequest);
       System.out.println("Successfully deleted the rule");
   }
```

- API-Details finden Sie DeleteRule in der AWS SDK for Java 2.x API-Referenz.

`DescribeRule`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       try {
           DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                   .name(eventRuleName)
                   .build();

           DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
           System.out.println("The state of the rule is " +
response.stateAsString());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeRule in der AWS SDK for Java 2.x API-Referenz.

`DisableRule`

Das folgende Codebeispiel zeigt die Verwendung`DisableRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Deaktivieren Sie eine Regel anhand ihres Regelnamens.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DisableRule in der AWS SDK for Java 2.x API-Referenz.

`EnableRule`

Das folgende Codebeispiel zeigt die Verwendung`EnableRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie EnableRule in der AWS SDK for Java 2.x API-Referenz.

`ListRuleNamesByTarget`

Das folgende Codebeispiel zeigt die Verwendung`ListRuleNamesByTarget`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Regelnamen mithilfe des Ziels auf.

```
   public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
       ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
               .targetArn(topicArn)
               .build();

       ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
       List<String> rules = response.ruleNames();
       for (String rule : rules) {
           System.out.println("The rule name is " + rule);
       }
   }
```

- API-Details finden Sie ListRuleNamesByTarget in der AWS SDK for Java 2.x API-Referenz.

`ListRules`

Das folgende Codebeispiel zeigt die Verwendung`ListRules`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktivieren Sie eine Regel anhand ihres Regelnamens.

```
   public static void listRules(EventBridgeClient eventBrClient) {
       try {
           ListRulesRequest rulesRequest = ListRulesRequest.builder()
                   .eventBusName("default")
                   .limit(10)
                   .build();

           ListRulesResponse response = eventBrClient.listRules(rulesRequest);
           List<Rule> rules = response.rules();
```

```
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListRules in der AWS SDK for Java 2.x API-Referenz.

`ListTargetsByRule`

Das folgende Codebeispiel zeigt die Verwendung`ListTargetsByRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Ziele für eine Regel mithilfe des Regelnamens auf.

```
    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
```

```
        }
    }
```

- API-Details finden Sie ListTargetsByRule in der AWS SDK for Java 2.x API-Referenz.

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
        String json = "{" +
                "\"UserEmail\": \"" + email + "\"," +
                "\"Message\": \"This event was generated by example code.\"," +
                "\"UtcTime\": \"Now.\"" +
                "}";

        PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
                .source("ExampleSource")
                .detail(json)
                .detailType("ExampleType")
                .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
                .entries(entry)
                .build();

        eventBrClient.putEvents(eventsRequest);
    }
```

- API-Details finden Sie PutEvents in der AWS SDK for Java 2.x API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
   public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {
       try {
           PutRuleRequest ruleRequest = PutRuleRequest.builder()
                   .name(ruleName)
                   .eventBusName("default")
                   .scheduleExpression(cronExpression)
                   .state("ENABLED")
                   .description("A test rule that runs on a schedule created by the
Java API")
                   .build();

           PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
           System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

       } catch (EventBridgeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Amazon-Simple-Storage-Service-Bucket hinzugefügt wird.

```
   // Create a new event rule that triggers when an Amazon S3 object is created in
```

```
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";

        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie PutRule in der AWS SDK for Java 2.x API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie ein Amazon-SNS-Thema als Ziel für eine Regel hinzu.

```
   // Add a rule which triggers an SNS target when a file is uploaded to an S3
   // bucket.
   public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
           String topicName, String eventRuleName, String bucketName) {
       String targetID = java.util.UUID.randomUUID().toString();
       Target myTarget = Target.builder()
               .id(targetID)
               .arn(topicArn)
               .build();

       List<Target> targets = new ArrayList<>();
       targets.add(myTarget);
       PutTargetsRequest request = PutTargetsRequest.builder()
               .eventBusName(null)
               .targets(targets)
               .rule(ruleName)
               .build();

       eventBrClient.putTargets(request);
       System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
               + bucketName + ".");
   }
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
   public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
           String ruleName) {
       String targetId = java.util.UUID.randomUUID().toString();
```

```
        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: sample event was received.\"")
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie PutTargets in der AWS SDK for Java 2.x API-Referenz.

`RemoveTargets`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTargets`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Entfernen Sie alle Ziele für eine Regel mithilfe des Regelnamens.

```
   public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
       // First, get all targets that will be deleted.
       ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
               .rule(eventRuleName)
               .build();

       ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
       List<Target> allTargets = response.targets();

       // Get all targets and delete them.
       for (Target myTarget : allTargets) {
           RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                   .rule(eventRuleName)
                   .ids(myTarget.id())
                   .build();

           eventBrClient.removeTargets(removeTargetsRequest);
           System.out.println("Successfully removed the target");
       }
   }
```

- API-Details finden Sie RemoveTargets in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Senden Sie Event-Benachrichtigungen an EventBridge

Das folgende Codebeispiel zeigt, wie ein Bucket aktiviert wird, um S3-Ereignisbenachrichtigungen an ein Amazon SNS-Thema EventBridge und eine Amazon SQS-Warteschlange zu senden und Benachrichtigungen an diese weiterzuleiten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
```

```
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge Scheduler-Beispiele, die SDK für Java 2.x verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von EventBridge Scheduler Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo Scheduler EventBridge

Das folgende Codebeispiel zeigt, wie Sie mit EventBridge Scheduler beginnen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
```

```
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient = SchedulerAsyncClient.create();

        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
        });

        // Wait for the asynchronous operation to complete.
        future.join();

        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n", results.size());
    }
}
```

- API-Details finden Sie ListSchedules in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`CreateSchedule`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchedule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new schedule for a target task.
    *
    * @param name                  the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
```

```
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
```

```
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }
```

- API-Details finden Sie CreateSchedule in der AWS SDK for Java 2.x API-Referenz.

`CreateScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateScheduleGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
```

```
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

- API-Details finden Sie CreateScheduleGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteSchedule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSchedule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
```

```
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

- API-Details finden Sie DeleteSchedule in der AWS SDK for Java 2.x API-Referenz.

`DeleteScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteScheduleGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- API-Details finden Sie DeleteScheduleGroup in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Geplante Ereignisse

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Stellen Sie einen CloudFormation Stack mit den erforderlichen Ressourcen bereit.

- Erstellen Sie eine EventBridge Scheduler-Zeitplangruppe.
- Erstellen Sie einen einmaligen EventBridge Scheduler-Zeitplan mit einem flexiblen Zeitfenster.
- Erstellen Sie einen wiederkehrenden EventBridge Scheduler-Zeitplan mit einer bestimmten Rate.
- Löschen Sie EventBridge Scheduler, den Zeitplan und die Zeitplangruppe.
- Bereinigen Sie die Ressourcen und löschen Sie den Stack.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie das Szenario aus.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for resource
 creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
```

```
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean up
 all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
 EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
```

```
            Amazon EventBridge Scheduler is a fully managed service that helps you
schedule and execute
            a wide range of tasks and events in the cloud. It's designed to simplify
the process of
            scheduling and managing recurring or one-time events, making it easier
for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its ability
to schedule events
            based on a variety of triggers, including time-based schedules, custom
event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or to
trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
                    A one-time schedule in Amazon EventBridge Scheduler is an event
trigger that allows
                    you to schedule a one-time event to run at a specific date and
time. This is useful for
                    executing a specific task or workflow at a predetermined time,
without the need for recurring
                    or complex scheduling.
```

```
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules for
your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources, enabling
you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
```

```
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part of
that group are deleted.");
        waitForInputToContinue();
        try {
            eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }

        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
```

```
    * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
    * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
    * in order to receive event emails.
    *
    * @return true if the application preparation was successful, false otherwise
    */
   public static boolean prepareApplication() {
       logger.info("""
           This example creates resources in a CloudFormation stack, including an
SNS topic
           that will be subscribed to the EventBridge Scheduler events.
           You will need to confirm the subscription in order to receive event
emails.
            """);

       String emailAddress = promptUserForEmail();
       logger.info("You entered {}", emailAddress);

       logger.info("Do you want to use a custom Stack name (y/n) ?");
       String ans = scanner.nextLine().trim();
       if (ans.equalsIgnoreCase("y")) {
           String newStackName = scanner.nextLine();
           logger.info("You entered {} for the new stack name", newStackName);
           waitForInputToContinue();
           STACK_NAME = newStackName;
       }

       logger.info("Get the roleArn and snsTopicArn values using a Cloudformation
template.");
       waitForInputToContinue();
       CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
       Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
       roleArn = stackOutputs.get("RoleARN");
       snsTopicArn = stackOutputs.get("SNStopicARN");

       logger.info("The roleARN is {}", roleArn);
       logger.info("The snsTopicArn is {}", snsTopicArn);

       try {
           eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
           logger.info("createScheduleGroupAsync completed successfully.");
```

```
        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
     * If the provided email address is invalid, the method will prompt the user to
try again.
     *
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
```

```
            return promptUserForEmail();
        }
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false} otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
 flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code false}
 otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the one-
time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial event
 in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) + ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
```

```
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired schedule
rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + " minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
```

```
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches the
 pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code false}
 otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

Wrapper für Service-Vorgänge.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
```

```
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           schedulerClient = SchedulerAsyncClient.builder()
               .region(Region.US_EAST_1)
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return schedulerClient;
   }


   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();
```

```
        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }


    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
```

```
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();
```

```
        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
```

```
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

Forecast-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Forecast Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDataset`

Das folgende Codebeispiel zeigt die Verwendung`CreateDataset`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateDatasetRequest;
import software.amazon.awssdk.services.forecast.model.Schema;
import software.amazon.awssdk.services.forecast.model.SchemaAttribute;
import software.amazon.awssdk.services.forecast.model.CreateDatasetResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataSet {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <name>\s

                Where:
                    name - The name of the data set.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String myDataSetARN = createForecastDataSet(forecast, name);
        System.out.println("The ARN of the new data set is " + myDataSetARN);
        forecast.close();
```

```
    }

    public static String createForecastDataSet(ForecastClient forecast, String name)
{
        try {
            Schema schema = Schema.builder()
                    .attributes(getSchema())
                    .build();

            CreateDatasetRequest datasetRequest = CreateDatasetRequest.builder()
                    .datasetName(name)
                    .domain("CUSTOM")
                    .datasetType("RELATED_TIME_SERIES")
                    .dataFrequency("D")
                    .schema(schema)
                    .build();

            CreateDatasetResponse response = forecast.createDataset(datasetRequest);
            return response.datasetArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }

    // Create a SchemaAttribute list required to create a data set.
    private static List<SchemaAttribute> getSchema() {

        List<SchemaAttribute> schemaList = new ArrayList<>();
        SchemaAttribute att1 = SchemaAttribute.builder()
                .attributeName("item_id")
                .attributeType("string")
                .build();

        SchemaAttribute att2 = SchemaAttribute.builder()
                .attributeName("timestamp")
                .attributeType("timestamp")
                .build();

        SchemaAttribute att3 = SchemaAttribute.builder()
                .attributeName("target_value")
```

```
                .attributeType("float")
                .build();

        // Push the SchemaAttribute objects to the List.
        schemaList.add(att1);
        schemaList.add(att2);
        schemaList.add(att3);
        return schemaList;
    }
}
```

- API-Details finden Sie CreateDataset in der AWS SDK for Java 2.x API-Referenz.

`CreateForecast`

Das folgende Codebeispiel zeigt die Verwendung`CreateForecast`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.CreateForecastRequest;
import software.amazon.awssdk.services.forecast.model.CreateForecastResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateForecast {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <name> <predictorArn>\s

                Where:
                    name - The name of the forecast.\s
                    predictorArn - The arn of the predictor to use.\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String predictorArn = args[1];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        String forecastArn = createNewForecast(forecast, name, predictorArn);
        System.out.println("The ARN of the new forecast is " + forecastArn);
        forecast.close();
    }

    public static String createNewForecast(ForecastClient forecast, String name,
String predictorArn) {
        try {
            CreateForecastRequest forecastRequest = CreateForecastRequest.builder()
                    .forecastName(name)
                    .predictorArn(predictorArn)
                    .build();

            CreateForecastResponse response =
forecast.createForecast(forecastRequest);
            return response.forecastArn();

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }
}
```

- API-Details finden Sie CreateForecast in der AWS SDK for Java 2.x API-Referenz.

`DeleteDataset`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDataset`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
```

```
                    datasetARN - The ARN of the data set to delete.\s
                """";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteDataset in der AWS SDK for Java 2.x API-Referenz.

`DeleteForecast`

Das folgende Codebeispiel zeigt die Verwendung`DeleteForecast`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataset {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <datasetARN>\s

                Where:
                    datasetARN - The ARN of the data set to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String datasetARN = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
```

```
                .build();

        deleteForecastDataSet(forecast, datasetARN);
        forecast.close();
    }

    public static void deleteForecastDataSet(ForecastClient forecast, String
 myDataSetARN) {
        try {
            DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                    .datasetArn(myDataSetARN)
                    .build();

            forecast.deleteDataset(deleteRequest);
            System.out.println("The Data Set was deleted");

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteForecast in der AWS SDK for Java 2.x API-Referenz.

`DescribeForecast`

Das folgende Codebeispiel zeigt die Verwendung`DescribeForecast`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DescribeForecastRequest;
import software.amazon.awssdk.services.forecast.model.DescribeForecastResponse;
```

```
import software.amazon.awssdk.services.forecast.model.ForecastException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeForecast {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <forecastarn>\s

                Where:
                    forecastarn - The arn of the forecast (for example,
 "arn:aws:forecast:us-west-2:xxxxx322:forecast/my_forecast)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String forecastarn = args[0];
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        describe(forecast, forecastarn);
        forecast.close();
    }

    public static void describe(ForecastClient forecast, String forecastarn) {
        try {
            DescribeForecastRequest request = DescribeForecastRequest.builder()
                    .forecastArn(forecastarn)
                    .build();

            DescribeForecastResponse response = forecast.describeForecast(request);
```

```
            System.out.println("The name of the forecast is " +
 response.forecastName());

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeForecast in der AWS SDK for Java 2.x API-Referenz.

`ListDatasetGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListDatasetGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.DatasetGroupSummary;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsRequest;
import software.amazon.awssdk.services.forecast.model.ListDatasetGroupsResponse;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListDataSetGroups {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listDataGroups(forecast);
        forecast.close();
    }

    public static void listDataGroups(ForecastClient forecast) {
        try {
            ListDatasetGroupsRequest group = ListDatasetGroupsRequest.builder()
                    .maxResults(10)
                    .build();

            ListDatasetGroupsResponse response = forecast.listDatasetGroups(group);
            List<DatasetGroupSummary> groups = response.datasetGroups();
            for (DatasetGroupSummary myGroup : groups) {
                System.out.println("The Data Set name is " +
 myGroup.datasetGroupName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListDatasetGroups in der AWS SDK for Java 2.x API-Referenz.

`ListForecasts`

Das folgende Codebeispiel zeigt die Verwendung`ListForecasts`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.forecast.ForecastClient;
import software.amazon.awssdk.services.forecast.model.ListForecastsResponse;
import software.amazon.awssdk.services.forecast.model.ListForecastsRequest;
import software.amazon.awssdk.services.forecast.model.ForecastSummary;
import software.amazon.awssdk.services.forecast.model.ForecastException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListForecasts {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        ForecastClient forecast = ForecastClient.builder()
                .region(region)
                .build();

        listAllForeCasts(forecast);
        forecast.close();
    }

    public static void listAllForeCasts(ForecastClient forecast) {
        try {
            ListForecastsRequest request = ListForecastsRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListForecastsResponse response = forecast.listForecasts(request);
            List<ForecastSummary> forecasts = response.forecasts();
            for (ForecastSummary forecastSummary : forecasts) {
                System.out.println("The name of the forecast is " +
 forecastSummary.forecastName());
            }

        } catch (ForecastException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListForecasts in der AWS SDK for Java 2.x API-Referenz.

Amazon Glacier-Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Glacier Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVault`

Das folgende Codebeispiel zeigt die Verwendung`CreateVault`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateVault {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateVault in der AWS SDK for Java 2.x API-Referenz.

`DeleteArchive`

Das folgende Codebeispiel zeigt die Verwendung`DeleteArchive`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
 String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
```

```
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();

            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteArchive in der AWS SDK for Java 2.x API-Referenz.

`DeleteVault`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVault`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteVault in der AWS SDK for Java 2.x API-Referenz.

`InitiateJob`

Das folgende Codebeispiel zeigt die Verwendung`InitiateJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Tresorinventur ab.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {
        try {
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
response.location());
            return response.jobId();
```

```
        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
 String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();
```

```
            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);

        }
    }
}
```

- API-Details finden Sie InitiateJob in der AWS SDK for Java 2.x API-Referenz.

`ListVaults`

Das folgende Codebeispiel zeigt die Verwendung`ListVaults`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }

                // Check for further results.
```

```
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListVaults in der AWS SDK for Java 2.x API-Referenz.

`UploadArchive`

Das folgende Codebeispiel zeigt die Verwendung`UploadArchive`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s

                Where:
                   strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                   vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }
```

```
    public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
            return res.archiveId();

        } catch (GlacierException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
```

```
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
```

```
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }
```

```
    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- API-Details finden Sie UploadArchive in der AWS SDK for Java 2.x API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
    }
}
```

- API-Details finden Sie ListJobs in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri> <bucketNameSc>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3 permissions.
\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, s3://<bucket name>/read).
                cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
                locationUri - The location of the database (you can find this file
 in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;
```

```
        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL) service
provided by Amazon
            Web Services (AWS). It is designed to simplify the process of building,
running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known as
the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists. Skipping
creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
```

```
   public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
       try {
           DatabaseInput input = DatabaseInput.builder()
               .description("Built with the AWS SDK for Java V2")
               .name(dbName)
               .locationUri(locationUri)
               .build();

           CreateDatabaseRequest request = CreateDatabaseRequest.builder()
               .databaseInput(input)
               .build();

           glueClient.createDatabase(request);
           System.out.println(dbName + " was successfully created");

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the data
source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                        String iam,
                                        String s3Path,
                                        String cron,
                                        String dbName,
                                        String crawlerName) {

       try {
           S3Target s3Target = S3Target.builder()
```

```
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
 in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
 throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
```

```
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
     */
```

```
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " + createDate);

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
           List<Table> tables = response.tableList();
           if (tables.isEmpty()) {
```

```
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());
```

```
        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
        }
    }
```

```
    /**
     * Retrieves and prints information about all the jobs in the Glue data catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
```

```
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
```

```
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
 the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
```

```
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the data
source
     * @param s3Path      the S3 path that the crawler will scan for data
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
```

```
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API-Details finden Sie CreateCrawler in der AWS SDK for Java 2.x API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
```

```
        }
    }
```

- API-Details finden Sie CreateJob in der AWS SDK for Java 2.x API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API-Details finden Sie DeleteCrawler in der AWS SDK for Java 2.x API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- API-Details finden Sie DeleteDatabase in der AWS SDK for Java 2.x API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a Glue job.
 *
 * @param glueClient the Glue client to use for the operation
 * @param jobName    the name of the job to be deleted
 * @throws GlueException if there is an error deleting the job
 */
public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        throw e;
    }
}
```

- API-Details finden Sie DeleteJob in der AWS SDK for Java 2.x API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
    *
    * @param glueClient  the AWS Glue client used to interact with the Glue service
    * @param crawlerName the name of the crawler to be retrieved
    */
   public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
       try {
           GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           boolean ready = false;
           while (!ready) {
               GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
               String status = response.crawler().stateAsString();
               if (status.compareTo("READY") == 0) {
                   ready = true;
               }
               Thread.sleep(3000);
           }

           System.out.println("The crawler is now ready");

       } catch (GlueException | InterruptedException e) {
           throw e;
       }
   }
```

- API-Details finden Sie GetCrawler in der AWS SDK for Java 2.x API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact with
the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " + createDate);

       } catch (GlueException e) {
           throw e;
       }
```

```
    }
```

- API-Details finden Sie GetDatabase in der AWS SDK for Java 2.x API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
```

```
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- API-Details finden Sie GetJobRuns in der AWS SDK for Java 2.x API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the names of the tables in the specified Glue database.
     *
     * @param glueClient the Glue client to use for the operation
     * @param dbName     the name of the Glue database to retrieve the table names
from
     * @return the name of the first table retrieved, or an empty string if no
tables were found
     */
    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                .databaseName(dbName)
                .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }
```

- API-Details finden Sie GetTables in der AWS SDK for Java 2.x API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client to use for the crawler operation
    * @param crawlerName the name of the crawler to start
    * @throws GlueException if there is an error starting the crawler
    */
   public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
       try {
           StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           glueClient.startCrawler(crawlerRequest);
           System.out.println(crawlerName + " was successfully started!");

       } catch (GlueException e) {
           throw e;
       }
   }
```

- API-Details finden Sie StartCrawler in der AWS SDK for Java 2.x API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                               String outBucket) {
       try {
           Map<String, String> myMap = new HashMap<>();
           myMap.put("--input_database", inputDatabase);
           myMap.put("--input_table", inputTable);
           myMap.put("--output_bucket_url", outBucket);

           StartJobRunRequest runRequest = StartJobRunRequest.builder()
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .arguments(myMap)
               .jobName(jobName)
               .build();

           StartJobRunResponse response = glueClient.startJobRun(runRequest);
           System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

       } catch (GlueException e) {
           throw e;
       }
```

```
    }
```

- API-Details finden Sie StartJobRun in der AWS SDK for Java 2.x API-Referenz.

HealthImaging Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CopyImageSet`

Das folgende Codebeispiel zeigt die Verwendung`CopyImageSet`.

SDK für Java 2.x

```
    /**
     * Copy an AWS HealthImaging image set.
     *
     * @param medicalImagingClient  - The AWS HealthImaging client object.
     * @param datastoreId           - The datastore ID.
     * @param imageSetId            - The image set ID.
     * @param latestVersionId       - The version ID.
```

```
    * @param destinationImageSetId - The optional destination image set ID, ignored
if null.
    * @param destinationVersionId  - The optional destination version ID, ignored
if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
               String subsetInstanceToCopy = getCopiableAttributesJSON(imageSetId,
subsets);
               copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                       .copiableAttributes(subsetInstanceToCopy)
                       .build());
           }

           CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                   .sourceImageSet(copySourceImageSetInformation.build());

           // Optionally designate a destination image set.
           if (destinationImageSetId != null) {
               copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                       .imageSetId(destinationImageSetId)
                       .latestVersionId(destinationVersionId)
```

```
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Hilfsfunktion zum Erstellen kopierbarer Attribute.

```
    /**
     * Create a JSON string of copiable image instances.
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                                "
                                 """
        );
```

```
        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
                                ": {
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                         }
                       }
                     }
                   }
                 }
                 """);
        return subsetInstanceToCopy.toString();
    }
```

- API-Details finden Sie CopyImageSet in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`CreateDatastore`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatastore`.

SDK für Java 2.x

```
    public static String createMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
            String datastoreName) {
        try {
```

```
            CreateDatastoreRequest datastoreRequest =
CreateDatastoreRequest.builder()
                    .datastoreName(datastoreName)
                    .build();
            CreateDatastoreResponse response =
medicalImagingClient.createDatastore(datastoreRequest);
            return response.datastoreId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- API-Details finden Sie CreateDatastore in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatastore`.

SDK für Java 2.x

```
    public static void deleteMedicalImagingDatastore(MedicalImagingClient
medicalImagingClient,
            String datastoreID) {
        try {
            DeleteDatastoreRequest datastoreRequest =
DeleteDatastoreRequest.builder()
                    .datastoreId(datastoreID)
                    .build();
            medicalImagingClient.deleteDatastore(datastoreRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
```

- API-Details finden Sie DeleteDatastore in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteImageSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteImageSet`.

SDK für Java 2.x

```
   public static void deleteMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId) {
       try {
           DeleteImageSetRequest deleteImageSetRequest =
DeleteImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId)
                   .build();

           medicalImagingClient.deleteImageSet(deleteImageSetRequest);

           System.out.println("The image set was deleted.");
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DeleteImageSet in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`GetDICOMImportJob`.

SDK für Java 2.x

```
   public static DICOMImportJobProperties getDicomImportJob(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String jobId) {

       try {
           GetDicomImportJobRequest getDicomImportJobRequest =
GetDicomImportJobRequest.builder()
                   .datastoreId(datastoreId)
                   .jobId(jobId)
                   .build();
           GetDicomImportJobResponse response =
medicalImagingClient.getDICOMImportJob(getDicomImportJobRequest);
           return response.jobProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- API-Details finden Sie GetDICOMImportJob in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDatastore`

Das folgende Codebeispiel zeigt die Verwendung`GetDatastore`.

SDK für Java 2.x

```
   public static DatastoreProperties getMedicalImageDatastore(MedicalImagingClient
medicalImagingClient,
           String datastoreID) {
       try {
           GetDatastoreRequest datastoreRequest = GetDatastoreRequest.builder()
                   .datastoreId(datastoreID)
                   .build();
           GetDatastoreResponse response =
medicalImagingClient.getDatastore(datastoreRequest);
           return response.datastoreProperties();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- API-Details finden Sie GetDatastore in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt die Verwendung`GetImageFrame`.

SDK für Java 2.x

```
       public static void getMedicalImageSetFrame(MedicalImagingClient
medicalImagingClient,
                       String destinationPath,
```

```
                        String datastoreId,
                        String imagesetId,
                        String imageFrameId) {

                try {
                        GetImageFrameRequest getImageSetMetadataRequest =
GetImageFrameRequest.builder()
                                        .datastoreId(datastoreId)
                                        .imageSetId(imagesetId)

.imageFrameInformation(ImageFrameInformation.builder()
                                                        .imageFrameId(imageFrameId)
                                                        .build())
                                        .build();

medicalImagingClient.getImageFrame(getImageSetMetadataRequest,

FileSystems.getDefault().getPath(destinationPath));

                        System.out.println("Image frame downloaded to " +
destinationPath);
                } catch (MedicalImagingException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
```

- API-Details finden Sie GetImageFrame in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSet`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSet`.

SDK für Java 2.x

```
   public static GetImageSetResponse getMedicalImageSet(MedicalImagingClient
medicalImagingClient,
           String datastoreId,
           String imagesetId,
           String versionId) {
       try {
           GetImageSetRequest.Builder getImageSetRequestBuilder =
GetImageSetRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetRequestBuilder =
getImageSetRequestBuilder.versionId(versionId);
           }

           return
medicalImagingClient.getImageSet(getImageSetRequestBuilder.build());
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- API-Details finden Sie GetImageSet in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSetMetadata`.

SDK für Java 2.x

```
   public static void getMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
           String destinationPath,
           String datastoreId,
           String imagesetId,
           String versionId) {

       try {
           GetImageSetMetadataRequest.Builder getImageSetMetadataRequestBuilder =
GetImageSetMetadataRequest.builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imagesetId);

           if (versionId != null) {
               getImageSetMetadataRequestBuilder =
getImageSetMetadataRequestBuilder.versionId(versionId);
           }


medicalImagingClient.getImageSetMetadata(getImageSetMetadataRequestBuilder.build(),
                   FileSystems.getDefault().getPath(destinationPath));

           System.out.println("Metadata downloaded to " + destinationPath);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie GetImageSetMetadata in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDICOMImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListDICOMImportJobs`.

SDK für Java 2.x

```
   public static List<DICOMImportJobSummary>
listDicomImportJobs(MedicalImagingClient medicalImagingClient,
           String datastoreId) {

       try {
           ListDicomImportJobsRequest listDicomImportJobsRequest =
ListDicomImportJobsRequest.builder()
                   .datastoreId(datastoreId)
                   .build();
           ListDicomImportJobsResponse response =
medicalImagingClient.listDICOMImportJobs(listDicomImportJobsRequest);
           return response.jobSummaries();
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return new ArrayList<>();
   }
```

- API-Details finden Sie ListDICOMImportJobs in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListDatastores`.

SDK für Java 2.x

```
   public static List<DatastoreSummary>
listMedicalImagingDatastores(MedicalImagingClient medicalImagingClient) {
       try {
           ListDatastoresRequest datastoreRequest = ListDatastoresRequest.builder()
                   .build();
           ListDatastoresIterable responses =
medicalImagingClient.listDatastoresPaginator(datastoreRequest);
           List<DatastoreSummary> datastoreSummaries = new ArrayList<>();

           responses.stream().forEach(response ->
datastoreSummaries.addAll(response.datastoreSummaries()));

           return datastoreSummaries;
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

- API-Details finden Sie ListDatastores in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListImageSetVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListImageSetVersions`.

SDK für Java 2.x

```
   public static List<ImageSetProperties>
listMedicalImageSetVersions(MedicalImagingClient medicalImagingClient,
           String datastoreId,
           String imagesetId) {
```

```
        try {
            ListImageSetVersionsRequest getImageSetRequest =
ListImageSetVersionsRequest.builder()
                    .datastoreId(datastoreId)
                    .imageSetId(imagesetId)
                    .build();

            ListImageSetVersionsIterable responses = medicalImagingClient
                    .listImageSetVersionsPaginator(getImageSetRequest);
            List<ImageSetProperties> imageSetProperties = new ArrayList<>();
            responses.stream().forEach(response ->
imageSetProperties.addAll(response.imageSetPropertiesList()));

            return imageSetProperties;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- API-Details finden Sie ListImageSetVersions in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für Java 2.x

```
    public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
```

```
                    .resourceArn(resourceArn)
                    .build();

            return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- API-Details finden Sie ListTagsForResource in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt die Verwendung`SearchImageSets`.

SDK für Java 2.x

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
    public static List<ImageSetsMetadataSummary> searchMedicalImagingImageSets(
            MedicalImagingClient medicalImagingClient,
            String datastoreId, SearchCriteria searchCriteria) {
        try {
            SearchImageSetsRequest datastoreRequest =
SearchImageSetsRequest.builder()
                    .datastoreId(datastoreId)
                    .searchCriteria(searchCriteria)
                    .build();
            SearchImageSetsIterable responses = medicalImagingClient
                    .searchImageSetsPaginator(datastoreRequest);
            List<ImageSetsMetadataSummary> imageSetsMetadataSummaries = new
ArrayList<>();
```

```
            responses.stream().forEach(response -> imageSetsMetadataSummaries
                    .addAll(response.imageSetsMetadataSummaries()));

            return imageSetsMetadataSummaries;
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

Anwendungsfall 1: EQUAL-Operator.

```
        List<SearchFilter> searchFilters =
 Collections.singletonList(SearchFilter.builder()
                .operator(Operator.EQUAL)
                .values(SearchByAttributeValue.builder()
                        .dicomPatientId(patientId)
                        .build())
                .build());

        SearchCriteria searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        List<ImageSetsMetadataSummary> imageSetsMetadataSummaries =
 searchMedicalImagingImageSets(
                medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets for patient " + patientId + " are:\n"
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudyDate und DICOMStudyTime.

```
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
        searchFilters = Collections.singletonList(SearchFilter.builder()
```

```
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate("19990101")
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build(),
                        SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
                                        .dicomStudyDate((LocalDate.now()
                                                .format(formatter)))
                                        .dicomStudyTime("000000.000")
                                        .build())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println(
                    "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
                            +
                            imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        searchFilters = Collections.singletonList(SearchFilter.builder()
                .operator(Operator.BETWEEN)
                .values(SearchByAttributeValue.builder()
                                .createdAt(Instant.parse("1985-04-12T23:20:50.52Z"))
                                .build(),
                        SearchByAttributeValue.builder()
```

```
                                .createdAt(Instant.now())
                                .build())
                .build());

        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .build();
        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with BETWEEN operator using
createdAt are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

Anwendungsfall #4: EQUAL-Operator für DICOMSeriesInstanceUID und BETWEEN für updatedAt und Sortierung der Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        Instant startDate = Instant.parse("1985-04-12T23:20:50.52Z");
        Instant endDate = Instant.now();

        searchFilters = Arrays.asList(
                SearchFilter.builder()
                        .operator(Operator.EQUAL)
                        .values(SearchByAttributeValue.builder()
                                .dicomSeriesInstanceUID(seriesInstanceUID)
                                .build())
                        .build(),
                SearchFilter.builder()
                        .operator(Operator.BETWEEN)
                        .values(

SearchByAttributeValue.builder().updatedAt(startDate).build(),

SearchByAttributeValue.builder().updatedAt(endDate).build()
                        ).build());

        Sort sort =
Sort.builder().sortOrder(SortOrder.ASC).sortField(SortField.UPDATED_AT).build();
```

```
        searchCriteria = SearchCriteria.builder()
                .filters(searchFilters)
                .sort(sort)
                .build();

        imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
                datastoreId, searchCriteria);
        if (imageSetsMetadataSummaries != null) {
            System.out.println("The image sets searched with EQUAL operator on
DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response\n" +
                    "in ASC order on updatedAt field are:\n "
                    + imageSetsMetadataSummaries);
            System.out.println();
        }
```

- API-Details finden Sie in der API-Referenz. SearchImageSets AWS SDK for Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartDICOMImportJob`.

SDK für Java 2.x

```
    public static String startDicomImportJob(MedicalImagingClient
medicalImagingClient,
            String jobName,
            String datastoreId,
            String dataAccessRoleArn,
            String inputS3Uri,
            String outputS3Uri) {

        try {
            StartDicomImportJobRequest startDicomImportJobRequest =
StartDicomImportJobRequest.builder()
```

```
                    .jobName(jobName)
                    .datastoreId(datastoreId)
                    .dataAccessRoleArn(dataAccessRoleArn)
                    .inputS3Uri(inputS3Uri)
                    .outputS3Uri(outputS3Uri)
                    .build();
            StartDicomImportJobResponse response =
medicalImagingClient.startDICOMImportJob(startDicomImportJobRequest);
            return response.jobId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
```

- API-Details finden Sie StartDICOMImportJob in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Java 2.x

```
    public static void tagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();
```

```
            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie TagResource in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für Java 2.x

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie UntagResource in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UpdateImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`UpdateImageSetMetadata`.

SDK für Java 2.x

```
   /**
    * Update the metadata of an AWS HealthImaging image set.
    *
    * @param medicalImagingClient - The AWS HealthImaging client object.
    * @param datastoreId          - The datastore ID.
    * @param imageSetId           - The image set ID.
    * @param versionId            - The version ID.
    * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
           UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                   .builder()
                   .datastoreId(datastoreId)
```

```
                    .imageSetId(imageSetId)
                    .latestVersionId(versionId)
                    .updateImageSetMetadataUpdates(metadataUpdates)
                    .force(force)
                    .build();

            UpdateImageSetMetadataResponse response =
medicalImagingClient.updateImageSetMetadata(updateImageSetMetadataRequest);

            System.out.println("The image set metadata was updated" + response);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es.

```
                final String insertAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
                final String removeAttributes = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "DICOM": {
                              "StudyDescription": "CT CHEST"
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeAttributes
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
                final String removeInstance = """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                              "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1":
{
                                "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                                }
                              }
                            }
                          }
                        }
                        """;
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .dicomUpdates(DICOMUpdates.builder()
```

```
                                .removableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(removeInstance
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- API-Details finden Sie UpdateImageSetMetadata in der AWS SDK for Java 2.x API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien

Taggen eines Datenspeichers

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Datenspeicher taggen.

SDK für Java 2.x

So taggen Sie einen Datenspeicher.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

So listen Sie Tags für einen Datenspeicher auf.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                datastoreArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
   public static ListTagsForResourceResponse
listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
           String resourceArn) {
       try {
           ListTagsForResourceRequest listTagsForResourceRequest =
ListTagsForResourceRequest.builder()
                   .resourceArn(resourceArn)
                   .build();

           return
medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
       } catch (MedicalImagingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }

       return null;
   }
```

So entfernen Sie Tags von einem Datenspeicher.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                               Collections.singletonList("Deployment"));
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
   public static void untagMedicalImagingResource(MedicalImagingClient
medicalImagingClient,
           String resourceArn,
           Collection<String> tagKeys) {
       try {
```

```
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Taggen eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Bildsatz taggen.

SDK für Java 2.x

So taggen Sie einen Imagesatz.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
    public static void tagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Map<String, String> tags) {
        try {
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tags(tags)
                    .build();

            medicalImagingClient.tagResource(tagResourceRequest);

            System.out.println("Tags have been added to the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

So listen Sie Tags für einen Imagesatz auf.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                ListTagsForResourceResponse result =
 ListTagsForResource.listMedicalImagingResourceTags(
                                medicalImagingClient,
                                imageSetArn);
                if (result != null) {
                        System.out.println("Tags for resource: " + result.tags());
                }
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
    public static ListTagsForResourceResponse
 listMedicalImagingResourceTags(MedicalImagingClient medicalImagingClient,
            String resourceArn) {
        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .build();

            return
 medicalImagingClient.listTagsForResource(listTagsForResourceRequest);
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

So entfernen Sie Tags von einem Imagesatz.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
    public static void untagMedicalImagingResource(MedicalImagingClient
 medicalImagingClient,
            String resourceArn,
            Collection<String> tagKeys) {
        try {
            UntagResourceRequest untagResourceRequest =
 UntagResourceRequest.builder()
                    .resourceArn(resourceArn)
                    .tagKeys(tagKeys)
```

```
                    .build();

            medicalImagingClient.untagResource(untagResourceRequest);

            System.out.println("Tags have been removed from the resource.");
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- API-Details finden Sie ListPolicies in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
/*
  To run this Java V2 code example, set up your development environment, including
 your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3 Service
 client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";
```

```
   public static String userArn;

   public static void main(String[] args) throws Exception {

       final String usage = """

               Usage:
                   <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

               Where:
                   username - The name of the IAM user to create.\s
                   policyName - The name of the policy to create.\s
                   roleName - The name of the role to create.\s
                   roleSessionName - The name of the session required for the
assumeRole operation.\s
                   bucketName - The name of the Amazon S3 bucket from which objects
are read.\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String userName = args[0];
       String policyName = args[1];
       String roleName = args[2];
       String roleSessionName = args[3];
       String bucketName = args[4];

       Region region = Region.AWS_GLOBAL;
       IamClient iam = IamClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the AWS IAM example scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println(" 1. Create the IAM user.");
       User createUser = createIAMUser(iam, userName);
```

```
        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
```

```
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();
```

```
            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String json)
{

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
```

```
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
```

```
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
```

```
                .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn) {

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);

            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
```

```
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
```

```
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }
```

```
    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK for Java 2.x API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateAccessKey in der AWS SDK for Java 2.x API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example, myawsaccount).
\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
```

```
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateAccountAlias in der AWS SDK for Java 2.x API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: " +
 result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreatePolicy in der AWS SDK for Java 2.x API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that represents
 the trust policy.\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " + response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- API-Details finden Sie CreateRole in der AWS SDK for Java 2.x API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
}
```

- API-Details finden Sie CreateUser in der AWS SDK for Java 2.x API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you want
 to delete.\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK for Java 2.x API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API-Details finden Sie DeleteAccountAlias in der AWS SDK for Java 2.x API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API-Details finden Sie DeletePolicy in der AWS SDK for Java 2.x API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s
```

```
                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteUser in der AWS SDK for Java 2.x API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];
```

```
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK for Java 2.x API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }
```

```
    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListAccessKeys in der AWS SDK for Java 2.x API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
```

```
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListAccountAliases in der AWS SDK for Java 2.x API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request = ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
```

```
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListUsers in der AWS SDK for Java 2.x API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.\s
                    accessId - The access key ID of the secret access key you want
 to update.\s
                    status - The status you want to assign to the secret access key.
\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
```

```
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s to"
 +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie UpdateAccessKey in der AWS SDK for Java 2.x API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String newName)
 {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s", newName);

        } catch (IamException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie UpdateUser in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-prof-
bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException, InterruptedException
 {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and Manage
 a Resilient Service!");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
```

```
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to create
several AWS resources
                        to set up a load-balanced web service endpoint and explore
some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances that
each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when an
instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged credentials.

                The template also defines an IAM policy that each instance uses to
assume a role that grants
                permissions to access the DynamoDB recommendation table and Systems
Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
```

```
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load balancer.
The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets, targetGroupArn,
lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
```

```
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
```

```
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
```

```
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       System.out.println(
               """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns a
static response.
               The service still reports as healthy because health checks are still
shallow.
               """);
       demoChoices(loadBalancer);

       System.out.println("Let's reinstate the recommendation service.");
       paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

       System.out.println("""
               Let's also substitute bad credentials for one of the instances in
the target group so that it can't
               access the DynamoDB recommendation table. We will get an instance id
value.
               """);

       LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
       AutoScaler autoScaler = new AutoScaler();
```

```
        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
```

```
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
```

```
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
```

```
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
```

```
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure 'newInstanceProfileName'
is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with profile
%s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
```

```
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
```

```
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
```

```
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
```

```
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();
```

```
        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();
```

```
        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()
                .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
```

```
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
```

```
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
```

```
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
```

```
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
 available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " + e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended, such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
```

```
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException {
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();
```

```
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws IOException
{
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable = enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
```

```
        System.out.println("Added all records to the " + tableName);
    }
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup

- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Arbeiten mit der IAM-Policy-Builder-API

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie IAM-Richtlinien mithilfe der objektorientierten API.
- Verwenden Sie die IAM-Policy-Builder-API mit dem IAM-Service.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In den Beispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

Erstellen Sie eine zeitbasierte Richtlinie.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1
```

```
.operator(IamConditionOperator.DATE_GREATER_THAN)

.key("aws:CurrentTime")

.value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_LESS_THAN)

.key("aws:CurrentTime")

.value("2020-06-30T23:59:59Z")))
                                .build();

               // Use an IamPolicyWriter to write out the JSON string to a more
readable
               // format.
               return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
       }
```

Erstellen Sie eine Richtlinie mit mehreren Bedingungen.

```
       public String multipleConditionsExample() {
               IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

.addAction("dynamodb:BatchWriteItem")

.addResource("arn:aws:dynamodb:*:*:table/table-name")

.addConditions(IamConditionOperator.STRING_EQUALS
```

```
 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                    List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))

 .key("dynamodb:Select")

 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Verwenden Sie Prinzipale in einer Richtlinie.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
```

```
                                .prettyPrint(true).build());
        }
```

Gewähren Sie kontoübergreifenden -Zugriff.

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addPrincipal(IamPrincipalType.AWS,
 "111122223333")
                                                .addAction("s3:PutObject")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                .key("s3:x-amz-acl")
                                                                .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

Erstellen und laden Sie eine `IamPolicy` hoch.

```
        public String createAndUploadPolicyExample(IamClient iam, String accountID,
 String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:PutItem")
                                                .addResource("arn:aws:dynamodb:us-
east-1:" + accountID
                                                                + ":table/
exampleTableName")
                                                .build())
                                .build();
```

```
                // Upload the policy.
                iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
                return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

Downloaden und arbeiten Sie mit einer `IamPolicy`.

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

                String policyVersion =
getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned from
IAM.
                String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));
```

```
                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

Amazon Inspector-Beispiele, die SDK für Java 2.x verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Inspector Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient = Inspector2Client.builder().build())
 {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");

        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
```

```
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }

    /**
     * Lists all member accounts associated with the current Inspector administrator
 account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
 Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
 administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- API-Details finden Sie ListMembers in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Überprüfen Sie den Status des Inspector-Kontos.
- Stellen Sie sicher, dass Inspector aktiviert ist.
- Analysieren Sie die Sicherheitsergebnisse.
- Überprüfen Sie die Scan-Abdeckung.
- Erstellen Sie einen Ergebnisfilter.
- Listet vorhandene Filter auf.
- Überprüfen Sie die Nutzung und die Kosten.
- Holen Sie sich Statistiken zur Netzabdeckung.
- Löschen Sie einen Filter.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die -Features gezeigt werden.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class InspectorScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorScenario.class);
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        InspectorActions inspectorActions = new InspectorActions();
        logger.info("Amazon Inspector Basics Scenario");

        logger.info("""
                    Amazon Inspector is a security assessment service provided by
 Amazon Web Services (AWS) that helps
                    improve the security and compliance of applications deployed on
 AWS. It automatically assesses
                    applications for vulnerabilities or deviations from best
 practices. By leveraging Amazon Inspector,
                    users can gain insights into the overall security state of their
 application and identify potential
                    security risks.

                    This service operates by conducting both network and host-based
 assessments, allowing it to detect a
                    wide range of security issues, including those related to
 operating systems, network configurations,
                    and application dependencies.
                    """);

        waitForInputToContinue();

        try {
            runScenario(inspectorActions);

            logger.info("");
            logger.info("Scenario completed successfully!");
            logger.info("");
            logger.info("What you learned:");
            logger.info("  - How to check Inspector account status");
            logger.info("  - How to enable Inspector");
            logger.info("  - How to list and analyze findings");
            logger.info("  - How to check coverage information");
            logger.info("  - How to create and manage filters");
```

```
            logger.info("  - How to track usage and costs");
            logger.info("  - How to clean up resources");
            logger.info("");

        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }

    /**
     * Runs the Inspector scenario in a step-by-step sequence.
     *
     * All InspectorActions methods are asynchronous and return CompletableFutures.
     * Each step ends with .join(). Any async exception thrown during .join() will
bubble up
     *
     */
    public static void runScenario(InspectorActions actions) {
        String filterArn = null;
        boolean inspectorEnabled = false;

        try {
            // Step 1
            logger.info(DASHES);
            logger.info("Step 1: Checking Inspector account status...");
            String status = actions.getAccountStatusAsync().join();
            logger.info(status);
            waitForInputToContinue();

            // Step 2
            logger.info(DASHES);
            logger.info("Step 2: Enabling Inspector...");
            String message = actions.enableInspectorAsync(null).join();
            logger.info(message);
            inspectorEnabled = true;  // track that Inspector was enabled
            waitForInputToContinue();

            // Step 3
            logger.info(DASHES);
```

```
            logger.info("Step 3: Listing LOW severity findings...");

            // Call the service method
            List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

            if (!allFindings.isEmpty()) {
                // Only proceed if there are findings
                String lastArn = allFindings.get(allFindings.size() - 1);
                logger.info("Look up details on: {}", lastArn);
                waitForInputToContinue();
                String details = actions.getFindingDetailsAsync(lastArn).join();
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName, "Suppress
low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
```

```
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();

            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(), ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
e.getMessage(), e);
                }
            }
```

```
            if (inspectorEnabled) {
                try {
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
 e.getMessage(), e);
                }
            }
        }
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

Eine Wrapper-Klasse für -SDK-Methoden.

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();
```

```
            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
```

```
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }
```

```
                    return summary.toString();
                });
    }


    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
```

```
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
```

```
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
```

```
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
```

```
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");
```

```
        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
```

```
    * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
    */
   public CompletableFuture<String> createLowSeverityFilterAsync(
           String filterName,
           String description) {

       // Define a filter to match LOW severity findings.
       StringFilter severityFilter = StringFilter.builder()
               .value(Severity.LOW.toString())
               .comparison(StringComparison.EQUALS)
               .build();

       // Create filter criteria.
       FilterCriteria filterCriteria = FilterCriteria.builder()
               .severity(Collections.singletonList(severityFilter))
               .build();

       // Build the filter creation request.
       CreateFilterRequest request = CreateFilterRequest.builder()
               .name(filterName)
               .filterCriteria(filterCriteria)
               .action(FilterAction.SUPPRESS)
               .description(description)
               .build();

       return getAsyncClient().createFilter(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof ValidationException ve) {
                           throw new CompletionException(
                                   "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                   ve
                           );
                       }

                       if (cause instanceof Inspector2Exception e) {
                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
```

```
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());
```

```
        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
 a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
 paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();
```

```
        ListCoveragePublisher paginator =
 getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();

            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
```

```
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }
```

```
                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }

    /**
     * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
     *
     * @param findingArn The ARN of the finding to look up.
     * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
     * containing all available details for the finding.
     * @throws RuntimeException if the async call to Inspector2 fails.
     */
    public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
        BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
                .findingArns(findingArn)
                .build();

        return getAsyncClient().batchGetFindingDetails(request)
                .thenApply(response -> {
                    if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                        return String.format("No details found for ARN: ",
findingArn);
                    }

                    StringBuilder sb = new StringBuilder();
                    response.findingDetails().forEach(detail -> {
                        sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
```

```
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
 ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
```

```
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
```

```
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
 account.accountId() : "Unknown";
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter
  - Aktivieren
  - ListCoverage
  - ListCoverageStatistics
  - ListFilters
  - ListFindings
  - ListUsageTotals

Aktionen

`BatchGetAccountStatus`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetAccountStatus`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the account status using the Inspector2Client.
    */
   public CompletableFuture<String> getAccountStatusAsync() {
       BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
               .accountIds(Collections.emptyList())
               .build();

       return getAsyncClient().batchGetAccountStatus(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof AccessDeniedException) {
                           throw new CompletionException(
                                   "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                   cause
                           );

                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
```

```
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }
```

```
                    return sb.toString();
                });
    }
```

- API-Details finden Sie BatchGetAccountStatus in der AWS SDK for Java 2.x API-Referenz.

`BatchGetFindingDetails`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetFindingDetails`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }
```

```
                    StringBuilder sb = new StringBuilder();
                    response.findingDetails().forEach(detail -> {
                        sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }
```

```
                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }
```

- API-Details finden Sie BatchGetFindingDetails in der AWS SDK for Java 2.x API-Referenz.

`CreateFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateFilter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
    *
    * @param filterName  the name of the filter to create
    * @param description a descriptive string explaining the purpose of the filter
    * @return a CompletableFuture that completes with the ARN of the created filter
    * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
    */
   public CompletableFuture<String> createLowSeverityFilterAsync(
           String filterName,
           String description) {

       // Define a filter to match LOW severity findings.
       StringFilter severityFilter = StringFilter.builder()
               .value(Severity.LOW.toString())
               .comparison(StringComparison.EQUALS)
               .build();

       // Create filter criteria.
       FilterCriteria filterCriteria = FilterCriteria.builder()
               .severity(Collections.singletonList(severityFilter))
               .build();

       // Build the filter creation request.
       CreateFilterRequest request = CreateFilterRequest.builder()
               .name(filterName)
               .filterCriteria(filterCriteria)
               .action(FilterAction.SUPPRESS)
               .description(description)
               .build();

       return getAsyncClient().createFilter(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof ValidationException ve) {
                           throw new CompletionException(
                                   "Validation error creating filter:
%s".formatted(ve.getMessage()),
```

```
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- API-Details finden Sie CreateFilter in der AWS SDK for Java 2.x API-Referenz.

`DeleteFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFilter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an AWS Inspector2 filter.
```

```
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

- API-Details finden Sie DeleteFilter in der AWS SDK for Java 2.x API-Referenz.

Deaktivieren

Das folgende Codebeispiel zeigt die Verwendung`Disable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
```

```
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
```

```
    }
```

- API-Details finden Sie unter Deaktivieren in der AWS SDK for Java 2.x API-Referenz.

Aktivieren

Das folgende Codebeispiel zeigt die Verwendung`Enable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Enables AWS Inspector for the provided account(s) and default resource types.
 *
 * @param accountIds Optional list of AWS account IDs.
 */
public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

    // The resource types to enable.
    List<ResourceScanType> resourceTypes = List.of(
            ResourceScanType.EC2,
            ResourceScanType.ECR,
            ResourceScanType.LAMBDA,
            ResourceScanType.LAMBDA_CODE
    );

    // Build the request.
    EnableRequest.Builder requestBuilder = EnableRequest.builder()
            .resourceTypes(resourceTypes);

    if (accountIds != null && !accountIds.isEmpty()) {
        requestBuilder.accountIds(accountIds);
    }

    EnableRequest request = requestBuilder.build();
```

```
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
```

```
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- API-Details finden Sie unter Aktivieren in der AWS SDK for Java 2.x API-Referenz.

`ListCoverage`

Das folgende Codebeispiel zeigt die Verwendung`ListCoverage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();
```

```
            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
```

```
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- API-Details finden Sie ListCoverage in der AWS SDK for Java 2.x API-Referenz.

`ListCoverageStatistics`

Das folgende Codebeispiel zeigt die Verwendung`ListCoverageStatistics`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
```

```
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }
```

- API-Details finden Sie ListCoverageStatistics in der AWS SDK for Java 2.x API-Referenz.

`ListFilters`

Das folgende Codebeispiel zeigt die Verwendung`ListFilters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
```

```
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- API-Details finden Sie ListFilters in der AWS SDK for Java 2.x API-Referenz.

`ListFindings`

Das folgende Codebeispiel zeigt die Verwendung`ListFindings`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
```

```
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
 a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
```

```
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
findings", ex);
                });
    }
```

- API-Details finden Sie ListFindings in der AWS SDK for Java 2.x API-Referenz.

`ListUsageTotals`

Das folgende Codebeispiel zeigt die Verwendung`ListUsageTotals`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
```

```
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }
```

- API-Details finden Sie ListUsageTotals in der AWS SDK for Java 2.x API-Referenz.

AWS IoT Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS IoT.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK for Java 2.x -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.

- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Shadow.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche eine AWS IoT Sache.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS IoT Funktionen demonstriert.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
```

```
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
 Internet of Things (IoT) Core service. The program guides you through a series of
 steps,
            including creating an IoT Thing, generating a device certificate,
 updating the Thing with attributes, and so on.
```

```
            It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service that
can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
```

```
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
```

```
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für AWS IoT SDK-Methoden.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
```

```
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
```

```
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
```

```
    *
    * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
```

```
    public void attachCertificateToThing(String thingName, String certificateArn) {
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
```

```
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument, StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
```

```
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
 getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
```

```
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
```

```
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
```

```
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
```

```
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest = CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
```

```
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
                List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
```

```
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
```

```
    * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed from
" + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
```

```
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration

- UpdateThing

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();

       CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
       future.whenComplete((attachResponse, ex) -> {
           if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
               System.out.println("Certificate attached to Thing successfully.");
```

```
                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- API-Details finden Sie AttachThingPrincipal in der AWS SDK for Java 2.x API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
```

```
    * This method initiates an asynchronous request to create an IoT certificate.
    * If the request is successful, it prints the certificate details and returns
the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
           }
       });

       future.join();
       return certificateArn[0];
   }
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS SDK for Java 2.x API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
    * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
           }
       });

       future.join();
```

```
    }
```

- API-Details finden Sie CreateThing in der AWS SDK for Java 2.x API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();
```

```
        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest = CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- API-Details finden Sie CreateTopicRule in der AWS SDK for Java 2.x API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
           .certificateId(extractCertificateId(certificateArn))
           .build();

       CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully deleted.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }
```

- API-Details finden Sie DeleteCertificate in der AWS SDK for Java 2.x API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
```

```
        });

        future.join();
    }
```

- API-Details finden Sie DeleteThing in der AWS SDK for Java 2.x API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";
```

```
                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK for Java 2.x API-Referenz.

`DescribeThing`

Das folgende Codebeispiel zeigt die Verwendung`DescribeThing`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
```

```
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }
```

- API-Details finden Sie DescribeThing in der AWS SDK for Java 2.x API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed from
" + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + ex.getMessage());
               }
           }
       });

       future.join();
   }
```

- API-Details finden Sie DetachThingPrincipal in der AWS SDK for Java 2.x API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
```

```
    }
```

- API-Details finden Sie ListCertificates in der AWS SDK for Java 2.x API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
```

```
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- API-Details finden Sie SearchIndex in der AWS SDK for Java 2.x API-Referenz.

AWS IoT data Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
       future.whenComplete((getThingShadowResponse, ex) -> {
           if (getThingShadowResponse != null) {
               // Extracting payload from response.
               SdkBytes payload = getThingShadowResponse.payload();
               String payloadString = payload.asUtf8String();
               System.out.println("Received Shadow Data: " + payloadString);
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to get Thing Shadow payload.");
               }
```

```
            }
        });

        future.join();
    }
```

- API-Details finden Sie GetThingShadow in der AWS SDK for Java 2.x API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument, StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();
```

```
        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- API-Details finden Sie UpdateThingShadow in der AWS SDK for Java 2.x API-Referenz.

AWS IoT SiteWise Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS IoT SiteWise.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT SiteWise

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT SiteWise beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest = ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
```

```
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- API-Details finden Sie ListAssetModels in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein AWS IoT SiteWise Asset-Modell.
- Erstellen Sie ein AWS IoT SiteWise Asset.
- Rufen Sie die Eigenschafts-ID-Werte ab.
- Daten an ein AWS IoT SiteWise Asset senden.
- Ruft den Wert der AWS IoT SiteWise Asset-Eigenschaft ab.
- Erstellen Sie ein AWS IoT SiteWise Portal.
- Erstellen Sie ein AWS IoT SiteWise Gateway.
- Beschreiben Sie das AWS IoT SiteWise Gateway.
- Löschen Sie die AWS IoT SiteWise Vermögenswerte.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS IoT SiteWise Funktionen demonstriert.

```
public class SitewiseScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0", "-");

   private static final Logger logger =
LoggerFactory.getLogger(SitewiseScenario.class);
   static Scanner scanner = new Scanner(System.in);

   private static final String ROLES_STACK = "RoleSitewise";

   static SitewiseActions sitewiseActions = new SitewiseActions();

   public static void main(String[] args) throws Throwable {
       Scanner scanner = new Scanner(System.in);
       String assetModelName = "MyAssetModel1";
       String assetName = "MyAsset1" ;
       String gatewayName = "MyGateway1" ;
       String myThing =  "MyThing1" ;

       logger.info("""
           AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
           makes it easy to collect, store, organize, and monitor data from
industrial equipment and processes.
           It is designed to help industrial and manufacturing organizations
collect data from their equipment and
           processes, and use that data to make informed decisions about their
operations.

           One of the key features of AWS IoT SiteWise is its ability to connect to
a wide range of industrial
           equipment and systems, including programmable logic controllers (PLCs),
sensors, and other
           industrial devices. It can collect data from these devices and organize
it into a unified data model,
           making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
           visualizing the data, setting up alarms and alerts, and generating
reports.

           Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
           It can collect and store data from thousands of devices and process
millions of data points per second,
           making it suitable for large-scale industrial operations. Additionally,
AWS IoT SiteWise is designed
```

```
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(assetModelName, assetName, gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName, String
gatewayName, String myThing) throws Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment. This
model provides a structured and
             hierarchical representation of these assets, allowing users to define
the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
            """);
        waitForInputToContinue(scanner);
```

```
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + cause.getMessage(),
cause);
                return;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure and
metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
```

```
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data from
            multiple sources, normalize the data into a standard format, and store
it in a
            centralized location. This makes it easier to analyze and gain insights
from the data.

            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process, and
analyze industrial data
            from connected equipment and sensors. One of the key benefits of reading
an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
```

```
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT SiteWise
platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
```

```
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}", response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
```

```
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
 message: {}, Error code {}",
                            notFoundException.getMessage(),
 notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
 cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
```

```
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

Eine Wrapper-Klasse für AWS IoT SiteWise SDK-Methoden.

```
public class SitewiseActions {

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
    }


    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
            .build();
```

```
        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();

        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
```

```
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
        Map<String, Double> sampleData = generateSampleData();
        long timestamp = Instant.now().toEpochMilli();

        TimeInNanos time = TimeInNanos.builder()
            .timeInSeconds(timestamp / 1000)
            .offsetInNanos((int) ((timestamp % 1000) * 1000000))
            .build();
```

```
        BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
            .entries(Arrays.asList(
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-3")
                    .assetId(assetId)
                    .propertyId(tempPropertyId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Temperature"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build(),
                PutAssetPropertyValueEntry.builder()
                    .entryId("entry-4")
                    .assetId(assetId)
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
```

```
    *
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
```

```
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
       ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
       return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
           .handle((response, throwable) -> {
               if (response != null) {
                   return response.assetModelPropertySummaries().stream()
                       .collect(Collectors
                           .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
               } else {
                   logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                   throw (CompletionException) throwable;
               }
           });
   }

   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
```

```
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
   }

   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
```

```
            });
    }

    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }


    /**
     * Creates a new IoT Sitewise gateway.
```

```
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");

       CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
           .gatewayName(gatewayName)
           .gatewayPlatform(platform)
           .tags(tag)
           .build();

       return getAsyncClient().createGateway(createGatewayRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Error creating the gateway.");
                   throw (CompletionException) exception;
               }
               logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
```

```
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
```

```
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
 method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
 calling
     *         {@link CompletionException#getCause()}, the calling code can access
 the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
 gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
 method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel

- DeleteGateway
- DescribeAssetModel
- DescribeGateway
- GetAssetPropertyValue
- ListAssetModelProperties
- ListAssetModels

Aktionen

`BatchPutAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchPutAssetPropertyValue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
                   .propertyId(humidityPropId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Humidity"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build()
           ))
           .build();
```

```
        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API-Details finden Sie BatchPutAssetPropertyValue in der AWS SDK for Java 2.x API-Referenz.

`CreateAsset`

Das folgende Codebeispiel zeigt die Verwendung`CreateAsset`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API-Details finden Sie CreateAsset in der AWS SDK for Java 2.x API-Referenz.

`CreateAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`CreateAssetModel`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
```

```
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

- API-Details finden Sie CreateAssetModel in der AWS SDK for Java 2.x API-Referenz.

`CreateGateway`

Das folgende Codebeispiel zeigt die Verwendung`CreateGateway`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- API-Details finden Sie CreateGateway in der AWS SDK for Java 2.x API-Referenz.

`DeleteAsset`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAsset`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
   }
```

- API-Details finden Sie DeleteAsset in der AWS SDK for Java 2.x API-Referenz.

`DeleteAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAssetModel`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
```

```
    }
```

- API-Details finden Sie DeleteAssetModel in der AWS SDK for Java 2.x API-Referenz.

`DeleteGateway`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGateway`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API-Details finden Sie DeleteGateway in der AWS SDK for Java 2.x API-Referenz.

`DescribeAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAssetModel`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
```

```
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- API-Details finden Sie DescribeAssetModel in der AWS SDK for Java 2.x API-Referenz.

`DescribeGateway`

Das folgende Codebeispiel zeigt die Verwendung`DescribeGateway`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
```

```
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }
```

- API-Details finden Sie DescribeGateway in der AWS SDK for Java 2.x API-Referenz.

`GetAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`GetAssetPropertyValue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Fetches the value of an asset property.
     *
```

```
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }
```

- API-Details finden Sie GetAssetPropertyValue in der AWS SDK for Java 2.x API-Referenz.

`ListAssetModels`

Das folgende Codebeispiel zeigt die Verwendung`ListAssetModels`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
       ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
       return getAsyncClient().listAssetModels(listAssetModelsRequest)
               .handle((listAssetModelsResponse, exception) -> {
                   if (exception != null) {
                       logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                       if (assetModelSummary.name().equals(assetModelName)) {
                           return assetModelSummary.id();
                       }
                   }
```

```
                    return null;
                });
    }
```

- API-Details finden Sie ListAssetModels in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Keyspaces unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Keyspaces Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Keyspaces

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von Amazon Keyspaces beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API-Details finden Sie ListKeyspaces in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die zeitpunktbezogene Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.
- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
```

```
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
 {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
```

```
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
```

```
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
 the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
 column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
 the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
 minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
```

```
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
ResourceNotFoundException is
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
```

```
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println("The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
                + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
```

```
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
                .setString("k0", titleUpdate)
                .setInt("k1", yearUpdate)
                .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute(
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
 Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
```

```
   public static void getMovieData(CqlSession session, String keyspaceName) {
       ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
       resultSet.forEach(item -> {
           System.out.println("The Movie title is " + item.getString("title"));
           System.out.println("The Movie year is " + item.getInt("year"));
           System.out.println("The plot is " + item.getString("plot"));
       });
   }

   // Load data into the table.
   public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
       String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
       JsonParser parser = new JsonFactory().createParser(new File(fileName));
       com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
       Iterator<JsonNode> iter = rootNode.iterator();
       ObjectNode currentNode;
       int t = 0;
       while (iter.hasNext()) {

           // Add 20 movies to the table.
           if (t == 20)
               break;
           currentNode = (ObjectNode) iter.next();

           int year = currentNode.path("year").asInt();
           String title = currentNode.path("title").asText();
           String plot = currentNode.path("info").path("plot").toString();

           // Insert the data into the Amazon Keyspaces table.
           BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
           builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
           PreparedStatement preparedStatement = session.prepare(sqlStatement);
           builder.addStatement(preparedStatement.boundStatementBuilder()
                   .setString("k0", title)
                   .setInt("k1", year)
                   .setString("k2", plot)
                   .build());

           BatchStatement batchStatement = builder.build();
```

```
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
```

```
                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace,
 String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
```

```
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
```

```
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
```

```
            System.out.println("The ARN of the KeySpace is " +
 response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyspace`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
           System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie CreateKeyspace in der AWS SDK for Java 2.x API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();
```

```
            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);

            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();
```

```
            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie CreateTable in der AWS SDK for Java 2.x API-Referenz.

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           keyClient.deleteKeyspace(deleteKeyspaceRequest);
```

```
        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DeleteKeyspace in der AWS SDK for Java 2.x API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK for Java 2.x API-Referenz.

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyspace`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
           String name = response.keyspaceName();
           System.out.println("The " + name + " KeySpace is ready");

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie GetKeyspace in der AWS SDK for Java 2.x API-Referenz.

`GetTable`

Das folgende Codebeispiel zeigt die Verwendung`GetTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
           }

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie GetTable in der AWS SDK for Java 2.x API-Referenz.

`ListKeyspaces`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyspaces`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
       try {
           ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                   .maxResults(10)
                   .build();

           ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
           listRes.stream()
                   .flatMap(r -> r.keyspaces().stream())
                   .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ListKeyspaces in der AWS SDK for Java 2.x API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
       try {
           ListTablesRequest tablesRequest = ListTablesRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
           listRes.stream()
                   .flatMap(r -> r.tables().stream())
                   .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                           " Table name: " + content.tableName()));

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ListTables in der AWS SDK for Java 2.x API-Referenz.

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung`RestoreTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                   .restoreTimestamp(myTime)
                   .sourceTableName("Movie")
                   .targetKeyspaceName(keyspaceName)
                   .targetTableName("MovieRestore")
                   .sourceKeyspaceName(keyspaceName)
                   .build();

           RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
           System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie RestoreTable in der AWS SDK for Java 2.x API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie UpdateTable in der AWS SDK for Java 2.x API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Kinesis Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`CreateStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateStream`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.CreateStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDataStream {
    public static void main(String[] args) {

        final String usage = """

                Usage:
```

```
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();
        createStream(kinesisClient, streamName);
        System.out.println("Done");
        kinesisClient.close();
    }

    public static void createStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            CreateStreamRequest streamReq = CreateStreamRequest.builder()
                    .streamName(streamName)
                    .shardCount(1)
                    .build();

            kinesisClient.createStream(streamReq);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateStream in der AWS SDK for Java 2.x API-Referenz.

`DeleteStream`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStream`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DeleteStreamRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDataStream {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream (for example,
 StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        deleteStream(kinesisClient, streamName);
        kinesisClient.close();
        System.out.println("Done");
    }

    public static void deleteStream(KinesisClient kinesisClient, String streamName)
 {
        try {
            DeleteStreamRequest delStream = DeleteStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            kinesisClient.deleteStream(delStream);

        } catch (KinesisException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteStream in der AWS SDK for Java 2.x API-Referenz.

`GetRecords`

Das folgende Codebeispiel zeigt die Verwendung`GetRecords`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.Shard;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorRequest;
import software.amazon.awssdk.services.kinesis.model.GetShardIteratorResponse;
import software.amazon.awssdk.services.kinesis.model.Record;
import software.amazon.awssdk.services.kinesis.model.GetRecordsRequest;
import software.amazon.awssdk.services.kinesis.model.GetRecordsResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetRecords {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to read from (for
 example, StockTradeStream).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
```

```
                .build();

        getStockTrades(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void getStockTrades(KinesisClient kinesisClient, String
streamName) {
        String shardIterator;
        String lastShardId = null;
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
                .streamName(streamName)
                .build();

        List<Shard> shards = new ArrayList<>();
        DescribeStreamResponse streamRes;
        do {
            streamRes = kinesisClient.describeStream(describeStreamRequest);
            shards.addAll(streamRes.streamDescription().shards());

            if (shards.size() > 0) {
                lastShardId = shards.get(shards.size() - 1).shardId();
            }
        } while (streamRes.streamDescription().hasMoreShards());

        GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
                .streamName(streamName)
                .shardIteratorType("TRIM_HORIZON")
                .shardId(lastShardId)
                .build();

        GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
        shardIterator = shardIteratorResult.shardIterator();

        // Continuously read data records from shard.
        List<Record> records;

        // Create new GetRecordsRequest with existing shardIterator.
        // Set maximum records to return to 1000.
        GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
                .shardIterator(shardIterator)
                .limit(1000)
```

```
                .build();

        GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

        // Put result into record list. Result may be empty.
        records = result.records();

        // Print records
        for (Record record : records) {
            SdkBytes byteBuffer = record.data();
            System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
 String(byteBuffer.asByteArray()));
        }
    }
}
```

- API-Details finden Sie GetRecords in der AWS SDK for Java 2.x API-Referenz.

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung`PutRecord`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kinesis.KinesisClient;
import software.amazon.awssdk.services.kinesis.model.PutRecordRequest;
import software.amazon.awssdk.services.kinesis.model.KinesisException;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamRequest;
import software.amazon.awssdk.services.kinesis.model.DescribeStreamResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StockTradesWriter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <streamName>

                Where:
                    streamName - The Amazon Kinesis data stream to which records are
 written (for example, StockTradeStream)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String streamName = args[0];
        Region region = Region.US_EAST_1;
        KinesisClient kinesisClient = KinesisClient.builder()
                .region(region)
                .build();

        // Ensure that the Kinesis Stream is valid.
        validateStream(kinesisClient, streamName);
        setStockData(kinesisClient, streamName);
        kinesisClient.close();
    }

    public static void setStockData(KinesisClient kinesisClient, String streamName)
 {
        try {
            // Repeatedly send stock trades with a 100 milliseconds wait in between.
            StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

            // Put in 50 Records for this example.
            int index = 50;
            for (int x = 0; x < index; x++) {
                StockTrade trade = stockTradeGenerator.getRandomTrade();
```

```
                sendStockTrade(trade, kinesisClient, streamName);
                Thread.sleep(100);
            }

        } catch (KinesisException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }

    private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
            String streamName) {
        byte[] bytes = trade.toJsonAsBytes();

        // The bytes could be null if there is an issue with the JSON serialization
by
        // the Jackson JSON library.
        if (bytes == null) {
            System.out.println("Could not get JSON bytes for stock trade");
            return;
        }

        System.out.println("Putting trade: " + trade);
        PutRecordRequest request = PutRecordRequest.builder()
                .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol
as the partition key, explained in
                                                       // the Supplemental
Information section below.
                .streamName(streamName)
                .data(SdkBytes.fromByteArray(bytes))
                .build();

        try {
            kinesisClient.putRecord(request);
        } catch (KinesisException e) {
            System.err.println(e.getMessage());
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String
streamName) {
        try {
```

```
            DescribeStreamRequest describeStreamRequest =
 DescribeStreamRequest.builder()
                    .streamName(streamName)
                    .build();

            DescribeStreamResponse describeStreamResponse =
 kinesisClient.describeStream(describeStreamRequest);

            if (!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
 {
                System.err.println("Stream " + streamName + " is not active. Please
 wait a few moments and try again.");
                System.exit(1);
            }

        } catch (KinesisException e) {
            System.err.println("Error found while describing the stream " +
 streamName);
            System.err.println(e);
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutRecord in der AWS SDK for Java 2.x API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }
```

```
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";
```

```
        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

AWS KMS Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS KMS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo AWS KMS

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();
```

```
        /*
         * The `subscribe` method is required when using paginator methods in the
AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
which is
         * based on a reactive stream. This allows asynchronous retrieval of
paginated
         * results as they become available. By subscribing to the stream, we can
process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API-Details finden Sie ListKeys in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines KMS-Schlüssels.

- Listen Sie die KMS-Schlüssel für Ihr Konto auf und rufen Sie Details über sie ab.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.
- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Taggen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String accountId = "";

    private static final Logger logger = LoggerFactory.getLogger(KMSScenario.class);

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    public static void main(String[] args) {
        final String usage = """
            Usage: <granteePrincipal>

            Where:
               granteePrincipal - The principal (user, service account, or group) to
 whom the grant or permission is being given.
            """;

        if (args.length != 1) {
            logger.info(usage);
            return;
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics scenario.

            This program demonstrates how to interact with AWS Key Management using
 the AWS SDK for Java (v2).
            The AWS Key Management Service (KMS) is a secure and highly available
 service that allows you to create
```

```
            and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
            KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
            data protection and regulatory compliance requirements.

            This Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " + targetKeyId);
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            2. Enable a KMS key

            By default, when the SDK creates an AWS key, it is enabled. The next bit
of code checks to
            determine if the key is enabled.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
```

```
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
```

```
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately 365
days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
```

```
            When authorizing access to a KMS key, grants are considered along with
key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
```

```
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and access
that the grant had provided.
            This means that any principal (user, role, or service) that was granted
access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
```

```
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
            access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is created
that
            gives the AWS account that owns the KMS key full access to the KMS key.
```

```
            Let's try to replace the automatically created policy with the following
policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make sure
it exists.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
             for your data signing needs. By using an AWS KMS key, you can leverage
the
             security controls and compliance features provided by AWS,
             which can help you meet various regulatory requirements and enhance the
overall security posture
             of your organization.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
```

```
   private static void requestDeleteResources(String aliasName, String targetKeyId)
{
       logger.info(DASHES);
       logger.info("15. Schedule the deletion of the KMS key\n");
       logger.info("""
           By default, KMS applies a waiting period of 30 days,
           but you can specify a waiting period of 7-30 days. When this operation
is successful,
           the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
           cryptographic operations. It remains in this state for the duration of
the waiting period.

           Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
           all data that was encrypted under the KMS key is unrecoverable.
           """);
       logger.info("Would you like to delete the Key Management resources? (y/n)");
       String delAns = scanner.nextLine().trim();
       if (delAns.equalsIgnoreCase("y")) {
           logger.info("You selected to delete the AWS KMS resources.");
           waitForInputToContinue(scanner);
           try {
               CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
               future.join();

           } catch (RuntimeException rt) {
               Throwable cause = rt.getCause();
               if (cause instanceof KmsException kmsEx) {
                   logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
               } else {
                   logger.info("An unexpected error occurred: " + rt.getMessage());
               }
           }
           waitForInputToContinue(scanner);
           try {
               CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
               future.join();

           } catch (RuntimeException rt) {
               Throwable cause = rt.getCause();
```

```
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
```

```
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }

        try {
            CompletableFuture<Void> future = kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
```

```
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Definieren Sie eine Klasse, die KMS-Aktionen umschließt.

```
public class KMSActions {
    private static final Logger logger = LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async client,
 with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
 creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
```

```
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();
```

```
        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
```

```
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
```

```
    * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
    */
   public CompletableFuture<Object> listAllAliasesAsync() {
       ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
           .limit(15)
           .build();

       ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
       return paginator.subscribe(response -> {
               response.aliases().forEach(alias ->
                   logger.info("The alias name is: " + alias.aliasName())
               );
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               if (ex.getCause() instanceof KmsException) {
                   KmsException e = (KmsException) ex.getCause();
                   throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
               }
           });
   }

   /**
    * Enables key rotation asynchronously for the specified key ID.
    *
    * @param keyId the ID of the key for which to enable key rotation
    * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
    * @throws RuntimeException if there was an error enabling key rotation, either
due to a KMS exception or an unexpected error
    */
   public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
       EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
           .keyId(keyId)
           .build();
```

```
        CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
```

```
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
```

```
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId   The ID or key ARN of the AWS KMS key.
     * @param grantId The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
     *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *         if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
```

```
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
```

```
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
```

```
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
```

```
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
 Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();
```

```
                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();
```

```
        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();
```

```
        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
            });
    }
```

```
    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
               } else if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
               }
           }
       });
```

```
        return responseFuture.thenApply(response -> null);
    }
```

- API-Details finden Sie CreateAlias in der AWS SDK for Java 2.x API-Referenz.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
```

```
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- API-Details finden Sie CreateGrant in der AWS SDK for Java 2.x API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- API-Details finden Sie CreateKey in der AWS SDK for Java 2.x API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
```

```
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
    */
   public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
       DecryptRequest decryptRequest = DecryptRequest.builder()
           .ciphertextBlob(encryptedData)
           .keyId(keyId)
           .build();

       CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
       responseFuture.whenComplete((decryptResponse, exception) -> {
           if (exception == null) {
               logger.info("Data decrypted successfully for key ID: " + keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
   }
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK for Java 2.x -API-Referenz.

`DeleteAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlias`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }
```

- API-Details finden Sie DeleteAlias in der AWS SDK for Java 2.x API-Referenz.

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- API-Details finden Sie DescribeKey in der AWS SDK for Java 2.x API-Referenz.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously disables the specified AWS Key Management Service (KMS) key.
    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
    * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
    */
   public CompletableFuture<Void> disableKeyAsync(String keyId) {
       DisableKeyRequest keyRequest = DisableKeyRequest.builder()
           .keyId(keyId)
           .build();

       return getAsyncClient().disableKey(keyRequest)
           .thenRun(() -> {
               logger.info("Key {} has been disabled successfully",keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
           });
   }
```

- API-Details finden Sie DisableKey in der AWS SDK for Java 2.x API-Referenz.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(response -> null);
   }
```

- API-Details finden Sie EnableKey in der AWS SDK for Java 2.x API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Encrypts the given text asynchronously using the specified KMS client and key
ID.
    *
    * @param keyId the ID of the KMS key to use for encryption
    * @param text the text to encrypt
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
           .plaintext(myBytes)
           .build();

       CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
       return responseFuture.whenComplete((response, ex) -> {
           if (response != null) {
               String algorithm = response.encryptionAlgorithm().toString();
               logger.info("The string was encrypted with algorithm {}.",
algorithm);
           } else {
               throw new RuntimeException(ex);
           }
       }).thenApply(EncryptResponse::ciphertextBlob);
   }
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK for Java 2.x -API-Referenz.

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAliases`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously lists all the aliases in the current AWS account.
    *
    * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
    */
   public CompletableFuture<Object> listAllAliasesAsync() {
       ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
           .limit(15)
           .build();

       ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
       return paginator.subscribe(response -> {
               response.aliases().forEach(alias ->
                   logger.info("The alias name is: " + alias.aliasName())
               );
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               if (ex.getCause() instanceof KmsException) {
                   KmsException e = (KmsException) ex.getCause();
                   throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
```

```
                }
            });
    }
```

- API-Details finden Sie ListAliases in der AWS SDK for Java 2.x API-Referenz.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
```

```
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }
```

- API-Details finden Sie ListGrants in der AWS SDK for Java 2.x API-Referenz.

`ListKeyPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyPolicies`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
```

```
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }
```

- API-Details finden Sie ListKeyPolicies in der AWS SDK for Java 2.x API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
```

```
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
which is
         * based on a reactive stream. This allows asynchronous retrieval of
paginated
         * results as they become available. By subscribing to the stream, we can
process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API-Details finden Sie ListKeys in der AWS SDK for Java 2.x API-Referenz.

`RevokeGrant`

Das folgende Codebeispiel zeigt die Verwendung`RevokeGrant`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
```

```
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- API-Details finden Sie RevokeGrant in der AWS SDK for Java 2.x API-Referenz.

`ScheduleKeyDeletion`

Das folgende Codebeispiel zeigt die Verwendung`ScheduleKeyDeletion`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
```

```
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();

       return getAsyncClient().scheduleKeyDeletion(deletionRequest)
           .thenRun(() -> {
               logger.info("Key {} will be deleted in 7 days", keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
           });
   }
```

- API-Details finden Sie ScheduleKeyDeletion in der AWS SDK for Java 2.x API-Referenz.

`Sign`

Das folgende Codebeispiel zeigt die Verwendung`Sign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
```

```
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
 Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();
```

```
                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }
```

- Weitere API-Informationen finden Sie unter Sign in der AWS SDK for Java 2.x -API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
```

```
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }
```

- API-Details finden Sie TagResource in der AWS SDK for Java 2.x API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren AWS Teams erstellt wurden und verwaltet werden. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
```

```
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListFunctions in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <functionName> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\s
                role - The AWS Identity and Access Management (IAM) service role
 that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
 example.Handler::handleRequest).\s
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
 name that contains the .zip or .jar used to update the Lambda function's code.\s
```

```
                key - The Amazon S3 key name that represents the .zip or .jar (for
example, LambdaHello-1.0-SNAPSHOT.jar).
                """;

        if (args.length != 5) {
              System.out.println(usage);
              return;
        }

        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
bucketName, role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {
```

```
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
```

```
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
```

```
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
```

```
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
                .waitUntilFunctionUpdated(getFunctionConfigRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);
```

```
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
 used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions

- UpdateFunctionCode
- UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String key,
                                              String bucketName,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
```

```
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateFunction in der AWS SDK for Java 2.x API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DeleteFunction in der AWS SDK for Java 2.x API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie GetFunction in der AWS SDK for Java 2.x API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Invokes a specific AWS Lambda function.
     *
```

```
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK for Java 2.x -API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
       try {
           LambdaWaiter waiter = awsLambda.waiter();
           UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
               .functionName(functionName)
               .publish(true)
               .s3Bucket(bucketName)
               .s3Key(key)
               .build();

           UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
           GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .build();

           WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
               .waitUntilFunctionUpdated(getFunctionConfigRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("The last modified value is " +
response.lastModified());

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK for Java 2.x API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the configuration of an AWS Lambda function.
    *
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .handler(handler)
               .runtime(Runtime.JAVA17)
               .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält

Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Zustandsmaschine erstellt wird, die AWS Lambda Funktionen nacheinander aufruft.

SDK für Java 2.x

Zeigt, wie ein AWS serverloser Workflow mithilfe von AWS Step Functions und erstellt wird. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step-Funktionen

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel

werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Verwenden der Neptune-API, um Grafikdaten abzufragen

Das folgende Codebeispiel zeigt, wie Sie mit der Neptune-API Grafikdaten abfragen.

SDK für Java 2.x

Zeigt, wie die Java-API von Amazon Neptune verwendet wird, um eine Lambda-Funktion zu erstellen, die Grafikdaten innerhalb der VPC abfragt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Neptune

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();

        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
```

```
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
+ ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();

        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();
```

```
        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;

public class Handler implements RequestHandler<KinesisEvent, Void> {
    @Override
    public Void handleRequest(final KinesisEvent event, final Context context) {
        LambdaLogger logger = context.getLogger();
        if (event.getRecords().isEmpty()) {
            logger.log("Empty Kinesis Event received");
```

```
            return null;
        }
        for (KinesisEvent.KinesisEventRecord record : event.getRecords()) {
            try {
                logger.log("Processed Event with EventId: "+record.getEventID());
                String data = new String(record.getKinesis().getData().array());
                logger.log("Data:"+ data);
                // TODO: Do interesting work based on the new data
            }
            catch (Exception ex) {
                logger.log("An error occurred:"+ex.getMessage());
                throw ex;
            }
        }
        logger.log("Successfully processed:"+event.getRecords().size()+" records");
        return null;
    }

}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
```

```
import
 com.amazonaws.services.lambda.runtime.events.DynamodbEvent.DynamodbStreamRecord;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class example implements RequestHandler<DynamodbEvent, Void> {

    private static final Gson GSON = new GsonBuilder().setPrettyPrinting().create();

    @Override
    public Void handleRequest(DynamodbEvent event, Context context) {
        System.out.println(GSON.toJson(event));
        event.getRecords().forEach(this::logDynamoDBRecord);
        return null;
    }

    private void logDynamoDBRecord(DynamodbStreamRecord record) {
        System.out.println(record.getEventID());
        System.out.println(record.getEventName());
        System.out.println("DynamoDB Record: " + GSON.toJson(record.getDynamodb()));
    }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Java.

```
import java.util.List;
```

```
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }
```

```
        return null;
    }
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
```

```
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
```

```
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
```

```
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KinesisEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessKinesisRecords implements RequestHandler<KinesisEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(KinesisEvent input, Context context) {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (KinesisEvent.KinesisEventRecord kinesisEventRecord :
 input.getRecords()) {
            try {
                //Process your record
                KinesisEvent.Record kinesisRecord = kinesisEventRecord.getKinesis();
                curRecordSequenceNumber = kinesisRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse(batchItemFailures);
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
```

```
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Java 2.x

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Java SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für Amazon Lex unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Lex Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Beispiele für Amazon Location unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Location Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Location

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Location Service beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    public static void main(String[] args) {
        final String usage = """

            Usage:
```

```
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list geofences
 from.
     * @return A {@link CompletableFuture} representing the result of the
 asynchronous operation.
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListGeofenceCollections
  - ListGeofences

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Location-Karte.
- Erstellen Sie einen API-Schlüssel für Amazon Location.
- Zeigen Sie die Karten-URL an.
- Erstellen Sie eine Geofence-Sammlung.
- Speichern einer Geofence-Geometrie
- Erstellen einer Tracker-Ressource
- Aktualisieren der Position eines Geräts
- Rufen Sie die neueste Positionsaktualisierung für ein bestimmtes Gerät ab.
- Erstellen Sie einen Routenrechner.
- Ermitteln der Entfernung zwischen Seattle und Vancouver
- Verwenden Sie Amazon-Location-APIs höherer Ebene.
- Löschen Sie die Amazon-Location-Komponenten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features von Amazon Location Service zeigt.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """

            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }
```

```
        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
        String trackerName = args[4];
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
            AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
            provides location-based services for developers. This service simplifies
            the integration of location-based features into applications, making it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.

            Tracking: The Location Service enables real-time tracking of mobile
devices,\s
            assets, or other entities, allowing developers to build applications\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),\s
            and vice versa, enabling developers to integrate location-based search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
            logger.info(e.getMessage());
        }
    }
```

```
   public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
       logger.info(DASHES);
       logger.info("1. Create a map");
       logger.info("""
            An AWS Location map can enhance the user experience of your
            application by providing accurate and personalized location-based
            features. For example, you could use the geocoding capabilities to
            allow users to search for and locate businesses, landmarks, or
            other points of interest within a specific region.
           """);

       waitForInputToContinue(scanner);
       String mapArn;
       try {
           mapArn = locationActions.createMap(mapName).join();
           logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof ServiceQuotaExceededException) {
               logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
           } else {
               logger.error("An unexpected error occurred while creating the map.",
cause);
           }
           return;
       }
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       logger.info(DASHES);
       logger.info("2. Create an AWS Location API key");
       logger.info("""
           When you embed a map in a web app or website, the API key is
           included in the map tile URL to authenticate requests. You can
           restrict API keys to specific AWS Location operations (e.g., only
           maps, not geocoding). API keys can expire, ensuring temporary
           access control.
           """);

       try {
```

```
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
 key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName+"/
tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
```

```
            } else {
                logger.error("An unexpected error occurred while creating the
geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a geographic
area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.

            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating geofence:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn = locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for deviceId,
but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
```

```
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
```

```
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver using
the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
            logger.info("Successfully calculated route. The distance in kilometers
is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
```

```
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.
           """);

       logger.info("First we will perform a Reverse Geocoding operation");
       waitForInputToContinue(scanner);
       try {
           locationActions.reverseGeocode().join();
           logger.info("Now we are going to perform a text search using coffee
shop.");
           waitForInputToContinue(scanner);
           locationActions.searchText("coffee shop").join();
           waitForInputToContinue(scanner);

           logger.info("Now we are going to perform a nearby Search.");
           //waitForInputToContinue(scanner);
           locationActions.searchNearBy().join();
           waitForInputToContinue(scanner);
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
               logger.error("A validation error occurred: {}", cause.getMessage(),
cause);
           } else {
               logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
           }
           return;
       }
       logger.info(DASHES);

       logger.info("12. Delete the AWS Location Services resources.");
       logger.info("Would you like to delete the AWS Location Services resources?
(y/n)");
       String delAns = scanner.nextLine().trim();
       if (delAns.equalsIgnoreCase("y")) {
           cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
       } else {
           logger.info("The AWS resources will not be deleted.");
       }
       waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence collection,
tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
```

```
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon Location Service SDK.

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
```

```
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();
```

```
        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
```

```
                    throw new CompletionException("Error performing place search",
exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
     * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
     * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
     */
    public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

```
   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
               }
           });
   }


   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
```

```
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
               }
           });
   }


   /**
    * Retrieves the position of a device using the provided LocationClient.
    *
    * @param trackerName The name of the tracker associated with the device.
    * @param deviceId    The ID of the device to retrieve the position for.
    * @throws RuntimeException If there is an error fetching the device position.
    */
   public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
       GetDevicePositionRequest request = GetDevicePositionRequest.builder()
           .trackerName(trackerName)
           .deviceId(deviceId)
           .build();

       return getClient().getDevicePosition(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;

        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
```

```
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }


    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }



    /**
```

```
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
```

```
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();
```

```
        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();
```

```
        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
```

```
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }




    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }



    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
```

```
    * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
    */
   public CompletableFuture<Void> deleteMap(String mapName) {
       DeleteMapRequest mapRequest = DeleteMapRequest.builder()
           .mapName(mapName)
           .build();

       return getClient().deleteMap(mapRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The map was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
               }
               logger.info("The map {} was deleted.", mapName);
           })
           .thenApply(response -> null);
   }


   /**
    * Deletes a tracker with the specified name.
    *
    * @param trackerName the name of the tracker to be deleted
    * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
    * @throws CompletionException if an error occurs while deleting the tracker
    *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
    *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
    */
   public CompletableFuture<Void> deleteTracker(String trackerName) {
       DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
           .trackerName(trackerName)
           .build();

       return getClient().deleteTracker(trackerRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }


    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
     * @throws CompletionException if an error occurs while deleting the route
calculator
     *                             - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
     *                             - If any other error occurs, a generic {@link
CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
```

```
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Aktionen

`BatchUpdateDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`BatchUpdateDevicePosition`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();

       CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof ResourceNotFoundException) {
```

```
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }
```

- API-Details finden Sie BatchUpdateDevicePosition in der AWS SDK for Java 2.x API-Referenz.

`CalculateRoute`

Das folgende Codebeispiel zeigt die Verwendung`CalculateRoute`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
```

```
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie CalculateRoute in der AWS SDK for Java 2.x API-Referenz.

`CreateGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateGeofenceCollection`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
```

```
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }
```

- API-Details finden Sie CreateGeofenceCollection in der AWS SDK for Java 2.x API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- API-Details finden Sie CreateKey in der AWS SDK for Java 2.x API-Referenz.

`CreateMap`

Das folgende Codebeispiel zeigt die Verwendung`CreateMap`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
```

```
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- API-Details finden Sie CreateMap in der AWS SDK for Java 2.x API-Referenz.

`CreateRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteCalculator`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
```

```
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie CreateRouteCalculator in der AWS SDK for Java 2.x API-Referenz.

`CreateTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateTracker`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();
```

```
        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }
```

- API-Details finden Sie CreateTracker in der AWS SDK for Java 2.x API-Referenz.

`DeleteGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGeofenceCollection`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
```

```
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

- API-Details finden Sie DeleteGeofenceCollection in der AWS SDK for Java 2.x API-Referenz.

`DeleteKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKey`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes the specified key from the key-value store.
```

```
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }
```

- API-Details finden Sie DeleteKey in der AWS SDK for Java 2.x API-Referenz.

`DeleteMap`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMap`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

- API-Details finden Sie DeleteMap in der AWS SDK for Java 2.x API-Referenz.

`DeleteRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRouteCalculator`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                             - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
    *                             - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The route calculator was not
found.", cause);
                   }
                   throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
               }
               logger.info("The route calculator {} was deleted.", calcName);
           })
           .thenApply(response -> null);
```

```
    }
```

- API-Details finden Sie DeleteRouteCalculator in der AWS SDK for Java 2.x API-Referenz.

`DeleteTracker`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTracker`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes a tracker with the specified name.
    *
    * @param trackerName the name of the tracker to be deleted
    * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
    * @throws CompletionException if an error occurs while deleting the tracker
    *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
    *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
    */
   public CompletableFuture<Void> deleteTracker(String trackerName) {
       DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
           .trackerName(trackerName)
           .build();

       return getClient().deleteTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The tracker was not found.",
cause);
```

```
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- API-Details finden Sie DeleteTracker in der AWS SDK for Java 2.x API-Referenz.

`GetDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`GetDevicePosition`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie GetDevicePosition in der AWS SDK for Java 2.x API-Referenz.

`PutGeofence`

Das folgende Codebeispiel zeigt die Verwendung`PutGeofence`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
```

```
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API-Details finden Sie PutGeofence in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Location Service Places unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Location Service Places Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ReverseGeocode`

Das folgende Codebeispiel zeigt die Verwendung`ReverseGeocode`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);
```

```
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
 cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
 exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- API-Details finden Sie ReverseGeocode in der AWS SDK for Java 2.x API-Referenz.

`SearchNearby`

Das folgende Codebeispiel zeigt die Verwendung`SearchNearby`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
```

```
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- API-Details finden Sie SearchNearby in der AWS SDK for Java 2.x API-Referenz.

`SearchText`

Das folgende Codebeispiel zeigt die Verwendung`SearchText`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Searches for a place using the provided search query and prints the detailed
information of the first result.
    *
    * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
    */
   public CompletableFuture<Void> searchText(String searchQuery) {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       List<Double> queryPosition = List.of(longitude, latitude);

       SearchTextRequest request = SearchTextRequest.builder()
               .queryText(searchQuery)
               .biasPosition(queryPosition)
               .build();

       return getGeoPlacesClient().searchText(request)
               .thenCompose(response -> {
                   if (response.resultItems().isEmpty()) {
                       logger.info("No places found.");
                       return CompletableFuture.completedFuture(null);
                   }

                   // Get the first place ID
                   String placeId = response.resultItems().get(0).placeId();
                   logger.info("Found Place with id: " + placeId);

                   // Fetch detailed info using getPlace
                   GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                           .placeId(placeId)
                           .build();
```

```
                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }
```

- API-Details finden Sie SearchText in der AWS SDK for Java 2.x API-Referenz.

AWS Marketplace Katalogisieren Sie API-Beispiele mithilfe des SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK for Java 2.x with AWS Marketplace Catalog API Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- AMI-Produkte
- Angebote von Channel-Partnern
- Container-Produkte
- Entitäten
- Angebote
- Produkte
- Wiederverkaufsgenehmigung
- SaaS--Produkte
- Dienstprogramme

AMI-Produkte

Hinzufügen einer Dimension zu einem vorhandenen AMI-Produkt und Aktualisieren der Preisbedingungen für das Angebot

Das folgende Codebeispiel zeigt, wie Sie eine Dimension zu einem vorhandenen AMI-Produkt hinzufügen und die Preisbedingungen für das Angebot aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
```

```
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Hinzufügen einer Region, in der ein AMI-Produkt bereitgestellt wird

Das folgende Codebeispiel zeigt, wie Sie eine Region hinzufügen, in der Sie ein AMI-Produkt bereitstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
```

```
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher jährlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
```

```
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
```

```
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
```

```
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher monatlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
```

```
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
```

```
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines AMI-Produktentwurf mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen AMI-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Einschränken einer Region, in der ein AMI-Produkt bereitgestellt wird

Das folgende Codebeispiel zeigt, wie Sie eine Region einschränken, in der Sie ein AMI-Produkt bereitstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
```

```
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Beschränken der Produktsichtbarkeit

Das folgende Codebeispiel zeigt, wie Sie die Sichtbarkeit von Produkten einschränken.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Angeben, ob AMI-Komponenten in neuen Regionen bereitgestellt werden

Das folgende Codebeispiel zeigt, wie angegeben wird, ob AMI-Assets in neuen Regionen bereitgestellt werden, die von AWS zur Unterstützung zukünftiger Regionen erstellt wurden.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Angebote von Channel-Partnern

Erstellen eines CPPO-Entwurfs für einen beliebigen Produkttyp

Das folgende Codebeispiel zeigt, wie Sie einen CPPO-Entwurf für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn für Käufer veröffentlichen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Ersatzangebot zur Wiederverkaufsgenehmigung mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein privates Ersatzangebot zur Genehmigung des Wiederverkaufs aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-1111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
```

```
                "Description": "Test private resale replacement offer with contract
pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Auflisten aller CPPOs, die von einem Channel-Partner erstellt wurden

Das folgende Codebeispiel zeigt, wie Sie alle CPPOs auflisten, die von einem Channel-Partner erstellt wurden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();
```

```
 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
  entityIdList.add(entitySummary.entityId());
 }

 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }
 }

 // filter for CPPO offers: ResaleAuthorizationId exists in Details

 List<String> cppoOfferIds = new ArrayList<String>();

 for (String entityId : entityIdList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entityId)
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);

  Document resaleAuthorizationDocument =
describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
  String resaleAuthorizationId = resaleAuthorizationDocument != null ?
resaleAuthorizationDocument.asString() : "";

  if (!resaleAuthorizationId.isEmpty()) {
      cppoOfferIds.add(resaleAuthorizationId);
  }
```

```
  }
  return cppoOfferIds;
 }

}
```

- API-Details finden Sie ListEntities in der AWS SDK for Java 2.x API-Referenz.

Auflisten aller gemeinsamen Genehmigungen des Wiederverkaufs, die einem Channel-Partner zur Verfügung stehen

Das folgende Codebeispiel zeigt, wie Sie alle gemeinsamen Genehmigung des Wiederverkaufs auflisten, die einem Channel-Partner zur Verfügung stehen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {
```

```
/*
 * list all resale authorizations shared to an account
 */
public static void main(String[] args) {

 List<ListEntitiesResponse> responseList = getListEntityResponseList();
 ReferenceCodesUtils.formatOutput(responseList);
}

public static List<ListEntitiesResponse> getListEntityResponseList() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 List<ListEntitiesResponse> responseList = new ArrayList<ListEntitiesResponse>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
   .maxResults(10)
   .ownershipType(OWNERSHIP_TYPE_SHARED)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 responseList.add(listEntitiesResponse);

 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest = ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(listEntitiesResponse.nextToken())
    .build();

  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
```

```
    responseList.add(listEntitiesResponse);
   }
   return responseList;
  }

}
```

- API-Details finden Sie ListEntities in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen eines CPPO und Anhängen einer Käufer-EULA

Das folgende Codebeispiel zeigt, wie Sie ein CPPO veröffentlichen und eine Käufer-EULA anhängen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
                 "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                }
               ]
           }
        ]
          }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
```

```
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen eines CPPO mithilfe einer einmaligen Wiederverkaufsgenehmigung und Aktualisieren des Preisaufschlags

Das folgende Codebeispiel zeigt, wie ein CPPO mithilfe einer einmaligen Wiederverkaufsgenehmigung für AMI-, SaaS- oder Container-Produkte veröffentlicht und der Preisaufschlag aktualisiert wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen eines CPPO-Entwurfs und Aktualisieren des Preisaufschlags

Das folgende Codebeispiel zeigt, wie Sie einen CPPO-Entwurf veröffentlichen und den Preisaufschlag aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren des Ablaufdatums eines CPPO

Das folgende Codebeispiel zeigt, wie das Ablaufdatum eines CPPO aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Container-Produkte

Erstellen eines Container-Produktentwurf mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen Container-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
```

```
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines limitierten Container-Produkts mit einem öffentlichen Angebot und Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein limitiertes Containerprodukt mit einem öffentlichen Angebot, Vertragspreisen und Standard-EULA erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Entitäten

Beschreiben aller Entitäten in einem einzigen Anruf

Das folgende Codebeispiel zeigt, wie Sie alle Entitäten in einem einzigen Aufruf beschreiben.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {

    /*
     * BatchDescribe my entities in a single call and
     *  check if it contains all the information I need to know about the entities.
     */
    public static void main(String[] args) {

        MarketplaceCatalogClient marketplaceCatalogClient =
                MarketplaceCatalogClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
                BatchDescribeEntitiesRequest.builder()
                        .entityRequestList(Arrays.asList(
                                EntityRequest.builder()
                                        .catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                        .build(),
                                EntityRequest.builder()

 .catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
                                        .build()))
                        .build();

        BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
 marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

        // Reading the successful entities response
        Map<String, EntityDetail> entityDetailsMap =
 batchDescribeEntitiesResponse.entityDetails();
```

```
        for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet()) {
            System.out.println("EntityId: " + entry.getKey());
            ReferenceCodesUtils.formatOutput(entry.getValue());
        }

        // Logging the failed entities error details
        Map<String, BatchDescribeErrorDetail> entityErrorsMap =
 batchDescribeEntitiesResponse.errors();
        for (Map.Entry<String, BatchDescribeErrorDetail> entry :
 entityErrorsMap.entrySet()) {
            System.out.println(String.format("EntityId: %s, ErrorCode: %s,
 ErrorMessage: %s", entry.getKey(),
                    entry.getValue().errorCode(), entry.getValue().errorMessage()));
        }
    }
}
```

- API-Details finden Sie BatchDescribeEntities in der AWS SDK for Java 2.x API-Referenz.

Auflisten und Beschreiben aller Angebote, die einem Produkt zugeordnet sind

Das folgende Codebeispiel zeigt, wie Sie alle mit einem Produkt verbundenen Angebote auflisten und beschreiben.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
 /*
  * retrieve all private offer information related to a single product
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryList();

  // for each offer id, output the offer detail using DescribeEntity API

  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   ReferenceCodesUtils.formatOutput(describeEntityResponse);
```

```
 }
}
public static List<EntitySummary> getEntitySummaryList() {
 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(50)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(50)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
```

```
    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
   }
   return entitySummaryList;
  }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DescribeEntity
  - ListEntities

Angebote

Erstellen einer benutzerdefinierten Dimension für ein SaaS-Produkt und Erstellen eines privaten Angebots

Das folgende Codebeispiel zeigt, wie Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt erstellen und ein privates Angebot erstellen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebotsentwurfs für ein AMI- oder SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie einen privaten Angebotsentwurf für ein beliebiges AMI- oder SaaS-Produkt erstellen, sodass Sie es intern überprüfen können, bevor Sie es für Käufer veröffentlichen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
```

```
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit Vertragspreisen für ein Container-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
 Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
 contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit Vertragspreisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
```

```
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit stündlichen jährlichen Preisen und einem flexiblen Zahlungsplan für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen und einem flexiblen Zahlungsplan für ein AMI-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit stündlichen jährlichen Preisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen für ein AMI-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.17"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit stündlichen Preisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen Preisen für ein AMI-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
```

```
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit Abonnementpreisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
```

```
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Angebots mit gestaffelten Preisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit gestaffelten Preisen für ein SaaS-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen kostenlosen Testangebots mit Abonnementpreisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
```

```
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines privaten Ersatzangebots mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein privates Ersatzangebot aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
 for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementEndDate": "2024-01-30"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "PaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
                            "ChargeDate": "2024-01-01",
                            "ChargeAmount": "0"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Beschreiben eines öffentlichen Angebots

Das folgende Codebeispiel veranschaulicht, wie Sie ein öffentliches Angebot beschreiben.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
```

```
 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- API-Details finden Sie DescribeEntity in der AWS SDK for Java 2.x API-Referenz.

Ablaufen des Entwurfs eines privaten Angebots

Das folgende Codebeispiel zeigt, wie Sie das Ablaufdatum eines privaten Angebots auf ein Datum in der Vergangenheit festlegen, sodass Käufer das Angebot nicht mehr sehen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Auflisten aller privaten Angebote

Das folgende Codebeispiel zeigt, wie Sie alle privaten Angebote auflisten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterDate
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date, Offer
 Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 public static void main(String[] args) {

  String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
```

```
 String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

 List<EntitySummary> entitySummaryList =
getEntitySummaryList(offerReleaseDateAfterValue, offerAvailableEndDateAfterValue);

 // for each offer id, output the offer detail using DescribeEntity API


 for (EntitySummary entitySummary : entitySummaryList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
       .build())
     .releaseDate(OfferReleaseDateFilter.builder()
       .dateRange(OfferReleaseDateFilterDateRange.builder()
         .afterValue(offerReleaseDateAfterValue)
         .build())
       .build())
     .build())
   .build();
```

```
  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER).maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }

  return entitySummaryList;
 }

}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Auflisten freigegebener öffentlicher und privater Angebote für eine bestimmte Produkt-ID

Das folgende Codebeispiel zeigt, wie Sie freigegebene öffentliche und private Angebote für eine bestimmte Produkt-ID auflisten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
  * To change to released private offers, change OFFER_TARGETING_NONE (None) to
 OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
  */
 public static void main(String[] args) {
```

```
 List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
 ReferenceCodesUtils.formatOutput(entitySummaryList);
}

public static List<EntitySummary> getEntitySummaryLIst() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_NONE)
       .build())
     .state(OfferStateFilter.builder()
       .valueListWithStrings(OFFER_STATE_RELEASED)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER)
   .maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();
```

```
  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .entityTypeFilters(entityTypeFilters)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren Sie ein Angebot, um einen Vertrag mit Pay-As-You-Go Preisangaben abzuschließen

Das folgende Codebeispiel zeigt, wie ein Angebot aktualisiert wird, um einen Vertrag mit Pay-As-You-Go Preisangaben abzuschließen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
```

```
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren eines Angebots, um einen stündlichen jährlichen Preis anzuwenden

Das folgende Codebeispiel zeigt, wie Sie ein Angebot aktualisieren, um einen stündlichen jährlichen Preis anzuwenden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
```

```
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren eines Angebots, um das Targeting auf bestimmte geografische Regionen anzuwenden

Das folgende Codebeispiel zeigt, wie Sie ein Angebot aktualisieren, um das Targeting auf bestimmte geografische Regionen anzuwenden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
```

```
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren des Namens und der Beschreibung eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie den Namen und die Beschreibung eines öffentlichen Angebots aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der EULA eines Angebots

Das folgende Codebeispiel zeigt, wie Sie die EULA eines Angebots aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren des Ablaufdatums eines privaten Angebots auf ein zukünftiges Datum

Das folgende Codebeispiel zeigt, wie das Ablaufdatum eines privaten Angebots auf ein zukünftiges Datum aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der Dauer der kostenlosen Testversion eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie die Dauer der kostenlosen Testversion eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
```

```
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der Rückerstattungsrichtlinie eines Angebots

Das folgende Codebeispiel zeigt, wie Sie die Rückerstattungsrichtlinie eines Angebots aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Produkte

Beschreiben eines AMI-, SaaS- oder Container-Produkts

Das folgende Codebeispiel zeigt, wie Sie ein AMI-, SaaS- oder Container-Produkt beschreiben und prüfen, ob es alle Informationen enthält, die Sie über das Produkt haben möchten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- API-Details finden Sie DescribeEntity in der AWS SDK for Java 2.x API-Referenz.

Auflisten aller AMI-, SaaS- oder Container-Produkte und zugehöriger öffentlicher Angebote

Das folgende Codebeispiel zeigt, wie Sie alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
```

```
 * List all my AMI or SaaS or Container products and associated public offers
 */
public static void main(String[] args) {

 Map<String, List<EntitySummary>> allProductsWithOffers =
getAllProductsWithOffers();

 ReferenceCodesUtils.formatOutput(allProductsWithOffers);
}

public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
List<EntitySummary>> ();

 // get all product entities
 List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(PRODUCT_TYPE_AMI)
   .maxResults(10)
   .nextToken(null)
   .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
```

```
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }

 // loop through each product entity and get the public released offers associated
using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();

  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

  offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
```

```
    while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
     listEntitiesRequest =
       ListEntitiesRequest.builder()
       .catalog(AWS_MP_CATALOG)
       .entityType(ENTITY_TYPE_OFFER)
       .maxResults(10)
       .entityTypeFilters(entityTypeFilters)
       .nextToken(listEntitiesResponse.nextToken())
       .build();
     listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
     offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
    }

    // save final results into map; key = product id; value = offer entity summary
 list

    allProductsWithOffers.put(productEntitySummary.entityId(),
 offerEntitySummaryList);
   }
   return allProductsWithOffers;
 }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DescribeEntity
  - ListEntities

Wiederverkaufsgenehmigung

Erstellen eines Entwurfs einer Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn bei einem Vertriebspartner veröffentlichen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Beschreiben einer Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie eine Wiederverkaufsgenehmigung beschreiben.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
```

```
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- API-Details finden Sie DescribeEntity in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaliger Wiederverkaufsgenehmigung mit einem privaten Angebot

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung mit einem privaten Angebot veröffentlichten sodass ein Channel-Partner damit ein Channel Partner Private Offer (CPPO) erstellen kann.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum

Das folgende Codebeispiel zeigt, wie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlicht wird, sodass ein Channel-Partner damit ein CPPO erstellen kann.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
```

```
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum und einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum und von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit Ablauf und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
```

```
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateBuyerTargetingTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "BuyerTargetingTerm",
                    "PositiveTargeting": {
                        "BuyerAccounts": [
                            "111111111111"
                        ]
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne ein Ablaufdatum

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne ein Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlichen, sodass ein Channel-Partner damit ein CPPO erstellen kann.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablaufdatum und einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablaufdatum und von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablauf und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
```

```
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen eines flexiblen Zahlungsplans

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und einen flexiblen Zahlungsplan hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
```

```
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
```

```
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen, ob es sich um eine Verlängerung handelt

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und hinzufügen, ob es sich um eine Verlängerung handelt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
```

```
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Beschränken der Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie eine Wiederverkaufsgenehmigung beschränken.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren des Namens und der Beschreibung der einmaligen oder mehrfach verwendbaren Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie den Namen und die Beschreibung einer einmaligen oder mehrfach verwendbaren Wiederverkaufsgenehmigung vor der Veröffentlichung für einen beliebigen Produkttyp aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
```

```
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

SaaS--Produkte

Erstellen eines SaaS-Produktentwurfs mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen SaaS-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen oder limitierten SaaS-Produkts und eines öffentlichen Angebots mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit nutzungsabhängiger Preisberechnung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
```

```
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
```

```
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
```

```
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen Sie ein öffentliches oder beschränktes SaaS-Produkt und ein öffentliches Angebot mit Vertrag und Pay-As-You-Go Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder beschränktes SaaS-Produkt und ein öffentliches Angebot mit einem Vertrag mit Pay-As-You-Go Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
```

```
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
```

```
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
```

```
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Erstellen eines öffentlichen oder limitierten SaaS-Produkts und eines öffentlichen Angebots mit Abonnementpreisen

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen erstellen. In diesen Beispielen wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
```

```
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
```

```
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen eines SaaS-Produkts und des zugehörigen öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot veröffentlichen. Das Produkt befindet sich standardmäßig in einem eingeschränkten Zustand.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
```

```
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
```

```
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen eines SaaS-Produkts und des zugehörigen öffentlichen Angebots anhand eines bestehenden Entwurfs

Das folgende Codebeispiel zeigt, wie Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot anhand eines bestehenden Entwurfs veröffentlichen. Das Produkt befindet sich standardmäßig in einem eingeschränkten Zustand.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren der Dimensionen auf einem AMI- oder SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie die Dimensionen auf einem AMI- oder SaaS-Produkt aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Um dieses Beispiel auszuführen, übergeben Sie das folgende JSON-Changeset an `RunChangesets` in Dienstprogramme, um ein Changeset zu starten aus dem Abschnitt Dienstprogramme.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

Dienstprogramme

Dienstprogramme zum Starten eines Changesets

Das folgende Codebeispiel zeigt, wie Sie Dienstprogramme zum Starten eines Changesets definieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Dienstprogramm zum Laden eines Changesets aus einer JSON-Datei und zum Starten der Verarbeitung.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);

  // process the changeset request
  try {
   StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
   ReferenceCodesUtils.formatOutput(result);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
```

```
public static StartChangeSetResponse getChangeSetRequestResult(String
changeSetsInput) throws IOException {

 //set up AWS credentials
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 //changeset list to save all the changesets in the changesets file
 List<Change> changeSetLists = new ArrayList<Change>();

 // read all changesets into object
 Root root = GSON.fromJson(changeSetsInput, Root.class);

 // process each changeset and add each changeset request to changesets list
 for (ChangeSet cs : root.changeSet) {

  ChangeSetEntity entity = cs.Entity;
  String entityType = entity.Type;
  String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
  Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

  Entity awsEntity =
    Entity.builder()
    .type(entityType)
    .identifier(entityIdentifier)
    .build();

  Change inputChangeRequest =
    Change.builder()
    .changeType(cs.ChangeType)
    .changeName(cs.ChangeName)
    .entity(awsEntity)
    .detailsDocument(detailsDocument)
    .build();

  changeSetLists.add(inputChangeRequest);
 }

 // process all changeset requests
 StartChangeSetRequest startChangeSetRequest =
   StartChangeSetRequest.builder()
```

```
    .catalog(root.catalog)
    .changeSet(changeSetLists)
    .build();

  StartChangeSetResponse result =
 marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

  return result;
 }

 public static Document getDocumentFromObject(Object detailsObject) {

  String detailsString = "{}";
  try {
   detailsString = IOUtils.toString(new
 ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  JsonNodeParser jsonNodeParser = JsonNodeParser.create();
  Document doc = jsonNodeParser.parse(detailsString).visit(new
 DocumentUnmarshaller());
  return doc;
 }


 public static String readChangeSetToString (String inputChangeSetFile) {

  InputStream changesetInputStream =
 RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

  String changeSetsInput = null;

  try {
   changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
  } catch (IOException e) {
   e.printStackTrace();
  }

  return changeSetsInput;

 }
}
```

- API-Details finden Sie StartChangeSet in der AWS SDK for Java 2.x API-Referenz.

AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK for Java 2.x with AWS Marketplace Agreement API Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Vereinbarungen

Vereinbarungen

Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung akzeptiert wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementCancellationReque

public class AcceptAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        acceptAgreementCancellationRequest();
    }

    private static void acceptAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementCancellationRequestRequest request =
                AcceptAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        AcceptAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.acceptAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Description: " + response.description());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- API-Details finden Sie AcceptAgreementCancellationRequest in der AWS SDK for Java 2.x API-Referenz.

Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung akzeptiert wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementPaymentRequestRes

public class AcceptAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER
 REFERENCE HERE>";

    public static void main(String[] args) {
        acceptAgreementPaymentRequest();
    }
```

```
    private static void acceptAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        AcceptAgreementPaymentRequestRequest request =
                AcceptAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                        .build();

        AcceptAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.acceptAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- API-Details finden Sie AcceptAgreementPaymentRequest in der AWS SDK for Java 2.x API-Referenz.

Ändern Sie die Verlängerungsdauer einer SaaS-Vertragsvereinbarung

Das folgende Codebeispiel zeigt, wie Sie eine SaaS-Vertragsvereinbarung ändern, um ihre Verlängerungsdauer hinzuzufügen oder zu aktualisieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RenewalTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
 * the auto-renewal setting using the AWS Marketplace Agreement Service APIs.
 *
```

```
 * <p>Scenario: A buyer subscribes to a SaaS product using a public offer that
 supports
 * auto-renewal. After acceptance, the buyer decides to amend the agreement to
 enable
 * auto-renewal via the RenewalTerm configuration.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P1M} for
 1 month).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer.</li>
 * </ul>
 */
public class AmendSaaSContractRenewalTerm {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the RenewalTerm in your offer.
    private static final String RENEWAL_TERM_ID = "<your-renewal-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";
```

```
    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    public static void main(String[] args) {
        amendSaaSContractAgreementRenewalTerm();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
disabled.
     * 2. Wait for entitlements to become active.
     * 3. Amend the agreement to enable auto-renewal.
     */
    private static void amendSaaSContractAgreementRenewalTerm() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        // Initial agreement: auto-renewal disabled.
        RequestedTerm renewalTerm = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)

.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(false)
                                .build()))
                .build();
```

```
        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm supportTerm =
RequestedTerm.builder().id(SUPPORT_TERM_ID).build();

        // --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, renewalTerm,
legalTerm, supportTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Amend: enable auto-renewal ---
        RequestedTerm renewalTermAmended = RequestedTerm.builder()
                .id(RENEWAL_TERM_ID)
```

```
.configuration(RequestedTermConfiguration.fromRenewalTermConfiguration(
                        RenewalTermConfiguration.builder()
                                .enableAutoRenew(true)
                                .build()))
                .build();

        // Use Intent.AMEND and sourceAgreementIdentifier to target the existing
agreement.
        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(configurableUpfrontPricingTerm,
renewalTermAmended, legalTerm, supportTerm)

.sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Amend agreement request created. AgreementRequestId: " +
carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Amendment accepted. Auto-renewal enabled. New
AgreementId: " + aarResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Ändern Sie ein AMI ConfigurableUpfrontPricingTerm für das Nutzungspreismodell

Das folgende Codebeispiel zeigt, wie eine AMI-Vereinbarung mit geändert werden kann, ConfigurableUpfrontPricingTerm indem das Modell für die Dimensionsmenge für das Nutzungspreismodell aktualisiert wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm
 and then amend the dimension quantity
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
```

```
 *   <li>An agreement with <b>usageBasedPricingTerm (UBPT)</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> — accepted
 after the UBPT agreement entitlements are active.</li>
 * </ol>
 * Once both agreement entitlements are available, this sample shows how to
 <b>amend</b> the agreement
 * with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} —  duration for the agreement
 *       (e.g., {@code P365D} for one year).</li>
 *   <li>{@code DIMENSION_1_KEY} — the dimension key defined in the offer (e.g.,
 instance type).</li>
 *   <li>{@code DIMENSION_1_VALUE} — initial quantity; {@code NEW_DIMENSION_1_VALUE}
 — amended quantity.</li>
 * </ul>
 */
public class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Initial quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;
```

```
    // Term ID for the UsageBasedPricingTerm in your offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // New quantity to use when amending the dimension of CUPT.
    private static final int NEW_DIMENSION_1_VALUE = 5;

    public static void main(String[] args) {
        amendAmiCUPTAgreement();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept an agreement request with usageBasedPricingTerm.
     * 2. Wait for entitlements to become active, then create and accept an
agreement request with configurableUpfrontPricingTerm (CUPT).
     * 3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
     */
    private static void amendAmiCUPTAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageTerm = RequestedTerm.builder()
                .id(USAGE_TERM_ID)
                .build();
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();
        RequestedTerm validityTerm = RequestedTerm.builder()
                .id(VALIDITY_TERM_ID)
                .build();

        // --- Agreement with UBPT ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
```

```
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageTerm, legalTerm, validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before creating the agreement with
CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                        .dimensionKey(DIMENSION_1_KEY)
```

```
.dimensionValue(DIMENSION_1_VALUE)
                                                        .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created. AgreementRequestId:
" + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

        // Wait for entitlements to become active before amending.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Amend Agreement with CUPT ---
        // Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        RequestedTerm newConfig = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)
```

```
 .configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

 .dimensionValue(NEW_DIMENSION_1_VALUE) // Increase quantity for this dimension key
                                                    .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carAmendRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.AMEND)
                        .requestedTerms(newConfig, legalTerm, validityTerm)
                        .sourceAgreementIdentifier(cuptAgreementId)
                        .build();
        CreateAgreementRequestResponse carAmendResponse =
                marketplaceAgreementClient.createAgreementRequest(carAmendRequest);
        System.out.println("Amendment of CUPT agreement request created.
 AgreementRequestId: " + carAmendResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarAmendRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carAmendResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarAmendResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarAmendRequest);
        System.out.println("Amendment accepted. New AgreementId: " +
 aarAmendResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion

Das folgende Codebeispiel zeigt, wie Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion erstellen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create an AMI Free Trial agreement
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to an AMI product that offers a free trial
 period.
 * The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
```

```
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 * </ul>
 */
public class NewAmiFreeTrial {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in your offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the SupportTerm in your offer.
    private static final String SUPPORT_TERM_ID = "<your-support-term-id>";

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    public static void main(String[] args) {
        createAndAcceptAmiFreeTrialAgreementRequest();
    }

    /**
     * Creates an AMI Free Trial agreement.
     * The FreeTrialPricingTerm grants access at no cost for the trial period.
     * The UsageBasedPricingTerm defines the charges that apply once the trial ends.
     */
    private static void createAndAcceptAmiFreeTrialAgreementRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm freeTrialPricingTerm = RequestedTerm.builder()
                .id(FREE_TRIAL_PRICING_TERM_ID)
                .build();
```

```
        RequestedTerm usageBasedPricingTerm = RequestedTerm.builder()
                .id(USAGE_BASED_PRICING_TERM_ID)
                .build();
        RequestedTerm supportTerm = RequestedTerm.builder()
                .id(SUPPORT_TERM_ID)
                .build();
        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, usageBasedPricingTerm,
 supportTerm, legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
 AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung

Das folgende Codebeispiel zeigt, wie Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung erstellen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import software.amazon.awssdk.services.marketplaceagreement.model.TaxConfiguration;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with CONTRACT pricing model with
 upfront payment
 * using the AWS Marketplace Agreement Service APIs.
```

```
 *
 * <p>Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
 * selecting an agreement duration and specifying quantities for multiple
 dimensions.
 * Tax estimation is enabled at the time of agreement creation.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code P12M} for
 12 months).</li>
 *   <li>{@code DIMENSION_1_KEY}, {@code DIMENSION_2_KEY} — dimension keys defined
 in the offer.</li>
 *   <li>{@code DIMENSION_1_VALUE}, {@code DIMENSION_2_VALUE} — quantities for each
 dimension.</li>
 * </ul>
 */
public class NewSaaSContractWithUpfrontPayment {

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P12M" for 12 months).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // First dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-1-key>";
    private static final int DIMENSION_1_VALUE = 10;

    // Second dimension key and quantity defined in your offer.
    private static final String DIMENSION_2_KEY = "<your-dimension-2-key>";
    private static final int DIMENSION_2_VALUE = 20;
```

```
    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Tax estimation setting: "ENABLED" to include estimated taxes in the
agreement.
    private static final String TAX_ESTIMATION = "ENABLED";

    public static void main(String[] args) {
        createSaaSContractAgreement();
    }

    /**
     * Creates a SaaS agreement with CONTRACT pricing model with configurable
upfront pricing
     * for multiple dimensions and tax estimation.
     */
    private static void createSaaSContractAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build(),
                                            Dimension.builder()
                                                    .dimensionKey(DIMENSION_2_KEY)

.dimensionValue(DIMENSION_2_VALUE)
                                                    .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder()
                .id(LEGAL_TERM_ID)
```

```
                .build();

        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm)

 .taxConfiguration(TaxConfiguration.builder().taxEstimation(TAX_ESTIMATION).build())
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
 createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("SaaS agreement request with CONTRACT pricing model
 accepted. AgreementId: " + acceptAgreementRequestResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Abrufen aller Vereinbarungs-IDs

Das folgende Codebeispiel zeigt, wie Sie alle Vereinbarungs-IDs abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();

  ReferenceCodesUtils.formatOutput(agreementIds);
```

```
}

public static List<String> getAllAgreementIds() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array
 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
```

```
      .nextToken(searchAgreementsResponse.nextToken())
      .filters(searchFilters)
      .build();
    searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
    agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }

  List<String> agreementIds = new ArrayList<String>();
  for (AgreementViewSummary summary : agreementSummaryList) {
   agreementIds.add(summary.agreementId());
  }
  return agreementIds;
 }

}
```

- API-Details finden Sie SearchAgreements in der AWS SDK for Java 2.x API-Referenz.

Abrufen aller Vereinbarungen

Das folgende Codebeispiel zeigt, wie Sie alle Vereinbarungen abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {

 /*
  * Get all purchase agreements for your PartyType (Proposer, or Acceptor)
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAllAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAllAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

        // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return
 agreements where you are the acceptor.
        // Change to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements where you
 are the proposer.
  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_ACCEPTOR).build();

  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();
```

```
  List<Filter> searchFilters = new ArrayList<Filter>();

  searchFilters.addAll(Arrays.asList(partyType, agreementType));

  // Save all results in a list array

  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters).build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- API-Details finden Sie SearchAgreements in der AWS SDK for Java 2.x API-Referenz.

Abrufen der Kunden-ID aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Kunden-ID aus einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {

 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
```

```
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Informieren Sie sich über die Einzelheiten einer Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Details zu einer bestimmten Anfrage zur Kündigung einer Vereinbarung abrufen können.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationRequestR
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementCancellationRequestR

public class GetAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementCancellationRequest();
    }

    private static void getAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementCancellationRequestRequest request =
                GetAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)

 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .build();

        GetAgreementCancellationRequestResponse response =
                marketplaceAgreementClient.getAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- API-Details finden Sie GetAgreementCancellationRequest in der AWS SDK for Java 2.x API-Referenz.

Rufen Sie die Einzelheiten einer Zahlungsanforderung für eine Vereinbarung ab

Das folgende Codebeispiel zeigt, wie Sie Details einer bestimmten Zahlungsaufforderung für eine Vereinbarung abrufen können.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestReques
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementPaymentRequestRespon

public class GetAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";

    public static void main(String[] args) {
        getAgreementPaymentRequest();
    }

    private static void getAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
```

```
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        GetAgreementPaymentRequestRequest request =
                GetAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .build();

        GetAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.getAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge ID: " + response.chargeId());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- API-Details finden Sie GetAgreementPaymentRequest in der AWS SDK for Java 2.x API-Referenz.

Abrufen finanzieller Angaben aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie finanzielle Angaben aus einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
  * Obtain financial details, such as Total Contract Value of the agreement from a
 given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
```

```
  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
   totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen von Einzelheiten zur kostenlosen Testversion aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Einzelheiten zur kostenlosen Testversion aus einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();
```

```
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
  }
  return freeTrialPricingTerms;
 }
}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen von Informationen über eine Vereinbarung

Das folgende Codebeispiel zeigt, wie Informationen über eine Vereinbarung abgerufen werden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {
```

```
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

  ReferenceCodesUtils.formatOutput(describeAgreementResponse);

 }

 public static DescribeAgreementResponse getResponse(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen von Produkt- und Angebotsdetails aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Produkt- und Angebotsdetails aus einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {

  // call Agreement API to get offer and product information for the agreement

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);
```

```
 for (DescribeEntityResponse response : entityResponseList) {
  ReferenceCodesUtils.formatOutput(response);
 }
}

public static List<DescribeEntityResponse> getEntities(String agreementId) {
 List<DescribeEntityResponse> entityResponseList = new
ArrayList<DescribeEntityResponse> ();

 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

 // get offer id for the given agreement

 String offerId = describeAgreementResponse.proposalSummary().offerId();

 // get all the product ids for this agreement

 List<String> productIds = new ArrayList<String>();
 for (Resource resource : describeAgreementResponse.proposalSummary().resources())
{
  productIds.add(resource.id());
 }

 // call Catalog API to get the details of the offer and products

 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
```

```
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId).build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen der EULA einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die EULA einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
```

```
  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
      .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
      .filter(docItem -> docItem.type() != null)
      .forEach(legalEulaArray::add);
  return legalEulaArray;
 }

}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen der Bedingungen für die automatische Verlängerung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Bedingungen für die automatische Verlängerung einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {

 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.renewalTerm() != null &&
 acceptedTerm.renewalTerm().configuration() != null
     && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
    autoRenewal =
 String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue())
    break;
   }
  }
```

```
  return autoRenewal;
 }

}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen der im Rahmen einer Vereinbarung gekauften Dimensionen

Das folgende Codebeispiel zeigt, wie Sie die im Rahmen einer Vereinbarung gekauften Dimensionen abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> dimensionKeys = getDimensionKeys(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionKeys);
 }

 public static List<String> getDimensionKeys(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<String> dimensionKeys = new ArrayList<String>();
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
     List<Dimension> dimensions =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     for (Dimension dimension : dimensions) {
      dimensionKeys.add(dimension.dimensionKey());
     }
    }

   }
  }
  return dimensionKeys;
```

```
 }
}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen der im Rahmen einer Vereinbarung gekauften Instances jeder Dimension

Das folgende Codebeispiel zeigt, wie Sie die im Rahmen einer Vereinbarung gekauften Instances jeder Dimensionen abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
public class GetAgreementTermsDimensionInstances {

 /*
  * get instances of each dimension that buyer has purchased in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionMap);
 }

 public static Map<String, List<Dimension>> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  Map<String, List<Dimension>> dimensionMap = new HashMap<String,
 List<Dimension>>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Dimension> dimensionsList = new ArrayList<Dimension>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    String selectorValue = "";
    if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
      selectorValue =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
     }
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions()) {
```

```
      dimensionsList =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     }
    }
    if (selectorValue.length() > 0) {
     dimensionMap.put(selectorValue, dimensionsList);
    }
   }
  }
  return dimensionMap;
 }
}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen des Zahlungsplans einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Zahlungsplan einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
```

```
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray = getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();

  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
```

```
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
       scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
       scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
       scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
       paymentScheduleArray.add(scheduleMap);
      }
     }
    }
   }
  }
  return paymentScheduleArray;
 }
}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen der Preise pro Dimension in einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Preise pro Dimension in einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<Object> dimensions = new ArrayList<Object>();
```

```
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
    }
    dimensions.add(rateInfo);
   }
  }
  return dimensions;
 }
}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen des Preistyps einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Preistyp einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
```

```
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;

import org.apache.commons.lang3.tuple.Triple;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
```

```
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
 private static final String CFPS = "Contract FPS";
 private static final String CCPFPS = "CCP with FPS";
 private static final String BYOL = "BYOL";
 private static final String FREE = "Free";
 private static final String FTH = "Free Trials and Hourly";

 // Agreement term pricing types
 private static final Set<String> LEGAL = Set.of("LegalTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT =
 Set.of("ConfigurableUpfrontPricingTerm");
 private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
 Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
 private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
 private static final Set<String> RECURRING_PAYMENT =
 Set.of("RecurringPaymentTerm");
 private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
 Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
 private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
 Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
```

```
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED
= Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and offer
types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();

 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
```

```
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
```

```
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
```

```
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE),
"Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).buil

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PURCHA

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());

 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
```

```
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
  return Objects.hash(first, second);
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null || getClass() != obj.getClass())
   return false;
  KeyPair other = (KeyPair) obj;
  return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
 }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {

 Set<String> offerTermTypes = new HashSet<String>();

 DescribeEntityRequest request =
```

```
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse result = marketplaceCatalogClient.describeEntity(request);

  String details = result.details();

  try {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode rootNode = objectMapper.readTree(details);
   JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

   for (JsonNode termNode : termsNode) {
    if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
     offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
    }
   }
  } catch (Exception e) {
   e.printStackTrace();
  }

  return offerTermTypes;

 }

 /*
  * Get all the agreement term types
  */
 public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
 agreementTerm) {
  Set<String> agreementTermTypes = new HashSet<String>();
  try {
   for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
    ObjectMapper objectMapper = new ObjectMapper();
    JsonNode termNode = objectMapper.readTree(getJson(term));
    Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
    while (fieldsIterator.hasNext()) {
     Map.Entry<String, JsonNode> entry = fieldsIterator.next();
     JsonNode value = entry.getValue();
     if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
      agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
     }
```

```
    }
   }
  } catch (Exception e) {
   e.printStackTrace();
  }
  return agreementTermTypes;

 }

 /*
  * make sure all elements in array2 exist in array1
  */
 public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
  for (String element : array2) {
   boolean found = false;
   for (String str : array1) {
    if (element.equals(str)) {
     found = true;
     break;
    }
   }
   if (!found) {
    return false;
   }
  }
  return true;
 }

 /*
  * Find the combinations of the agreement term types for the agreement
  */
 public static Set<String> getMatchedTermTypesCombination(Set<String>
 agreementTermTypes) {
  Set<String> matchedCombination = new HashSet<String>();
  for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
   if (allElementsExist(agreementTermTypes, element)) {
    matchedCombination = element;
   }
  }
  return matchedCombination;
 }

 public static void main(String[] args) {
```

```
 List<AgreementViewSummary> agreements = getAgreementsById();

 for (AgreementViewSummary summary : agreements) {
  String pricingType = "";
  String agreementId = summary.agreementId();
  System.out.println(agreementId);
  String offerId = summary.proposalSummary().offerId();

  //get all pricing term types for the offer in the agreement
  Set<String> offerTermTypes = getOfferTermTypes(offerId);
  String productType = summary.proposalSummary().resources().get(0).type();

  //get all pricing term types for the agreement
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
  GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  Set<String> agreementTermTypes =
getAgreementTermTypes(getAgreementTermsResponse);

  //get matched pricing term type combination set
  Set<String> agreementMatchedTermType =
getMatchedTermTypesCombination(agreementTermTypes);

  //check to see if this agreement pricing term combination needs additional check
on offer pricing terms
  String needToCheckOfferType = needToCheckOfferTermsType(productType,
agreementMatchedTermType);

  // get the pricing type for the agreement based on the product type, agreement
term types and offer term types if needed
  if (needToCheckOfferType != null) {
   Set<String> offerMatchedTermType =
getMatchedTermTypesCombination(offerTermTypes);
   pricingType = getPricingType(productType, agreementMatchedTermType,
offerMatchedTermType);
  } else if (agreementMatchedTermType == LEGAL) {
   pricingType = FREE;
  } else {
   pricingType = getPricingType(productType, agreementMatchedTermType, new
HashSet());
  }
  System.out.println("Pricing type is " + pricingType);
```

```
  }
 }

 private static String getJson(Object result) {
  String json = "";

  try {
   ObjectMapper om = new ObjectMapper();
   om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
   om.registerModule(new JavaTimeModule());
   ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

   json = ow.writeValueAsString(result);
  } catch (JsonProcessingException e) {
   e.printStackTrace();
  }
  return json;
 }

}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen des Produkttyps einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Produkttyp einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
```

```
  List<String> productIds = new ArrayList<String>();
  for (Resource resource : describeAgreementResponse.proposalSummary().resources())
 {
   productIds.add(resource.id() + ":" + resource.type());
  }
  return productIds;
 }
}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen des Status einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Status einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementStatus {

 public static void main(String[] args) {
```

```
 String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

 DescribeAgreementResponse describeAgreementResponse =
getDescribeAgreementResponse(agreementId);

 System.out.println("Agreement status is " + describeAgreementResponse.status());

}

public static DescribeAgreementResponse getDescribeAgreementResponse(String
agreementId) {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
 return describeAgreementResponse;
}

}
```

- API-Details finden Sie DescribeAgreement in der AWS SDK for Java 2.x API-Referenz.

Abrufen der Support-Bedingungen einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Support-Bedingungen einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }
```

```
 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Abrufen der Bedingungen einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Bedingungen einer Vereinbarung abrufen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
```

```
 }

}
```

- API-Details finden Sie GetAgreementTerms in der AWS SDK for Java 2.x API-Referenz.

Listen Sie Anfragen zur Kündigung von Vereinbarungen auf

Das folgende Codebeispiel zeigt, wie Anfragen zur Kündigung von Vereinbarungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementCancellationRequestSumm
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequest
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementCancellationRequest

public class ListAgreementCancellationRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementCancellationRequests();
    }
```

```
    private static void listAgreementCancellationRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        String nextToken = null;

        do {
            ListAgreementCancellationRequestsRequest request =
                    ListAgreementCancellationRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementCancellationRequestsResponse response =

 marketplaceAgreementClient.listAgreementCancellationRequests(request);

            for (AgreementCancellationRequestSummary summary : response.items()) {
                System.out.println("Cancellation Request ID: " +
 summary.agreementCancellationRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Reason Code: " + summary.reasonCodeAsString());
                System.out.println("Agreement Type: " + summary.agreementType());
                System.out.println("Catalog: " + summary.catalog());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- API-Details finden Sie ListAgreementCancellationRequests in der AWS SDK for Java 2.x API-Referenz.

Zahlungsanfragen für Vereinbarungen auflisten

Das folgende Codebeispiel zeigt, wie Zahlungsanforderungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsRequ
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementPaymentRequestsResp
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentRequestSummary;

public class ListAgreementPaymentRequests {

    private static final String PARTY_TYPE = "Proposer";

    public static void main(String[] args) {
        listAgreementPaymentRequests();
    }

    private static void listAgreementPaymentRequests() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
```

```
        String nextToken = null;

        do {
            ListAgreementPaymentRequestsRequest request =
                    ListAgreementPaymentRequestsRequest.builder()
                            .partyType(PARTY_TYPE)
                            .nextToken(nextToken)
                            .build();

            ListAgreementPaymentRequestsResponse response =

 marketplaceAgreementClient.listAgreementPaymentRequests(request);

            for (PaymentRequestSummary summary : response.items()) {
                System.out.println("Payment Request ID: " +
 summary.paymentRequestId());
                System.out.println("Agreement ID: " + summary.agreementId());
                System.out.println("Status: " + summary.statusAsString());
                System.out.println("Name: " + summary.name());
                System.out.println("Charge ID: " + summary.chargeId());
                System.out.println("Charge Amount: " + summary.chargeAmount());
                System.out.println("Currency Code: " + summary.currencyCode());
                System.out.println("Created At: " + summary.createdAt());
                System.out.println("Updated At: " + summary.updatedAt());
                System.out.println("---");
            }

            nextToken = response.nextToken();
        } while (nextToken != null);
    }
}
```

- API-Details finden Sie ListAgreementPaymentRequests in der AWS SDK for Java 2.x API-Referenz.

Lehnen Sie eine Anfrage zur Kündigung einer Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung abgelehnt wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.agreementCancellation;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationReque
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementCancellationReque

public class RejectAgreementCancellationRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT
 CANCELLATION REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementCancellationRequest();
    }

    private static void rejectAgreementCancellationRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementCancellationRequestRequest request =
                RejectAgreementCancellationRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
```

```
 .agreementCancellationRequestId(AGREEMENT_CANCELLATION_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementCancellationRequestResponse response =

 marketplaceAgreementClient.rejectAgreementCancellationRequest(request);

        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Cancellation Request ID: " +
 response.agreementCancellationRequestId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Reason Code: " + response.reasonCodeAsString());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
    }
}
```

- API-Details finden Sie RejectAgreementCancellationRequest in der AWS SDK for Java 2.x API-Referenz.

Lehnen Sie eine Zahlungsanforderung für eine Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung abgelehnt wird.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer.paymentRequest;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequestReq
import
 software.amazon.awssdk.services.marketplaceagreement.model.RejectAgreementPaymentRequestRes

public class RejectAgreementPaymentRequest {

    private static final String AGREEMENT_ID = "<AGREEMENT ID HERE>";
    private static final String PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>";
    private static final String REJECTION_REASON = "<REJECTION REASON HERE>";

    public static void main(String[] args) {
        rejectAgreementPaymentRequest();
    }

    private static void rejectAgreementPaymentRequest() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RejectAgreementPaymentRequestRequest request =
                RejectAgreementPaymentRequestRequest.builder()
                        .agreementId(AGREEMENT_ID)
                        .paymentRequestId(PAYMENT_REQUEST_ID)
                        .rejectionReason(REJECTION_REASON)
                        .build();

        RejectAgreementPaymentRequestResponse response =
                marketplaceAgreementClient.rejectAgreementPaymentRequest(request);

        System.out.println("Payment Request ID: " + response.paymentRequestId());
        System.out.println("Agreement ID: " + response.agreementId());
        System.out.println("Status: " + response.statusAsString());
        System.out.println("Status Message: " + response.statusMessage());
        System.out.println("Name: " + response.name());
        System.out.println("Charge Amount: " + response.chargeAmount());
        System.out.println("Currency Code: " + response.currencyCode());
        System.out.println("Created At: " + response.createdAt());
        System.out.println("Updated At: " + response.updatedAt());
```

```
    }
}
```

- API-Details finden Sie RejectAgreementPaymentRequest in der AWS SDK for Java 2.x API-Referenz.

Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vertragsbasiertes Angebot

Das folgende Codebeispiel zeigt, wie eine SaaS-Vertragsvereinbarung durch ein neues vertragsbasiertes Angebot ersetzt wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing
 model with a new
 * Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
 * Agreement-Based Offer from the seller. The buyer replaces the existing agreement
 * with the new ABO offer, which may include updated pricing terms and payment
 schedule.
 *
 * <p>Note: Unlike other Replace samples, this example starts from an already-active
 * {@code EXISTING_AGREEMENT_ID} rather than creating a new agreement first. Use
 this
 * pattern when you already have an agreement ID and want to replace it directly.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code EXISTING_AGREEMENT_ID} — the agreement ID of the active agreement to
 replace (starts with {@code agmt-}).</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new ABO offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the new offer's term
 list.</li>
 * </ul>
 */
public class ReplaceSaaSContractWithAgreementBasedOffer {

    // The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
    private static final String EXISTING_AGREEMENT_ID = "<your-existing-agreement-
id>";

    // The agreementProposalId from the new Agreement-Based Offer.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the LegalTerm in the new offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";
```

```
    // Term ID for the FixedUpfrontPricingTerm in the new offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the new offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-
term-id>";

    public static void main(String[] args) {
        replaceExistingAgreement();
    }

    /**
     * Replaces an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.
     * Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
     */
    private static void replaceExistingAgreement() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
 RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
 RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();

        // Replace the agreement with the new offer
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(fixedUpfrontPricingTerm,
 paymentScheduleTerm, legalTerm, validityTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(EXISTING_AGREEMENT_ID)
                        .build();
```

```
        CreateAgreementRequestResponse createAgreementRequestResponse =

 marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

 .agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

 marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement replaced with ABO. New AgreementId: " +
 acceptAgreementRequestResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Vertrag mit Verbrauchspreisen

Das folgende Codebeispiel zeigt, wie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing (CCP) ersetzt wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS free trial agreement and then replace it with a
 * paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.
 *
 * <p>Scenario: A buyer first starts a free trial on a SaaS product. Once the trial
 is active,
 * they decide to convert by replacing it with a paid offer that includes a
 * FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create a SaaS free trial agreement with freeTrialPricingTerm.</li>
 *   <li>Wait for freeTrialPricingTerm agreement entitlements to become active.</li>
 *   <li>Replace the free trial agreement with the paid CCP offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 free trial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the paid CCP offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
```

```
 * </ul>
 */
public class ReplaceSaaSFreeTrialWithCCP {

    // The agreementProposalId from the free trial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-
agreement-proposal-identifier>";

    // Term ID for the FreeTrialPricingTerm in the free trial offer.
    private static final String FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-
pricing-term-id>";

    // Term ID for the LegalTerm in the free trial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the paid CCP offer to convert to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-
agreement-proposal-identifier>";

    // Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    private static final String FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-
upfront-pricing-term-id>";

    // Term ID for the PaymentScheduleTerm in the CCP offer.
    private static final String PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-
term-id>";

    // Term ID for the UsageBasedPricingTerm in the CCP offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";

    // Term ID for the ValidityTerm in the CCP offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the CCP offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        createSaaSFreeTrialAndReplaceWithCCP();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS free trial agreement with freeTrialPricingTerm.
```

```
     * 2. Wait for freeTrialPricingTerm agreement entitlements to become active.
     * 3. Replace the free trial agreement with the paid CCP offer.
     */
    private static void createSaaSFreeTrialAndReplaceWithCCP() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
        RequestedTerm freeTrialPricingTerm =
RequestedTerm.builder().id(FREE_TRIAL_PRICING_TERM_ID).build();

        // --- Step 1: Agreement with freeTrialPricingTerm ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(freeTrialPricingTerm, legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with freeTrialPricingTerm accepted.
AgreementId: " + acceptAgreementRequestResponse.agreementId());

        // Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        System.out.println("Waiting for freeTrialPricingTerm agreement entitlements
to become active...");
```

```
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("freeTrialPricingTerm agreement entitlements are now
active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the free
trial agreement.
        RequestedTerm usageBasedPricingTerm =
RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm fixedUpfrontPricingTerm =
RequestedTerm.builder().id(FIXED_UPFRONT_PRICING_TERM_ID).build();
        RequestedTerm paymentScheduleTerm =
RequestedTerm.builder().id(PAYMENT_SCHEDULE_TERM_ID).build();
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(usageBasedPricingTerm,
fixedUpfrontPricingTerm, paymentScheduleTerm, validityTerm, newLegalTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

.sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
+ carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
```

```
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        System.out.println("Agreement with freeTrialPricingTerm replaced with paid
 CCP offer. New AgreementId: " + aarResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Ersetzen Sie eine nutzungsabhängige SaaS-Preisklausel und kündigen Sie die vorherige Vereinbarung

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige SaaS-Preisbedingung ersetzt und die vorherige Vereinbarung gekündigt wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CancelAgreementRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import utils.AgreementApiUtils;

/**
 * Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
 * with a new offer using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
 * The buyer then converts to a different offer by replacing the existing agreement.
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with UBPT.</li>
 *   <li>Wait for entitlements to become active.</li>
 *   <li>Replace the agreement with a new offer using Intent.REPLACE.</li>
 *   <li>Cancel the new agreement using CancelAgreement.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
 * </ul>
 */
public class ReplaceSaaSUsageBasedPricingTermAndCancel {

    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-
pricing-term-id>";
```

```
    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-
based-pricing-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    public static void main(String[] args) {
        replaceSaaSUbptAndCancel();
    }

    /**
     * Full end-to-end flow:
     * 1. Create and accept the initial agreement request with
 UsageBasedPricingTerm.
     * 2. Wait for entitlements to become active.
     * 3. Replace the agreement with a new offer.
     * 4. Cancel the new agreement.
     */
    private static void replaceSaaSUbptAndCancel() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm usageBasedPricingTerm =
 RequestedTerm.builder().id(USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm validityTerm =
 RequestedTerm.builder().id(VALIDITY_TERM_ID).build();
```

```
        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();

        // --- Step 1: Create and accept the initial UBPT agreement request ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(usageBasedPricingTerm, validityTerm,
legalTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
acceptAgreementRequestResponse.agreementId());

        // Wait for entitlements to become active before replacing.
        System.out.println("Waiting for entitlements to become active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient,
acceptAgreementRequestResponse.agreementId());
        System.out.println("Entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Replace the UBPT agreement with a new offer ---
        // Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
agreement.
        RequestedTerm newUsageBasedPricingTerm =
RequestedTerm.builder().id(NEW_USAGE_BASED_PRICING_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();
```

```
        RequestedTerm newLegalTerm =
 RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageBasedPricingTerm, newValidityTerm,
 newLegalTerm)

 .agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)

 .sourceAgreementIdentifier(acceptAgreementRequestResponse.agreementId())
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String agreementIdWithTheNewOffer = aarResponse.agreementId();
        System.out.println("UBPT agreement replaced. New AgreementId: " +
 agreementIdWithTheNewOffer);

        // --- Step 3: Cancel the new agreement ---
        CancelAgreementRequest cancelAgreementRequest =
                CancelAgreementRequest.builder()
                        .agreementId(agreementIdWithTheNewOffer)
                        .build();
        marketplaceAgreementClient.cancelAgreement(cancelAgreementRequest);
        System.out.println("The new agreement has been cancelled. AgreementId: " +
 agreementIdWithTheNewOffer);
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Ersetzen Sie eine nutzungsabhängige AMI-Preisklausel, behalten Sie jedoch die ConfigurableUpfrontPricingTerm

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige AMI-Preisklausel ersetzt und gleichzeitig die bestehende beibehalten wird. ConfigurableUpfrontPricingTerm

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementEntitlementsResponse
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a
 new offer while
 * the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using
 the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: An AMI product with USAGE pricing requires two agreements:
 * <ol>
 *   <li>An agreement with <b>usageBasedPricingTerm</b> — accepted first to
 establish the base agreement.</li>
 *   <li>An agreement with <b>configurableUpfrontPricingTerm (CUPT)</b> — accepted
 after the usageBasedPricingTerm agreement entitlements are active.</li>
 * </ol>
 * <p> This sample shows how to replace only the agreement with
 usageBasedPricingTerm with a
 * new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).
 *
 * <p>Flow:
 * <ol>
 *   <li>Create and accept the initial agreement request with
 usageBasedPricingTerm.</li>
 *   <li>Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT) (after usageBasedPricingTerm agreement entitlements are active).</li>
 *   <li>Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.</li>
 * </ol>
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offers:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 initial offer.</li>
 *   <li>{@code NEW_AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from
 the new offer to replace to.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in each offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement (e.g., {@code
 P365D}).</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the CUPT term.</li>
 * </ul>
 */
public class ReplaceAmiUsageBasedPricingTermButNotCupt {
```

```
    // The agreementProposalId from the initial offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P365D" for 365 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key defined in the CUPT term.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";

    // Quantity for the dimension.
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the UsageBasedPricingTerm in the initial offer.
    private static final String USAGE_TERM_ID = "<your-usage-term-id>";

    // Term ID for the LegalTerm in the initial offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in the initial offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    // The agreementProposalId from the new offer to replace to.
    private static final String NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-
agreement-proposal-identifier>";

    // Term ID for the UsageBasedPricingTerm in the new offer.
    private static final String NEW_USAGE_TERM_ID = "<your-new-usage-term-id>";

    // Term ID for the LegalTerm in the new offer.
    private static final String NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>";

    // Term ID for the ValidityTerm in the new offer.
    private static final String NEW_VALIDITY_TERM_ID = "<your-new-validity-term-
id>";

    public static void main(String[] args) {
        replaceAmiUsageWhenCuptExists();
    }
```

```
   /**
    * Full end-to-end flow:
    * 1. Create and accept an agreement request with usageBasedPricingTerm, then
wait for entitlements.
    * 2. Create and accept an agreement request with CUPT, then wait for
entitlements.
    * 3. Replace the agreement with usageBasedPricingTerm with a new offer
(agreement with CUPT is unaffected).
    */
   private static void replaceAmiUsageWhenCuptExists() {
       MarketplaceAgreementClient marketplaceAgreementClient =
               MarketplaceAgreementClient.builder()
                       .httpClient(ApacheHttpClient.builder().build())
                       .credentialsProvider(ProfileCredentialsProvider.create())
                       .build();

       RequestedTerm usageTerm = RequestedTerm.builder().id(USAGE_TERM_ID).build();
       RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
       RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

       // --- Step 1: Agreement with UBPT ---
       CreateAgreementRequestRequest createAgreementRequestRequest =
               CreateAgreementRequestRequest.builder()
                       .clientToken(AgreementApiUtils.generateClientToken())
                       .intent(Intent.NEW)
                       .requestedTerms(usageTerm, legalTerm, validityTerm)
                       .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                       .build();
       CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
       System.out.println("Agreement request with UBPT created. AgreementRequestId:
" + createAgreementRequestResponse.agreementRequestId());

       AcceptAgreementRequestRequest acceptAgreementRequestRequest =
               AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                       .build();
       AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
```

```
        final String usageAgreementId =
acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request with UBPT accepted. AgreementId: " +
usageAgreementId);

        // Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
        System.out.println("Waiting for UBPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse entitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, usageAgreementId);
        System.out.println("UBPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(entitlementsResponse);

        // --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        CreateAgreementRequestRequest carRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse carResponse =
                marketplaceAgreementClient.createAgreementRequest(carRequest);
        System.out.println("Agreement request with CUPT created. AgreementRequestId:
" + carResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarRequest =
```

```
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarResponse =
                marketplaceAgreementClient.acceptAgreementRequest(aarRequest);
        final String cuptAgreementId = aarResponse.agreementId();
        System.out.println("Agreement request with CUPT accepted. AgreementId: " +
cuptAgreementId);

        // Wait for CUPT agreement entitlements to become active before replacing
usage.
        System.out.println("Waiting for CUPT agreement entitlements to become
active...");
        GetAgreementEntitlementsResponse cuptEntitlementsResponse =
AgreementApiUtils.pollUntilEntitlementsAvailable(
                marketplaceAgreementClient, cuptAgreementId);
        System.out.println("CUPT agreement entitlements are now active.");
        AgreementApiUtils.formatOutput(cuptEntitlementsResponse);

        // --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
---
        RequestedTerm newUsageTerm =
RequestedTerm.builder().id(NEW_USAGE_TERM_ID).build();
        RequestedTerm newLegalTerm =
RequestedTerm.builder().id(NEW_LEGAL_TERM_ID).build();
        RequestedTerm newValidityTerm =
RequestedTerm.builder().id(NEW_VALIDITY_TERM_ID).build();

        // Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
usageBasedPricingTerm agreement only.
        // The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
by this replacement.
        CreateAgreementRequestRequest carReplaceRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.REPLACE)
                        .requestedTerms(newUsageTerm, newLegalTerm, newValidityTerm)

.agreementProposalIdentifier(NEW_AGREEMENT_PROPOSAL_IDENTIFIER)
                        .sourceAgreementIdentifier(usageAgreementId)
                        .build();
        CreateAgreementRequestResponse carReplaceResponse =

marketplaceAgreementClient.createAgreementRequest(carReplaceRequest);
```

```
        System.out.println("Replace agreement request created. AgreementRequestId: "
 + carReplaceResponse.agreementRequestId());

        AcceptAgreementRequestRequest aarReplaceRequest =
                AcceptAgreementRequestRequest.builder()
                        .agreementRequestId(carReplaceResponse.agreementRequestId())
                        .build();
        AcceptAgreementRequestResponse aarReplaceResponse =

 marketplaceAgreementClient.acceptAgreementRequest(aarReplaceRequest);
        System.out.println("Agreement with UBPT replaced. New AgreementId: " +
 aarReplaceResponse.agreementId());
    }
}
```

- API-Details finden Sie CreateAgreementRequest in der AWS SDK for Java 2.x API-Referenz.

Suchen nach Vereinbarungen anhand des Enddatums

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand des Enddatums suchen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
 where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
 the acceptor.
  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();
```

```
 Filter customizeFilter =
Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

 // search agreement with filters

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- API-Details finden Sie SearchAgreements in der AWS SDK for Java 2.x API-Referenz.

Suchen nach Vereinbarungen mit einem benutzerdefinierten Filter

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen mit einem benutzerdefinierten Filter suchen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */
```

```
public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";

 private static final String FILTER_VALUE = "SaaSProduct";

 /*
  * search agreements by one customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
 where you are the proposer.
  // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
 the acceptor.
  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
 Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
 customizeFilter));

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
```

```
    .filters(filters)
    .build();
  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .filters(filters)
     .nextToken(searchAgreementsResponse.nextToken())
     .build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- API-Details finden Sie SearchAgreements in der AWS SDK for Java 2.x API-Referenz.

Suchen nach Vereinbarungen mit zwei benutzerdefinierten Filtern

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen mit zwei benutzerdefinierten Filtern suchen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.seller;

import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * All filter combinations we support for Proposer and Acceptor:
 * https://docs.aws.amazon.com/marketplace/latest/APIReference/API_marketplace-
agreements_SearchAgreements.html
 */

public class SearchAgreementsByTwoFilters {

 public static final String FILTER_1_NAME = "ResourceType";
```

```
public static final String FILTER_1_VALUE = "SaaSProduct";

public static final String FILTER_2_NAME = "Status";

public static final String FILTER_2_VALUE = "ACTIVE";

/*
 * search agreements by two customize filter
 */
public static void main(String[] args) {

 List<AgreementViewSummary> agreementSummaryList = getAgreements();

 ReferenceCodesUtils.formatOutput(agreementSummaryList);

}

public static List<AgreementViewSummary> getAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // Set PartyType filter to PARTY_TYPE_FILTER_VALUE_PROPOSER to return agreements
where you are the proposer.
 // Change to PARTY_TYPE_FILTER_VALUE_ACCEPTOR to return agreements where you are
the acceptor.
 Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 Filter customizeFilter1 =
Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

 Filter customizeFilter2 =
Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();


 List<Filter> filters = new ArrayList<Filter>();
```

```
 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- API-Details finden Sie SearchAgreements in der AWS SDK for Java 2.x API-Referenz.

Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung

Das folgende Codebeispiel zeigt, wie Bestellungen aktualisiert werden, nachdem die Vereinbarung akzeptiert wurde.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
package com.example.awsmarketplace.agreementapi.buyer;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AcceptAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Charge;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ConfigurableUpfrontPricingTermCo
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.CreateAgreementRequestResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import software.amazon.awssdk.services.marketplaceagreement.model.Intent;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.ListAgreementChargesResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.PurchaseOrder;
import software.amazon.awssdk.services.marketplaceagreement.model.RequestedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.RequestedTermConfiguration;
import
 software.amazon.awssdk.services.marketplaceagreement.model.UpdatePurchaseOrdersRequest;
import utils.AgreementApiUtils;

/**
```

```
 * Demonstrates how to associate a purchase order reference with a SaaS agreement
 with CONTRACT pricing model
 * using the AWS Marketplace Agreement Service APIs.
 *
 * <p>Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model
 and provides a purchase order
 * reference in AcceptAgreementRequest. After acceptance, the buyer lists the
 resulting
 * charges via ListAgreementCharges and associates the purchase order reference with
 a
 * specific charge via UpdatePurchaseOrders.
 *
 * <p>Before running this sample, replace the placeholder constants below with
 values from
 * your AWS Marketplace offer:
 * <ul>
 *   <li>{@code AGREEMENT_PROPOSAL_IDENTIFIER} — the agreementProposalId from the
 offer.</li>
 *   <li>Term IDs (starting with {@code term-}) — found in the offer's term list.</
li>
 *   <li>{@code SELECTOR_VALUE} — duration for the agreement.</li>
 *   <li>{@code DIMENSION_1_KEY} — dimension key defined in the offer.</li>
 *   <li>{@code PURCHASE_ORDER_REFERENCE} — your internal purchase order number
 (e.g., {@code po-123456}).</li>
 * </ul>
 */
public class UpdatePurchaseOrdersAfterAgreementAcceptance {

    // Your internal purchase order reference number (e.g., "po-123456").
    private static final String PURCHASE_ORDER_REFERENCE = "po-123456";

    // The agreementProposalId from the offer.
    private static final String AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-
proposal-identifier>";

    // Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    private static final String CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-
configurable-upfront-pricing-term-id>";

    // Duration for the agreement (e.g., "P366D" for 366 days).
    private static final String SELECTOR_VALUE = "<your-selector-value>";

    // Dimension key and quantity defined in your offer.
    private static final String DIMENSION_1_KEY = "<your-dimension-key>";
```

```
    private static final int DIMENSION_1_VALUE = 1;

    // Term ID for the LegalTerm in your offer.
    private static final String LEGAL_TERM_ID = "<your-legal-term-id>";

    // Term ID for the ValidityTerm in your offer.
    private static final String VALIDITY_TERM_ID = "<your-validity-term-id>";

    public static void main(String[] args) {
        listAgreementChargesAndUpdatePurchaseOrders();
    }

    /**
     * Full end-to-end flow:
     * 1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
reference.
     * 2. List charges to retrieve charge IDs and revisions.
     * 3. Associate the purchase order reference with a specific charge via
UpdatePurchaseOrders.
     * 4. List charges again to confirm the update.
     */
    private static void listAgreementChargesAndUpdatePurchaseOrders() {
        MarketplaceAgreementClient marketplaceAgreementClient =
                MarketplaceAgreementClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        RequestedTerm configurableUpfrontPricingTerm = RequestedTerm.builder()
                .id(CONFIGURABLE_UPFRONT_PRICING_TERM_ID)

.configuration(RequestedTermConfiguration.fromConfigurableUpfrontPricingTermConfiguration(
                        ConfigurableUpfrontPricingTermConfiguration.builder()
                                .selectorValue(SELECTOR_VALUE)
                                .dimensions(Dimension.builder()
                                                    .dimensionKey(DIMENSION_1_KEY)

.dimensionValue(DIMENSION_1_VALUE)
                                                    .build())
                                .build()))
                .build();

        RequestedTerm legalTerm = RequestedTerm.builder().id(LEGAL_TERM_ID).build();
```

```
        RequestedTerm validityTerm =
RequestedTerm.builder().id(VALIDITY_TERM_ID).build();

        // --- Create Agreement ---
        CreateAgreementRequestRequest createAgreementRequestRequest =
                CreateAgreementRequestRequest.builder()
                        .clientToken(AgreementApiUtils.generateClientToken())
                        .intent(Intent.NEW)
                        .requestedTerms(configurableUpfrontPricingTerm, legalTerm,
validityTerm)
                        .agreementProposalIdentifier(AGREEMENT_PROPOSAL_IDENTIFIER)
                        .build();
        CreateAgreementRequestResponse createAgreementRequestResponse =

marketplaceAgreementClient.createAgreementRequest(createAgreementRequestRequest);
        System.out.println("Agreement request created. AgreementRequestId: " +
createAgreementRequestResponse.agreementRequestId());

        // --- Accept Agreement Request with Purchase Order ---
        // The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        String chargeId =
createAgreementRequestResponse.chargeSummary().expectedCharges().get(0).id();
        PurchaseOrder purchaseOrderAtAcceptance = PurchaseOrder.builder()
                .chargeId(chargeId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .build();
        AcceptAgreementRequestRequest acceptAgreementRequestRequest =
                AcceptAgreementRequestRequest.builder()

.agreementRequestId(createAgreementRequestResponse.agreementRequestId())
                        .purchaseOrders(purchaseOrderAtAcceptance)
                        .build();
        AcceptAgreementRequestResponse acceptAgreementRequestResponse =

marketplaceAgreementClient.acceptAgreementRequest(acceptAgreementRequestRequest);
        final String agreementId = acceptAgreementRequestResponse.agreementId();
        System.out.println("Agreement request accepted with purchase order reference
'" + PURCHASE_ORDER_REFERENCE
                + "'. AgreementId: " + agreementId);

        // --- List Agreement Charges ---
        ListAgreementChargesRequest listAgreementChargesRequest =
                ListAgreementChargesRequest.builder()
```

```
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse listAgreementChargesResponse =

 marketplaceAgreementClient.listAgreementCharges(listAgreementChargesRequest);

        System.out.println("All charges for agreement " + agreementId + ":");
        AgreementApiUtils.formatOutput(listAgreementChargesResponse);

        // --- Update Purchase Order ---
        Charge firstCharge = listAgreementChargesResponse.items().get(0);
        PurchaseOrder purchaseOrder = PurchaseOrder.builder()
                .agreementId(agreementId)
                .purchaseOrderReference(PURCHASE_ORDER_REFERENCE)
                .chargeRevision(firstCharge.revision())
                .chargeId(firstCharge.id())
                .build();
        UpdatePurchaseOrdersRequest updatePurchaseOrdersRequest =
                UpdatePurchaseOrdersRequest.builder()
                        .purchaseOrders(purchaseOrder)
                        .build();

 marketplaceAgreementClient.updatePurchaseOrders(updatePurchaseOrdersRequest);
        System.out.println("Purchase order reference '" + PURCHASE_ORDER_REFERENCE
                + "' updated for ChargeId: " + firstCharge.id());

        // --- Verify Update ---
        ListAgreementChargesRequest lacRequest =
                ListAgreementChargesRequest.builder()
                        .agreementId(agreementId)
                        .build();
        ListAgreementChargesResponse lacResponse =
                marketplaceAgreementClient.listAgreementCharges(lacRequest);
        System.out.println("Verified updated charge:");
        AgreementApiUtils.formatOutput(lacResponse.items().get(0));
    }
}
```

- API-Details finden Sie UpdatePurchaseOrders in der AWS SDK for Java 2.x API-Referenz.

MediaConvert Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren MediaConvert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
```

```
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
```

```
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
 where the input file is located.\s
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String mcRoleARN = args[0];
                String fileInput = args[1];
                Region region = Region.US_WEST_2;
                MediaConvertClient mc = MediaConvertClient.builder()
                                .region(region)
                                .build();

                String id = createMediaJob(mc, mcRoleARN, fileInput);
```

```
                System.out.println("MediaConvert job created. Job Id = " + id);
                mc.close();
        }

        public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
 String fileInput) {

                String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
 1) + "javasdk/out/";
                String fileOutput = s3path + "index";
                String thumbsOutput = s3path + "thumbs/";
                String mp4Output = s3path + "mp4/";

                try {
                        System.out.println("MediaConvert role arn: " + mcRoleARN);
                        System.out.println("MediaConvert input file: " + fileInput);
                        System.out.println("MediaConvert output path: " + s3path);

                        // output group Preset HLS low profile
                        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                        // output group Preset HLS media profile
                        Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
                        // output group Preset HLS high profole
                        Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
 3500000, 8, 1920, 1080, 1920);

                        OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

    HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

    HlsManifestDurationFormat.INTEGER)
```

```
.streamInfResolution(
   HlsStreamInfResolution.INCLUDE)
.clientCache(HlsClientCache.ENABLED)
.captionLanguageSetting(
   HlsCaptionLanguageSetting.OMIT)
.manifestCompression(
   HlsManifestCompression.NONE)
.codecSpecification(
   HlsCodecSpecification.RFC_4281)
.outputSelection(
   HlsOutputSelection.MANIFESTS_AND_SEGMENTS)
.programDateTime(HlsProgramDateTime.EXCLUDE)
.programDateTimePeriod(600)
.timedMetadataId3Frame(
   HlsTimedMetadataId3Frame.PRIV)
.timedMetadataId3Period(10)
.destination(fileOutput)
.segmentControl(HlsSegmentControl.SEGMENTED_FILES)
.minFinalSegmentLength((double) 0)
.segmentLength(4).minSegmentLength(0).build())
                                                        .build())
                                        .outputs(hlsLow, hlsMedium,
hlsHigh).build();
```

```
                        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                                .build())
                                                .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280)
                                                                                .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.H_264)

   .h264Settings(H264Settings

                    .builder()
```

```
                .rateControlMode(
                        H264RateControlMode.QVBR)
                .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                .qualityTuningLevel(
                        H264QualityTuningLevel.SINGLE_PASS)
                .qvbrSettings(
                        H264QvbrSettings.builder()
                                .qvbrQualityLevel(
                                        8)
                                .build())
                .codecLevel(H264CodecLevel.AUTO)
                .codecProfile(H264CodecProfile.MAIN)
                .maxBitrate(2400000)
                .framerateControl(
                        H264FramerateControl.INITIALIZE_FROM_SOURCE)
                .gopSize(2.0)
                .gopSizeUnits(H264GopSizeUnits.SECONDS)
                .numberBFramesBetweenReferenceFrames(
                        2)
                .gopClosedCadence(
                        1)
                .gopBReference(H264GopBReference.DISABLED)
```

```
                .slowPal(H264SlowPal.DISABLED)
                .syntax(H264Syntax.DEFAULT)
                .numberReferenceFrames(
                        3)
                .dynamicSubGop(H264DynamicSubGop.STATIC)
                .fieldEncoding(H264FieldEncoding.PAFF)
                .sceneChangeDetect(
                        H264SceneChangeDetect.ENABLED)
                .minIInterval(0)
                .telecine(H264Telecine.NONE)
                .framerateConversionAlgorithm(
                        H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                .entropyEncoding(
                        H264EntropyEncoding.CABAC)
                .slices(1)
                .unregisteredSeiTimecode(
                        H264UnregisteredSeiTimecode.DISABLED)
                .repeatPps(H264RepeatPps.DISABLED)
                .adaptiveQuantization(
                        H264AdaptiveQuantization.HIGH)
                .spatialAdaptiveQuantization(
                        H264SpatialAdaptiveQuantization.ENABLED)
```

```
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
   .build())
                                                                    .build())
.audioDescriptions(AudioDescription.builder()
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
.languageCodeControl(
   AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
   .codec(AudioCodec.AAC)
   .aacSettings(AacSettings
                    .builder()
                    .codecProfile(AacCodecProfile.LC)
                    .rateControlMode(
                                    AacRateControlMode.CBR)
                    .codingMode(AacCodingMode.CODING_MODE_2_0)
                    .sampleRate(44100)
```

```
                    .bitrate(160000)

                    .rawFormat(AacRawFormat.NONE)

                    .specification(AacSpecification.MPEG4)

                    .audioDescriptionBroadcasterMix(

                                    AacAudioDescriptionBroadcasterMix.NORMAL)

                    .build())

   .build())
                                                                        .build())
                                                        .build())
                                        .build();
                       OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                        .build())
                                        .outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)
```

```
.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.FRAME_CAPTURE)

   .frameCaptureSettings(

                   FrameCaptureSettings

                                   .builder()

                                   .framerateNumerator(

                                                   1)

                                   .framerateDenominator(

                                                   1)

                                   .maxCaptures(10000000)

                                   .quality(80)

                                   .build())

   .build())
                                                                                .build())
                                                                   .build())
                                       .build();

                       Map<String, AudioSelector> audioSelectors = new HashMap<>();
                       audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                                   .offset(0).build());

                       JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                       .audioSelectors(audioSelectors)
                                       .videoSelector(
```

```
VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                        .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                        .outputGroups(appleHLS, thumbs,
fileMp4).build();

                        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
mc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
```

```
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

   489, 490, 491, 492)
                                                                                .build())
                                                                .build())
                                        .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)
```

```
.timecodeInsertion(
   VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
   .codec(VideoCodec.H_264)
   .h264Settings(H264Settings
                   .builder()
                   .rateControlMode(
                                   H264RateControlMode.QVBR)
                   .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                   .qualityTuningLevel(
                                   H264QualityTuningLevel.SINGLE_PASS)
                   .qvbrSettings(H264QvbrSettings
                                   .builder()
                                   .qvbrQualityLevel(
                                                   qvbrQualityLevel)
                                   .build())
                   .codecLevel(H264CodecLevel.AUTO)
                   .codecProfile((targetHeight > 720
```

```
                                && targetWidth > 1280)
                                                ? H264CodecProfile.HIGH
                                                : H264CodecProfile.MAIN)
                        .maxBitrate(qvbrMaxBitrate)
                        .framerateControl(
                                H264FramerateControl.INITIALIZE_FROM_SOURCE)
                        .gopSize(2.0)
                        .gopSizeUnits(H264GopSizeUnits.SECONDS)
                        .numberBFramesBetweenReferenceFrames(
                                2)
                        .gopClosedCadence(
                                1)
                        .gopBReference(H264GopBReference.DISABLED)
                        .slowPal(H264SlowPal.DISABLED)
                        .syntax(H264Syntax.DEFAULT)
                        .numberReferenceFrames(
                                3)
                        .dynamicSubGop(H264DynamicSubGop.STATIC)
                        .fieldEncoding(H264FieldEncoding.PAFF)
                        .sceneChangeDetect(
                                H264SceneChangeDetect.ENABLED)
                        .minIInterval(0)
```

```
                    .telecine(H264Telecine.NONE)
                    .framerateConversionAlgorithm(
                                    H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                    .entropyEncoding(
                                    H264EntropyEncoding.CABAC)
                    .slices(1)
                    .unregisteredSeiTimecode(
                                    H264UnregisteredSeiTimecode.DISABLED)
                    .repeatPps(H264RepeatPps.DISABLED)
                    .adaptiveQuantization(
                                    H264AdaptiveQuantization.HIGH)
                    .spatialAdaptiveQuantization(
                                    H264SpatialAdaptiveQuantization.ENABLED)
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
    .build())
                                                                    .build())
```

```
 .audioDescriptions(AudioDescription.builder()

 .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

 .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

 .codecSettings(AudioCodecSettings.builder()

 .codec(AudioCodec.AAC).aacSettings(AacSettings

    .builder()

    .codecProfile(AacCodecProfile.LC)

    .rateControlMode(

                    AacRateControlMode.CBR)

    .codingMode(AacCodingMode.CODING_MODE_2_0)

    .sampleRate(44100)

    .bitrate(96000)

    .rawFormat(AacRawFormat.NONE)

    .specification(AacSpecification.MPEG4)

    .audioDescriptionBroadcasterMix(

                    AacAudioDescriptionBroadcasterMix.NORMAL)

    .build())
                                                                    .build())
                                                        .build())
                                        .build();
                } catch (MediaConvertException e) {
                        e.printStackTrace();
                        System.exit(0);
                }
                return output;
        }
}
```

- API-Details finden Sie CreateJob in der AWS SDK for Java 2.x API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- API-Details finden Sie GetJob in der AWS SDK for Java 2.x API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
```

```
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- API-Details finden Sie ListJobs in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Migration Hub unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Migration Hub Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteProgressUpdateStream`

Das folgende Codebeispiel zeigt die Verwendung`DeleteProgressUpdateStream`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DeleteProgressUpdateStreamRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteProgressStream {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <progressStream>\s

                Where:
                    progressStream - the name of a progress stream to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String progressStream = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
```

```
                .build();

        deleteStream(migrationClient, progressStream);
        migrationClient.close();
    }

    public static void deleteStream(MigrationHubClient migrationClient, String
 streamName) {
        try {
            DeleteProgressUpdateStreamRequest deleteProgressUpdateStreamRequest =
 DeleteProgressUpdateStreamRequest
                    .builder()
                    .progressUpdateStreamName(streamName)
                    .build();


 migrationClient.deleteProgressUpdateStream(deleteProgressUpdateStreamRequest);
            System.out.println(streamName + " is deleted");

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteProgressUpdateStream in der AWS SDK for Java 2.x API-Referenz.

`DescribeApplicationState`

Das folgende Codebeispiel zeigt die Verwendung`DescribeApplicationState`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeApplicationStateResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAppState {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeAppState <appId>\s

                Where:
                    appId -  the application id value.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeApplicationState(migrationClient, appId);
        migrationClient.close();
    }
```

```
    public static void describeApplicationState(MigrationHubClient migrationClient,
 String appId) {
        try {
            DescribeApplicationStateRequest applicationStateRequest =
 DescribeApplicationStateRequest.builder()
                    .applicationId(appId)
                    .build();

            DescribeApplicationStateResponse applicationStateResponse =
 migrationClient
                    .describeApplicationState(applicationStateRequest);
            System.out.println("The application status is " +
 applicationStateResponse.applicationStatusAsString());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeApplicationState in der AWS SDK for Java 2.x API-Referenz.

`DescribeMigrationTask`

Das folgende Codebeispiel zeigt die Verwendung`DescribeMigrationTask`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskRequest;
import
 software.amazon.awssdk.services.migrationhub.model.DescribeMigrationTaskResponse;
```

```
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeMigrationTask {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    DescribeMigrationTask <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        describeMigTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void describeMigTask(MigrationHubClient migrationClient, String
 migrationTask,
            String progressStream) {
        try {
```

```
            DescribeMigrationTaskRequest migrationTaskRequestRequest =
 DescribeMigrationTaskRequest.builder()
                    .progressUpdateStream(progressStream)
                    .migrationTaskName(migrationTask)
                    .build();

            DescribeMigrationTaskResponse migrationTaskResponse = migrationClient
                    .describeMigrationTask(migrationTaskRequestRequest);
            System.out.println("The name is " +
 migrationTaskResponse.migrationTask().migrationTaskName());

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeMigrationTask in der AWS SDK for Java 2.x API-Referenz.

`ImportMigrationTask`

Das folgende Codebeispiel zeigt die Verwendung`ImportMigrationTask`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import
 software.amazon.awssdk.services.migrationhub.model.CreateProgressUpdateStreamRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ImportMigrationTaskRequest;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportMigrationTask {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <migrationTask> <progressStream>\s

                Where:
                    migrationTask - the name of a migration task.\s
                    progressStream - the name of a progress stream.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String migrationTask = args[0];
        String progressStream = args[1];
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        importMigrTask(migrationClient, migrationTask, progressStream);
        migrationClient.close();
    }

    public static void importMigrTask(MigrationHubClient migrationClient, String
 migrationTask, String progressStream) {
        try {
            CreateProgressUpdateStreamRequest progressUpdateStreamRequest =
 CreateProgressUpdateStreamRequest.builder()
                    .progressUpdateStreamName(progressStream)
                    .dryRun(false)
                    .build();
```

```
            migrationClient.createProgressUpdateStream(progressUpdateStreamRequest);
            ImportMigrationTaskRequest migrationTaskRequest =
 ImportMigrationTaskRequest.builder()
                    .migrationTaskName(migrationTask)
                    .progressUpdateStream(progressStream)
                    .dryRun(false)
                    .build();

            migrationClient.importMigrationTask(migrationTaskRequest);

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ImportMigrationTask in der AWS SDK for Java 2.x API-Referenz.

`ListApplications`

Das folgende Codebeispiel zeigt die Verwendung`ListApplications`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ApplicationState;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListApplicationStatesResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListApplications {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listApps(migrationClient);
        migrationClient.close();
    }

    public static void listApps(MigrationHubClient migrationClient) {
        try {
            ListApplicationStatesRequest applicationStatesRequest =
 ListApplicationStatesRequest.builder()
                    .maxResults(10)
                    .build();

            ListApplicationStatesResponse response =
 migrationClient.listApplicationStates(applicationStatesRequest);
            List<ApplicationState> apps = response.applicationStateList();
            for (ApplicationState appState : apps) {
                System.out.println("App Id is " + appState.applicationId());
                System.out.println("The status is " +
 appState.applicationStatus().toString());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListApplications in der AWS SDK for Java 2.x API-Referenz.

`ListCreatedArtifacts`

Das folgende Codebeispiel zeigt die Verwendung`ListCreatedArtifacts`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
```

```
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListCreatedArtifacts in der AWS SDK for Java 2.x API-Referenz.

`ListMigrationTasks`

Das folgende Codebeispiel zeigt die Verwendung`ListMigrationTasks`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListMigrationTasksResponse;
```

```
import software.amazon.awssdk.services.migrationhub.model.MigrationTaskSummary;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMigrationTasks {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listMigrTasks(migrationClient);
        migrationClient.close();
    }

    public static void listMigrTasks(MigrationHubClient migrationClient) {
        try {
            ListMigrationTasksRequest listMigrationTasksRequest =
 ListMigrationTasksRequest.builder()
                    .maxResults(10)
                    .build();

            ListMigrationTasksResponse response =
 migrationClient.listMigrationTasks(listMigrationTasksRequest);
            List<MigrationTaskSummary> migrationList =
 response.migrationTaskSummaryList();
            for (MigrationTaskSummary migration : migrationList) {
                System.out.println("Migration task name is " +
 migration.migrationTaskName());
                System.out.println("The Progress update stream is " +
 migration.progressUpdateStream());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API-Details finden Sie ListMigrationTasks in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon MSK unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

Neptune-Beispiele unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Neptune Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Neptune

Das folgende Codebeispiel zeigt die ersten Schritte mit Neptune.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
        describeDbCluster(neptuneClient).join(); // This ensures the async code runs
 to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
```

```
    *
    * @param neptuneClient the Neptune asynchronous client used to make the request
    * @return a {@link CompletableFuture} that completes when the operation is
finished
    */
   public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
neptuneClient) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .maxRecords(20)
               .build();

       SdkPublisher<DescribeDbClustersResponse> paginator =
neptuneClient.describeDBClustersPaginator(request);
       CompletableFuture<Void> future = new CompletableFuture<>();

       paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
           private Subscription subscription;

           @Override
           public void onSubscribe(Subscription s) {
               this.subscription = s;
               s.request(Long.MAX_VALUE); // request all items
           }

           @Override
           public void onNext(DescribeDbClustersResponse response) {
               response.dbClusters().forEach(cluster -> {
                   System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                   System.out.println("Status: " + cluster.status());
               });
           }

           @Override
           public void onError(Throwable t) {
               future.completeExceptionally(t);
           }

           @Override
           public void onComplete() {
               future.complete(null);
           }
       });
```

```
        return future.whenComplete((result, throwable) -> {
            neptuneClient.close();
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
throwable.getMessage());
            }
        });
    }
```

- API-Details finden Sie DescribeDBClustersPaginator in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Subnetzgruppe von Amazon Neptune.
- Erstellen Sie einen Neptun-Cluster.
- Erstellen Sie eine Neptun-Instance.
- Prüfen des Status einer Neptune-Instance.
- Zeigen Sie Details zum Neptun-Cluster an.
- Stoppen Sie den Neptune-Cluster.
- Starten Sie den Neptune-Cluster.
- Löschen Sie die Neptune-Komponenten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Neptune-Features gezeigt werden.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
 group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
 cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
 instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";

        logger.info("""
                   Amazon Neptune is a fully managed graph
                   database service by AWS, designed specifically
                   for handling complex relationships and connected
                   datasets at scale. It supports two popular graph models:
                   property graphs (via openCypher and Gremlin) and RDF
                   graphs (via SPARQL). This makes Neptune ideal for
                   use cases such as knowledge graphs, fraud detection,
                   social networking, recommendation engines, and
                   network management, where relationships between
                   entities are central to the data.

                   Being fully managed, Neptune handles database
                   provisioning, patching, backups, and replication,
                   while also offering high availability and durability
                   within AWS's infrastructure.

                   For developers, programming with Neptune allows
```

```
                    for building intelligent, relationship-aware
                    applications that go beyond traditional tabular
                    databases. Developers can use the AWS SDK for Java
                    to automate infrastructure operations (via NeptuneClient).

                     Let's get started...
                     """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }

    public static void runScenario(String subnetGroupName, String dbInstanceId,
String clusterName) {
        logger.info(DASHES);
        logger.info("1. Create a Neptune DB Subnet Group");
        logger.info("The Neptune DB subnet group is used when launching a Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a Neptune Cluster");
        logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
        waitForInputToContinue(scanner);
        String dbClusterId;
        try {
            dbClusterId = neptuneActions.createDBClusterAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
```

```
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
```

```
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
                Thank you for checking out the Amazon Neptune Service Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Neptune-SDK-Methoden.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
```

```
   private static NeptuneAsyncClient neptuneAsyncClient;
   private final Region region = Region.US_EAST_1;
   private static final Logger logger =
LoggerFactory.getLogger(NeptuneActions.class);
   private final NeptuneClient neptuneClient =
NeptuneClient.builder().region(region).build();

   /**
    * Retrieves an instance of the NeptuneAsyncClient.
    * <p>
    * This method initializes and returns a singleton instance of the
NeptuneAsyncClient. The client
    * is configured with the following settings:
    * <ul>
    *     <li>Maximum concurrency: 100</li>
    *     <li>Connection timeout: 60 seconds</li>
    *     <li>Read timeout: 60 seconds</li>
    *     <li>Write timeout: 60 seconds</li>
    *     <li>API call timeout: 2 minutes</li>
    *     <li>API call attempt timeout: 90 seconds</li>
    *     <li>Retry strategy: STANDARD</li>
    * </ul>
    * The client is built using the NettyNioAsyncHttpClient.
    *
    * @return the singleton instance of the NeptuneAsyncClient
    */
   private static NeptuneAsyncClient getAsyncClient() {
       if (neptuneAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                   .maxConcurrency(100)
                   .connectionTimeout(Duration.ofSeconds(60))
                   .readTimeout(Duration.ofSeconds(60))
                   .writeTimeout(Duration.ofSeconds(60))
                   .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                   .apiCallTimeout(Duration.ofMinutes(2))
                   .apiCallAttemptTimeout(Duration.ofSeconds(90))
                   .retryStrategy(RetryMode.STANDARD)
                   .build();

           neptuneAsyncClient = NeptuneAsyncClient.builder()
                   .httpClient(httpClient)
```

```
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return neptuneAsyncClient;
    }

    /**
     * Asynchronously deletes a set of Amazon Neptune resources in a defined order.
     * <p>
     * The method performs the following operations in sequence:
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code dbInstanceId}.</
li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code dbClusterId}.</
li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
 subnetGroupName}.</li>
     * </ol>
     * <p>
     * If any step fails, the subsequent operations are not performed, and the
 exception
     * is logged. This method blocks the calling thread until all operations
 complete.
     *
     * @param dbInstanceId      the ID of the Neptune DB instance to delete
     * @param dbClusterId       the ID of the Neptune DB cluster to delete
     * @param subnetGroupName   the name of the Neptune DB subnet group to delete
     */
    public void deleteNeptuneResourcesAsync(String dbInstanceId, String dbClusterId,
 String subnetGroupName) {
        deleteDBInstanceAsync(dbInstanceId)
                .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
                .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
                .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
                .whenComplete((v, ex) -> {
                    if (ex != null) {
                        logger.info("Failed to delete Neptune resources: " +
 ex.getMessage());
                    } else {
                        logger.info("Neptune resources deleted successfully.");
                    }
                })
                .join(); // Waits for the entire async chain to complete
```

```
    }

    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String instanceId)
{
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
```

```
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }


    private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof NeptuneException &&
                                ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                            long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                            logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                            future.complete(null);
                            return;
                        }
                        future.completeExceptionally(new CompletionException("Error
polling DB instance", cause));
                        return;
```

```
                    }

                    String status =
response.dbInstances().get(0).dbInstanceStatus();
                    long elapsed = (System.currentTimeMillis() - startTime) / 1000;
                    System.out.printf("\r  Waiting: Instance %s status: %-10s (%ds
elapsed)", instanceId, status, elapsed);
                    System.out.flush();

                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() -> checkInstanceDeletedRecursive(instanceId,
startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
```

```
                        future.completeExceptionally(new RuntimeException("Cluster
not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached desired status
'%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
```

```
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }

    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
```

```
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
                       cluster.vpcSecurityGroups().forEach(vpcGroup ->
                               logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                       logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                       logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                       logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                       logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                       logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                       logger.info("------");
                   }
               })
               .exceptionally(ex -> {
                   Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
```

```
                        );
                        return;
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }
```

```
   /**
    * Creates a new Amazon Neptune DB instance asynchronously.
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }


   /**
```

```
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
               });
   }


   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
```

```
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();
```

```
            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup

- CreateGraph
- DeleteDBCluster
- DeleteDBInstance
- DeleteDBSubnetGroup
- DescribeDBClusters
- DescribeDBInstances
- ExecuteGremlinProfileQuery
- ExecuteGremlinQuery
- ExecuteOpenCypherExplainQuery
- ExecuteQuery
- Starten Sie DBCluster
- Stoppen Sie DBCluster

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDBCluster`

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates a new Amazon Neptune DB cluster asynchronously.
     *
     * @param dbName the name of the DB cluster to be created
     * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
     * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
     */
```

```
    public CompletableFuture<String> createDBClusterAsync(String dbName) {
        CreateDbClusterRequest request = CreateDbClusterRequest.builder()
                .dbClusterIdentifier(dbName)
                .engine("neptune")
                .deletionProtection(false)
                .backupRetentionPeriod(1)
                .build();

        return getAsyncClient().createDBCluster(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String clusterId = response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der AWS SDK for Java 2.x - API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new Amazon Neptune DB instance asynchronously.
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`CreateDBSubnetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSubnetGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();
```

```
        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }
```

- API-Details finden Sie CreateDBSubnetGroup in der AWS SDK for Java 2.x API-Referenz.

`CreateGraph`

Das folgende Codebeispiel zeigt die Verwendung`CreateGraph`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Executes the process of creating a new Neptune graph.
     *
     * @param client        the Neptune graph client used to interact with the
Neptune service
```

```
     * @param graphName     the name of the graph to be created
     * @throws NeptuneGraphException if an error occurs while creating the graph
     */
    public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
        try {
            // Create the graph request
            CreateGraphRequest request = CreateGraphRequest.builder()
                    .graphName(graphName)
                    .provisionedMemory(16)
                    .build();

            // Create the graph
            CreateGraphResponse response = client.createGraph(request);

            // Extract the graph name and ARN
            String createdGraphName = response.name();
            String graphArn = response.arn();
            String graphEndpoint = response.endpoint();

            System.out.println("Graph created successfully!");
            System.out.println("Graph Name: " + createdGraphName);
            System.out.println("Graph ARN: " + graphArn);
            System.out.println("Graph Endpoint: " +graphEndpoint );

        } catch (NeptuneGraphException e) {
            System.err.println("Failed to create graph: " +
e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
   }
```

- API-Details finden Sie CreateGraph in der AWS SDK for Java 2.x API-Referenz.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der AWS SDK for Java 2.x -API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`DeleteDBSubnetGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBSubnetGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
```

```
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }
```

- API-Details finden Sie DeleteDBSubnetGroup in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
```

```
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " + cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK for Java 2.x - API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
    *
    * @param instanceId     the ID of the Neptune instance to check
    * @param desiredStatus  the desired status of the Neptune instance
    * @param startTime      the start time of the operation, used to calculate the
elapsed time
    * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
    */
   private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
```

```
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

`ExecuteGremlinProfileQuery`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteGremlinProfileQuery`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
    * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
    *
    * @param client the {@link NeptunedataClient} instance to use for executing the
Gremlin query
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Querying Neptune...");
           ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                   .gremlinQuery("g.V().has('code', 'ANC')")
                   .build();

           ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

           System.out.println("Full Response:");
           System.out.println(response);

           // Retrieve and print the result
           if (response.result() != null) {
               System.out.println("Query Result:");
               System.out.println(response.result().toString());
           } else {
               System.out.println("No result returned from the query.");
           }
       } catch (NeptunedataException e) {
           System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
       } catch (Exception e) {
           System.err.println("Unexpected error: " + e.getMessage());
       } finally {
           client.close();
       }
   }
```

- API-Details finden Sie ExecuteGremlinProfileQuery in der AWS SDK for Java 2.x API-Referenz.

`ExecuteGremlinQuery`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteGremlinQuery`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
    *
    * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
    */
   private static void executeGremlinProfileQuery(NeptunedataClient client) {
       System.out.println("Executing Gremlin PROFILE query...");

       ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
               .gremlinQuery("g.V().has('code', 'ANC')")
               .build();

       ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
       if (response.output() != null) {
           System.out.println("Query Profile Output:");
           System.out.println(response.output());
       } else {
           System.out.println("No output returned from the profile query.");
       }
   }
```

- API-Details finden Sie ExecuteGremlinQuery in der AWS SDK for Java 2.x API-Referenz.

`ExecuteOpenCypherExplainQuery`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteOpenCypherExplainQuery`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Executes an OpenCypher EXPLAIN query using the provided Neptune data client.
     *
     * @param client The Neptune data client to use for the query execution.
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Executing OpenCypher EXPLAIN query...");
            ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                    .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                    .explainMode("debug")
                    .build();

            ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

            if (response.results() != null) {
                System.out.println("Explain Results:");
                System.out.println(response.results().asUtf8String());
            } else {
                System.out.println("No explain results returned.");
            }

        } catch (NeptunedataException e) {
            System.err.println("Neptune error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- API-Details finden Sie ExecuteOpenCypherExplainQuery in der AWS SDK for Java 2.x API-Referenz.

`ExecuteQuery`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteQuery`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Executes a Gremlin profile query on the Neptune Analytics graph.
    *
    * @param client       the {@link NeptuneGraphClient} instance to use for the
query
    * @param graphId      the identifier of the graph to execute the query on
    *
    * @throws NeptuneGraphException if an error occurs while executing the query on
the Neptune Graph
    * @throws Exception if an unexpected error occurs
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client, String
graphId) {

       try {
           System.out.println("Running openCypher query on Neptune Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
```

```
                String result = reader.lines().collect(Collectors.joining("\n"));
                System.out.println("Query Result:");
                System.out.println(result);
            } catch (Exception e) {
                System.err.println("Error reading response: " + e.getMessage());
            }

        } catch (NeptuneGraphException e) {
            System.err.println("NeptuneGraph error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- API-Details finden Sie ExecuteQuery in der AWS SDK for Java 2.x API-Referenz.

`StartDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`StartDBCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
```

```
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }
```

- Weitere API-Informationen finden Sie unter StartDBCluster in der AWS SDK for Java 2.x -API-Referenz.

`StopDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`StopDBCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
```

```
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }
```

- Weitere API-Informationen finden Sie unter StopDBCluster in der AWS SDK for Java 2.x -API-Referenz.

Szenarien

Verwenden der Neptune-API, um Grafikdaten abzufragen

Das folgende Codebeispiel zeigt, wie Sie mit der Neptune-API Grafikdaten abfragen.

SDK für Java 2.x

Zeigt, wie die Java-API von Amazon Neptune verwendet wird, um eine Lambda-Funktion zu erstellen, die Grafikdaten innerhalb der VPC abfragt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Lambda
- Neptune

Beispiele für Partner Central unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Partner Central Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AssignOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`AssignOpportunity`.

SDK für Java 2.x

Weisen Sie eine bestehende Opportunity einem anderen Benutzer neu zu.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }
```

```
 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
          .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie AssignOpportunity in der AWS SDK for Java 2.x API-Referenz.

`AssociateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`AssociateOpportunity`.

SDK für Java 2.x

Erstellen Sie eine formelle Zuordnung zwischen einer Opportunity und verschiedenen verwandten Entitäten.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId, entityType,
 entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
```

```
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie AssociateOpportunity in der AWS SDK for Java 2.x API-Referenz.

`CreateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`CreateOpportunity`.

SDK für Java 2.x

Erstellen Sie eine Opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
```

```
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 public static void main(String[] args) {

  String inputFile = "CreateOpportunity2.json";

  if (args.length > 0)
   inputFile = args[0];

  CreateOpportunityResponse response = createOpportunity(inputFile);

  client.close();
 }
```

```
static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
   .stage(root.lifeCycle.stage)
   .targetCloseDate(root.lifeCycle.targetCloseDate)
   .build();
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
    .awsFundingUsed(root.marketing.awsFundingUsed)
    .campaignName(root.marketing.campaignName)
    .channels(root.marketing.channels)
    .source(root.marketing.source)
    .useCases(root.marketing.useCases)
    .build();

 }
```

```
 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
   .city(root.customer.account.address.city)
               .postalCode(root.customer.account.address.postalCode)
               .stateOrRegion(root.customer.account.address.stateOrRegion)
               .countryCode(root.customer.account.address.countryCode)
               .streetAddress(root.customer.account.address.streetAddress)
               .build();
 }

 Account account = null;
 if ( root.customer != null && root.customer.account!= null) {
  account = Account.builder()
            .address(address)
            .awsAccountId(root.customer.account.awsAccountId)
               .duns(root.customer.account.duns)
               .industry(root.customer.account.industry)
               .otherIndustry(root.customer.account.otherIndustry)
               .companyName(root.customer.account.companyName)
               .websiteUrl(root.customer.account.websiteUrl)
               .build();
 }

 List<Contact> contacts = new ArrayList<Contact>();
 if ( root.customer != null && root.customer.contacts != null) {
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
   Contact contact = Contact.builder()
                 .email(jsonContact.email)
                 .firstName(jsonContact.firstName)
                 .lastName(jsonContact.lastName)
                 .phone(jsonContact.phone)
                 .businessTitle(jsonContact.businessTitle)
                 .build();
   contacts.add(contact);
  }
 }

 Customer customer = Customer.builder()
   .account(account)
   .contacts(contacts)
   .build();
```

```
 Contact oportunityTeamContact = null;
 if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
  oportunityTeamContact = Contact.builder()
               .firstName(root.opportunityTeam.get(0).firstName)
               .lastName(root.opportunityTeam.get(0).lastName)
               .email(root.opportunityTeam.get(0).email)
               .phone(root.opportunityTeam.get(0).phone)
               .businessTitle(root.opportunityTeam.get(0).businessTitle)
               .build();
 }

 List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
 if ( root.project != null && root.project.expectedCustomerSpend != null) {
  for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
   ExpectedCustomerSpend expectedCustomerSpend = null;
   expectedCustomerSpend = ExpectedCustomerSpend.builder()
     .amount(expectedCustomerSpendJson.amount)
     .currencyCode(expectedCustomerSpendJson.currencyCode)
     .frequency(expectedCustomerSpendJson.frequency)
     .targetCompany(expectedCustomerSpendJson.targetCompany)
     .build();
   expectedCustomerSpends.add(expectedCustomerSpend);
  }
       }

       Project project = null;
       if ( root.project != null) {
        project = Project.builder()
               .title(root.project.title)
               .customerBusinessProblem(root.project.customerBusinessProblem)
               .customerUseCase(root.project.customerUseCase)
               .deliveryModels(root.project.deliveryModels)
               .expectedCustomerSpend(expectedCustomerSpends)
               .salesActivities(root.project.salesActivities)
               .competitorName(root.project.competitorName)
               .otherSolutionDescription(root.project.otherSolutionDescription)
               .build();
       }

       SoftwareRevenue softwareRevenue = null;
       if ( root.softwareRevenue != null) {
        MonetaryValue monetaryValue = null;
```

```
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
            .amount(root.softwareRevenue.value.amount)
            .currencyCode(root.softwareRevenue.value.currencyCode)
            .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

  // Building the Actual CreateOpportunity Request
  CreateOpportunityRequest createOpportunityRequest =
 CreateOpportunityRequest.builder()
    .catalog(CATALOG_TO_USE)
    .clientToken(root.clientToken)
    .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
    .opportunityType(root.opportunityType)
    .lifeCycle(lifeCycle)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .origin(root.origin)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- API-Details finden Sie CreateOpportunity in der AWS SDK for Java 2.x API-Referenz.

`DisassociateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateOpportunity`.

SDK für Java 2.x

Entfernen Sie eine bestehende Verknüpfung zwischen einer Opportunity und verwandten Entitäten.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
```

```
    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String entityType = "Solutions";

    String entityIdentifier = "S-0000000";

    DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

    ReferenceCodesUtils.formatOutput(response);
   }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie DisassociateOpportunity in der AWS SDK for Java 2.x API-Referenz.

`GetAwsOpportunitySummary`

Das folgende Codebeispiel zeigt die Verwendung`GetAwsOpportunitySummary`.

SDK für Java 2.x

Ruft eine Zusammenfassung einer AWS Opportunity ab.

```
package org.example;
```

```
import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequest
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRespons

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId) {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();
```

```
  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie GetAwsOpportunitySummary in der AWS SDK for Java 2.x API-Referenz.

`GetEngagementInvitation`

Das folgende Codebeispiel zeigt die Verwendung`GetEngagementInvitation`.

SDK für Java 2.x

Ruft die Details einer Einladung zur Verlobung ab, die von AWS einem Partner geteilt wurde.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {
```

```
 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie GetEngagementInvitation in der AWS SDK for Java 2.x API-Referenz.

`GetOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`GetOpportunity`.

SDK für Java 2.x

Rufen Sie eine Opportunity ab.

```
package org.example;
```

```
import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
```

```
            .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie GetOpportunity in der AWS SDK for Java 2.x API-Referenz.

`ListEngagementInvitations`

Das folgende Codebeispiel zeigt die Verwendung`ListEngagementInvitations`.

SDK für Java 2.x

Ruft eine Liste mit Einladungen zur Interaktion ab, die an den Partner gesendet wurden.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReques
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRespon
import software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {
```

```
 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();

        return opportunitySummaries;
 }
}
```

- API-Details finden Sie ListEngagementInvitations in der AWS SDK for Java 2.x API-Referenz.

`ListOpportunities`

Das folgende Codebeispiel zeigt die Verwendung`ListOpportunities`.

SDK für Java 2.x

Listen Sie Opportunitys auf.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
```

```
    List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

    ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

    opportunitySummaries.addAll(response.opportunitySummaries());

    while (response.nextToken() != null && response.nextToken().length() > 0) {
     listOpportunityRequest = ListOpportunitiesRequest.builder()
                   .catalog(CATALOG_TO_USE)
             .maxResults(5)
             .nextToken(response.nextToken())
             .build();
     response = client.listOpportunities(listOpportunityRequest);
     opportunitySummaries.addAll(response.opportunitySummaries());
    }

    client.close();

       return opportunitySummaries;
 }
}
```

- API-Details finden Sie ListOpportunities in der AWS SDK for Java 2.x API-Referenz.

`ListSolutions`

Das folgende Codebeispiel zeigt die Verwendung`ListSolutions`.

SDK für Java 2.x

Ruft eine Liste von Partner Solutions ab, die der Partner bei Partner Central registriert hat.

```
package org.example;
```

```
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);
```

```
     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- API-Details finden Sie ListSolutions in der AWS SDK for Java 2.x API-Referenz.

`RejectEngagementInvitation`

Das folgende Codebeispiel zeigt die Verwendung`RejectEngagementInvitation`.

SDK für Java 2.x

Lehnt einen AWS geteilten EngagementInvitation Inhalt ab.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRespo

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
```

```
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie RejectEngagementInvitation in der AWS SDK for Java 2.x API-Referenz.

`StartEngagementByAcceptingInvitationTask`

Das folgende Codebeispiel zeigt die Verwendung`StartEngagementByAcceptingInvitationTask`.

SDK für Java 2.x

Beginnt das Engagement mit der Annahme einer EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse
import software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;
```

```
    StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

    if ( response == null) {
     System.out.println("Opportunity is not AWS Originated.");
    } else {
     ReferenceCodesUtils.formatOutput(response);
    }
   }

   private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {

    GetEngagementInvitationRequest getRequest =
 GetEngagementInvitationRequest.builder()
         .catalog(Constants.CATALOG_TO_USE)
         .identifier(invitationId)
         .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getRequest);

       return response;
 }

 static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
 invitationId) {

  if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
   StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
    StartEngagementByAcceptingInvitationTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .clientToken(clientToken)
          .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
        return response;
  }
  return null;
 }
}
```

- API-Details finden Sie StartEngagementByAcceptingInvitationTask in der AWS SDK for Java 2.x API-Referenz.

StartEngagementFromOpportunityTask

Das folgende Codebeispiel zeigt die VerwendungStartEngagementFromOpportunityTask.

SDK für Java 2.x

Initiiert den Interaktionsprozess von einer bestehenden Opportunity aus, indem die Einladung zur Interaktion angenommen und eine entsprechende Opportunity im System des Partners erstellt wird.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */
```

```
public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String opportunityId)
 {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).visibi
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- API-Details finden Sie StartEngagementFromOpportunityTask in der AWS SDK for Java 2.x API-Referenz.

`UpdateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`UpdateOpportunity`.

SDK für Java 2.x

Aktualisieren Sie eine Opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted or
 In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

 public static void main(String[] args) {

  String inputFile = "updateOpportunity.json";

  if (args.length > 0)
   inputFile = args[0];

  UpdateOpportunityResponse response = updateOpportunity(inputFile);

  client.close();
 }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
```

```
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
        System.out.println(opportunityId + ":" + response);
        return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
```

```
      .stage(root.lifeCycle.stage)
      .targetCloseDate(root.lifeCycle.targetCloseDate)
      .build();
    }

    Marketing marketing = null;
    if ( root.marketing != null ) {
     marketing = Marketing.builder()
       .awsFundingUsed(root.marketing.awsFundingUsed)
       .campaignName(root.marketing.campaignName)
       .channels(root.marketing.channels)
       .source(root.marketing.source)
       .useCases(root.marketing.useCases)
       .build();

    }

    Address address = null;
    if (root.customer != null && root.customer.account != null &&
  root.customer.account.address != null) {
     address = Address.builder().postalCode(root.customer.account.address.postalCode)
       .stateOrRegion(root.customer.account.address.stateOrRegion)
       .countryCode(root.customer.account.address.countryCode).build();
    }

    Account account = null;
    if (root.customer != null && root.customer.account != null) {
     account = Account.builder().address(address).duns(root.customer.account.duns)

  .industry(root.customer.account.industry).companyName(root.customer.account.companyName)
       .websiteUrl(root.customer.account.websiteUrl).build();
    }

    List<Contact> contacts = new ArrayList<Contact>();
    if ( root.customer != null && root.customer.contacts != null) {
     for (org.example.entity.Contact jsonContact : root.customer.contacts) {
      Contact contact = Contact.builder()
                     .email(jsonContact.email)
                     .firstName(jsonContact.firstName)
                     .lastName(jsonContact.lastName)
                     .phone(jsonContact.phone)
                     .businessTitle(jsonContact.businessTitle)
                     .build();
      contacts.add(contact);
```

```
    }
   }

   Customer customer =
Customer.builder().account(account).contacts(contacts).build();

   List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
   if ( root.project != null && root.project.expectedCustomerSpend != null) {
    for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
     ExpectedCustomerSpend expectedCustomerSpend = null;
     expectedCustomerSpend = ExpectedCustomerSpend.builder()
       .amount(expectedCustomerSpendJson.amount)
       .currencyCode(expectedCustomerSpendJson.currencyCode)
       .frequency(expectedCustomerSpendJson.frequency)
       .targetCompany(expectedCustomerSpendJson.targetCompany)
       .build();
     expectedCustomerSpends.add(expectedCustomerSpend);
    }
         }

   Project project = null;
   if (root.project != null) {
    project = Project.builder().title(root.project.title)
      .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModels)
      .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorName)
      .otherSolutionDescription(root.project.otherSolutionDescription).build();
   }

   // Building the Actual CreateOpportunity Request
   UpdateOpportunityRequest updateOpportunityRequest =
UpdateOpportunityRequest.builder().catalog(root.catalog)

.identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

.primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportunityT
     .lifeCycle(lifeCycle)
     .customer(customer)
     .project(project)
```

```
      .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
      .marketing(marketing)
      .nationalSecurity(root.nationalSecurity)
      .opportunityType(root.opportunityType)
      .build();

    UpdateOpportunityResponse updateResponse =
  client.updateOpportunity(updateOpportunityRequest);
    System.out.println("Successfully updated opportunity: " + updateResponse);

    return updateResponse;
   } else {
    System.out.println("Opportunity cannot be updated.");
    return null;
   }
     }
}
```

- API-Details finden Sie UpdateOpportunity in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Aktualisieren der zugehörigen Entität einer Opportunity

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Trennen Sie die Zuordnung einer alten Entität.
- Ordnen Sie eine neue Entität zu.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie es im Scenarios-Repository eingerichtet und ausgeführt wird.

Aktualisieren der zugehörigen Entität einer Opportunity

```
package org.example;
```

```
import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";

     disassociateOppornititityResponse(opportunityId, entityType,
 originalEntityIdentifier );
```

```
    AssociateOpportunityResponse associateOpportunityResponse =
associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

    ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
   }

private static AssociateOpportunityResponse associateOpportunityResponse(String
opportunityId, String entityType, String entityIdentifier) {

       AssociateOpportunityRequest associateOpportunityRequest =
AssociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       AssociateOpportunityResponse response =
client.associateOpportunity(associateOpportunityRequest);

       return response;
}

private static DisassociateOpportunityResponse
disassociateOppornitityResponse(String opportunityId, String entityType, String
entityIdentifier) {
 PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

       DisassociateOpportunityRequest disassociateOpportunityRequest =
DisassociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       DisassociateOpportunityResponse response =
client.disassociateOpportunity(disassociateOpportunityRequest);

       return response;
```

```
  }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AssociateOpportunity
  - DisassociateOpportunity

Beispiele für Amazon Personalize unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Personalize Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateBatchInferenceJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateBatchInferenceJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                        String solutionVersionArn,
                        String jobName,
                        String s3InputDataSourcePath,
                        String s3DataDestinationPath,
                        String roleArn,
                        String explorationWeight,
                        String explorationItemAgeCutOff) {

                long waitInMilliseconds = 60 * 1000;
                String status;
                String batchInferenceJobArn;

                try {

                        // Set up data input and output parameters.
                        S3DataConfig inputSource = S3DataConfig.builder()
                                        .path(s3InputDataSourcePath)
                                        .build();

                        S3DataConfig outputDestination = S3DataConfig.builder()
                                        .path(s3DataDestinationPath)
                                        .build();

                        BatchInferenceJobInput jobInput =
BatchInferenceJobInput.builder()
                                        .s3DataSource(inputSource)
                                        .build();

                        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
                                        .s3DataDestination(outputDestination)
                                        .build();

                        // Optional code to build the User-Personalization specific
item exploration
                        // config.
                        HashMap<String, String> explorationConfig = new HashMap<>();

                        explorationConfig.put("explorationWeight",
explorationWeight);
```

```
                        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

                        BatchInferenceJobConfig jobConfig =
BatchInferenceJobConfig.builder()
                                        .itemExplorationConfig(explorationConfig)
                                        .build();

                        // End optional User-Personalization recipe specific code.

                        CreateBatchInferenceJobRequest
createBatchInferenceJobRequest = CreateBatchInferenceJobRequest
                                        .builder()
                                        .solutionVersionArn(solutionVersionArn)
                                        .jobInput(jobInput)
                                        .jobOutput(jobOutputLocation)
                                        .jobName(jobName)
                                        .roleArn(roleArn)
                                        .batchInferenceJobConfig(jobConfig) //
Optional
                                        .build();

                        batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
                                        .batchInferenceJobArn();

                        DescribeBatchInferenceJobRequest
describeBatchInferenceJobRequest = DescribeBatchInferenceJobRequest
                                        .builder()
                                        .batchInferenceJobArn(batchInferenceJobArn)
                                        .build();

                        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
                        while (Instant.now().getEpochSecond() < maxTime) {

                                BatchInferenceJob batchInferenceJob =
personalizeClient

.describeBatchInferenceJob(describeBatchInferenceJobRequest)
                                                .batchInferenceJob();

                                status = batchInferenceJob.status();
                                System.out.println("Batch inference job status: " +
status);
```

```
                                if (status.equals("ACTIVE") || status.equals("CREATE
FAILED")) {
                                        break;
                                }
                                try {
                                        Thread.sleep(waitInMilliseconds);
                                } catch (InterruptedException e) {
                                        System.out.println(e.getMessage());
                                }
                        }
                        return batchInferenceJobArn;

                } catch (PersonalizeException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return "";
        }
```

- API-Details finden Sie CreateBatchInferenceJob in der AWS SDK for Java 2.x API-Referenz.

`CreateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`CreateCampaign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalCompaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
            String name) {

        try {
            CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                    .minProvisionedTPS(1)
```

```
                    .solutionVersionArn(solutionVersionArn)
                    .name(name)
                    .build();

            CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
            System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
            return campaignResponse.campaignArn();
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- API-Details finden Sie CreateCampaign in der AWS SDK for Java 2.x API-Referenz.

`CreateDataset`

Das folgende Codebeispiel zeigt die Verwendung`CreateDataset`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createDataset(PersonalizeClient personalizeClient,
            String datasetName,
            String datasetGroupArn,
            String datasetType,
            String schemaArn) {
        try {
            CreateDatasetRequest request = CreateDatasetRequest.builder()
                    .name(datasetName)
                    .datasetGroupArn(datasetGroupArn)
                    .datasetType(datasetType)
                    .schemaArn(schemaArn)
```

```
                .build();

        String datasetArn = personalizeClient.createDataset(request)
                .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API-Details finden Sie CreateDataset in der AWS SDK for Java 2.x API-Referenz.

`CreateDatasetExportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetExportJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
        String jobName,
        String datasetArn,
        IngestionMode ingestionMode,
        String roleArn,
        String s3BucketPath,
        String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {
```

```
            S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
            DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig)
                    .build();

            CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
                    .jobName(jobName)
                    .datasetArn(datasetArn)
                    .ingestionMode(ingestionMode)
                    .jobOutput(jobOutput)
                    .roleArn(roleArn)
                    .build();

            String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

            DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
                    .datasetExportJobArn(datasetExportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetExportJob datasetExportJob = personalizeClient
                        .describeDatasetExportJob(describeDatasetExportJobRequest)
                        .datasetExportJob();

                status = datasetExportJob.status();
                System.out.println("Export job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    return status;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        } catch (PersonalizeException e) {
```

```
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- API-Details finden Sie CreateDatasetExportJob in der AWS SDK for Java 2.x API-Referenz.

`CreateDatasetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

       try {
           CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                   .name(datasetGroupName)
                   .build();
           return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
       } catch (PersonalizeException e) {
           System.out.println(e.awsErrorDetails().errorMessage());
       }
       return "";
   }
```

Erstellen Sie eine Domain-Datensatzgruppe.

```
   public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
```

```
            String datasetGroupName,
            String domain) {

        try {
            CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
                    .name(datasetGroupName)
                    .domain(domain)
                    .build();
            return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- API-Details finden Sie CreateDatasetGroup in der AWS SDK for Java 2.x API-Referenz.

`CreateDatasetImportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatasetImportJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
            String jobName,
            String datasetArn,
            String s3BucketPath,
            String roleArn) {

        long waitInMilliseconds = 60 * 1000;
        String status;
        String datasetImportJobArn;
```

```
        try {
            DataSource importDataSource = DataSource.builder()
                    .dataLocation(s3BucketPath)
                    .build();

            CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
                    .datasetArn(datasetArn)
                    .dataSource(importDataSource)
                    .jobName(jobName)
                    .roleArn(roleArn)
                    .build();

            datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
                    .datasetImportJobArn();
            DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
                    .datasetImportJobArn(datasetImportJobArn)
                    .build();

            long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                DatasetImportJob datasetImportJob = personalizeClient
                        .describeDatasetImportJob(describeDatasetImportJobRequest)
                        .datasetImportJob();

                status = datasetImportJob.status();
                System.out.println("Dataset import job status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return datasetImportJobArn;
```

```
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

- API-Details finden Sie CreateDatasetImportJob in der AWS SDK for Java 2.x API-Referenz.

`CreateEventTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateEventTracker`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName,
           String datasetGroupArn) {

       String eventTrackerId = "";
       String eventTrackerArn;
       long maxTime = 3 * 60 * 60; // 3 hours
       long waitInMilliseconds = 20 * 1000; // 20 seconds
       String status;

       try {

           CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
                   .name(eventTrackerName)
                   .datasetGroupArn(datasetGroupArn)
                   .build();

           CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient
```

```
                    .createEventTracker(createEventTrackerRequest);

            eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
            eventTrackerId = createEventTrackerResponse.trackingId();
            System.out.println("Event tracker ARN: " + eventTrackerArn);
            System.out.println("Event tracker ID: " + eventTrackerId);

            maxTime = Instant.now().getEpochSecond() + maxTime;

            DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();

            while (Instant.now().getEpochSecond() < maxTime) {

                status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
                System.out.println("EventTracker status: " + status);

                if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
            return eventTrackerId;
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- API-Details finden Sie CreateEventTracker in der AWS SDK for Java 2.x API-Referenz.

`CreateFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateFilter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public static String createFilter(PersonalizeClient personalizeClient,
        String filterName,
        String datasetGroupArn,
        String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .datasetGroupArn(datasetGroupArn)
                .filterExpression(filterExpression)
                .build();

        return personalizeClient.createFilter(request).filterArn();
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- API-Details finden Sie CreateFilter in der AWS SDK for Java 2.x API-Referenz.

`CreateRecommender`

Das folgende Codebeispiel zeigt die Verwendung`CreateRecommender`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

            maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

            while (Instant.now().getEpochSecond() < maxTime) {

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

                if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
```

```
                    System.out.println(e.getMessage());
                }
            }
            return recommenderArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateRecommender in der AWS SDK for Java 2.x API-Referenz.

`CreateSchema`

Das folgende Codebeispiel zeigt die Verwendung`CreateSchema`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .schema(schema)
                    .build();
```

```
            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

Erstellen Sie ein Schema mit einer Domain.

```
    public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain,
            String filePath) {

        String schema = null;
        try {
            schema = new String(Files.readAllBytes(Paths.get(filePath)));
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

        try {
            CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
                    .name(schemaName)
                    .domain(domain)
                    .schema(schema)
                    .build();

            String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

            System.out.println("Schema arn: " + schemaArn);

            return schemaArn;
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateSchema in der AWS SDK for Java 2.x API-Referenz.

`CreateSolution`

Das folgende Codebeispiel zeigt die Verwendung`CreateSolution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
           String datasetGroupArn,
           String solutionName,
           String recipeArn) {

       try {
           CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
                   .name(solutionName)
                   .datasetGroupArn(datasetGroupArn)
                   .recipeArn(recipeArn)
                   .build();

           CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
           return solutionResponse.solutionArn();

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
        return "";
    }
```

- API-Details finden Sie CreateSolution in der AWS SDK for Java 2.x API-Referenz.

`CreateSolutionVersion`

Das folgende Codebeispiel zeigt die Verwendung`CreateSolutionVersion`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
       long maxTime = 0;
       long waitInMilliseconds = 30 * 1000; // 30 seconds
       String solutionStatus = "";
       String solutionVersionStatus = "";
       String solutionVersionArn = "";

       try {
           DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

           // Wait until solution is active.
           while (Instant.now().getEpochSecond() < maxTime) {

               solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
               System.out.println("Solution status: " + solutionStatus);
```

```
                if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                    break;
                }
                try {
                    Thread.sleep(waitInMilliseconds);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }

            if (solutionStatus.equals("ACTIVE")) {

                CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
                        .solutionArn(solutionArn)
                        .build();

                CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient
                        .createSolutionVersion(createSolutionVersionRequest);
                solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

                System.out.println("Solution version ARN: " + solutionVersionArn);

                DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
                        .solutionVersionArn(solutionVersionArn)
                        .build();

                while (Instant.now().getEpochSecond() < maxTime) {

                    solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
                            .solutionVersion().status();
                    System.out.println("Solution version status: " +
solutionVersionStatus);

                    if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                        break;
                    }
                    try {
```

```
                        Thread.sleep(waitInMilliseconds);
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                return solutionVersionArn;
            }
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateSolutionVersion in der AWS SDK for Java 2.x API-Referenz.

`DeleteCampaign`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCampaign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {
        try {
            DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);
            System.out.println("Delete request sent successfully.");
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
e.awsErrorDetails().errorMessage());
```

```
            throw new RuntimeException(e);
        }
    }
}
```

- API-Details finden Sie DeleteCampaign in der AWS SDK for Java 2.x API-Referenz.

`DeleteEventTracker`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEventTracker`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
       try {
           DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
                   .eventTrackerArn(eventTrackerArn)
                   .build();

           int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCod

           System.out.println("Status code:" + status);

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DeleteEventTracker in der AWS SDK for Java 2.x API-Referenz.

`DeleteSolution`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSolution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void deleteGivenSolution(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           personalizeClient.deleteSolution(solutionRequest);
           System.out.println("Done");

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DeleteSolution in der AWS SDK for Java 2.x API-Referenz.

`DescribeCampaign`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCampaign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {

       try {
           DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                   .campaignArn(campaignArn)
                   .build();

           DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
           Campaign myCampaign = campaignResponse.campaign();
           System.out.println("The Campaign name is " + myCampaign.name());
           System.out.println("The Campaign status is " + myCampaign.status());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeCampaign in der AWS SDK for Java 2.x API-Referenz.

`DescribeRecipe`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRecipe`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

       try {
           DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
                   .recipeArn(recipeArn)
                   .build();

           DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
           System.out.println("The recipe name is " +
recipeResponse.recipe().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeRecipe in der AWS SDK for Java 2.x API-Referenz.

`DescribeSolution`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSolution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {

       try {
           DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
                   .solutionArn(solutionArn)
                   .build();

           DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
           System.out.println("The Solution name is " +
response.solution().name());

       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeSolution in der AWS SDK for Java 2.x API-Referenz.

`ListCampaigns`

Das folgende Codebeispiel zeigt die Verwendung`ListCampaigns`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

       try {
           ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
                   .maxResults(10)
```

```
                    .solutionArn(solutionArn)
                    .build();

            ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
            List<CampaignSummary> campaigns = response.campaigns();
            for (CampaignSummary campaign : campaigns) {
                System.out.println("Campaign name is : " + campaign.name());
                System.out.println("Campaign ARN is : " + campaign.campaignArn());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListCampaigns in der AWS SDK for Java 2.x API-Referenz.

`ListDatasetGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListDatasetGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listDSGroups(PersonalizeClient personalizeClient) {

        try {
            ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
                    .maxResults(15)
                    .build();

            ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
```

```
            List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
            for (DatasetGroupSummary group : groups) {
                System.out.println("The DataSet name is : " + group.name());
                System.out.println("The DataSet ARN is : " +
group.datasetGroupArn());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListDatasetGroups in der AWS SDK for Java 2.x API-Referenz.

`ListRecipes`

Das folgende Codebeispiel zeigt die Verwendung`ListRecipes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listAllRecipes(PersonalizeClient personalizeClient) {

        try {
            ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
                    .maxResults(15)
                    .build();

            ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
            List<RecipeSummary> recipes = response.recipes();
            for (RecipeSummary recipe : recipes) {
                System.out.println("The recipe ARN is: " + recipe.recipeArn());
                System.out.println("The recipe name is: " + recipe.name());
            }
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListRecipes in der AWS SDK for Java 2.x API-Referenz.

`ListSolutions`

Das folgende Codebeispiel zeigt die Verwendung`ListSolutions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listAllSolutions(PersonalizeClient personalizeClient, String
datasetGroupArn) {

        try {
            ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()
                    .maxResults(10)
                    .datasetGroupArn(datasetGroupArn)
                    .build();

            ListSolutionsResponse response =
personalizeClient.listSolutions(solutionsRequest);
            List<SolutionSummary> solutions = response.solutions();
            for (SolutionSummary solution : solutions) {
                System.out.println("The solution ARN is: " +
solution.solutionArn());
                System.out.println("The solution name is: " + solution.name());
            }

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListSolutions in der AWS SDK for Java 2.x API-Referenz.

`UpdateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`UpdateCampaign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String updateCampaign(PersonalizeClient personalizeClient,
            String campaignArn,
            String solutionVersionArn,
            Integer minProvisionedTPS) {

        try {
            // build the updateCampaignRequest
            UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .solutionVersionArn(solutionVersionArn)
                    .minProvisionedTPS(minProvisionedTPS)
                    .build();

            // update the campaign
            personalizeClient.updateCampaign(updateCampaignRequest);

            DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();
```

```
            DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
            Campaign updatedCampaign = campaignResponse.campaign();

            System.out.println("The Campaign status is " +
updatedCampaign.status());
            return updatedCampaign.status();

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie UpdateCampaign in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Personalize Events unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Personalize Events Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                        String datasetArn,
                        String item1Id,
                        String item1PropertyName,
                        String item1PropertyValue,
                        String item2Id,
                        String item2PropertyName,
                        String item2PropertyValue) {

                int responseCode = 0;
                ArrayList<Item> items = new ArrayList<>();

                try {
                        Item item1 = Item.builder()
                                        .itemId(item1Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        item1PropertyName,
 item1PropertyValue))
                                        .build();

                        items.add(item1);

                        Item item2 = Item.builder()
                                        .itemId(item2Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        item2PropertyName,
 item2PropertyValue))
                                        .build();

                        items.add(item2);

                        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
```

```
                                        .datasetArn(datasetArn)
                                        .items(items)
                                        .build();

                        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- API-Details finden Sie PutEvents in der AWS SDK for Java 2.x API-Referenz.

`PutUsers`

Das folgende Codebeispiel zeigt die Verwendung`PutUsers`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                        String datasetArn,
                        String user1Id,
                        String user1PropertyName,
                        String user1PropertyValue,
                        String user2Id,
                        String user2PropertyName,
                        String user2PropertyValue) {

                int responseCode = 0;
                ArrayList<User> users = new ArrayList<>();
```

```
                try {
                        User user1 = User.builder()
                                        .userId(user1Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user1PropertyName,
 user1PropertyValue))
                                        .build();

                        users.add(user1);

                        User user2 = User.builder()
                                        .userId(user2Id)
                                        .properties(String.format("{\"%1$s\": \"%2$s
\"}",
                                                        user2PropertyName,
 user2PropertyValue))
                                        .build();

                        users.add(user2);

                        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
                                        .datasetArn(datasetArn)
                                        .users(users)
                                        .build();

                        responseCode =
 personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
                        System.out.println("Response code: " + responseCode);
                        return responseCode;

                } catch (PersonalizeEventsException e) {
                        System.out.println(e.awsErrorDetails().errorMessage());
                }
                return responseCode;
        }
```

- API-Details finden Sie PutUsers in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Personalize Runtime unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Personalize Runtime Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetPersonalizedRanking`

Das folgende Codebeispiel zeigt die Verwendung`GetPersonalizedRanking`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
           String campaignArn,
           String userId,
           ArrayList<String> items) {

       try {
           GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
                   .campaignArn(campaignArn)
```

```
                    .userId(userId)
                    .inputList(items)
                    .build();

            GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient
                    .getPersonalizedRanking(rankingRecommendationsRequest);
            List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
            int rank = 1;
            for (PredictedItem item : rankedItems) {
                System.out.println("Item ranked at position " + rank + " details");
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
                System.out.println("---------------------------------------------");
                rank++;
            }
            return rankedItems;
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- API-Details finden Sie GetPersonalizedRanking in der AWS SDK for Java 2.x API-Referenz.

`GetRecommendations`

Das folgende Codebeispiel zeigt die Verwendung`GetRecommendations`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Liste von empfohlenen Artikeln ab.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .campaignArn(campaignArn)
                   .numResults(20)
                   .userId(userId)
                   .build();

           GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                   .getRecommendations(recommendationsRequest);
           List<PredictedItem> items = recommendationsResponse.itemList();
           for (PredictedItem item : items) {
               System.out.println("Item Id is : " + item.itemId());
               System.out.println("Item score is : " + item.score());
           }

       } catch (AwsServiceException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

Rufen Sie eine Liste von empfohlenen Artikeln von einem Empfehler ab, der in einer Domain-Datensatzgruppe erstellt wurde.

```
   public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn,
           String userId) {

       try {
           GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                   .recommenderArn(recommenderArn)
                   .numResults(20)
                   .userId(userId)
                   .build();
```

```
            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Verwenden Sie einen Filter, wenn Sie Empfehlungen anfordern.

```
    public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
            String campaignArn,
            String userId,
            String filterArn,
            String parameter1Name,
            String parameter1Value1,
            String parameter1Value2,
            String parameter2Name,
            String parameter2Value) {

        try {

            Map<String, String> filterValues = new HashMap<>();

            filterValues.put(parameter1Name, String.format("\"%1$s\",\"%2$s\"",
                    parameter1Value1, parameter1Value2));
            filterValues.put(parameter2Name, String.format("\"%1$s\"",
                    parameter2Value));

            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .campaignArn(campaignArn)
                    .numResults(20)
                    .userId(userId)
```

```
                    .filterArn(filterArn)
                    .filterValues(filterValues)
                    .build();

            GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient
                    .getRecommendations(recommendationsRequest);
            List<PredictedItem> items = recommendationsResponse.itemList();

            for (PredictedItem item : items) {
                System.out.println("Item Id is : " + item.itemId());
                System.out.println("Item score is : " + item.score());
            }
        } catch (PersonalizeRuntimeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie GetRecommendations in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Pinpoint unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateApp`

Das folgende Codebeispiel zeigt die Verwendung`CreateApp`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateAppRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateAppResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateApplicationRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateApp {
    public static void main(String[] args) {
        final String usage = """

                  Usage:  <appName>

                  Where:
                   appName - The name of the application to create.

                 """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }
        String appName = args[0];
        System.out.println("Creating an application with name: " + appName);

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String appID = createApplication(pinpoint, appName);
        System.out.println("App ID is: " + appID);
        pinpoint.close();
    }

    public static String createApplication(PinpointClient pinpoint, String appName)
 {
        try {
            CreateApplicationRequest appRequest = CreateApplicationRequest.builder()
                    .name(appName)
                    .build();

            CreateAppRequest request = CreateAppRequest.builder()
                    .createApplicationRequest(appRequest)
                    .build();

            CreateAppResponse result = pinpoint.createApp(request);
            return result.applicationResponse().id();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateApp in der AWS SDK for Java 2.x API-Referenz.

`CreateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`CreateCampaign`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Kampagne.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.Message;
import software.amazon.awssdk.services.pinpoint.model.Schedule;
import software.amazon.awssdk.services.pinpoint.model.Action;
import software.amazon.awssdk.services.pinpoint.model.MessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.WriteCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignResponse;
import software.amazon.awssdk.services.pinpoint.model.CreateCampaignRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCampaign {
    public static void main(String[] args) {

        final String usage = """

                Usage:   <appId> <segmentId>

                Where:
                  appId - The ID of the application to create the campaign in.
                  segmentId - The ID of the segment to create the campaign from.
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String segmentId = args[1];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPinCampaign(pinpoint, appId, segmentId);
        pinpoint.close();
    }

    public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
        CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
        System.out.println("Campaign " + result.name() + " created.");
        System.out.println(result.description());
    }

    public static CampaignResponse createCampaign(PinpointClient client, String
appID, String segmentID) {

        try {
            Schedule schedule = Schedule.builder()
                    .startTime("IMMEDIATE")
                    .build();

            Message defaultMessage = Message.builder()
                    .action(Action.OPEN_APP)
                    .body("My message body.")
                    .title("My message title.")
                    .build();

            MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
                    .defaultMessage(defaultMessage)
                    .build();

            WriteCampaignRequest request = WriteCampaignRequest.builder()
                    .description("My description")
                    .schedule(schedule)
                    .name("MyCampaign")
```

```
                    .segmentId(segmentID)
                    .messageConfiguration(messageConfiguration)
                    .build();

            CreateCampaignResponse result =
 client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build());

            System.out.println("Campaign ID: " + result.campaignResponse().id());
            return result.campaignResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
}
```

- API-Details finden Sie CreateCampaign in der AWS SDK for Java 2.x API-Referenz.

`CreateExportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateExportJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Exportieren Sie einen Endpunkt.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.ExportJobRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.GetExportJobRequest;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;

/**
 * To run this code example, you need to create an AWS Identity and Access
 * Management (IAM) role with the correct policy as described in this
 * documentation:
 * https://docs.aws.amazon.com/pinpoint/latest/developerguide/audience-data-
export.html
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ExportEndpoints {
    public static void main(String[] args) {
        final String usage = """

                This program performs the following steps:

                1. Exports the endpoints to an Amazon S3 bucket.
```

```
                2. Downloads the exported endpoints files from Amazon S3.
                3. Parses the endpoints files to obtain the endpoint IDs and prints
them.
                Usage: ExportEndpoints <applicationId> <s3BucketName>
<iamExportRoleArn> <path>

                Where:
                  applicationId - The ID of the Amazon Pinpoint application that has
the endpoint.
                  s3BucketName - The name of the Amazon S3 bucket to export the JSON
file to.\s
                  iamExportRoleArn - The ARN of an IAM role that grants Amazon
Pinpoint write permissions to the S3 bucket.  path - The path where the files
downloaded from the Amazon S3 bucket are written (for example, C:/AWS/).
                """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String applicationId = args[0];
       String s3BucketName = args[1];
       String iamExportRoleArn = args[2];
       String path = args[3];
       System.out.println("Deleting an application with ID: " + applicationId);

       Region region = Region.US_EAST_1;
       PinpointClient pinpoint = PinpointClient.builder()
               .region(region)
               .build();

       S3Client s3Client = S3Client.builder()
               .region(region)
               .build();

       exportAllEndpoints(pinpoint, s3Client, applicationId, s3BucketName, path,
iamExportRoleArn);
       pinpoint.close();
       s3Client.close();
   }

   public static void exportAllEndpoints(PinpointClient pinpoint,
           S3Client s3Client,
```

```
            String applicationId,
            String s3BucketName,
            String path,
            String iamExportRoleArn) {

        try {
            List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
 s3BucketName, iamExportRoleArn,
                    applicationId);
            List<String> endpointFileKeys = objectKeys.stream().filter(o ->
 o.endsWith(".gz"))
                    .collect(Collectors.toList());
            downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
 s3Client, String s3BucketName,
            String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
 dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
 "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                    .roleArn(iamExportRoleArn)
                    .s3UrlPrefix(s3UrlPrefix)
                    .build();

            CreateExportJobRequest exportJobRequest =
 CreateExportJobRequest.builder()
                    .applicationId(applicationId)
                    .exportJobRequest(jobRequest)
```

```
                    .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
                    "bucket %s . . .\n", applicationId, s3BucketName);

            CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                    .bucket(s3BucketName)
                    .prefix(endpointsKeyPrefix)
                    .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
            List<S3Object> objects = v2Response.contents();
            for (S3Object object : objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
            String applicationId,
            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
```

```
                    .jobId(jobId)
                    .applicationId(applicationId)
                    .build();

            do {
                getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }

        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                        .bucket(s3BucketName)
                        .key(key)
                        .build();

                ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
```

```
                String fileSuffix = new
 SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix + ".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");

        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateExportJob in der AWS SDK for Java 2.x API-Referenz.

`CreateImportJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateImportJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Importieren Sie ein Segment.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.ImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.ImportJobRequest;
import software.amazon.awssdk.services.pinpoint.model.Format;
import software.amazon.awssdk.services.pinpoint.model.CreateImportJobResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportSegment {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <bucket> <key> <roleArn>\s

                Where:
                  appId - The application ID to create a segment for.
                  bucket - The name of the Amazon S3 bucket that contains the
 segment definitons.
                  key - The key of the S3 object.
                  roleArn - ARN of the role that allows Amazon Pinpoint to
 access S3. You need to set trust management for this to work. See https://
docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html
                  """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String bucket = args[1];
        String key = args[2];
        String roleArn = args[3];

        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        ImportJobResponse response = createImportSegment(pinpoint, appId, bucket,
 key, roleArn);
        System.out.println("Import job for " + bucket + " submitted.");
        System.out.println("See application " + response.applicationId() + " for
 import job status.");
```

```
        System.out.println("See application " + response.jobStatus() + " for import
 job status.");
        pinpoint.close();
    }

    public static ImportJobResponse createImportSegment(PinpointClient client,
            String appId,
            String bucket,
            String key,
            String roleArn) {

        try {
            ImportJobRequest importRequest = ImportJobRequest.builder()
                    .defineSegment(true)
                    .registerEndpoints(true)
                    .roleArn(roleArn)
                    .format(Format.JSON)
                    .s3Url("s3://" + bucket + "/" + key)
                    .build();

            CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
                    .importJobRequest(importRequest)
                    .applicationId(appId)
                    .build();

            CreateImportJobResponse jobResponse =
 client.createImportJob(jobRequest);
            return jobResponse.importJobResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- API-Details finden Sie CreateImportJob in der AWS SDK for Java 2.x API-Referenz.

`CreateSegment`

Das folgende Codebeispiel zeigt die Verwendung`CreateSegment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AttributeDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.AttributeType;
import software.amazon.awssdk.services.pinpoint.model.RecencyDimension;
import software.amazon.awssdk.services.pinpoint.model.SegmentBehaviors;
import software.amazon.awssdk.services.pinpoint.model.SegmentDemographics;
import software.amazon.awssdk.services.pinpoint.model.SegmentLocation;
import software.amazon.awssdk.services.pinpoint.model.SegmentDimensions;
import software.amazon.awssdk.services.pinpoint.model.WriteSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentRequest;
import software.amazon.awssdk.services.pinpoint.model.CreateSegmentResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSegment {
        public static void main(String[] args) {
                final String usage = """

                                Usage:   <appId>

                                Where:
                                  appId - The application ID to create a segment
 for.
```

```
                                """;

                if (args.length != 1) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String appId = args[0];
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                SegmentResponse result = createSegment(pinpoint, appId);
                System.out.println("Segment " + result.name() + " created.");
                System.out.println(result.segmentType());
                pinpoint.close();
        }

        public static SegmentResponse createSegment(PinpointClient client, String
appId) {
                try {
                        Map<String, AttributeDimension> segmentAttributes = new
HashMap<>();
                        segmentAttributes.put("Team", AttributeDimension.builder()
                                        .attributeType(AttributeType.INCLUSIVE)
                                        .values("Lakers")
                                        .build());

                        RecencyDimension recencyDimension =
RecencyDimension.builder()
                                        .duration("DAY_30")
                                        .recencyType("ACTIVE")
                                        .build();

                        SegmentBehaviors segmentBehaviors =
SegmentBehaviors.builder()
                                        .recency(recencyDimension)
                                        .build();

                        SegmentDemographics segmentDemographics =
SegmentDemographics
                                        .builder()
                                        .build();
```

```
                        SegmentLocation segmentLocation = SegmentLocation
                                        .builder()
                                        .build();

                        SegmentDimensions dimensions = SegmentDimensions
                                        .builder()
                                        .attributes(segmentAttributes)
                                        .behavior(segmentBehaviors)
                                        .demographic(segmentDemographics)
                                        .location(segmentLocation)
                                        .build();

                        WriteSegmentRequest writeSegmentRequest =
 WriteSegmentRequest.builder()
                                        .name("MySegment")
                                        .dimensions(dimensions)
                                        .build();

                        CreateSegmentRequest createSegmentRequest =
 CreateSegmentRequest.builder()
                                        .applicationId(appId)
                                        .writeSegmentRequest(writeSegmentRequest)
                                        .build();

                        CreateSegmentResponse createSegmentResult =
 client.createSegment(createSegmentRequest);
                        System.out.println("Segment ID: " +
 createSegmentResult.segmentResponse().id());
                        System.out.println("Done");
                        return createSegmentResult.segmentResponse();

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
                return null;
        }
}
```

- API-Details finden Sie CreateSegment in der AWS SDK for Java 2.x API-Referenz.

`DeleteApp`

Das folgende Codebeispiel zeigt die Verwendung`DeleteApp`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Anwendung.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteAppResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteApp {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                 appId - The ID of the application to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String appId = args[0];
        System.out.println("Deleting an application with ID: " + appId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinApp(pinpoint, appId);
        System.out.println("Done");
        pinpoint.close();
    }

    public static void deletePinApp(PinpointClient pinpoint, String appId) {
        try {
            DeleteAppRequest appRequest = DeleteAppRequest.builder()
                    .applicationId(appId)
                    .build();

            DeleteAppResponse result = pinpoint.deleteApp(appRequest);
            String appName = result.applicationResponse().name();
            System.out.println("Application " + appName + " has been deleted.");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteApp in der AWS SDK for Java 2.x API-Referenz.

`DeleteEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEndpoint`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Endpunktes

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.DeleteEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appName> <endpointId >

                Where:
                  appId - The id of the application to delete.
                  endpointId - The id of the endpoint to delete.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpointId = args[1];
        System.out.println("Deleting an endpoint with id: " + endpointId);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }
```

```
    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API-Details finden Sie DeleteEndpoint in der AWS SDK for Java 2.x API-Referenz.

`GetEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`GetEndpoint`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.google.gson.FieldNamingPolicy;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
```

```
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class LookUpEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId> <endpoint>

                Where:
                  appId - The ID of the application to delete.
                  endpoint - The ID of the endpoint.\s
                  """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        String endpoint = args[1];
        System.out.println("Looking up an endpoint point with ID: " + endpoint);
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        lookupPinpointEndpoint(pinpoint, appId, endpoint);
        pinpoint.close();
    }

    public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
 String endpoint) {
        try {
            GetEndpointRequest appRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpoint)
```

```
                    .build();

            GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
            EndpointResponse endResponse = result.endpointResponse();

            // Uses the Google Gson library to pretty print the endpoint JSON.
            Gson gson = new GsonBuilder()
                    .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                    .setPrettyPrinting()
                    .create();

            String endpointJson = gson.toJson(endResponse);
            System.out.println(endpointJson);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API-Details finden Sie GetEndpoint in der AWS SDK for Java 2.x API-Referenz.

`GetSegments`

Das folgende Codebeispiel zeigt die Verwendung`GetSegments`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Segmente auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsRequest;
```

```
import software.amazon.awssdk.services.pinpoint.model.GetSegmentsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SegmentResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSegments {
    public static void main(String[] args) {
        final String usage = """

                Usage:   <appId>

                Where:
                  appId - The ID of the application that contains a segment.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSegs(pinpoint, appId);
        pinpoint.close();
    }

    public static void listSegs(PinpointClient pinpoint, String appId) {
        try {
            GetSegmentsRequest request = GetSegmentsRequest.builder()
                    .applicationId(appId)
                    .build();
```

```
            GetSegmentsResponse response = pinpoint.getSegments(request);
            List<SegmentResponse> segments = response.segmentsResponse().item();
            for (SegmentResponse segment : segments) {
                System.out
                        .println("Segement " + segment.id() + " " + segment.name() +
 " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetSegments in der AWS SDK for Java 2.x API-Referenz.

`GetSmsChannel`

Das folgende Codebeispiel zeigt die Verwendung`GetSmsChannel`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelResponse;
import software.amazon.awssdk.services.pinpoint.model.GetSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.SMSChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateSmsChannelResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateChannel {
    public static void main(String[] args) {
        final String usage = """

                Usage: CreateChannel <appId>

                Where:
                  appId - The name of the application whose channel is updated.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        SMSChannelResponse getResponse = getSMSChannel(pinpoint, appId);
        toggleSmsChannel(pinpoint, appId, getResponse);
        pinpoint.close();
    }

    private static SMSChannelResponse getSMSChannel(PinpointClient client, String
 appId) {
        try {
            GetSmsChannelRequest request = GetSmsChannelRequest.builder()
                    .applicationId(appId)
                    .build();

            SMSChannelResponse response =
 client.getSmsChannel(request).smsChannelResponse();
            System.out.println("Channel state is " + response.enabled());
            return response;

        } catch (PinpointException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void toggleSmsChannel(PinpointClient client, String appId,
 SMSChannelResponse getResponse) {
        boolean enabled = !getResponse.enabled();
        try {
            SMSChannelRequest request = SMSChannelRequest.builder()
                    .enabled(enabled)
                    .build();

            UpdateSmsChannelRequest updateRequest =
 UpdateSmsChannelRequest.builder()
                    .smsChannelRequest(request)
                    .applicationId(appId)
                    .build();

            UpdateSmsChannelResponse result =
 client.updateSmsChannel(updateRequest);
            System.out.println("Channel state: " +
 result.smsChannelResponse().enabled());

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetSmsChannel in der AWS SDK for Java 2.x API-Referenz.

`GetUserEndpoints`

Das folgende Codebeispiel zeigt die Verwendung`GetUserEndpoints`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsRequest;
import software.amazon.awssdk.services.pinpoint.model.GetUserEndpointsResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListEndpointIds {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <applicationId> <userId>

                Where:
                   applicationId - The ID of the Amazon Pinpoint application that
 has the endpoint.
                   userId - The user id applicable to the endpoints""";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String applicationId = args[0];
        String userId = args[1];
```

```
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllEndpoints(pinpoint, applicationId, userId);
        pinpoint.close();
    }

    public static void listAllEndpoints(PinpointClient pinpoint,
            String applicationId,
            String userId) {

        try {
            GetUserEndpointsRequest endpointsRequest =
 GetUserEndpointsRequest.builder()
                    .userId(userId)
                    .applicationId(applicationId)
                    .build();

            GetUserEndpointsResponse response =
 pinpoint.getUserEndpoints(endpointsRequest);
            List<EndpointResponse> endpoints = response.endpointsResponse().item();

            // Display the results.
            for (EndpointResponse endpoint : endpoints) {
                System.out.println("The channel type is: " +
 endpoint.channelType());
                System.out.println("The address is  " + endpoint.address());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetUserEndpoints in der AWS SDK for Java 2.x API-Referenz.

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmailPart;
import software.amazon.awssdk.services.pinpoint.model.SimpleEmail;
import software.amazon.awssdk.services.pinpoint.model.EmailMessage;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessage {

        // The character encoding the you want to use for the subject line and
```

```
        // message body of the email.
        public static String charset = "UTF-8";

    // The body of the email for recipients whose email clients support HTML
content.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
for Java 2.x

        """;

        public static void main(String[] args) {
                final String usage = """

                                Usage:    <subject> <appId> <senderAddress>
<toAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in Amazon
Pinpoint in the region you're using to send email\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendEmail(pinpoint, subject, senderAddress, toAddress);
        System.out.println("Email was sent");
        pinpoint.close();
    }
```

```
    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
 subject, String senderAddress, String toAddress) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Senden einer E-Mail-Nachricht mit CC-Werten.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import  software.amazon.awssdk.services.pinpointemail.PinpointEmailClient;
import software.amazon.awssdk.services.pinpointemail.model.Body;
import software.amazon.awssdk.services.pinpointemail.model.Content;
import software.amazon.awssdk.services.pinpointemail.model.Destination;
import software.amazon.awssdk.services.pinpointemail.model.EmailContent;
import software.amazon.awssdk.services.pinpointemail.model.Message;
import software.amazon.awssdk.services.pinpointemail.model.SendEmailRequest;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendEmailMessageCC {

    // The body of the email.
    static final String body = """
        Amazon Pinpoint test (AWS SDK for Java 2.x)

        This email was sent through the Amazon Pinpoint Email API using the AWS SDK
 for Java 2.x

        """;
    public static void main(String[] args) {
        final String usage = """

            Usage:    <subject> <senderAddress> <toAddress> <ccAddress>

            Where:
               subject - The email subject to use.
               senderAddress - The from address. This address has to be verified in
 Amazon Pinpoint in the region you're using to send email\s
               toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email\s
               ccAddress - The CC address.
            """;
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String subject = args[0];
        String senderAddress = args[1];
        String toAddress = args[2];
        String ccAddress = args[3];

        System.out.println("Sending a message");
        PinpointEmailClient pinpoint = PinpointEmailClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ArrayList<String> ccList = new ArrayList<>();
        ccList.add(ccAddress);
        sendEmail(pinpoint, subject, senderAddress, toAddress, ccList);
        pinpoint.close();
    }

    public static void sendEmail(PinpointEmailClient pinpointEmailClient, String
 subject, String senderAddress, String toAddress, ArrayList<String> ccAddresses) {
        try {
            Content content = Content.builder()
                .data(body)
                .build();

            Body messageBody = Body.builder()
                .text(content)
                .build();

            Message message = Message.builder()
                .body(messageBody)
                .subject(Content.builder().data(subject).build())
                .build();

            Destination destination = Destination.builder()
                .toAddresses(toAddress)
                .ccAddresses(ccAddresses)
                .build();

            EmailContent emailContent = EmailContent.builder()
                .simple(message)
```

```
                .build();

            SendEmailRequest sendEmailRequest = SendEmailRequest.builder()
                .fromEmailAddress(senderAddress)
                .destination(destination)
                .content(emailContent)
                .build();

            pinpointEmailClient.sendEmail(sendEmailRequest);
            System.out.println("Message Sent");

        } catch (PinpointException e) {
            // Handle exception
            e.printStackTrace();
        }
    }
}
```

Senden Sie eine SMS-Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
                                  message - The body of the message to send.
                                  appId - The Amazon Pinpoint project/application ID
 to use when you send this message.
                                  originationNumber - The phone number or short code
 that you specify has to be associated with your Amazon Pinpoint account. For best
 results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                  destinationNumber - The recipient's phone number.
 For best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                                  """;

                if (args.length != 4) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
```

```
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
                pinpoint.close();
        }

        public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request = SendMessagesRequest.builder()
                                        .applicationId(appId)
```

```
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
 pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

Senden Sie Batch-SMS-Nachrichten.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
    public static String senderId = "MySenderID";

    public static void main(String[] args) {
        final String usage = """

                Usage:   <message> <appId> <originationNumber> <destinationNumber>
 <destinationNumber1>\s

                Where:
                  message - The body of the message to send.
                  appId - The Amazon Pinpoint project/application ID to use when you
 send this message.
                  originationNumber - The phone number or short code that you
 specify has to be associated with your Amazon Pinpoint account. For best results,
 specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
 results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).
                  destinationNumber1 - The second recipient's phone number.  For
 best results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                      String originationNumber,
                                      String destinationNumber, String
destinationNumber1) {
        try {
            Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
            AddressConfiguration addConfig = AddressConfiguration.builder()
                    .channelType(ChannelType.SMS)
                    .build();

            // Add an entry to the Map object for each number to whom you want to
send a
            // message.
            addressMap.put(destinationNumber, addConfig);
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                    .body(message)
                    .messageType(messageType)
                    .originationNumber(originationNumber)
                    .senderId(senderId)
                    .keyword(registeredKeyword)
                    .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                    .smsMessage(smsMessage)
                    .build();
```

```
            MessageRequest msgReq = MessageRequest.builder()
                    .addresses(addressMap)
                    .messageConfiguration(direct)
                    .build();

            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                    .applicationId(appId)
                    .messageRequest(msgReq)
                    .build();

            SendMessagesResponse response = pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SendMessages in der AWS SDK for Java 2.x API-Referenz.

`UpdateEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`UpdateEndpoint`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.EndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.EndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.UpdateEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointRequest;
import software.amazon.awssdk.services.pinpoint.model.GetEndpointResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import software.amazon.awssdk.services.pinpoint.model.EndpointDemographic;
import software.amazon.awssdk.services.pinpoint.model.EndpointLocation;
import software.amazon.awssdk.services.pinpoint.model.EndpointUser;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.UUID;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Date;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateEndpoint {
    public static void main(String[] args) {
        final String usage = """

                Usage:  <appId>

                Where:
                  appId - The ID of the application to create an endpoint for.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String appId = args[0];
        PinpointClient pinpoint = PinpointClient.builder()
                .region(Region.US_EAST_1)
                .build();

        EndpointResponse response = createEndpoint(pinpoint, appId);
        System.out.println("Got Endpoint: " + response.id());
        pinpoint.close();
    }

    public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
        String endpointId = UUID.randomUUID().toString();
        System.out.println("Endpoint ID: " + endpointId);

        try {
            EndpointRequest endpointRequest = createEndpointRequestData();
            UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .endpointRequest(endpointRequest)
                    .build();

            UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
            System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

            GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
            System.out.println(getEndpointResponse.endpointResponse().address());

System.out.println(getEndpointResponse.endpointResponse().channelType());

System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
            System.out.println(getEndpointResponse.endpointResponse().requestId());
```

```
            System.out.println(getEndpointResponse.endpointResponse().user());

            return getEndpointResponse.endpointResponse();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static EndpointRequest createEndpointRequestData() {
        try {
            List<String> favoriteTeams = new ArrayList<>();
            favoriteTeams.add("Lakers");
            favoriteTeams.add("Warriors");
            HashMap<String, List<String>> customAttributes = new HashMap<>();
            customAttributes.put("team", favoriteTeams);

            EndpointDemographic demographic = EndpointDemographic.builder()
                    .appVersion("1.0")
                    .make("apple")
                    .model("iPhone")
                    .modelVersion("7")
                    .platform("ios")
                    .platformVersion("10.1.1")
                    .timezone("America/Los_Angeles")
                    .build();

            EndpointLocation location = EndpointLocation.builder()
                    .city("Los Angeles")
                    .country("US")
                    .latitude(34.0)
                    .longitude(-118.2)
                    .postalCode("90068")
                    .region("CA")
                    .build();

            Map<String, Double> metrics = new HashMap<>();
            metrics.put("health", 100.00);
            metrics.put("luck", 75.00);

            EndpointUser user = EndpointUser.builder()
                    .userId(UUID.randomUUID().toString())
```

```
                    .build();

            DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
 "Z" to indicate UTC, no timezone
                                                                            // offset
            String nowAsISO = df.format(new Date());

            return EndpointRequest.builder()
                    .address(UUID.randomUUID().toString())
                    .attributes(customAttributes)
                    .channelType("APNS")
                    .demographic(demographic)
                    .effectiveDate(nowAsISO)
                    .location(location)
                    .metrics(metrics)
                    .optOut("NONE")
                    .requestId(UUID.randomUUID().toString())
                    .user(user)
                    .build();

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- API-Details finden Sie UpdateEndpoint in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Pinpoint SMS und Voice API unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon Pinpoint SMS and Voice API AWS SDK for Java 2.x Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendVoiceMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendVoiceMessage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```

```
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
            + "<emphasis>Amazon Pinpoint</emphasis> "
            + "using the <break strength='weak'/>AWS "
            + "SDK for Java. "
            + "<amazon:effect phonation='soft'>Thank "
            + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = """
                Usage:   <originationNumber> <destinationNumber>\s

                Where:
                  originationNumber - The phone number or short code that you
 specify has to be associated with your Amazon Pinpoint account. For best results,
 specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
 results, you should specify the phone number in E.164 format (for example,
 +1-555-555-5654).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
        String originationNumber = args[0];
        String destinationNumber = args[1];
```

```
        System.out.println("Sending a voice message");

        // Set the content type to application/json.
        List<String> listVal = new ArrayList<>();
        listVal.add("application/json");
        Map<String, List<String>> values = new HashMap<>();
        values.put("Content-Type", listVal);

        ClientOverrideConfiguration config2 = ClientOverrideConfiguration.builder()
                .headers(values)
                .build();

        PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
                .overrideConfiguration(config2)
                .region(Region.US_EAST_1)
                .build();

        sendVoiceMsg(client, originationNumber, destinationNumber);
        client.close();
    }

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
                                    String destinationNumber) {
        try {
            SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                    .languageCode(languageCode)
                    .text(ssmlMessage)
                    .voiceId(voiceName)
                    .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                    .ssmlMessage(ssmlMessageType)
                    .build();

            SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
                    .destinationPhoneNumber(destinationNumber)
                    .originationPhoneNumber(originationNumber)
                    .content(content)
                    .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");
```

```
        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SendVoiceMessage in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Polly unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.Voice;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeVoicesSample {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        describeVoice(polly);
        polly.close();
    }

    public static void describeVoice(PollyClient polly) {
        try {
            DescribeVoicesRequest voicesRequest = DescribeVoicesRequest.builder()
                    .languageCode("en-US")
                    .build();

            DescribeVoicesResponse enUsVoicesResult =
 polly.describeVoices(voicesRequest);
```

```
            List<Voice> voices = enUsVoicesResult.voices();
            for (Voice myVoice : voices) {
                System.out.println("The ID of the voice is " + myVoice.id());
                System.out.println("The gender of the voice is " +
 myVoice.gender());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeVoices in der AWS SDK for Java 2.x API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.ListLexiconsResponse;
import software.amazon.awssdk.services.polly.model.ListLexiconsRequest;
import software.amazon.awssdk.services.polly.model.LexiconDescription;
import software.amazon.awssdk.services.polly.model.PollyException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLexicons {
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listLexicons(polly);
        polly.close();
    }

    public static void listLexicons(PollyClient client) {
        try {
            ListLexiconsRequest listLexiconsRequest = ListLexiconsRequest.builder()
                    .build();

            ListLexiconsResponse listLexiconsResult =
 client.listLexicons(listLexiconsRequest);
            List<LexiconDescription> lexiconDescription =
 listLexiconsResult.lexicons();
            for (LexiconDescription lexDescription : lexiconDescription) {
                System.out.println("The name of the Lexicon is " +
 lexDescription.name());
            }

        } catch (PollyException e) {
            System.err.println("Exception caught: " + e);
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListLexicons in der AWS SDK for Java 2.x API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import javazoom.jl.decoder.JavaLayerException;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.polly.PollyClient;
import software.amazon.awssdk.services.polly.model.DescribeVoicesRequest;
import software.amazon.awssdk.services.polly.model.Voice;
import software.amazon.awssdk.services.polly.model.DescribeVoicesResponse;
import software.amazon.awssdk.services.polly.model.OutputFormat;
import software.amazon.awssdk.services.polly.model.PollyException;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechRequest;
import software.amazon.awssdk.services.polly.model.SynthesizeSpeechResponse;
import java.io.IOException;
import java.io.InputStream;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PollyDemo {
    private static final String SAMPLE = "Congratulations. You have successfully
 built this working demo " +
            " of Amazon Polly in Java Version 2. Have fun building voice enabled
 apps with Amazon Polly (that's me!), and always "
            +
            " look at the AWS website for tips and tricks on using Amazon Polly and
 other great services from AWS";
```

```
    public static void main(String args[]) {
        PollyClient polly = PollyClient.builder()
                .region(Region.US_WEST_2)
                .build();

        talkPolly(polly);
        polly.close();
    }

    public static void talkPolly(PollyClient polly) {
        try {
            DescribeVoicesRequest describeVoiceRequest =
DescribeVoicesRequest.builder()
                    .engine("standard")
                    .build();

            DescribeVoicesResponse describeVoicesResult =
polly.describeVoices(describeVoiceRequest);
            Voice voice = describeVoicesResult.voices().stream()
                    .filter(v -> v.name().equals("Joanna"))
                    .findFirst()
                    .orElseThrow(() -> new RuntimeException("Voice not found"));
            InputStream stream = synthesize(polly, SAMPLE, voice, OutputFormat.MP3);
            AdvancedPlayer player = new AdvancedPlayer(stream,

javazoom.jl.player.FactoryRegistry.systemRegistry().createAudioDevice());
            player.setPlayBackListener(new PlaybackListener() {
                public void playbackStarted(PlaybackEvent evt) {
                    System.out.println("Playback started");
                    System.out.println(SAMPLE);
                }

                public void playbackFinished(PlaybackEvent evt) {
                    System.out.println("Playback finished");
                }
            });

            // play it!
            player.play();

        } catch (PollyException | JavaLayerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static InputStream synthesize(PollyClient polly, String text, Voice
 voice, OutputFormat format)
            throws IOException {
        SynthesizeSpeechRequest synthReq = SynthesizeSpeechRequest.builder()
                .text(text)
                .voiceId(voice.id())
                .outputFormat(format)
                .build();

        ResponseInputStream<SynthesizeSpeechResponse> synthRes =
 polly.synthesizeSpeech(synthReq);
        return synthRes;
    }
}
```

- API-Details finden Sie SynthesizeSpeech in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon RDS unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
```

```
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie mehrere Operationen aus.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
 software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySql database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
```

```
                    <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

               Where:
                   dbGroupName - The database group name.\s
                   dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
                   dbInstanceIdentifier - The database instance identifier\s
                   dbName - The database name.\s
                   dbSnapshotIdentifier - The snapshot identifier.\s
                   secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
               """;

       if (args.length != 6) {
           System.out.println(usage);
           System.exit(1);
       }

       String dbGroupName = args[0];
       String dbParameterGroupFamily = args[1];
       String dbInstanceIdentifier = args[2];
       String dbName = args[3];
       String dbSnapshotIdentifier = args[4];
       String secretName = args[5];

       Gson gson = new Gson();
       User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
       String masterUsername = user.getUsername();
       String masterUserPassword = user.getPassword();

       Region region = Region.US_WEST_2;
       RdsClient rdsClient = RdsClient.builder()
               .region(region)
               .build();
       System.out.println(DASHES);
       System.out.println("Welcome to the Amazon RDS example scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Return a list of the available DB engines");
       describeDBEngines(rdsClient);
       System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbParameterGroups(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbParameters(rdsClient, dbGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBParas(rdsClient, dbGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated value");
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
 the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "9. Create an RDS database instance that contains a MySql database
 and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
 dbInstanceIdentifier, dbName, masterUsername,
                masterUserPassword);
```

```
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
```

```
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }
```

```
            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
```

```
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   // Create a database instance and return the ARN of the database.
   public static String createDatabaseInstance(RdsClient rdsClient,
           String dbGroupName,
           String dbInstanceIdentifier,
           String dbName,
           String userName,
           String userPassword) {

       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .allocatedStorage(100)
               .dbName(dbName)
               .engine("mysql")
               .dbInstanceClass("db.t3.medium") // Updated to a supported class
               .engineVersion("8.0.32")         // Updated to a supported version
               .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
               .masterUsername(userName)
               .masterUserPassword(userPassword)
               .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }

       return "";
   }

   // Get a list of micro instances.
   public static void getMicroInstances(RdsClient rdsClient) {
       try {
           DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                   .builder()
                   .engine("mysql")
```

```
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
{
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
```

```
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
```

```
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
```

```
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
```

```
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }
```

```
    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbName,
            String userName,
            String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
       try {
           CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                   .dbParameterGroupName(dbGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK for Java 2.x API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Create an Amazon RDS snapshot.
   public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
       try {
           CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .dbSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
           System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK for Java 2.x - API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void deleteDatabaseInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
 DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Delete the parameter group after database has been deleted.
   // An exception is thrown if you attempt to delete the para group while database
   // exists.
   public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
           while (!isDataDel) {
               DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
               List<DBInstance> instanceList = response.dbInstances();
               int listSize = instanceList.size();
               didFind = false;
               int index = 1;
               for (DBInstance instance : instanceList) {
                   instanceARN = instance.dbInstanceArn();
                   if (instanceARN.compareTo(dbARN) == 0) {
                       System.out.println(dbARN + " still exists");
                       didFind = true;
                   }
                   if ((index == listSize) && (!didFind)) {
                       // Went through the entire list and did not find the
database ARN.
                       isDataDel = true;
                   }
                   Thread.sleep(sleepTime * 1000);
                   index++;
               }
           }
```

```
            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
 DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK for Java 2.x API-Referenz.

`DescribeAccountAttributes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAccountAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.AccountQuota;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeAccountAttributesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccountAttributes {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        getAccountAttributes(rdsClient);
        rdsClient.close();
    }

    public static void getAccountAttributes(RdsClient rdsClient) {
        try {
            DescribeAccountAttributesResponse response =
 rdsClient.describeAccountAttributes();
            List<AccountQuota> quotasList = response.accountQuotas();
            for (AccountQuota quotas : quotasList) {
                System.out.println("Name is: " + quotas.accountQuotaName());
                System.out.println("Max value is " + quotas.max());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeAccountAttributes in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .defaultOnly(true)
                   .engine("mysql")
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.RdsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeDBInstances {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeInstances(rdsClient);
        rdsClient.close();
    }

    public static void describeInstances(RdsClient rdsClient) {
        try {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
```

```
            for (DBInstance instance : instanceList) {
                System.out.println("Instance ARN is: " + instance.dbInstanceArn());
                System.out.println("The Engine is " + instance.engine());
                System.out.println("Connection endpoint is" +
 instance.endpoint().address());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK for Java 2.x -API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeDbParameterGroups(RdsClient rdsClient, String
 dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
 DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
 rdsClient.describeDBParameterGroups(groupsRequest);
```

```
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK for Java 2.x API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
```

```
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK for Java 2.x -API-Referenz.

`GenerateRDSAuthToken`

Das folgende Codebeispiel zeigt die Verwendung`GenerateRDSAuthToken`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die RdsUtilities Klasse, um ein Authentifizierungstoken zu generieren.

```
public class GenerateRDSAuthToken {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <masterUsername>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUsername - The master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUsername = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        String token = getAuthToken(rdsClient, dbInstanceIdentifier,
 masterUsername);
        System.out.println("The token response is " + token);
    }

    public static String getAuthToken(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUsername) {

        RdsUtilities utilities = rdsClient.utilities();
```

```
        try {
            GenerateAuthenticationTokenRequest tokenRequest =
 GenerateAuthenticationTokenRequest.builder()
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .username(masterUsername)
                    .port(3306)
                    .hostname(dbInstanceIdentifier)
                    .build();

            return utilities.generateAuthenticationToken(tokenRequest);

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie GenerateRDSAuthToken in der AWS SDK for Java 2.x API-Referenz.

`ModifyDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.ModifyDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ModifyDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbSnapshotIdentifier>\s
                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    masterUserPassword - The updated password that corresponds to
 the master user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String masterUserPassword = args[1];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        updateIntance(rdsClient, dbInstanceIdentifier, masterUserPassword);
        rdsClient.close();
    }

    public static void updateIntance(RdsClient rdsClient, String
 dbInstanceIdentifier, String masterUserPassword) {
        try {
            // For a demo - modify the DB instance by modifying the master password.
            ModifyDbInstanceRequest modifyDbInstanceRequest =
 ModifyDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .publiclyAccessible(true)
                    .masterUserPassword(masterUserPassword)
                    .build();
```

```
            ModifyDbInstanceResponse instanceResponse =
 rdsClient.modifyDBInstance(modifyDbInstanceRequest);
            System.out.print("The ARN of the modified database is: " +
 instanceResponse.dbInstance().dbInstanceArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter ModifyDBInstance in der AWS SDK for Java 2.x - API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
```

```
            ModifyDbParameterGroupRequest groupRequest =
 ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
 rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
 response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK for Java 2.x API-Referenz.

`RebootDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`RebootDBInstance`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.RebootDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RebootDBInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier>\s

                Where:
                    dbInstanceIdentifier - The database instance identifier\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        rebootInstance(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }

    public static void rebootInstance(RdsClient rdsClient, String
 dbInstanceIdentifier) {
        try {
            RebootDbInstanceRequest rebootDbInstanceRequest =
 RebootDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            RebootDbInstanceResponse instanceResponse =
 rdsClient.rebootDBInstance(rebootDbInstanceRequest);
            System.out.print("The database " +
 instanceResponse.dbInstance().dbInstanceArn() + " was rebooted");

        } catch (RdsException e) {
```

```
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter RebootDBInstance in der AWS SDK for Java 2.x - API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Java.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;
import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.rdsdata.model.Field;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RdsLambdaHandler implements RequestHandler<APIGatewayProxyRequestEvent,
 APIGatewayProxyResponseEvent> {

    @Override
    public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent
 event, Context context) {
        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent();
```

```
        try {
            // Obtain auth token
            String token = createAuthToken();

            // Define connection configuration
            String connectionString = String.format("jdbc:mysql://%s:%s/%s?
useSSL=true&requireSSL=true",
                    System.getenv("ProxyHostName"),
                    System.getenv("Port"),
                    System.getenv("DBName"));

            // Establish a connection to the database
            try (Connection connection =
 DriverManager.getConnection(connectionString, System.getenv("DBUserName"), token);
                 PreparedStatement statement = connection.prepareStatement("SELECT ?
 + ? AS sum")) {

                statement.setInt(1, 3);
                statement.setInt(2, 2);

                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        int sum = resultSet.getInt("sum");
                        response.setStatusCode(200);
                        response.setBody("The selected sum is: " + sum);
                    }
                }
            }

        } catch (Exception e) {
            response.setStatusCode(500);
            response.setBody("Error: " + e.getMessage());
        }

        return response;
    }

    private String createAuthToken() {
        // Create RDS Data Service client
        RdsDataClient rdsDataClient = RdsDataClient.builder()
                .region(Region.of(System.getenv("AWS_REGION")))
                .credentialsProvider(DefaultCredentialsProvider.create())
                .build();
```

```
        // Define authentication request
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
                .resourceArn(System.getenv("ProxyHostName"))
                .secretArn(System.getenv("DBUserName"))
                .database(System.getenv("DBName"))
                .sql("SELECT 'RDS IAM Authentication'")
                .build();

        // Execute request and obtain authentication token
        ExecuteStatementResponse response = rdsDataClient.executeStatement(request);
        Field tokenField = response.records().get(0).get(0);

        return tokenField.stringValue();
    }
}
```

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK for Java 2.x Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Redshift unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Redshift

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Redshift beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.paginators.DescribeClustersIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloRedshift {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RedshiftClient redshiftClient = RedshiftClient.builder()
            .region(region)
            .build();

        listClustersPaginator(redshiftClient);
    }

    public static void listClustersPaginator(RedshiftClient redshiftClient) {
```

```
        DescribeClustersIterable clustersIterable =
 redshiftClient.describeClustersPaginator();
        clustersIterable.stream()
            .flatMap(r -> r.clusters().stream())
            .forEach(cluster -> System.out
                .println(" Cluster identifier: " + cluster.clusterIdentifier() + "
 status = " + cluster.clusterStatus()));
    }
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Redshift-Clusters.
- Listen Sie die Datenbanken im Cluster auf.
- Erstellen einer Tabelle mit dem Namen „Movies“.
- Befüllen der Tabelle „Movies“.
- Abfragen der Tabelle „Movies“ nach Jahr.
- Ändern des Redshift-Clusters.
- Löschen Sie den Amazon-Redshift-Cluster.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-Redshift-Features gezeigt werden.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
```

```
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
 file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret Manager
 that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift by
 using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
 service hosted in the cloud.
```

```
            The program's primary functionalities include cluster creation,
 verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
 execution of SQL statements.
            Furthermore, it demonstrates the process of querying data from the Movie
 table.\s

            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information, such
 as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
```

```
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
        logger.info("Enter a cluster id value or accept the default by hitting Enter
(default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error code
{}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default and
 used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
 redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation to
complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between 2012
and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(), movieYear,
clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
```

```
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
```

```
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon Redshift SDK.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;
```

```
    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
```

```
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
```

```
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
```

```
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
   }

   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records added
to the Movies table
```

```
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = iter.next();
                   int year = currentNode.get("year").asInt();
                   String title = currentNode.get("title").asText();

                   // Use SqlParameter to avoid SQL injection.
                   List<SqlParameter> parameterList = new ArrayList<>();
                   String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                   SqlParameter idParam = SqlParameter.builder()
                       .name("id")
```

```
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();

                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
```

```
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
```

```
    */
   public CompletableFuture<Void> getResultsAsync(String statementId) {
       GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
           .id(statementId)
           .build();

       return getAsyncDataClient().getStatementResult(resultRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.info("Error getting statement result {} ",
exception.getMessage());
                   throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
               }

               // Extract and print the field values using streams if the response
is valid.
               response.records().stream()
                   .flatMap(List::stream)
                   .map(Field::stringValue)
                   .filter(value -> value != null)
                   .forEach(value -> System.out.println("The Movie title field is "
+ value));

               return response;
           }).thenAccept(response -> {
               // Optionally add more logic here if needed after handling the
response
           });
   }


   /**
    * Asynchronously queries movies by a given year from a Redshift database.
    *
    * @param database    the name of the database to query
    * @param dbUser      the user to connect to the database with
    * @param year        the year to filter the movies by
    * @param clusterId   the identifier of the Redshift cluster to connect to
    * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
    */
   public CompletableFuture<String> queryMoviesByYearAsync(String database,
```

```
                                                                    String dbUser,
                                                                    int year,
                                                                    String clusterId)
{

       String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
       SqlParameter yearParam = SqlParameter.builder()
           .name("year")
           .value(String.valueOf(year))
           .build();

       ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .database(database)
           .dbUser(dbUser)
           .parameters(yearParam)
           .sql(sqlStatement)
           .build();

       return CompletableFuture.supplyAsync(() -> {
           try {
               ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
               return response.id();
           } catch (RedshiftDataException e) {
               throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
           }
       }).exceptionally(exception -> {
           logger.info("Error: {}", exception.getMessage());
           return "";
       });
   }

   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
```

```
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
```

```
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
    /**
```

```
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
           .nodeType("ra3.4xlarge")
           .publiclyAccessible(true)
           .numberOfNodes(2)
           .build();

       return getAsyncClient().createCluster(clusterRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Created cluster ");
               } else {
                   throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
               }
           });
   }
```

- API-Details finden Sie CreateCluster in der AWS SDK for Java 2.x API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {}", exception.getMessage());
                   }
               } else {
                   // Handle successful response
                   logger.info("The status is {}",
response.cluster().clusterStatus());
               }
           });
   }
```

- API-Details finden Sie DeleteCluster in der AWS SDK for Java 2.x API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
```

```
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- API-Details finden Sie DescribeClusters in der AWS SDK for Java 2.x API-Referenz.

`DescribeStatement`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStatement`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
    *
    * @param sqlId the ID of the SQL statement to check
    * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
    */
   public CompletableFuture<Void> checkStatementAsync(String sqlId) {
       DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
           .id(sqlId)
           .build();

       return getAsyncDataClient().describeStatement(statementRequest)
           .thenCompose(response -> {
               String status = response.statusAsString();
               logger.info("... Status: {} ", status);

               if ("FAILED".equals(status)) {
                   throw new RuntimeException("The Query Failed. Ending program");
               } else if ("FINISHED".equals(status)) {
                   return CompletableFuture.completedFuture(null);
               } else {
                   // Sleep for 1 second and recheck status
                   return CompletableFuture.runAsync(() -> {
                       try {
                           TimeUnit.SECONDS.sleep(1);
                       } catch (InterruptedException e) {
                           throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                       }
                   }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
```

```
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- API-Details finden Sie DescribeStatement in der AWS SDK for Java 2.x API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führt eine SQL-Anweisung aus, um eine Datenbanktabelle zu erstellen.

```
    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
```

```
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

Führt eine SQL-Anweisung aus, um Daten in eine Datenbanktabelle einzufügen.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records added
to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
```

```
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
                        .value(title)
                        .build();
```

```
                    SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                    parameterList.add(idParam);
                    parameterList.add(titleParam);
                    parameterList.add(yearParam);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

Führt eine SQL-Anweisung aus, um eine Datenbanktabelle abzufragen.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
```

```
                                                                    String dbUser,
                                                                    int year,
                                                                    String clusterId)
{

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK for Java 2.x API-Referenz.

`GetStatementResult`

Das folgende Codebeispiel zeigt die Verwendung`GetStatementResult`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Überprüfen Sie das Ergebnis der Anweisung.

```
    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
```

```
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- API-Details finden Sie GetStatementResult in der AWS SDK for Java 2.x API-Referenz.

`ListDatabases`

Das folgende Codebeispiel zeigt die Verwendung`ListDatabases`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
```

```
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }
```

- API-Details finden Sie ListDatabases in der AWS SDK for Java 2.x API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();
```

```
        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }
```

- API-Details finden Sie ModifyCluster in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Redshift-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Beispiele für Amazon Rekognition unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
     }

    /**
     * Compares two faces from images stored in an Amazon S3 bucket using AWS
 Rekognition.
```

```
    *
    * <p>This method takes two image keys from an S3 bucket and compares the faces
within them.
    * It prints out the confidence level of matched faces and reports the number of
unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the S3
bucket.
    * @param targetKey   The object key (file path) for the target image in the S3
bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(targetKey)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                   .sourceImage(sourceImage)
                   .targetImage(targetImage)
                   .similarityThreshold(similarityThreshold)
                   .build();

           // Compare the two images.
```

```
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces = compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- API-Details finden Sie CompareFaces in der AWS SDK for Java 2.x API-Referenz.

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateCollection`.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
```

```
     * @param rekClient    the Amazon Rekognition client used to interact with the
 Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateCollection in der AWS SDK for Java 2.x API-Referenz.

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCollection`.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
```

```
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteCollection in der AWS SDK for Java 2.x API-Referenz.

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFaces`.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
```

```
     * @param collectionId  the ID of the collection from which the face should be
 deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteFaces in der AWS SDK for Java 2.x API-Referenz.

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCollection`.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
```

```
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeCollection in der AWS SDK for Java 2.x API-Referenz.

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung`DetectFaces`.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """

            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source image
 is stored.
                sourceImage - The name of the source image file in the Amazon S3
 bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }
```

```
   /**
    * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
    *
    * @param rekClient    The Amazon Rekognition client used to interact with the
Rekognition service.
    * @param bucketName   The name of the Amazon S3 bucket where the source image
is stored.
    * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
    */
   public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
               .bucket(bucketName)
               .name(sourceImage)
               .build();

           Image targetImage = Image.builder()
               .s3Object(s3ObjectTarget)
               .build();

           DetectFacesRequest facesRequest = DetectFacesRequest.builder()
               .attributes(Attribute.ALL)
               .image(targetImage)
               .build();

           DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
           List<FaceDetail> faceDetails = facesResponse.faceDetails();
           for (FaceDetail face : faceDetails) {
               AgeRange ageRange = face.ageRange();
               System.out.println("The detected face is estimated to be between "
                       + ageRange.low().toString() + " and " +
ageRange.high().toString()
                       + " years old.");

               System.out.println("There is a smile : " +
face.smile().value().toString());
           }

       } catch (RekognitionException e) {
           System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectFaces in der AWS SDK for Java 2.x API-Referenz.

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
```

```
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
stored
                sourceImage - The name of the image file (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the detection
request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
```

```
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectLabels in der AWS SDK for Java 2.x API-Referenz.

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectModerationLabels`.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
```

```
    * Detects moderation labels in an image stored in an Amazon S3 bucket.
    *
    * @param rekClient      the Amazon Rekognition client to use for the detection
    * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
    * @param sourceImage    the name of the image file to be analyzed
    *
    * @throws RekognitionException if there is an error during the image detection
process
    */
   public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                   .image(targetImage)
                   .minConfidence(60F)
                   .build();

           DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                   .detectModerationLabels(moderationLabelsRequest);
           List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
           System.out.println("Detected labels for image");
           for (ModerationLabel label : labels) {
               System.out.println("Label: " + label.name()
                       + "\n Confidence: " + label.confidence().toString() + "%"
                       + "\n Parent:" + label.parentName());
           }

       } catch (RekognitionException e) {
           e.printStackTrace();
           System.exit(1);
       }
   }
```

```
}
```

- API-Details finden Sie DetectModerationLabels in der AWS SDK for Java 2.x API-Referenz.

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung`DetectText`.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
```

```
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored\n"
+
            "   sourceImage - The path to the image that contains text (for example,
pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();
```

```
            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " + text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectText in der AWS SDK for Java 2.x API-Referenz.

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung`IndexFaces`.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
```

```
    * @param bucketName    the name of the Amazon S3 bucket containing the image
    * @param sourceImage   the name of the image file to add to the collection
    * @throws RekognitionException if there is an error while interacting with the
Amazon Rekognition service
    */
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                   .collectionId(collectionId)
                   .image(targetImage)
                   .maxFaces(1)
                   .qualityFilter(QualityFilter.AUTO)
                   .detectionAttributes(Attribute.DEFAULT)
                   .build();

           IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
           System.out.println("Results for the image");
           System.out.println("\n Faces indexed:");
           List<FaceRecord> faceRecords = facesResponse.faceRecords();
           for (FaceRecord faceRecord : faceRecords) {
               System.out.println("  Face ID: " + faceRecord.face().faceId());
               System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
           }

           List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
           System.out.println("Faces not indexed:");
           for (UnindexedFace unindexedFace : unindexedFaces) {
               System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
               System.out.println("  Reasons:");
               for (Reason reason : unindexedFace.reasons()) {
                   System.out.println("Reason:  " + reason);
               }
```

```
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie IndexFaces in der AWS SDK for Java 2.x API-Referenz.

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung`ListCollections`.

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListCollections in der AWS SDK for Java 2.x API-Referenz.

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung`ListFaces`.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListFaces in der AWS SDK for Java 2.x API-Referenz.

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung`RecognizeCelebrities`.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }
```

```
        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source image
is stored
     * @param sourceImage  the name of the source image file stored in the Amazon S3
bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
```

```
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie RecognizeCelebrities in der AWS SDK for Java 2.x API-Referenz.

`SearchFaces`

Das folgende Codebeispiel zeigt die Verwendung`SearchFaces`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
```

```
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SearchFaces in der AWS SDK for Java 2.x API-Referenz.

`SearchFacesByImage`

Das folgende Codebeispiel zeigt die Verwendung`SearchFacesByImage`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SearchFacesByImage in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von PSA in Bildern

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Informationen in Videos erkennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie Amazon-Rekognition-Aufträge, um Elemente wie Personen, Objekte und Text in Videos zu erkennen.
- Überprüfen Sie den Auftragsstatus, bis die Aufträge abgeschlossen sind.
- Gibt die Liste der von jedem Auftrag erkannten Elemente aus.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen von Informationen aus einem Video, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
```

```
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;
```

```
                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
 recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Labels in einem Video mithilfe einer Labelerkennung.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
```

```
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 5) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();
```

```
            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }
```

```
   public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
       List<Message> messages;
       ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
               .queueUrl(queueUrl)
               .build();

       try {
           messages = sqs.receiveMessage(messageRequest).messages();

           if (!messages.isEmpty()) {
               for (Message message : messages) {
                   String notification = message.body();

                   // Get the status and job id from the notification
                   ObjectMapper mapper = new ObjectMapper();
                   JsonNode jsonMessageTree = mapper.readTree(notification);
                   JsonNode messageBodyText = jsonMessageTree.get("Message");
                   ObjectMapper operationResultMapper = new ObjectMapper();
                   JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                   JsonNode operationJobId = jsonResultTree.get("JobId");
                   JsonNode operationStatus = jsonResultTree.get("Status");
                   System.out.println("Job found in JSON is " + operationJobId);

                   DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                           .queueUrl(queueUrl)
                           .build();

                   String jobId = operationJobId.textValue();
                   if (startJobId.compareTo(jobId) == 0) {
                       System.out.println("Job id: " + operationJobId);
                       System.out.println("Status : " +
operationStatus.toString());

                       if (operationStatus.asText().equals("SUCCEEDED"))
                           getResultsLabels(rekClient);
                       else
                           System.out.println("Video analysis failed");

                       sqs.deleteMessage(deleteMessageRequest);
                   } else {
```

```
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
```

```
                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
```

```
    }
}
```

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetect {
    private static String startJobId = "";
```

```
   public static void main(String[] args) {
       final String usage = """

               Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

               Where:
                  bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                  video - The name of the video (for example, people.mp4).\s
                  queueUrl- The URL of a SQS queue.\s
                  topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                  roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucket = args[0];
       String video = args[1];
       String queueUrl = args[2];
       String topicArn = args[3];
       String roleArn = args[4];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       SqsClient sqs = SqsClient.builder()
               .region(Region.US_EAST_1)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startLabels(rekClient, channel, bucket, video);
       getLabelJob(rekClient, sqs, queueUrl);
       System.out.println("This example is done!");
```

```
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();
```

```
                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);
```

```
                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
```

```
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
```

```
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
```

```
                .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
```

```
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
```

```
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie technische Signal-Segmente und Einstellungserkennungssegmente in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startSegmentDetection(rekClient, channel, bucket, video);
        getSegmentResults(rekClient);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startSegmentDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();
```

```
            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();
```

```
                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
```

```
                    System.out.println("\tDuration : " + seconds + " milliseconds");
                    System.out.println("\tStart time code: " +
 detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
 detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
 detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
 segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Text in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
```

```
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();
```

```
                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                    System.out.println("Type: " +
detectedText.textDetection().type());
```

```
                    System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
 textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen Sie Personen in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";
```

```
    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
```

```
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
```

```
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Codebeispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe der Route 53-Domainregistrierung Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Route-53-Domainregistrierung

Das folgende Codebeispiel zeigt, wie Sie mit der Route 53-Domainregistrierung beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.route53domains.Route53DomainsClient;
import software.amazon.awssdk.services.route53.model.Route53Exception;
import software.amazon.awssdk.services.route53domains.model.DomainPrice;
import software.amazon.awssdk.services.route53domains.model.ListPricesRequest;
import software.amazon.awssdk.services.route53domains.model.ListPricesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code examples performs the following operation:
 *
 * 1. Invokes ListPrices for at least one domain type, such as the “com” type
 * and displays the prices for Registration and Renewal.
 *
 */
public class HelloRoute53 {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <hostedZoneId> \n\n" +
                "Where:\n" +
                "    hostedZoneId - The id value of an existing hosted zone. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Invokes ListPrices for at least one domain type.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
 domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .maxItems(10)
                    .tld(domainType)
```

```
                    .build();

            ListPricesResponse response =
 route53DomainsClient.listPrices(pricesRequest);
            List<DomainPrice> prices = response.prices();
            for (DomainPrice pr : prices) {
                System.out.println("Name: " + pr.name());
                System.out.println(
                        "Registration: " + pr.registrationPrice().price() + " " +
 pr.registrationPrice().currency());
                System.out.println("Renewal: " + pr.renewalPrice().price() + " " +
 pr.renewalPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
                System.out.println("Transfer: " + pr.transferPrice().price() + " " +
 pr.transferPrice().currency());
                System.out.println("Change Ownership: " +
 pr.changeOwnershipPrice().price() + " "
                        + pr.changeOwnershipPrice().currency());
                System.out.println(
                        "Restoration: " + pr.restorationPrice().price() + " " +
 pr.restorationPrice().currency());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListPrices in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Auflisten der aktuellen Domains und der Vorgänge des letzten Jahres

- Anzeigen der Abrechnung für das vergangene Jahr und der Preise für Domaintypen
- Abrufen von Domainvorschlägen
- Überprüfen der Verfügbarkeit und Übertragbarkeit von Domains
- Optional: Anfordern einer Domainregistrierung
- Abrufen eines Vorgangsdetails
- Optional: Abrufen eines Domaindetails

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
```

```
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainType> <phoneNumber> <email> <domainSuggestion>
 <firstName> <lastName> <city>

                Where:
                    domainType - The domain type (for example, com).\s
                    phoneNumber - The phone number to use (for example,
 +91.9966564xxx)      email - The email address to use.      domainSuggestion - The
 domain suggestion (for example, findmy.accountants).\s
                    firstName - The first name to use to register a domain.\s
                    lastName -  The last name to use to register a domain.\s
                    city - the city to use to register a domain.\s
                    """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        String phoneNumber = args[1];
        String email = args[2];
        String domainSuggestion = args[3];
        String firstName = args[4];
        String lastName = args[5];
        String city = args[6];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
```

```
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
```

```
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
```

```
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
```

```
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " + content.registrationPrice().price()
+ " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           LocalDateTime localDateTime2 = localDateTime.minusYears(1);
           Instant myStartTime = localDateTime2.toInstant(zoneOffset);
           Instant myEndTime = localDateTime.toInstant(zoneOffset);

           ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                   .start(myStartTime)
                   .end(myEndTime)
                   .build();

           ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
           listRes.stream()
                   .flatMap(r -> r.billingRecords().stream())
                   .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Aktionen

`CheckDomainAvailability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainAvailability`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                   .domainName(domainSuggestion)
```

```
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie CheckDomainAvailability in der AWS SDK for Java 2.x API-Referenz.

`CheckDomainTransferability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainTransferability`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie CheckDomainTransferability in der AWS SDK for Java 2.x API-Referenz.

`GetDomainDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainDetail`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
       try {
           GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                   .domainName(domainSuggestion)
                   .build();

           GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
           System.out.println("The contact first name is " +
response.registrantContact().firstName());
           System.out.println("The contact last name is " +
response.registrantContact().lastName());
           System.out.println("The contact org name is " +
response.registrantContact().organizationName());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- API-Details finden Sie GetDomainDetail in der AWS SDK for Java 2.x API-Referenz.

`GetDomainSuggestions`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainSuggestions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
       try {
           GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                   .domainName(domainSuggestion)
                   .suggestionCount(5)
                   .onlyAvailable(true)
                   .build();

           GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
           List<DomainSuggestion> suggestions = response.suggestionsList();
           for (DomainSuggestion suggestion : suggestions) {
               System.out.println("Suggestion Name: " + suggestion.domainName());
               System.out.println("Availability: " + suggestion.availability());
               System.out.println(" ");
           }

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie GetDomainSuggestions in der AWS SDK for Java 2.x API-Referenz.

`GetOperationDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetOperationDetail`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
       try {
           GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                   .operationId(operationId)
                   .build();

           GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
           System.out.println("Operation detail message is " + response.message());

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie GetOperationDetail in der AWS SDK for Java 2.x API-Referenz.

`ListDomains`

Das folgende Codebeispiel zeigt die Verwendung`ListDomains`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListDomains in der AWS SDK for Java 2.x API-Referenz.

`ListOperations`

Das folgende Codebeispiel zeigt die Verwendung`ListOperations`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void listOperations(Route53DomainsClient route53DomainsClient) {
```

```
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListOperations in der AWS SDK for Java 2.x API-Referenz.

`ListPrices`

Das folgende Codebeispiel zeigt die Verwendung`ListPrices`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
       try {
           ListPricesRequest pricesRequest = ListPricesRequest.builder()
                   .tld(domainType)
                   .build();

           ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
           listRes.stream()
                   .flatMap(r -> r.prices().stream())
                   .forEach(content -> System.out.println(" Name: " +
content.name() +
                           " Registration: " + content.registrationPrice().price()
+ " "
                           + content.registrationPrice().currency() +
                           " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie ListPrices in der AWS SDK for Java 2.x API-Referenz.

`RegisterDomain`

Das folgende Codebeispiel zeigt die Verwendung`RegisterDomain`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
           String domainSuggestion,
           String phoneNumber,
           String email,
           String firstName,
           String lastName,
           String city) {

       try {
           ContactDetail contactDetail = ContactDetail.builder()
                   .contactType(ContactType.COMPANY)
                   .state("LA")
                   .countryCode(CountryCode.IN)
                   .email(email)
                   .firstName(firstName)
                   .lastName(lastName)
                   .city(city)
                   .phoneNumber(phoneNumber)
                   .organizationName("My Org")
                   .addressLine1("My Address")
                   .zipCode("123 123")
                   .build();

           RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                   .adminContact(contactDetail)
                   .registrantContact(contactDetail)
                   .techContact(contactDetail)
                   .domainName(domainSuggestion)
                   .autoRenew(true)
                   .durationInYears(1)
                   .build();

           RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
           System.out.println("Registration requested. Operation Id: " +
response.operationId());
           return response.operationId();

       } catch (Route53Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
        return "";
    }
```

- API-Details finden Sie RegisterDomain in der AWS SDK for Java 2.x API-Referenz.

`ViewBilling`

Das folgende Codebeispiel zeigt die Verwendung`ViewBilling`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
       try {
           Date currentDate = new Date();
           LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
           ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
           LocalDateTime localDateTime2 = localDateTime.minusYears(1);
           Instant myStartTime = localDateTime2.toInstant(zoneOffset);
           Instant myEndTime = localDateTime.toInstant(zoneOffset);

           ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                   .start(myStartTime)
                   .end(myEndTime)
                   .build();

           ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
           listRes.stream()
                   .flatMap(r -> r.billingRecords().stream())
                   .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                           " Operation: " + content.operationAsString() +
```

```
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ViewBilling in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon S3 unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
```

```
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListBuckets in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ein Beispielszenario.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>
```

```
            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3 bucket.
                objectPath - The full file path of the object within the S3 bucket
(e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be downloaded
and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to store
and retrieve
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set
of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets and
objects as
            part of their application deployment pipelines. With S3, teams can focus
on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.
```

```
            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future = s3Actions.multipartUpload(bucketName,
multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}' and
key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future = s3Actions.performMultiCopy(toBucket,
bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
```

```
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
 s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse, die die Operationen enthält.

```
public class S3Actions {

    private static final Logger logger = LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

```
    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
     * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }


    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the key (or name) of the S3 object to retrieve
     * @param path       the local file path where the object's bytes will be
written
     * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
     */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
```

```
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }


    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
```

```
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }


    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

    /**
     * Performs a multipart upload to an Amazon S3 bucket.
     *
```

```
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (name) of the file to be uploaded
     * @return a {@link CompletableFuture} that completes when the multipart upload
is successful
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key) {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
```

```
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload completed
successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

```
   /**
    * Deletes an object from an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has been
deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during delete",
ex);
           }
       });

       return response.thenApply(r -> null);
   }


   /**
    * Deletes an S3 bucket asynchronously.
    *
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();
```

```
        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
```

```
                    logger.error("Error during upload part copy: " +
 exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`AbortMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`AbortMultipartUpload`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
```

```
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
```

```
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
            }
        }
    }

    /**
     * Aborts incomplete multipart uploads older than the specified point in time.
     *
     * @param bucketName  the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName, Instant
pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
```

```
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key) {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }

    /**
```

```
    * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
    *
    * @param bucketName the name of the S3 bucket
    */
   static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
       Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
           .abortIncompleteMultipartUpload(b -> b.
               daysAfterInitiation(7))
           .status("Enabled")
           .filter(SdkBuilder::build) // Filter element is required.
           .build());

       PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
           .bucket(bucketName)
           .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

       if (response.sdkHttpResponse().isSuccessful()) {
           logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
       } else {
           logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
       }
   }

   /************************
    Multipart upload methods
    ***********************/

   static void initiateAndInterruptMultiPartUpload(String bucketName, String key,
String threadName) {
       Runnable upload = () -> {
           try {
               doMultipartUpload(bucketName, key);
           } catch (SdkException e) {
               logger.error(e.getMessage());
           }
       };
       Thread uploadThread = new Thread(upload, threadName);
       uploadThread.start();
       try {
           Thread.sleep(Duration.ofSeconds(1).toMillis());
```

```
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key) {
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
        List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
        step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
    }

    static String step1CreateMultipartUpload(String bucketName, String key) {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
```

```
                long read = file.getChannel().read(bb);

                bb.flip();
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String bucketName, String key, String
uploadId, List<CompletedPart> completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId(String bucketName, String key) {
        String uploadId = step1CreateMultipartUpload(bucketName, key);
```

```
        doMultipartUploadWithUploadId(bucketName, key, uploadId);
        return uploadId;
    }

    static void doMultipartUploadWithUploadId(String bucketName, String key, String
uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /************************
     Resource handling methods
     ************************/

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
```

```
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL = PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
 Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}
```

- API-Details finden Sie AbortMultipartUpload in der AWS SDK for Java 2.x API-Referenz.

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren eines Objekts mit einem S3Client.

```
    /**
     * Asynchronously copies an object from one S3 bucket to another.
```

```
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

Verwenden Sie an S3TransferManager, um ein Objekt von einem Bucket in einen anderen zu kopieren. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
```

```
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- API-Details finden Sie CopyObject in der AWS SDK for Java 2.x API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket.

```
    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
```

```
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- API-Details finden Sie CreateBucket in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
```

```
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
 getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
 deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }
```

- API-Details finden Sie DeleteBucket in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from (for
 example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
     * @throws S3Exception if there is an error deleting the bucket policy
     */
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();
```

```
        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketWebsite`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
 configuration from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
```

```
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteBucketWebsite in der AWS SDK for Java 2.x API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has been
 deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
 String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        CompletableFuture<DeleteObjectResponse> response =
 getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during delete",
 ex);
            }
        });

        return response.thenApply(r -> null);
    }
```

- API-Details finden Sie DeleteObject in der AWS SDK for Java 2.x API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName>

            Where:
               bucketName - the Amazon S3 bucket name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;
```

```
        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteObjects in der AWS SDK for Java 2.x API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <objectKey>

            Where:
              bucketName - The Amazon S3 bucket to get the access control list (ACL)
 for.
              objectKey - The object to get the ACL for.\s
            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an empty
string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
                grantee = grant.grantee().id();
```

```
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- API-Details finden Sie GetBucketAcl in der AWS SDK for Java 2.x API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the policy
     * @return the policy text for the specified bucket, or an empty string if an
 error occurs
     */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`GetBucketReplication`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketReplication`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client           the S3 client used to interact with the S3 service
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
```

```
            .bucket(sourceBucketName)
            .build();

        try {
            ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
            ReplicationRule rule = replicationConfig.rules().get(0);
            System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
            System.out.println("Retrieved priority: " + rule.priority());
            System.out.println("Retrieved source-bucket replication rule status: " +
rule.status());

        } catch (S3Exception e) {
            System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
        }
    }
```

- API-Details finden Sie GetBucketReplication in der AWS SDK for Java 2.x API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lesen Sie Daten als Byte-Array mit einem S3Client.

```
    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
```

```
     * @param keyName    the key (or name) of the S3 object to retrieve
     * @param path       the local file path where the object's bytes will be
 written
     * @return a {@link CompletableFuture} that completes when the object bytes have
 been written to the local file
     */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
 keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
 getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
 ex);
            }
        });
    }
```

Verwenden Sie an S3TransferManager, um ein Objekt in einem S3-Bucket in eine lokale Datei herunterzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
```

```
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Lesen Sie Tags, die zu einem Objekt gehören, mit einem S3Client.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
```

```
    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie eine URL für ein Objekt mit einem S3Client ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
```

```
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be retrieved
     * @throws S3Exception if there is an error retrieving the URL for the specified
object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
```

```
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie ein Objekt mithilfe des S3Presigner-Client-Objekts mit einem S3Client ab.

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
                <bucketName> <keyName>\s

            Where:
```

```
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating the
pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
service.
     * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();
```

```
            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream = connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

Rufen Sie ein Objekt ab, indem Sie ein ResponseTransformer Objekt und S3Client verwenden.

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves them
to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
local file.
     * @throws S3Exception If an error occurs while retrieving the object from the
S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();
```

```
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetObject in der AWS SDK for Java 2.x API-Referenz.

`GetObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLegalHold`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
```

```
            System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
        }

        return null;
    }
```

- API-Details finden Sie GetObjectLegalHold in der AWS SDK for Java 2.x API-Referenz.

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLockConfiguration`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
    }
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS SDK for Java 2.x API-Referenz.

`GetObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectRetention`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get the retention period for an S3 object.
   public ObjectLockRetention getObjectRetention(String bucketName, String key){
       try {
           GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

           GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
           System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
           return response.retention();

       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           return null;
       }
   }
```

- API-Details finden Sie GetObjectRetention in der AWS SDK for Java 2.x API-Referenz.

`HeadObject`

Das folgende Codebeispiel zeigt die Verwendung`HeadObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Bestimmen Sie den Inhaltstyp eines Objekts.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Rufen Sie den Wiederherstellungsstatus eines Objekts ab.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to interact
 with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param keyName    the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon S3
 service
     */
    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
```

```
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie HeadObject in der AWS SDK for Java 2.x API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- API-Details finden Sie ListBuckets in der AWS SDK for Java 2.x API-Referenz.

`ListMultipartUploads`

Das folgende Codebeispiel zeigt die Verwendung`ListMultipartUploads`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
```

```
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon S3
 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
```

```
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
 "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListMultipartUploads in der AWS SDK for Java 2.x API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
```

```
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

Listet Objekte mithilfe der Paginierung auf.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>\s
```

```
            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + "
 size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK for Java 2.x API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
```

```
              <bucketName> <id>\s

            Where:
              bucketName - The Amazon S3 bucket to grant permissions on.\s
              id - The ID of the owner of this bucket (you can get this value from
the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full control
of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();
```

```
            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutBucketAcl in der AWS SDK for Java 2.x API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
```

```
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
```

```
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an Amazon
S3 bucket.
     *
     * @param s3            the {@link S3Client} instance used to interact with the
Amazon S3 service
     * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
     * @param accountId     the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS configuration
for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
     *
     * @param s3 the S3Client instance to use for the operation
     * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
     * @param accountId the expected bucket owner's account ID
     *
     * @throws S3Exception if there is an error retrieving the CORS configuration
```

```
     */
    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                .allowedMethods(allowMethods)
```

```
                .allowedOrigins(allowOrigins)
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutBucketCors in der AWS SDK for Java 2.x API-Referenz.

`PutBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketLifecycleConfiguration`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <accountId>\s

            Where:
              bucketName - The Amazon Simple Storage Service (Amazon S3) bucket to
 upload an object into.
              accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix to
the
            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
```

```
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the LifecycleRule objects to an ArrayList.
            ArrayList<LifecycleRule> ruleList = new ArrayList<>();
            ruleList.add(rule1);
            ruleList.add(rule2);

            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a new
lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
```

```
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
```

```
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
 DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutBucketLifecycleConfiguration in der AWS SDK for Java 2.x API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
```

```
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
 Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }
```

```
    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
```

```
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- API-Details finden Sie PutBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`PutBucketReplication`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketReplication`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Sets the replication configuration for an Amazon S3 bucket.
     *
     * @param s3Client             the S3Client instance to use for the operation
     * @param sourceBucketName     the name of the source bucket
```

```
    * @param destBucketName       the name of the destination bucket
    * @param destinationBucketARN the Amazon Resource Name (ARN) of the destination
bucket
    * @param roleARN              the ARN of the IAM role to use for the
replication configuration
    */
   public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
       try {
           Destination destination = Destination.builder()
               .bucket(destinationBucketARN)
               .storageClass(StorageClass.STANDARD)
               .build();

           // Define a prefix filter for replication.
           ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
               .prefix("documents/")
               .build();

           // Define delete marker replication setting.
           DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
               .status(DeleteMarkerReplicationStatus.DISABLED)
               .build();

           // Create the replication rule.
           ReplicationRule replicationRule = ReplicationRule.builder()
               .priority(1)
               .filter(ruleFilter)
               .status(ReplicationRuleStatus.ENABLED)
               .deleteMarkerReplication(deleteMarkerReplication)
               .destination(destination)
               .build();

           List<ReplicationRule> replicationRuleList = new ArrayList<>();
           replicationRuleList.add(replicationRule);

           // Define the replication configuration with IAM role.
           ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
               .role(roleARN)
               .rules(replicationRuleList)
               .build();
```

```
            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- API-Details finden Sie PutBucketReplication in der AWS SDK for Java 2.x API-Referenz.

`PutBucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketVersioning`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
```

```
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String bucketName){
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- API-Details finden Sie PutBucketVersioning in der AWS SDK for Java 2.x API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName> [indexdoc]\s

            Where:
               bucketName   - The Amazon S3 bucket to set the website configuration
 on.\s
               indexdoc - The index document, ex. 'index.html'
                          If not specified, 'index.html' will be set.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
```

```
     * @param indexDoc The name of the index document to use for the website
configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutBucketWebsite in der AWS SDK for Java 2.x API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine Datei mit einem S3Client in einen Bucket hoch.

```
   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to upload file", ex);
           }
       });
   }
```

Verwenden Sie an S3TransferManager, um eine Datei in einen Bucket hochzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
```

```
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie Tags fest.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);
```

```
            Tagging allTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact with
the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }
```

```
            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                .key("Tag 3")
                .value("This is tag 3")
                .build();

            Tag tag4 = Tag.builder()
                .key("Tag 4")
                .value("This is tag 4")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
```

```
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an error
 occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie Metadaten fest.

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
              <bucketName> <objectKey> <objectPath>\s

            Where:
              bucketName - The Amazon S3 bucket to upload an object into.
              objectKey - The object to upload (for example, book.pdf).
              objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
```

```
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Laden Sie ein Objekt mit einem S3Client in einen Bucket hoch und legen Sie einen Wert für die Objektaufbewahrung fest.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
```

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object (for
 example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
```

```
     *
     * @param s3     the S3Client object used to interact with the Amazon S3 service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support
object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutObject in der AWS SDK for Java 2.x API-Referenz.

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLegalHold`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
       System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
   }
```

- API-Details finden Sie PutObjectLegalHold in der AWS SDK for Java 2.x API-Referenz.

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLockConfiguration`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
```

```
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();
```

```
        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS SDK for Java 2.x API-Referenz.

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectRetention`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
```

```
            .build();

        PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
 until "+ humanReadableDate +".");
    }
```

- API-Details finden Sie PutObjectRetention in der AWS SDK for Java 2.x API-Referenz.

`RestoreObject`

Das folgende Codebeispiel zeigt die Verwendung`RestoreObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
```

```
 *
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
 Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
```

```
     * @param s3                   an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName           the name of the S3 bucket where the object is
 stored
     * @param keyName              the key (object name) of the S3 object to be
 restored
     * @param expectedBucketOwner  the AWS account ID of the expected bucket owner
     */
    public static void restoreS3Object(S3Client s3, String bucketName, String
 keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

 .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie RestoreObject in der AWS SDK for Java 2.x API-Referenz.

`SelectObjectContent`

Das folgende Codebeispiel zeigt die Verwendung`SelectObjectContent`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das folgende Beispiel zeigt eine Abfrage mit einem JSON-Objekt. Das vollständige Beispiel zeigt auch die Verwendung eines CSV-Objekts.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

```
public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
 < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
```

```
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
```

```
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
```

```
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- API-Details finden Sie SelectObjectContent in der AWS SDK for Java 2.x API-Referenz.

`UploadPartCopy`

Das folgende Codebeispiel zeigt die Verwendung`UploadPartCopy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
           .bucket(toBucket)
           .key(key)
           .build();
```

```
        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- API-Details finden Sie UploadPartCopy in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Prüfen, ob ein Bucket vorhanden ist

Das folgende Codebeispiel zeigt, wie Sie überprüfen, ob ein Bucket vorhanden ist.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sie können die folgende `doesBucketExists` Methode als Ersatz für die SDK for Java V1 Amazons3Client #does BucketExist V2 (String) -Methode verwenden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the bucket
 name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName, s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't be
 used to create a new bucket.
     * <p>
     * Internally this method uses the <a
```

```
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks for
 the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the value
 false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket exists
 but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}
```

- API-Details finden Sie GetBucketAcl in AWS SDK for Java 2.x der API-Referenz.

Prüfen Sie, ob ein Objekt existiert

Das folgende Codebeispiel zeigt, wie überprüft wird, ob ein Objekt existiert.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sie können die folgende `doesObjectExist` Methode als Ersatz für das SDK für die Java-V1-Methode AmazonS3Client #does ObjectExist (String, String) verwenden.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
 LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
 name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
 s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
```

```
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with 404
 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may return
 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName   The name of the bucket containing the object.
     * @param key          The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
 exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
 s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
 string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .build());
            return true;
        } catch (NoSuchKeyException e) {
            return false;
        }
    }
}
```

- API-Details finden Sie HeadObject in AWS SDK for Java 2.x der API-Referenz.

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im Folgenden werden drei Beispiele gezeigt, wie vorsignierte URLs erstellt und die URLs mit HTTP-Clientbibliotheken verwendet werden:

- Eine HTTP-GET-Anfrage, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-GET-Anfrage mit Metadaten in den Headern, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-PUT-Anfrage mit Abfrageparametern, die die URL mit einer HTTP-Clientbibliothek verwendet

Generieren Sie eine vorsignierte URL für ein Objekt und laden Sie sie dann herunter (GET-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
```

```
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Generieren Sie die URL.

```
    /* Create a pre-signed URL to download an object in a subsequent GET request. */
    public String createPresignedGetUrl(String bucketName, String keyName) {
        try (S3Presigner presigner = S3Presigner.create()) {

            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            GetObjectPresignRequest presignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10))  // The URL will
 expire in 10 minutes.
                    .getObjectRequest(objectRequest)
                    .build();

            PresignedGetObjectRequest presignedRequest =
 presigner.presignGetObject(presignRequest);
            logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
            logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
```

```
    }
```

Laden Sie das Objekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `HttpURLConnection` (seit v1.1), um den Download durchzuführen.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
    public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Download durchzuführen.

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
```

```
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Verwenden Sie die AWS `SdkHttpClient` SDK-Klasse für Java, um den Download durchzuführen.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
```

```
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

Generieren Sie eine vorsignierte URL mit Metadaten in den Headern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
```

```
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generieren Sie die URL.

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
 String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
 PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL expires
 in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
 presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
 presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
```

```
    }
```

Laden Sie ein Dateiobjekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `HttpURLConnection` (seit v1.1), um den Upload durchzuführen.

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-meta-" +
k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Upload durchzuführen.

```
   /* Use the JDK HttpClient (since v11) class to do the upload. */
   public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           final HttpResponse<Void> response = httpClient.send(requestBuilder
                           .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                           .build(),
                   HttpResponse.BodyHandlers.discarding());

           logger.info("HTTP response code is " + response.statusCode());

       } catch (URISyntaxException | InterruptedException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
```

```
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" + k,
 v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
 FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Generieren Sie eine vorsignierte URL mit Abfrageparametern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;
```

```
import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Generieren Sie die URL.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
adds query parameters
    *  to the request instead of using headers. By using query parameters, you do
not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder =
AwsRequestOverrideConfiguration.builder();

           queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .overrideConfiguration(overrideConfigurationBuilder.build()) //
Add the override configuration.
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
```

```
                    .signatureDuration(Duration.ofMinutes(10))  // The URL expires
in 10 minutes.
                    .putObjectRequest(objectRequest)
                    .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

```
   /**
    * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT request.
Since the
    * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
    *
    * @param presignedUrlString The URL for the PUT request.
    * @param fileToPut File to uplaod
    */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());

           SdkHttpRequest request = requestBuilder.build();

           HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                   .request(request)
```

```
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Löschen von unvollständigen mehrteiligen Uploads

Das folgende Codebeispiel zeigt, wie Sie unvollständige mehrteilige Amazon-S3-Uploads löschen oder beenden.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Um mehrteilige Uploads zu beenden, die gerade bearbeitet werden oder aus irgendeinem Grund unvollständig sind, können Sie eine Liste mit Uploads abrufen und sie dann löschen, wie im folgenden Beispiel gezeigt.

```
    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();
```

```
            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
            }
        }
    }
```

Um unvollständige mehrteilige Uploads zu löschen, die vor oder nach einem Datum initiiert wurden, können Sie mehrteilige Uploads anhand eines bestimmten Zeitpunkts selektiv löschen, wie im folgenden Beispiel gezeigt.

```
    /**
     * Aborts incomplete multipart uploads older than the specified point in time.
     *
     * @param bucketName  the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName, Instant
pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();
```

```
                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
```

Wenn Sie nach dem Start eines mehrteiligen Uploads Zugriff auf die Upload-ID haben, können Sie den laufenden Upload mithilfe der ID löschen.

```
    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key) {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }
```

Um unvollständige mehrteilige Uploads, die älter als eine bestimmte Anzahl von Tagen sind, dauerhaft zu löschen, richten Sie eine Bucket-Lebenszykluskonfiguration für den Bucket ein. Das folgende Beispiel zeigt, wie Sie eine Regel erstellen, um unvollständige Uploads zu löschen, die älter als 7 Tage sind.

```
    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
```

```
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

Erkennen von PSA in Bildern

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Herunterladen von S3-Verzeichnissen

Das folgende Codebeispiel zeigt, wie Sie den Inhalt der Verzeichnisse von Amazon-S3-Buckets herunterladen und filtern.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie das S3TransferManager in verwenden AWS SDK for Java 2.x , um „Verzeichnisse“ aus einem Amazon S3-Bucket herunterzuladen. Es zeigt auch, wie man es DownloadFilters in der Anfrage verwendet.

```
    /**
     * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
     * an object to an S3 bucket, you specify a key, which is essentially the "path"
to the object. The key can contain
```

```
    * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
    * logical representation, not an actual directory.
    * <p><pre>
    * In this example, our S3 bucket contains the following objects:
    *
    * folder1/file1.txt
    * folder1/file2.txt
    * folder1/file3.txt
    * folder2/file1.txt
    * folder2/file2.txt
    * folder2/file3.txt
    * folder3/file1.txt
    * folder3/file2.txt
    * folder3/file3.txt
    *
    * When method `downloadS3Directories` is invoked with
    * `destinationPathURI` set to `/test`, the downloaded
    * directory looks like:
    *
    * |- test
    *    |- folder1
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    *    |- folder3
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
    * @param bucketName         The S3 bucket that contains the 'directories' to
download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
   public Integer downloadS3Directories(S3TransferManager transferManager,
                                        URI destinationPathURI, String bucketName)
{

       // Define the filters for which 'directories' we want to download.
```

```
        DownloadFilter folder1Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
 folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2` was
 not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }
```

Herunterladen von Objekten in ein lokales Verzeichnis

Das folgende Codebeispiel zeigt, wie Sie alle Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis herunterladen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie an S3TransferManager, um alle S3-Objekte im selben S3-Bucket herunterzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
```

```
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- API-Details finden Sie DownloadDirectory in der AWS SDK for Java 2.x API-Referenz.

Sperren von Amazon-S3-Objekten

Das folgende Codebeispiel zeigt, wie Sie mit S3-Object-Lock-Features arbeiten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
```

```
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
 Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
```

```
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
```

```
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
```

```
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
```

```
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking features.
                """);

        System.out.println("S3 buckets can be created either with or without object
lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
```

```
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on individual
files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
```

```
                                System.out.println("**** You have selected to put a
 legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
 exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance retention
 period to %s in %s (y/n)?
                                Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
```

```
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey) {
        // Calculate the instant one day from now.
```

```
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time zone.
        ZonedDateTime zonedDateTime = futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
```

```
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" + ex.getMessage() +
"'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
```

```
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the updated
value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();
```

```
        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();
```

```
            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath) {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
```

```
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
```

```
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Verwalten großer Nachrichten mit S3

Das folgende Codebeispiel zeigt, wie Sie die erweiterte Clientbibliothek von Amazon SQS verwenden, um mit großen Amazon-SQS-Nachrichten zu arbeiten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
```

```
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
```

```
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
```

```
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x -API-Referenz.

- CreateBucket
- PutBucketLifecycleConfiguration
- ReceiveMessage
- SendMessage

URIs analysieren

Das folgende Codebeispiel zeigt, wie Sie Amazon S3-URIs analysieren, um wichtige Komponenten wie den Bucket-Namen und Objektschlüssel zu extrahieren.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Analysieren Sie eine Amazon S3-URI mithilfe der S3Uri-Klasse.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate S3Uri
     *                    capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
```

```
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri() method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
 returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123], partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
 s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.
```

```
        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters, must
 be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".", the
 dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

Process-S3-Ereignisbenachrichtigungen

Das folgende Codebeispiel zeigt, wie Sie mit S3-Ereignisbenachrichtigungen auf objektorientierte Weise arbeiten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie ein S3-Benachrichtigungsereignis mithilfe von Amazon SQS verarbeiten.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
```

```
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                       .join();

               // Receive the messages.
               ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                       .queueUrl(queueUrl)
               ).get();
               logger.info("Count of received messages: {}",
response.messages().size());
               didReceiveMessages = !response.messages().isEmpty();

               // Create a collection to hold the received message for deletion
```

```
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Senden Sie Event-Benachrichtigungen an EventBridge

Das folgende Codebeispiel zeigt, wie ein Bucket aktiviert wird, um S3-Ereignisbenachrichtigungen an ein Amazon SNS-Thema EventBridge und eine Amazon SQS-Warteschlange zu senden und Benachrichtigungen an diese weiterzuleiten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
```

```
                            .eventBridgeConfiguration(
                                    SdkBuilder::build)
                    ).build()).join();

            // Create an EventBridge rule to route Object Created notifications.
            PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                    .name(RULE_NAME)
                    .eventPattern("""
                            {
                              "source": ["aws.s3"],
                              "detail-type": ["Object Created"],
                              "detail": {
                                "bucket": {
                                  "name": ["%s"]
                                }
                              }
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
```

```
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

Verfolgen von Uploads und Downloads

Das folgende Codebeispiel zeigt, wie Sie einen Upload oder Download in ein bzw. von einem Amazon-S3-Objekt verfolgen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verfolgen Sie den Fortschritt eines Datei-Uploads.

```
    public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                                String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b.bucket(bucketName).key(key))
```

```
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .source(Paths.get(filePathURI))
                .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        fileUpload.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3 MB
file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

Verfolgen Sie den Fortschritt eines Datei-Downloads.

```
   public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                            String key, String downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

       CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>

                   <Loggers>
                       <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                           <AppenderRef ref="AlignedConsoleAppender"/>
                       </logger>
                   </Loggers>
               </Configuration>

           Log4J2 logs the progress. The following is example output for a 21.3 MB
file download.
               Transfer initiated...
               |=======             | 39.4%
               |===============     | 78.8%
               |====================| 100.0%
```

```
                Transfer complete!
        */
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetObject
  - PutObject

Hochladen eines Verzeichnisses in einen Bucket

Das folgende Codebeispiel zeigt, wie Sie ein lokales Verzeichnis rekursiv in einen Amazon Simple Storage Service (Amazon S3)-Bucket hochladen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie an S3TransferManager, um ein lokales Verzeichnis hochzuladen. Sehen Sie sich die vollständige Datei an und testen Sie sie.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
```

```
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- API-Details finden Sie UploadDirectory in der AWS SDK for Java 2.x API-Referenz.

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien von und zu einem S3-Bucket übertragen, mit dem S3TransferManager.

```
    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
```

```
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

Laden Sie eine einzelne Datei hoch.

```
    public String uploadFile(S3TransferManager transferManager, String bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
```

```
        return uploadResult.response().eTag();
    }
```

In den Codebeispielen werden folgende Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Verwenden Sie den S3 Transfer Manager zusätzlich zum AWS CRT-based S3-Client, um transparent einen mehrteiligen Upload durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

```
    /**
```

```
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the file path of the file to be uploaded
     */
    public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

Verwenden Sie die S3Client-API, um einen mehrteiligen Upload durchzuführen.

```
    /**
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the object key
     * @param filePath   the path to the file to be uploaded
     */
    public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
```

```
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

Verwenden Sie die S3AsyncClient API mit aktivierter mehrteiliger Unterstützung, um einen mehrteiligen Upload durchzuführen.

```
   /**
    * Uploads a file to an S3 bucket using the S3AsyncClient and enabling multipart
support.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the object key
    * @param filePath   the local file path of the file to be uploaded
    */
   public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
       // Enable multipart support.
       S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
           .multipartEnabled(true)
           .build();

       CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b
               .bucket(bucketName)
               .key(key),
           Paths.get(filePath));

       response.join();
       logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
   }
```

Stream unbekannter Größe hochladen

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe in ein Amazon S3-Objekt hochladen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie den AWS CRT-based S3-Client.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
```

```
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
 use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Verwenden Sie den standardmäßigen asynchronen S3-Client mit aktiviertem mehrteiligem Upload.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
 LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
 S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
```

```
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size, use
 can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key), body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Verwenden Sie den Amazon-S3-Transfer-Manager.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
 LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
 example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
 completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size, you
 can use the S3TransferManager based on the AWS CRT-based S3 client.
```

```
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

Verwenden der Prüfsummen

Das folgende Codebeispiel zeigt, wie Sie Prüfsummen verwenden, um mit einem Amazon-S3-Objekt zu arbeiten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In den Codebeispielen wird eine Teilmenge der folgenden Importe verwendet.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
```

```
import java.util.UUID;
```

Geben Sie einen Prüfsummenalgorithmus für die `putObject`-Methode an, wenn Sie `PutObjectRequest` erstellen.

```
    public void putObjectWithChecksum(String bucketName, String key) {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
            RequestBody.fromString("This is a test"));
    }
```

Überprüfen Sie die Prüfsumme für die `getObject` Methode, wenn Sie die GetObjectRequest erstellen.

```
    public GetObjectResponse getObjectWithChecksum(String bucketName, String key) {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
            .response();
    }
```

Pre-calculate eine Prüfsumme für die `putObject` Methode, wenn Sie die erstellen. `PutObjectRequest`

```
    public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
 String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
            RequestBody.fromFile(Paths.get(filePath)));
    }
```

Verwenden Sie den S3 Transfer Manager zusätzlich zum AWS CRT-based S3-Client, um transparent einen mehrteiligen Upload durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

Sie können einen Prüfsummenalgorithmus angeben, den das SDK verwenden soll. Standardmäßig verwendet das SDK den CRC32-Algorithmus.

```
   public void multipartUploadWithChecksumTm(String bucketName, String key, String
filePath) {
       S3TransferManager transferManager = S3TransferManager.create();
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b
               .bucket(bucketName)
               .key(key)
               .checksumAlgorithm(ChecksumAlgorithm.SHA1))
           .source(Paths.get(filePath))
           .build();
       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
       fileUpload.completionFuture().join();
       transferManager.close();
   }
```

Verwenden Sie die S3Client-API oder (S3AsyncClient API), um einen mehrteiligen Upload durchzuführen. Wenn Sie eine zusätzliche Prüfsumme angeben, müssen Sie den Algorithmus angeben, der bei der Initiierung des Uploads verwendet werden soll. Sie müssen auch den Algorithmus für jede Teilanforderung angeben und die für jedes Teil nach dem Hochladen berechnete Prüfsumme bereitstellen.

```
   public void multipartUploadWithChecksumS3Client(String bucketName, String key,
String filePath) {
       ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

       // Initiate the multipart upload.
       CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
           .bucket(bucketName)
           .key(key)
           .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
       String uploadId = createMultipartUploadResponse.uploadId();

       // Upload the parts of the file.
```

```
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .checksumAlgorithm(algorithm) // Checksum specified on each
part.
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
```

```
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
   }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Java

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotificat

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

Beispiele für Amazon S3 Control unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Control Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon S3 Control

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon S3 Control.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
```

```
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have not
 configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/latest/
guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }
```

```
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
```

```
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation is
finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- API-Details finden Sie ListJobs in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel zeigt, wie Sie die Kernoperationen von Amazon S3 Control erlernen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erlernen Sie die Kernoperationen.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing of
 large-scale
            data stored in Amazon S3. It automatically scales resources to handle
 varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
 operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove tags
 on thousands or millions of
```

```
            objects in a single operation, streamlining the management and
 organization of their data.

            This can be particularly useful for tasks such as cost allocation,
 lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
 essential.
            S3 Batch's scalability and serverless nature make it an ideal solution
 for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this scenario.
        System.out.println("Use CloudFormation to stand up the resource required for
 this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
 CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
```

```
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the number,
the higher the priority.
             So, a job with a priority of `30` would have a higher priority than a
job with
             a priority of `20`. This is a common way to represent the priority of a
task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
```

```
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " + ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
```

```
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
```

```
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }


}
```

Erstellen Sie eine Klasse, die Operationen umschließt.

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
```

```
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
```

```
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
```

```
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
               future.complete(null); // Complete the CompletableFuture on
successful execution
           })
           .exceptionally(ex -> {
               System.err.println("Failed to update job priority: " +
ex.getMessage());
               future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
               return null; // Return null to handle the exception
           });

       return future;
   }

   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return asyncClient.getJobTagging(request)
```

```
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
```

```
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of the
 S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
```

```
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
```

```
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3 bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
```

```
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
```

```
        Path filePath = Paths.get("src/main/resources/batch/",
 fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
 getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into bucket
 " + bucketName);
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
   /**
    * Deletes an object from an Amazon S3 bucket asynchronously.
    *
    * @param bucketName The name of the S3 bucket where the object is stored.
    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has been
deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
deletion.
    */
   public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
       ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
       toDelete.add(ObjectIdentifier.builder()
           .key(objectName)
           .build());

       DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
           .bucket(bucketName)
           .delete(Delete.builder()
               .objects(toDelete).build())
           .build();

       return getS3AsyncClient().deleteObjects(dor)
           .thenAccept(result -> {
               System.out.println("The object was deleted!");
           })
           .exceptionally(ex -> {
               throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
           });
   }

   /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
```

```
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }
```

```
    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has been
deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " + bucketName,
ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or uploading
the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " + bucketName);
    }

    /**
```

```
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                   actions.deleteBucketObjects(bucketName, fileName)
                  .thenRun(() -> System.out.println("Object deletion completed"))
                  .exceptionally(ex -> {
                      System.err.println("Error occurred: " + ex.getMessage());
                      return null;
                  });
        }
        System.out.println("All files have been deleted from the bucket " +
bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " + path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
filePath.toString());

                    // Create and write content to the new file.
```

```
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority

- UpdateJobStatus

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen asynchronen S3-Auftrag.

```
   /**
    * Creates an asynchronous S3 job using the AWS Java SDK.
    *
    * @param accountId         the AWS account ID associated with the job
    * @param iamRoleArn        the ARN of the IAM role to be used for the job
    * @param manifestLocation  the location of the job manifest file in S3
    * @param reportBucketName  the name of the S3 bucket to store the job report
    * @param uuid              a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of the
S3 job.
    *         The CompletableFuture will return the job ID if the job is created
successfully,
    *         or throw an exception if there is an error.
    */
   public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                     String manifestLocation,
String reportBucketName, String uuid) {

       String[] bucketName = new String[]{""};
       String[] parts = reportBucketName.split(":::");
       if (parts.length > 1) {
           bucketName[0] = parts[1];
       } else {
```

```
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();
```

```
                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }
```

Erstellen Sie einen Auftrag zur Compliance-Aufbewahrung.

```
    /**
```

```
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();
```

```
        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Erstellen Sie einen Auftrag für eine gesetzliche Aufbewahrungsfrist.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
with
```

```
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
```

```
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

Erstellen Sie einen neuen Auftrag zur Governance-Aufbewahrung.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
 <manifestObjectVersionId>
```

```
            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3 object
that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will be
stored.
                roleArn - The ARN of the IAM role that will be used to perform the
governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the `eTag`
property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException {
        final JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
```

```
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
```

```
    }
}
```

- API-Details finden Sie CreateJob in der AWS SDK for Java 2.x API-Referenz.

`DeleteJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJobTagging`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously deletes the tags associated with a specific batch job.
    *
    * @param jobId     The ID of the batch job whose tags should be deleted.
    * @param accountId The ID of the account associated with the batch job.
    * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
    */
   public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
       DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .build();

       return asyncClient.deleteJobTagging(jobTaggingRequest)
           .thenAccept(response -> {
               System.out.println("You have successfully deleted " + jobId + "
tagging.");
           })
           .exceptionally(ex -> {
               System.err.println("Failed to delete job tags: " + ex.getMessage());
```

```
                throw new RuntimeException(ex);
            });
    }
```

- API-Details finden Sie DeleteJobTagging in der AWS SDK for Java 2.x API-Referenz.

`DescribeJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeJob`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
```

```
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- API-Details finden Sie DescribeJob in der AWS SDK for Java 2.x API-Referenz.

`GetJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`GetJobTagging`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- API-Details finden Sie GetJobTagging in der AWS SDK for Java 2.x API-Referenz.

`PutJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`PutJobTagging`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId     the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
   public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
       S3Tag departmentTag = S3Tag.builder()
           .key("department")
           .value("Marketing")
           .build();

       S3Tag fiscalYearTag = S3Tag.builder()
           .key("FiscalYear")
           .value("2020")
           .build();

       PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .tags(departmentTag, fiscalYearTag)
           .build();

       return asyncClient.putJobTagging(putJobTaggingRequest)
           .thenRun(() -> {
```

```
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- API-Details finden Sie PutJobTagging in der AWS SDK for Java 2.x API-Referenz.

`UpdateJobPriority`

Das folgende Codebeispiel zeigt die Verwendung`UpdateJobPriority`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
```

```
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- API-Details finden Sie UpdateJobPriority in der AWS SDK for Java 2.x API-Referenz.

`UpdateJobStatus`

Das folgende Codebeispiel zeigt die Verwendung`UpdateJobStatus`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
```

```
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- API-Details finden Sie UpdateJobStatus in der AWS SDK for Java 2.x API-Referenz.

Beispiele für S3 Directory Buckets unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von S3 Directory Buckets Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon S3 directory buckets

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Verzeichnis-Buckets von Amazon S3.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.s3.directorybucket;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
```

```
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected error:
{}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);
```

```
        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
```

```
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error code:
 " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateBucket
  - ListDirectoryBuckets

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.
- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.
- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;
```

```
   private static String stackName = "cfn-stack-s3-express-basics--" +
UUID.randomUUID();

   private static String regularUser = "";
   private static String vpcId = "";
   private static String expressUser = "";

   private static String vpcEndpointId = "";

   private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

   public static void main(String[] args) {
       try {
           s3ExpressScenario();
       } catch (RuntimeException e) {
           logger.info(e.getMessage());
       }
   }

   // Runs the scenario.
   private static void s3ExpressScenario() {
       logger.info(DASHES);
       logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK for
Java V2.");
       logger.info("""
           Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
           specifically when working in the same Availability Zone (AZ). To see the
best results in this example and when you implement
           directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
           bucket.
           """);
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       // Create an optional VPC and create 2 IAM users.
       UserNames userNames = createVpcUsers();
       String expressUserName = userNames.getExpressUserName();
       String regularUserName = userNames.getRegularUserName();

       //  Set up two S3 clients, one regular and one express,
```

```
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate how
to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3 Express
One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
```

```
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient, mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
```

```
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ", e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from a
directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and measures
the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this example
is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.
```

```
            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + " for
the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}", downloads);
        }
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
```

```
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
```

```
        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if you
are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }
```

```
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with permission
to
            work with directory buckets.

            It's important to remember the user permissions when interacting with
            directory buckets. Instead of validating permissions on every call as
            regular buckets do, directory buckets utilize the user credentials and
session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions and
one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
        */
        logger.info(DASHES);
        logger.info("""
```

```
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");
```

```
        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the stack
outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express permissions
     * @param regularUserName the username for the user with regular S3 permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
```

```
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and one
uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore here.
We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
```

```
            """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-s3"; //
Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
```

```
    * Creates an asynchronous S3 client with the specified access key and secret
access key.
    *
    * @param accessKeyId     the AWS access key ID
    * @param secretAccessKey the AWS secret access key
    * @return a {@link CompletableFuture} that asynchronously creates the S3 client
    * @throws IllegalArgumentException if the access key ID or secret access key is
null
    */
   public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
       return CompletableFuture.supplyAsync(() -> {
           // Validate input parameters
           if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey ==
null || secretAccessKey.isBlank()) {
               throw new IllegalArgumentException("Access Key ID and Secret Access
Key must not be null or empty");
           }

           AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
           return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
               .region(Region.US_WEST_2)
               .build();
       });
   }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           logger.info("");
           logger.info("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               logger.info("Continuing with the program...");
               logger.info("");
               break;
           } else {
               logger.info("Invalid input. Please try again.");
           }
       }
   }
```

```
}
```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
```

```
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or its
objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
        ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3AsyncClient.listObjectsV2(listRequest)
            .thenCompose(listResponse -> {
                if (!listResponse.contents().isEmpty()) {
                    List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                        .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
```

```
                        .collect(Collectors.toList());

                    DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                        .bucket(bucketName)
                        .delete(Delete.builder().objects(objectIdentifiers).build())
                        .build();

                    return s3AsyncClient.deleteObjects(deleteRequest)
                        .thenAccept(deleteResponse -> {
                            if (!deleteResponse.errors().isEmpty()) {
                                deleteResponse.errors().forEach(error ->
                                    logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                            }
                        });
                }
                return CompletableFuture.completedFuture(null);
            })
            .thenCompose(ignored -> {
                DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3AsyncClient.deleteBucket(deleteBucketRequest);
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
```

```
    }

    /**
     *  Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys in
the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient s3Client,
String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in bucket:
" + bucketName, exception);
                }
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client   the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
     */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();
```

```
        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest, AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object. Reason:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client           the S3 async client to use for the copy operation
     * @param sourceBucket       the name of the source bucket
     * @param sourceKey          the key of the object to be copied in the source
bucket
     * @param destinationBucket  the name of the destination bucket
     * @param destinationKey     the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation is
finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
```

```
                        throw new CompletionException("Couldn't copy object '" +
sourceKey + "' from bucket '" + sourceBucket + "' to bucket '" + destinationBucket
+ "'. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the session
     * @param bucketName the name of the S3 bucket for which to create the session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse> createSessionAsync(S3AsyncClient
s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the session.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
```

```
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String zone)
{
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists: "
+ ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
```

```
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3Waiter.waitUntilBucketExists(bucketRequestWait);
            })
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                    }
                }
                logger.info(bucketName + " is ready");
            });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client     the S3 async client to use for the upload
     * @param bucketName   the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
```

```
     * @param text         the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest, AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not exist:
" + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(userName)
            .build();

        return getIAMAsyncClient().createAccessKey(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Access Key Created.");
                } else {
                    if (exception == null) {
                        Throwable cause = exception.getCause();
```

```
                        if (cause instanceof IamException) {
                            throw new CompletionException("IAM error while creating
 access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create access
 key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2 zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
 Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
 processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
 DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return null
 if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g., "usw2-
az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
 promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
```

```
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
    private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String> zoneIds)
{
        Scanner scanner = new Scanner(System.in);
        int index = -1;

        while (index < 0 || index >= zoneIds.size()) {
            logger.info("Select an availability zone:");
            IntStream.range(0, zoneIds.size()).forEach(i ->
                logger.info(i + ": " + zoneIds.get(i))
            );

            logger.info("Enter the number corresponding to your choice: ");
            if (scanner.hasNextInt()) {
                index = scanner.nextInt();
            } else {
```

```
                scanner.next();
            }
        }

        AvailabilityZone selectedZone = zonesList.get(index);
        logger.info("You selected: " + selectedZone.zoneId());
        return selectedZone;
    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
     *         VPC ID and VPC endpoint ID.
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();
```

```
                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found for
VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId, routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

                CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId, vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: " +
 cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Aktionen

`AbortMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`AbortMultipartUpload`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Brechen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;


    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
 bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
 objectKey);
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API-Details finden Sie AbortMultipartUpload in der AWS SDK for Java 2.x API-Referenz.

`CompleteMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CompleteMultipartUpload`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Schließen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                    .parts(uploadParts)
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
 CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
 s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to complete multipart upload: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API-Details finden Sie CompleteMultipartUpload in der AWS SDK for Java 2.x API-Referenz.

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt aus einem Verzeichnis-Bucket in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}", objectKey,
 sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
 CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
 copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie CopyObject in der AWS SDK for Java 2.x API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
 String zone) throws S3Exception {
```

```
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie CreateBucket in der AWS SDK for Java 2.x API-Referenz.

`CreateMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultipartUpload`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;
```

```
        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie CreateMultipartUpload in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie einen S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
```

```
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie DeleteBucket in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketEncryption`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration für einen Verzeichnis-Bucket.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie DeleteBucketEncryption in der AWS SDK for Java 2.x API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Bucket-Richtlinie für einen Verzeichnis-Bucket.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
 DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();
```

```
            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Objekts in einem Verzeichnis-Bucket

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie DeleteObject in der AWS SDK for Java 2.x API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie mehrere Objekte in einem Verzeichnis-Bucket

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
 bucketName, List<String> objectKeys) {
```

```
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted -> logger.info("Deleted
object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie DeleteObjects in der AWS SDK for Java 2.x API-Referenz.

`GetBucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketEncryption`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration eines Verzeichnis-Buckets.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
            GetBucketEncryptionRequest getRequest =
 GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket encryption configuration
```

```
            GetBucketEncryptionResponse response =
 s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
 response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
 rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
 rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie GetBucketEncryption in der AWS SDK for Java 2.x API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Richtlinie eines Verzeichnis-Buckets ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response = s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt aus einem Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves an object from the specified S3 directory bucket.
```

```
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();

            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API-Details finden Sie GetObject in der AWS SDK for Java 2.x API-Referenz.

`GetObjectAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Objektattribute aus einem eines Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);
```

```
        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API-Details finden Sie GetObjectAttributes in der AWS SDK for Java 2.x API-Referenz.

`HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung`HeadBucket`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Prüfen Sie, ob das angegebene S3-Verzeichnis-Bucket existiert und zugänglich ist.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String bucketName)
 {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
```

```
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie HeadBucket in der AWS SDK for Java 2.x API-Referenz.

`HeadObject`

Das folgende Codebeispiel zeigt die Verwendung`HeadObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Metadaten eines Objekts in einem Verzeichnis-Bucket ab.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with ETag:
 \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API-Details finden Sie HeadObject in der AWS SDK for Java 2.x API-Referenz.

`ListDirectoryBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListDirectoryBuckets`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie alle Verzeichnis-Buckets auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
```

```
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- API-Details finden Sie ListDirectoryBuckets in der AWS SDK for Java 2.x API-Referenz.

`ListMultipartUploads`

Das folgende Codebeispiel zeigt die Verwendung`ListMultipartUploads`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie mehrteilige Uploads in einen Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * Lists multipart uploads for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of MultipartUpload objects representing the multipart uploads
     */
    public static List<MultipartUpload> listDirectoryBucketMultipartUploads(S3Client
 s3Client, String bucketName) {
        logger.info("Listing in-progress multipart uploads for bucket: {}",
 bucketName);

        try {
            // Create a ListMultipartUploadsRequest
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            // List the multipart uploads
            ListMultipartUploadsResponse response =
 s3Client.listMultipartUploads(listMultipartUploadsRequest);
```

```
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                logger.info("In-progress multipart upload: Upload ID: {}, Key: {},
 Initiated: {}", upload.uploadId(),
                        upload.key(), upload.initiated());
            }
            return uploads;

        } catch (S3Exception e) {
            logger.error("Failed to list multipart uploads: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- API-Details finden Sie ListMultipartUploads in der AWS SDK for Java 2.x API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte in einem Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

```
import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);

        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
 ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
 s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
            logger.error("Failed to list objects: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK for Java 2.x API-Referenz.

`ListParts`

Das folgende Codebeispiel zeigt die Verwendung`ListParts`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Teile eines mehrteiligen Uploads in einen Verzeichnis-Bucket auf.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Lists the parts of a multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object being uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @return A list of Part representing the parts of the multipart upload
     */
    public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
            String objectKey, String uploadId) {
        logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

        try {
            // Create a ListPartsRequest
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                    .bucket(bucketName)
                    .uploadId(uploadId)
                    .key(objectKey)
                    .build();

            // List the parts of the multipart upload
            ListPartsResponse response = s3Client.listParts(listPartsRequest);
            List<Part> parts = response.parts();
            for (Part part : parts) {
                logger.info("Uploaded part: Part number = \"{}\", etag = {}",
 part.partNumber(), part.eTag());
            }
            return parts;

        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- API-Details finden Sie ListParts in der AWS SDK for Java 2.x API-Referenz.

`PutBucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketEncryption`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie die Bucket-Verschlüsselung auf einen Verzeichnis-Bucket fest.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
```

```
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For directory
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();

        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- API-Details finden Sie PutBucketEncryption in der AWS SDK for Java 2.x API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wenden Sie eine Bucket-Richtlinie auf einen Verzeichnis-Bucket an.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
```

```
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the same
 account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie PutBucketPolicy in der AWS SDK for Java 2.x API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie ein Objekts in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey, Path filePath) {
```

```
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- API-Details finden Sie PutObject in der AWS SDK for Java 2.x API-Referenz.

`UploadPart`

Das folgende Codebeispiel zeigt die Verwendung`UploadPart`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie einen Teil eines mehrteiligen Uploads in einen Verzeichnis-Bucket hoch.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the completed
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
```

```
    */
   public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
           String objectKey, String uploadId, Path filePath) throws IOException {
       logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

       int partNumber = 1;
       List<CompletedPart> uploadedParts = new ArrayList<>();
       ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

       // Read the local file, break down into chunks and process
       try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(), "r")) {
           long fileSize = file.length();
           int position = 0;

           // Sequentially upload parts of the file
           while (position < fileSize) {
               file.seek(position);
               int read = file.getChannel().read(bb);

               bb.flip(); // Swap position and limit before reading from the buffer
               UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                       .bucket(bucketName)
                       .key(objectKey)
                       .uploadId(uploadId)
                       .partNumber(partNumber)
                       .build();

               UploadPartResponse partResponse = s3Client.uploadPart(
                       uploadPartRequest,
                       RequestBody.fromByteBuffer(bb));

               // Build the uploaded part
               CompletedPart uploadedPart = CompletedPart.builder()
                       .partNumber(partNumber)
                       .eTag(partResponse.eTag())
                       .build();

               // Add the uploaded part to the list
               uploadedParts.add(uploadedPart);

               // Log to indicate the part upload is done
```

```
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- API-Details finden Sie UploadPart in der AWS SDK for Java 2.x API-Referenz.

`UploadPartCopy`

Das folgende Codebeispiel zeigt die Verwendung`UploadPartCopy`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Kopierteile basierend auf der Größe des Quellobjekts und kopieren Sie einzelne Teile in einen Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart> multipartUploadCopyForDirectoryBucket(S3Client
 s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
```

```
                .build();
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" + lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code: {}",
partNum,
```

```
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
    }
```

- API-Details finden Sie UploadPartCopy in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer vorsignierten URL, um ein Objekt abzurufen

Die folgenden Codebeispiele zeigen, wie Sie eine vorsignierte URL für S3-Verzeichnis-Buckets erstellen und ein Objekt hochladen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte GET-URL für den Zugriff auf ein Objekt in einem S3-Verzeichnis-Bucket.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
```

```
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Generates a presigned URL for accessing an object in the specified S3
     * directory bucket.
     *
     * @param s3Presigner The S3 presigner client used to generate the presigned URL
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to access
     * @return A presigned URL for accessing the specified object
     */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
            String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
 valid for 10 minutes
                    .getObjectRequest(getObjectRequest)
                    .build();

            // Generate the presigned URL
```

```
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API-Details finden Sie GetObject in der AWS SDK for Java 2.x API-Referenz.

SageMaker KI-Beispiele, die das SDK für Java 2.x verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von SageMaker KI Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo SageMaker KI

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von SageMaker KI beginnen können.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSageMaker {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        listBooks(sageMakerClient);
        sageMakerClient.close();
    }

    public static void listBooks(SageMakerClient sageMakerClient) {
        try {
            ListNotebookInstancesResponse notebookInstancesResponse =
 sageMakerClient.listNotebookInstances();
            List<NotebookInstanceSummary> items =
 notebookInstancesResponse.notebookInstances();
            for (NotebookInstanceSummary item : items) {
                System.out.println("The notebook name is: " +
 item.notebookInstanceName());
```

```
            }

        } catch (SageMakerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`CreatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`CreatePipeline`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
 filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
```

```
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }
```

- API-Details finden Sie CreatePipeline in der AWS SDK for Java 2.x API-Referenz.

`DeletePipeline`

Das folgende Codebeispiel zeigt die Verwendung`DeletePipeline`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
```

```
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }
```

- API-Details finden Sie DeletePipeline in der AWS SDK for Java 2.x API-Referenz.

`DescribePipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribePipelineExecution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }
```

- API-Details finden Sie DescribePipelineExecution in der AWS SDK for Java 2.x API-Referenz.

StartPipelineExecution

Das folgende Codebeispiel zeigt die VerwendungStartPipelineExecution.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
```

```
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();
```

```
        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
 StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }
```

- API-Details finden Sie StartPipelineExecution in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erste Schritte mit Geodatenaufträgen und Pipelines

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.

- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter Erstellen und Ausführen von SageMaker Pipelines mithilfe von AWS SDKs. Community.aws

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
 +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service (Amazon
 SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
                "    lnglatData - The file location of the latlongtest.csv file
 required for this use case.\n\n" +
```

```
                "    spatialPipelinePath - The file location of the
GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for example,
sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store the
results in an export file.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
        String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
        String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

        String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
                handlerName);
        String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
        System.out.println("The queue URL is " + queueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setting up bucket " + bucketName);
        if (!checkBucket(s3Client, bucketName)) {
            setupBucket(s3Client, bucketName);
            System.out.println("Put " + lnglatData + " into " + bucketName);
            putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now we can create and run our pipeline.");
        setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
        String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
                pipelineName);
```

```
        System.out.println("The pipeline execution ARN value is " +
 pipelineExecutionARN);
        waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
        System.out.println("Getting output results " + bucketName);
        getOutputResults(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
 {
        System.out.println("Output file contents: \n");
```

```
        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
        String text = new String(byteArray, StandardCharsets.UTF_8);
        System.out.println("Text output: " + text);
    }

    // Display some results from the output directory.
    public static void getOutputResults(S3Client s3Client, String bucketName) {
        System.out.println("Getting output results {bucketName}.");
        ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
                .bucket(bucketName)
                .prefix("outputfiles/")
                .build();

        ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
        List<S3Object> s3Objects = response.contents();
        for (S3Object object : s3Objects) {
            readObject(s3Client, bucketName, object.key());
        }
    }

    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
```

```
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
 pipelineName) {
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
 filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();
```

```
            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";
```

```
        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();

        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();
```

```
        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);

        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
```

```
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
```

```
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
```

```
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
```

```
   public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
           String lambdaName) {
       System.out.println("Setting up queue named " + queueName);
       try {
           Map<QueueAttributeName, String> queueAtt = new HashMap<>();
           queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
           queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
           queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(queueAtt)
                   .build();

           sqsClient.createQueue(createQueueRequest);
           System.out.println("\nGet queue url");
           GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
           TimeUnit.SECONDS.sleep(15);

           connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
           System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
           return getQueueUrlResponse.queueUrl();

       } catch (SqsException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       } catch (InterruptedException e) {
           throw new RuntimeException(e);
       }
       return "";
   }

   // Connect the queue to the Lambda function as an event source.
   public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
           String lambdaName) {
       System.out.println("Connecting the Lambda function and queue for the
pipeline.");
       String queueArn = "";
```

```
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
```

```
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
                    .build();

            // Create a Lambda function using a waiter.
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The function ARN is " +
functionResponse.functionArn());
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSageMakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        System.out.println("Creating a role to use with SageMaker.");
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
```

```
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
```

```
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
```

```
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this workflow.");
        String functionArn;
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }
```

```
    // Checks to see if the Amazon SQS queue exists. If not, this method creates a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }
```

```
    // Checks to see if the SageMaker role exists. If not, this method creates it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Beispiele für Secrets Manager unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Secrets Manager Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * We recommend that you cache your secret values by using client-side caching.
 *
 * Caching secrets improves speed and reduces your costs. For more information,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-
secrets.html
 */
public class GetSecretValue {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <secretName>\s

                Where:
                    secretName - The name of the secret (for example, tutorials/
MyFirstSecret).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String secretName = args[0];
        Region region = Region.US_EAST_1;
        SecretsManagerClient secretsClient = SecretsManagerClient.builder()
                .region(region)
```

```
                .build();

        getValue(secretsClient, secretName);
        secretsClient.close();
    }

    public static void getValue(SecretsManagerClient secretsClient, String
 secretName) {
        try {
            GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                    .secretId(secretName)
                    .build();

            GetSecretValueResponse valueResponse =
 secretsClient.getSecretValue(valueRequest);
            String secret = valueResponse.secretString();
            System.out.println(secret);

        } catch (SecretsManagerException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetSecretValue in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon SES unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();
```

```
        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListIdentities in der AWS SDK for Java 2.x API-Referenz.

`ListTemplates`

Das folgende Codebeispiel zeigt die Verwendung`ListTemplates`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
```

```
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListTemplates in der AWS SDK for Java 2.x API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
```

```
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();
```

```
        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use as
 an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p>Please see the attached file for a " + "list of customers to
 contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
```

```
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
```

```
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);
```

```
            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- API-Details finden Sie SendEmail in der AWS SDK for Java 2.x API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;
```

```
/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }
```

```
    public static void send(SesV2Client client, String sender, String recipient,
 String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        /*
         * Specify both name and favorite animal (favoriteanimal) in your code when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
 doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
 using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SendTemplatedEmail in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Redshift-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift

- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Java 2.x

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter. GitHub

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen eines Beispiels, das die JDBC-API verwendet, finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Erkennen von PSA in Bildern

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service

(Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Zustandsmaschine erstellt wird, die AWS Lambda Funktionen nacheinander aufruft.

SDK für Java 2.x

Zeigt, wie ein AWS serverloser Workflow mithilfe von AWS Step Functions und erstellt wird. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step-Funktionen

Beispiele für Amazon SES API v2 unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon SES-API v2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateContact`

Das folgende Codebeispiel zeigt die Verwendung`CreateContact`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
try {
  // Create a new contact with the provided email address in the
  CreateContactRequest contactRequest = CreateContactRequest.builder()
      .contactListName(CONTACT_LIST_NAME)
      .emailAddress(emailAddress)
      .build();

  sesClient.createContact(contactRequest);
  contacts.add(emailAddress);

  System.out.println("Contact created: " + emailAddress);
```

```
        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }
```

- API-Details finden Sie CreateContact in der AWS SDK for Java 2.x API-Referenz.

`CreateContactList`

Das folgende Codebeispiel zeigt die Verwendung`CreateContactList`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }
```

- API-Details finden Sie CreateContactList in der AWS SDK for Java 2.x API-Referenz.

`CreateEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailIdentity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   try {
     CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
         .emailIdentity(verifiedEmail)
         .build();
     sesClient.createEmailIdentity(createEmailIdentityRequest);
     System.out.println("Email identity created: " + verifiedEmail);
   } catch (AlreadyExistsException e) {
     System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
   } catch (NotFoundException e) {
     System.err.println("The provided email address is not verified: " +
verifiedEmail);
     throw e;
   } catch (LimitExceededException e) {
     System.err
         .println("You have reached the limit for email identities. Please remove
some identities and try again.");
     throw e;
   } catch (SesV2Exception e) {
     System.err.println("Error creating email identity: " + e.getMessage());
     throw e;
   }
```

- API-Details finden Sie CreateEmailIdentity in der AWS SDK for Java 2.x API-Referenz.

`CreateEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailTemplate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
 remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
 e.getMessage());
      throw e;
```

```
    }
```

- API-Details finden Sie CreateEmailTemplate in der AWS SDK for Java 2.x API-Referenz.

`DeleteContactList`

Das folgende Codebeispiel zeigt die Verwendung`DeleteContactList`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }
```

- API-Details finden Sie DeleteContactList in der AWS SDK for Java 2.x API-Referenz.

`DeleteEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailIdentity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Delete the email identity
      DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
          .emailIdentity(this.verifiedEmail)
          .build();

      sesClient.deleteEmailIdentity(deleteIdentityRequest);

      System.out.println("Email identity deleted: " + this.verifiedEmail);
    } catch (NotFoundException e) {
      // If the email identity does not exist, log the error and proceed
      System.out.println("Email identity not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
      e.printStackTrace();
    }
  } else {
    System.out.println("Skipping email identity deletion.");
  }
```

- API-Details finden Sie DeleteEmailIdentity in der AWS SDK for Java 2.x API-Referenz.

`DeleteEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailTemplate`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

- API-Details finden Sie DeleteEmailTemplate in der AWS SDK for Java 2.x API-Referenz.

`ListContacts`

Das folgende Codebeispiel zeigt die Verwendung`ListContacts`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }
```

- API-Details finden Sie ListContacts in der AWS SDK for Java 2.x API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
```

```
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
                                    recipient - An email address that represents the
 recipient.\s
                                    subject - The subject line.\s
                                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        // The HTML body of the email.
```

```
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML) {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
```

```
            System.out.println("Attempting to send an email through Amazon SES "
                    + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Sendet eine Nachricht unter Verwendung einer Vorlage.

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }
```

Sendet eine Nachricht mit Header-Informationen.

```
public class SendwithHeader {

    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
```

```
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c -> c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " + response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- API-Details finden Sie SendEmail in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Newsletter-Szenario

Das folgende Codebeispiel zeigt, wie Sie das Newsletter-Szenario von Amazon SES API v2 ausführen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
 CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
```

```
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
```

```
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }

    try {
```

```
      CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
 verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please remove
 some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating email identity: " + e.getMessage());
      throw e;
    }

    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
    } catch (AlreadyExistsException e) {
```

```
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
e.getMessage());
      throw e;
    }

    try {
      // Delete the contact list
      DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      sesClient.deleteContactList(deleteContactListRequest);

      System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
    } catch (NotFoundException e) {
      // If the contact list does not exist, log the error and proceed
      System.out.println("Contact list not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }

      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
```

```
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity

- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Beispiele für Amazon SNS unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SNS beginnen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`CheckIfPhoneNumberIsOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`CheckIfPhoneNumberIsOptedOut`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CheckIfPhoneNumberIsOptedOut in der AWS SDK for Java 2.x API-Referenz.

`ConfirmSubscription`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSubscription`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ConfirmSubscription in der AWS SDK for Java 2.x API-Referenz.

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
```

```
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API-Details finden Sie CreateTopic in der AWS SDK for Java 2.x API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API-Details finden Sie DeleteTopic in der AWS SDK for Java 2.x API-Referenz.

`GetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetSMSAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK for Java 2.x - API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " + topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn) {
        try {
            GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
 snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
 result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK for Java 2.x API-Referenz.

`ListPhoneNumbersOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`ListPhoneNumbersOptedOut`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " + result.sdkHttpResponse().statusCode()
 + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListPhoneNumbersOptedOut in der AWS SDK for Java 2.x API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListSubscriptions in der AWS SDK for Java 2.x API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK for Java 2.x API-Referenz.

Veröffentlichen

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK for Java 2.x -API-Referenz.

`SetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSMSAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK for Java 2.x - API-Referenz.

`SetSubscriptionAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSubscriptionAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.
```

```
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
```

```
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SetSubscriptionAttributes in der AWS SDK for Java 2.x API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() + "\n
\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie SetTopicAttributes in der AWS SDK for Java 2.x API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>
```

```
                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie einen HTTP-Endpunkt für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url) {
        try {
```

```
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>
```

```
                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK for Java 2.x -API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie TagResource in der AWS SDK for Java 2.x API-Referenz.

Abonnement beenden

Das folgende Codebeispiel zeigt die VerwendungUnsubscribe.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK for Java 2.x -API-Referenz.

Szenarien

Erstellen Sie eine App zum Senden von Daten an eine DynamoDB-Tabelle

Die folgenden Codebeispiele zeigen, wie eine Anwendung erstellt wird, die Daten an eine Amazon-DynamoDB-Tabelle übermittelt und Sie benachrichtigt, wenn ein Benutzer die Tabelle aktualisiert.

SDK für Java 2.x

Zeigt, wie man eine dynamische Webanwendung erstellt, die Daten über die Amazon-DynamoDB-Java-API übermittelt und eine Textnachricht über die Amazon Simple Notification Service Java API sendet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SNS

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen des Beispiels, das die Java Async-API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen eines Plattformendpunkts für Push-Benachrichtigungen

Das folgende Codebeispiel zeigt, wie ein Plattformendpunkt für Amazon-SNS-Push-Benachrichtigungen erstellt wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
```

```
                for Android devices) when the mobile app is registered to receive
 push notifications.

                platformApplicationArn - The ARN value of platform application. You
 can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.

- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to create.
 \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created for
 the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
 created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
 URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
```

```
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
```

```
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition

- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Veröffentlichen einer SMS-Nachricht zu einem Thema

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-SNS-Thema.
- Verknüpfen Sie Telefonnummern mit dem Thema.
- Veröffentlichen Sie SMS-Nachrichten im Thema, damit alle abonnierten Telefonnummern die Nachricht gleichzeitig empfangen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Thema und geben Sie seinen ARN zurück.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

Abonnieren eines Endpunkts für ein Thema.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives notifications
 (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
```

```
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Legen Sie Attribute für die Nachricht fest, z. B. die ID des Senders, den Höchstpreis und seinen Typ. Nachrichtenattribute sind optional.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String, String>
 attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
 was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen einer Nachricht für ein Thema. Die Nachricht wird an jeden Teilnehmer gesendet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());
```

```
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>
```

```
                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is sent
 (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK for Java 2.x -API-Referenz.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
```

```
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
```

```
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";
```

```
        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
```

```
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");
```

```
        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
```

```
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
```

```
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
```

```
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }
```

```
            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

```
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
```

```
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
```

```
   public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
       try {
           SubscribeRequest request;
           if (filterList.isEmpty()) {
               // No filter subscription is added.
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());
               return result.subscriptionArn();
           } else {
               request = SubscribeRequest.builder()
                   .protocol("sqs")
                   .endpoint(queueArn)
                   .returnSubscriptionArn(true)
                   .topicArn(topicArn)
                   .build();

               SubscribeResponse result = snsClient.subscribe(request);
               System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                   "with the subscription ARN " + result.subscriptionArn());

               String attributeName = "FilterPolicy";
               Gson gson = new Gson();
               String jsonString = "{\"tone\": []}";
               JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
               JsonArray toneArray = jsonObject.getAsJsonArray("tone");
               for (String value : filterList) {
                   toneArray.add(new JsonPrimitive(value));
               }

               String updatedJsonString = gson.toJson(jsonObject);
               System.out.println(updatedJsonString);
               SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
```

```
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
```

```
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
```

```
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();
```

```
            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion mithilfe der Lambda Java Runtime API erstellt wird. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine Lambda-Funktion erstellt, die von Amazon API Gateway aufgerufen wird und eine Amazon-DynamoDB-Tabelle nach

Arbeitsjubiläen durchsucht und Amazon Simple Notification Service (Amazon SNS) verwendet, um eine Textnachricht an Ihre Mitarbeiter zu senden, die ihnen zu ihrem einjährigen Jubiläum gratuliert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Java 2.x

Zeigt, wie ein von Amazon EventBridge geplantes Ereignis erstellt wird, das eine AWS Lambda Funktion aufruft. Konfigurieren Sie EventBridge die Konfiguration so, dass ein Cron-Ausdruck verwendet wird, um zu planen, wann die Lambda-Funktion aufgerufen wird. In diesem Beispiel erstellen Sie eine Lambda-Funktion mithilfe der Lambda-Java-Laufzeit-API. In diesem Beispiel werden verschiedene AWS Dienste aufgerufen, um einen bestimmten Anwendungsfall auszuführen. Dieses Beispiel zeigt, wie man eine App erstellt, die eine mobile Textnachricht an Ihre Mitarbeiter sendet, um ihnen zum einjährigen Jubiläum zu gratulieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
```

```
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

Beispiele für Amazon SQS unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SqsException;
import software.amazon.awssdk.services.sqs.paginators.ListQueuesIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloSQS {
    public static void main(String[] args) {
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        listQueues(sqsClient);
        sqsClient.close();
    }

    public static void listQueues(SqsClient sqsClient) {
```

```
        try {
            ListQueuesIterable listQueues = sqsClient.listQueuesPaginator();
            listQueues.stream()
                    .flatMap(r -> r.queueUrls().stream())
                    .forEach(content -> System.out.println(" Queue URL: " +
 content.toLowerCase()));

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListQueues in der AWS SDK for Java 2.x API-Referenz.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
```

```
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");
```

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
```

```
        }

        System.out.println("\nSend message");
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
```

```
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
```

```
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateQueue in der AWS SDK for Java 2.x API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- API-Details finden Sie DeleteMessage in der AWS SDK for Java 2.x API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteQueue {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName>

                Where:
                   queueName - The name of the Amazon SQS queue to delete.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String queueName = args[0];
        SqsClient sqs = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        deleteSQSQueue(sqs, queueName);
        sqs.close();
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteQueue in der AWS SDK for Java 2.x API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- API-Details finden Sie GetQueueUrl in der AWS SDK for Java 2.x API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

- API-Details finden Sie ListQueues in der AWS SDK for Java 2.x API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

- API-Details finden Sie ReceiveMessage in der AWS SDK for Java 2.x API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Es folgen zwei Beispiele für die `SendMessage`-Operation:

- Senden einer Nachricht mit Text und einer Verzögerung
- Senden einer Nachricht mit Text und Nachrichtenattributen

Senden einer Nachricht mit Text und einer Verzögerung.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessages {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <queueName> <message>

                Where:
                   queueName - The name of the queue.
                   message - The message to send.
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String queueName = args[0];
        String message = args[1];
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();
        sendMessage(sqsClient, queueName, message);
        sqsClient.close();
    }

    public static void sendMessage(SqsClient sqsClient, String queueName, String
 message) {
        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(message)
                    .delaySeconds(5)
                    .build();

            sqsClient.sendMessage(sendMsgRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Senden einer Nachricht mit Text und Nachrichtenattributen.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java Map
 to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary". You
 can optionally
     * define a custom extension by using a "." and your extension.</p>
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

 .binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
 e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
```

```
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}", e.getMessage(),
e);
            throw new RuntimeException(e);
        }
    }
```

- API-Details finden Sie SendMessage in der AWS SDK for Java 2.x API-Referenz.

`SendMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`SendMessageBatch`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- API-Details finden Sie SendMessageBatch in der AWS SDK for Java 2.x API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie ein Amazon SQS für die serverseitige Verschlüsselung (SSE) mit einem benutzerdefinierten KMS-Schlüssel.

```
    public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
        SqsClient sqsClient = SqsClient.create();

        GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

        GetQueueUrlResponse getQueueUrlResponse;
        try {
            getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
        } catch (QueueDoesNotExistException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
        String queueUrl = getQueueUrlResponse.queueUrl();


        Map<QueueAttributeName, String> attributes = Map.of(
                QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
                QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set
the data key reuse period to 140 seconds.
        );                                                                  // This
is how long SQS can reuse the data key before requesting a new one from KMS.
```

```
        SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
        } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Java 2.x

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel

- erstellt ein Amazon-SNS-FIFO-Thema, zwei Amazon SQS-FIFO-Warteschlangen und eine Standard-Warteschlange.
- abonniert die Warteschlangen für das Thema und veröffentlicht eine Nachricht zu dem Thema.

Der Test überprüft den Eingang der Nachricht in jeder Warteschlange. Das vollständige Beispiel zeigt auch das Hinzufügen von Zugriffsrichtlinien und löscht die Ressourcen am Ende.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
            "Where:\n" +
            "   fifoTopicName - The name of the FIFO topic that you want to create.
 \n\n" +
            "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
            +
            "   retailQueueARN - The name of a SQS FIFO queue that will created for
 the retail consumer. \n\n" +
            "   analyticsQueueARN - The name of a SQS standard queue that will be
 created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
```

```
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                "FifoTopic", "true",
                "ContentBasedDeduplication", "false",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
```

```
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
```

```
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Verwalten großer Nachrichten mit S3

Das folgende Codebeispiel zeigt, wie Sie die erweiterte Clientbibliothek von Amazon SQS verwenden, um mit großen Amazon-SQS-Nachrichten zu arbeiten.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
```

```
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
```

```
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- Weitere Informationen finden Sie im AWS SDK for Java 2.x -Entwicklerhandbuch.

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

Process-S3-Ereignisbenachrichtigungen

Das folgende Codebeispiel zeigt, wie Sie mit S3-Ereignisbenachrichtigungen auf objektorientierte Weise arbeiten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie ein S3-Benachrichtigungsereignis mithilfe von Amazon SQS verarbeiten.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
```

```
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",
```

```
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
```

```
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns;
```

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
```

```
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
```

```
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);
```

```
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
```

```
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
```

```
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
```

```
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
```

```
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();
```

```
            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
```

```
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
```

```
                                        String message,
                                        String topicArn,
                                        String msgAttValue,
                                        String duplication,
                                        String groupId,
                                        String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
```

```
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
```

```
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
```

```
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Senden und Empfangen von Stapeln von Nachrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Amazon SQS-Warteschlange
- Senden Sie Stapel von Nachrichten an die Warteschlange.
- Empfangen Sie Stapel von Nachrichten aus der Warteschlange.
- Löschen Sie Stapel von Nachrichten aus der Warteschlange.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wie in den folgenden Beispielen gezeigt, können Sie Batch-Nachrichtenoperationen mit Amazon SQS auf zwei verschiedene Arten mit dem AWS SDK for Java 2.x durchführen.

SendRecvBatch.javaverwendet explizite Batch-Operationen. Sie erstellen Nachrichtenstapel manuell und rufen `sendMessageBatch()` und `deleteMessageBatch()` direkt auf. Sie bearbeiten auch Batch-Antworten, einschließlich fehlgeschlagener Nachrichten. Dieser Ansatz gibt Ihnen die volle Kontrolle über die Batchgröße und die Fehlerbehandlung. Es erfordert jedoch mehr Code, um die Logik der Batchverarbeitung zu verwalten.

SimpleProducerConsumer.javaverwendet die `SqsAsyncBatchManager` High-Level-Bibliothek für die automatische Batchverarbeitung von Anfragen. Sie führen individuelle Aufrufe von `sendMessage()` und `deleteMessage()` durch und verwenden dieselben Methodensignaturen wie der Standardclient. Das SDK puffert diese Aufrufe automatisch und sendet sie als Batch-

Operationen. Dieser Ansatz erfordert nur minimale Codeänderungen und bietet gleichzeitig Leistungsvorteile bei der Batchverarbeitung.

Verwenden Sie die explizite Batchverarbeitung, wenn Sie eine genaue Kontrolle über die Batchzusammensetzung und die Fehlerbehandlung benötigen. Verwenden Sie die automatisches Batchverarbeitung, wenn Sie die Leistung mit minimalen Codeänderungen optimieren möchten.

SendRecvBatch.java - Verwendet explizite Batch-Operationen mit Nachrichten.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
 sent.
     * The caller must inspect the Successful and Failed lists in the response and
     * resend any failed messages.
     *
     * @param queueUrl  The URL of the queue to receive the messages.
     * @param messages  The messages to send to the queue. Each message contains a
 body and attributes.
     * @return The response from SQS that contains the list of successful and failed
 messages.
     */
    public static SendMessageBatchResponse sendMessages(
            String queueUrl, List<MessageEntry> messages) {

        try {
            List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                MessageEntry message = messages.get(i);
                entries.add(SendMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .messageBody(message.getBody())
                        .messageAttributes(message.getAttributes())
                        .build());
```

```
            }

            SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

            if (!response.successful().isEmpty()) {
                for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                    LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
                }
            }

            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10 by
SQS).
     *                   The actual number of messages received might be less.
```

```
    * @param waitTime   The maximum time to wait (in seconds) before returning.
When
    *                  this number is greater than zero, long polling is used.
This
    *                  can result in reduced costs and fewer false empty
responses.
    * @return The list of Message objects received. These each contain the body
    *         of the message and metadata and custom attributes.
    */
   public static List<Message> receiveMessages(String queueUrl, int maxNumber, int
waitTime) {
       try {
           ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
                   .queueUrl(queueUrl)
                   .maxNumberOfMessages(maxNumber)
                   .waitTimeSeconds(waitTime)
                   .messageAttributeNames("All")
                   .build();

           List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

           for (Message message : messages) {
               LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
           }

           return messages;

       } catch (SqsException e) {
           LOGGER.error("Couldn't receive messages from queue: {}", queueUrl, e);
           throw e;
       }
   }

   /**
    * Delete a batch of messages from a queue in a single request.
    *
    * @param queueUrl  The URL of the queue from which to delete the messages.
    * @param messages  The list of messages to delete.
    * @return The response from SQS that contains the list of successful and failed
    *         message deletions.
    */
```

```
   public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
       try {
           List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               entries.add(DeleteMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .receiptHandle(messages.get(i).receiptHandle())
                       .build());
           }

           DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

           if (!response.successful().isEmpty()) {
               for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
               }
           }

           if (!response.failed().isEmpty()) {
               for (BatchResultErrorEntry errorEntry : response.failed()) {
                   LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
               }
           }

           return response;

       } catch (SqsException e) {
           LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
           throw e;
       }
   }
```

```
    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }

    /**
     * Shows how to:
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
     * * Reassemble the lines of the file and verify they match the original file.
     */
    public static void usageDemo() {
        LOGGER.info("-".repeat(88));
        LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
        LOGGER.info("-".repeat(88));

        String queueUrl = null;
        try {
            // Create a queue for the demo.
            String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
            CreateQueueRequest createRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            queueUrl = sqsClient.createQueue(createRequest).queueUrl();
```

```
            LOGGER.info("Created queue: {}", queueUrl);

            try (InputStream inputStream =
SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                 BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream))) {

                List<String> lines = reader.lines().toList();

                // Send file lines in batches.
                int batchSize = 10;
                LOGGER.info("Sending file lines in batches of {} as messages.",
batchSize);

                for (int i = 0; i < lines.size(); i += batchSize) {
                    List<MessageEntry> messageBatch = new ArrayList<>();

                    for (int j = i; j < Math.min(i + batchSize, lines.size()); j++)
{
                        String line = lines.get(j);
                        if (line == null || line.trim().isEmpty()) {
                            continue; // Skip empty lines.
                        }

                        Map<String, MessageAttributeValue> attributes = new
HashMap<>();
                        attributes.put("line", MessageAttributeValue.builder()
                                .dataType("String")
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
```

```
                LOGGER.info("Receiving, handling, and deleting messages in batches
of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");

                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
```

```
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
 DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
 }
```

SimpleProducerConsumer.java - Verwendet automatisches Stapeln von Nachrichten.

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;
```

```
import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for Amazon
 SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
 * - Automatic batching: The SDK automatically buffers individual requests and sends
 them
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-blocking
 execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better connection
 reuse
 *
 * This example compares:
 * 1. Single-message operations using SqsAsyncClient directly
```

```
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
 LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
     * Prompts for queue name, thread counts, batch size, message size, and runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();
```

```
        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl, batchSize,
                        messageSizeByte, producedCount, stop);
            }
            producers[i].start();
        }

        // Start the consumers.
```

```
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl, consumedCount,
stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl, batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }

        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }

    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
```

```
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS, providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
```

```
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends messages until the stop flag is set.
         *
         * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call per
message.
         * This approach provides baseline performance metrics for comparison with
batching.
         * Each call blocks until the individual message is sent, demonstrating
traditional
         * one-request-per-operation behavior.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Sends messages using SqsAsyncBatchManager for automatic request batching and
optimization.
     *
     * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
     * The SqsAsyncBatchManager automatically buffers individual sendMessage() calls
and
     * sends them as batches when thresholds are reached:
     * - maxBatchSize: Maximum 10 messages per batch (default)
```

```
    * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
    *
    * Key advantages of the batching approach:
    * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
    * - Automatic optimization: No code changes needed to benefit from batching
    * - Transparent buffering: The SDK handles batching logic internally
    * - Reduced API calls: Multiple messages sent in single batch requests
    * - Lower costs: Fewer API calls result in reduced SQS charges
    * - Higher throughput: Batch operations process significantly more messages per
second
    */
   private static class BatchProducer extends Thread {
       final SqsAsyncBatchManager batchManager;
       final String queueUrl;
       final int batchSize;
       final AtomicInteger producedCount;
       final AtomicBoolean stop;
       final String theMessage;

       /**
        * Creates a producer thread for batch operations.
        *
        * @param batchManager the batch manager for efficient message sending
        * @param queueUrl the URL of the target queue
        * @param batchSize the number of messages to send per batch
        * @param messageSizeByte the size of messages to generate
        * @param producedCount shared counter for tracking sent messages
        * @param stop shared flag to signal thread termination
        */
       BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                     int messageSizeByte, AtomicInteger producedCount,
                     AtomicBoolean stop) {
           this.batchManager = batchManager;
           this.queueUrl = queueUrl;
           this.batchSize = batchSize;
           this.producedCount = producedCount;
           this.stop = stop;
           this.theMessage = makeRandomString(messageSizeByte);
       }

       /**
```

```
        * Continuously sends batches of messages using the high-level batching
library.
        *
        * Notice how batchManager.sendMessage() uses the exact same method
signature
        * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
        * the drop-in replacement capability of the SqsAsyncBatchManager.
        *
        * The SDK automatically:
        * - Buffers individual sendMessage() calls internally
        * - Groups them into batch requests when thresholds are met
        * - Sends SendMessageBatchRequest operations to SQS
        * - Returns individual CompletableFuture responses for each message
        *
        * This transparent batching provides significant performance improvements
        * without requiring changes to application logic or error handling
patterns.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   // Send multiple messages using the high-level batch manager.
                   // Each batchManager.sendMessage() call uses identical syntax to
                   // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                   // The SDK automatically buffers these calls and sends them as
                   // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                   // thresholds are reached, significantly improving throughput.
                   for (int i = 0; i < batchSize; i++) {
                       CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                                SendMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .messageBody(theMessage)
                                        .build());

                       // Handle the response asynchronously
                       future.whenComplete((response, throwable) -> {
                           if (throwable == null) {
                               producedCount.incrementAndGet();
                           } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
```

```
                                        !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchProducer: Failed to send message",
throwable);
                            }
                            // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                        });
                    }

                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Receives and deletes messages individually using SqsAsyncClient for baseline
measurement.
     *
     * This consumer demonstrates traditional single-message operations without
batching.
     * Each receiveMessage() and deleteMessage() call results in separate API
requests,
     * providing a performance baseline for comparison with batch operations.
     *
     * The method signatures are identical to SqsAsyncBatchManager methods:
     * - receiveMessage() matches batchManager.receiveMessage()
     * - deleteMessage() matches batchManager.deleteMessage()
     *
     * This API consistency allows easy migration to the high-level batching
library.
     */
    private static class Consumer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger consumedCount;
```

```
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for receiving messages
         * @param queueUrl the URL of the source queue
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
 consumedCount,
                 AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using traditional single-
request operations.
         *
         * Uses SqsAsyncClient methods directly:
         * - receiveMessage(): One API call per receive operation
         * - deleteMessage(): One API call per delete operation
         *
         * This approach demonstrates the baseline performance without batching
 optimization.
         * Compare these method calls with the identical signatures used in
 BatchConsumer
         * to see how the high-level batching library maintains API compatibility.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    try {
                        final ReceiveMessageResponse result =
 sqsAsyncClient.receiveMessage(
                                ReceiveMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .build()).join();

                        if (!result.messages().isEmpty()) {
```

```
                            final Message m = result.messages().get(0);
                            // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .receiptHandle(m.receiptHandle())
                                    .build()).join();
                            consumedCount.incrementAndGet();
                        }
                    } catch (SdkException | java.util.concurrent.CompletionException
e) {
                        log.error(e.getMessage());
                    }
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Consumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
     *
     * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
     * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
     *
     * 1. Receive optimization: Maintains an internal buffer of messages fetched in
the
     *    background, so receiveMessage() calls return immediately from the buffer
     * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
     *    as DeleteMessageBatchRequest operations when thresholds are reached
     *
     * Key features:
     * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
```

```
     *   as SqsAsyncClient methods, making this a true drop-in replacement
     * - Background fetching: The batch manager continuously fetches messages to
keep
     *   the internal buffer populated, reducing receive latency
     * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
     *   sent as batch operations (up to 10 per batch, 200ms frequency)
     * - Transparent optimization: No application logic changes needed to benefit
     *
     * Performance benefits:
     * - Reduced API calls through automatic batching of delete operations
     * - Lower latency for receives due to background message buffering
     * - Higher overall throughput with fewer network round trips
     */
    private static class BatchConsumer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger consumedCount;
        final AtomicBoolean stop;

        /**
         * Creates a consumer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message processing
         * @param queueUrl the URL of the source queue
         * @param batchSize the maximum number of messages to receive per batch
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level batching
library.
         *
```

```
        * Demonstrates the key advantage of SqsAsyncBatchManager: identical method
signatures
        * with automatic optimization. Notice how:
        *
        * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
        * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
        *
        * Behind the scenes, the batch manager:
        * 1. Maintains an internal message buffer populated by background fetching
        * 2. Returns messages immediately from the buffer (reduced latency)
        * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
        * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
        *
        * This provides significant performance improvements with zero code changes
        * compared to traditional SqsAsyncClient usage patterns.
        */
       public void run() {
           try {
               while (!stop.get()) {
                   // Receive messages using the high-level batch manager.
                   // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
                   // but benefits from internal message buffering for improved
performance.
                   final ReceiveMessageResponse result =
batchManager.receiveMessage(
                           ReceiveMessageRequest.builder()
                                   .queueUrl(queueUrl)
                                   .maxNumberOfMessages(Math.min(batchSize, 10))
                                   .build()).join();

                   if (!result.messages().isEmpty()) {
                       final List<Message> messages = result.messages();

                       // Delete messages using the batch manager.
                       // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                       // but the SDK automatically buffers these calls and sends
them
```

```
                        // as DeleteMessageBatchRequest operations for optimal
performance.
                        for (Message message : messages) {
                            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                    DeleteMessageRequest.builder()
                                            .queueUrl(queueUrl)
                                            .receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                          !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
                    }

                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
```

```
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Logs throughput statistics every second until stopped.
         *
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Verwenden der Amazon SQS Java Messaging Library, um mit der JMS-Schnittstelle zu arbeiten

Das folgende Codebeispiel zeigt, wie Sie die Amazon SQS Java Messaging Library verwenden, um mit der JMS-Schnittstelle zu arbeiten.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die folgenden Beispiele funktionieren mit standardmäßigen Amazon-SQS-Warteschlangen und beinhalten:

- Senden einer Testnachricht
- Synchrones Empfangen von Nachrichten
- Asynchrones Empfangen von Nachrichten
- Empfangen von Nachrichten im CLIENT_ACKNOWLEDGE-Modus.
- Empfangen von Nachrichten im UNORDERED_ACKNOWLEDGE-Modus.
- Verwenden von Spring, um Abhängigkeiten einzufügen.
- Eine Dienstprogramm-Klasse, die allgemeine Methoden bereitstellt, die in den anderen Beispielen verwendet werden.

Weitere Informationen zur Verwendung von JMS mit Amazon SQS finden Sie im Entwicklerhandbuch für Amazon SQS.

Senden einer Testnachricht

```
   /**
    * This method establishes a connection to a standard Amazon SQS queue using the
Amazon SQS
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery, and
automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending messages
to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection()) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session that uses the JMS auto-acknowledge mode.
           Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

           createAndSendMessages(session, producer);
       } // The connection closes automatically. This also closes the session.
       LOGGER.info("Connection closed");
   }

   /**
```

```
    * This method reads text input from the keyboard and sends each line as a
separate message
    * to a standard Amazon SQS queue using the Amazon SQS Java Messaging Library.
It continues
    * to accept input until the user enters an empty line, using JMS (Java Message
Service) API to
    * handle the message delivery.
    *
    * @param session The JMS session used to create messages
    * @param producer The JMS message producer used to send messages to the queue
    */
   private static void createAndSendMessages(Session session, MessageProducer
producer) {
       BufferedReader inputReader = new BufferedReader(
               new InputStreamReader(System.in, Charset.defaultCharset()));

       try {
           String input;
           while (true) {
               LOGGER.info("Enter message to send (leave empty to exit): ");
               input = inputReader.readLine();
               if (input == null || input.isEmpty()) break;

               TextMessage message = session.createTextMessage(input);
               producer.send(message);
               LOGGER.info("Send message {}", message.getJMSMessageID());
           }
       } catch (EOFException e) {
           // Just return on EOF
       } catch (IOException e) {
           LOGGER.error("Failed reading input: {}", e.getMessage(), e);
       } catch (JMSException e) {
           LOGGER.error("Failed sending message: {}", e.getMessage(), e);
       }
   }
```

Synchrones Empfangen von Nachrichten

```
   /**
    * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
```

```
    * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
    * waits for messages to arrive, and processes them one at a time. The method
handles all
    * necessary setup and cleanup of messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
   public static void doReceiveMessageSync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           connection.start();

           receiveMessages(consumer);
       }  // The connection closes automatically. This also closes the session.
       LOGGER.info("Connection closed");
   }

   /**
    * This method continuously checks for new messages from a standard Amazon SQS
queue using
    * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
    * it using JMS (Java Message Service), and confirms receipt. The method stops
checking for
    * messages after 20 seconds of no activity.
```

```
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}", message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

Asynchrones Empfangen von Nachrichten

```
    /**
     * This method sets up automatic message handling for a standard Amazon SQS
queue using the
     * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
     * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
     * messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageAsync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
```

```
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            try {
                // Create a consumer for the queue.
                MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
                // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
                // We use a lambda expression for the implementation.
                consumer.setMessageListener(message -> {
                    try {
                        SqsJmsExampleUtils.handleMessage(message);
                        message.acknowledge();
                    } catch (JMSException e) {
                        LOGGER.error("Error processing message: {}",
e.getMessage());
                    }
                });
                // Start receiving incoming messages.
                connection.start();
                LOGGER.info("Waiting for messages...");
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

Empfangen von Nachrichten im CLIENT_ACKNOWLEDGE-Modus.

```
   /**
    * This method demonstrates how message acknowledgment affects message
processing in a standard
    * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
    * then shows how JMS (Java Message Service) client acknowledgment mode handles
both explicit
    * and implicit message confirmations, including how acknowledging one message
can automatically
    * acknowledge previous messages.
    *
    * @throws JMSException If there is a problem with the messaging operations
    */
   public static void doReceiveMessagesSyncClientAcknowledge() throws JMSException
{
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with client acknowledge mode.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           // Create a producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
```

```
            // Open the connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the second
one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
```

```
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based on
the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from the
queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after processing
     * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Empfangen von Nachrichten im UNORDERED_ACKNOWLEDGE-Modus.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message acknowledges
all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try( SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

           // Create a session with unordered acknowledge mode.
           Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

           // Create the producer and consumer.
           MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
           MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

           // Open a connection.
```

```
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, and we'll get the first
message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its own
separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
```

```
    private static void sendMessage(MessageProducer producer, Session session,
 String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
 acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
 print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
 message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Verwenden von Spring, um Abhängigkeiten einzufügen.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;
```

```
import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java Messaging
 Library
 * with Spring Framework integration. This example connects to a standard Amazon SQS
 message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
     * Demonstrates sending and receiving messages through a standard Amazon SQS
 message queue
     * using Spring Framework configuration. This method loads connection settings
 from an XML file,
     * establishes a messaging session using the Amazon SQS Java Messaging Library,
 and processes
     * messages using JMS (Java Message Service) operations. If the queue doesn't
 exist, it will
     * be created automatically.
     *
     * @param args Command line arguments (not used)
     */
    public static void main(String[] args) {

        URL resource =
 SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
        File springFile = new File(resource.getFile());
        if (!springFile.exists() || !springFile.canRead()) {
            LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or isn't
 readable.");
            System.exit(1);
```

```
        }

        try (FileSystemXmlApplicationContext context =
                     new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

            Connection connection;
            try {
                connection = context.getBean(Connection.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
                System.exit(2);
                return;
            }

            String queueName;
            try {
                queueName = context.getBean("queueName", String.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
                System.exit(3);
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection) connection,
queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
```

```
        }   // Spring context autocloses. Managed Spring beans that implement
 AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
 message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
 method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
 (Java Message
     * Service) operations. The method will stop checking for messages after 15
 seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
 queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Spring-Bean-Definitionen.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder" class="software.amazon.awssdk.services.sqs.SqsClient"
 factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>
```

```
    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

Eine Dienstprogramm-Klasse, die allgemeine Methoden bereitstellt, die in den anderen Beispielen verwendet werden.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple Queue
 Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common operations
 for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
```

```
   private static final Logger LOGGER =
LoggerFactory.getLogger(SqsJmsExampleUtils.class);
   public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

   /**
    * This method verifies that a message queue exists and creates it if necessary.
The method checks for
    * an existing queue first to optimize performance.
    *
    * @param connection The active connection to the messaging service
    * @param queueName The name of the queue to verify or create
    * @param visibilityTimeout The duration in seconds that messages will be hidden
after being received
    * @throws JMSException If there is an error accessing or creating the queue
    */
   public static void ensureQueueExists(SQSConnection connection, String queueName,
Long visibilityTimeout) throws JMSException {
       AmazonSQSMessagingClientWrapper client =
connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where the
queue
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
```

```
     * @param session The active messaging session used to create and send the
message
     * @param queueName The name of the queue where the message will be sent
     */
    public static void sendTextMessage(Session session, String queueName) {
        // Rest of implementation...

        try {
            MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
            Message message = session.createTextMessage("Hello world!");
            producer.send(message);
        } catch (JMSException e) {
            LOGGER.error( "Error receiving from SQS", e );
        }
    }

    /**
     * This method processes incoming messages and logs their content based on the
message type.
     * It supports text messages, binary data, and Java objects.
     *
     * @param message The message to be processed and logged
     * @throws JMSException If there is an error reading the message content
     */
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to 256k
so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }
```

```
    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
    public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
        // Rest of implementation...
        try {
            // Create a consumer for the queue.
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
            // Provide an implementation of the MessageListener interface, which has
a single 'onMessage' method.
            // We use a lambda expression for the implementation.
            consumer.setMessageListener(message -> {
                try {
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSException e) {
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * This method performs cleanup operations after message processing is complete.
It receives
```

```
     * any messages in the specified queue, removes the message queue and closes all
     * active connections to prevent resource leaks.
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);

            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
```

```
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain hidden
 after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long secondsToWait,
 Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
 connectionFactory.createConnection()) {
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " + i +
 "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateQueue
  - DeleteQueue

Arbeiten Sie mit Warteschlangen-Tags

Das folgende Codebeispiel zeigt, wie Sie Tagging-Operationen mit Amazon SQS durchführen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Im folgenden Beispiel werden Tags für eine Warteschlange erstellt, Tags aufgelistet und ein Tag entfernt.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
```

```
    static final String queueName = "TagExamples-queue-" +
UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER = LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
```

```
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a Consumer<UntagQueueRequest.Builder>
parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
```

```
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - ListQueueTags
  - TagQueue
  - UntagQueue

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSEvent.SQSMessage;

public class Function implements RequestHandler<SQSEvent, Void> {
    @Override
    public Void handleRequest(SQSEvent sqsEvent, Context context) {
        for (SQSMessage msg : sqsEvent.getRecords()) {
            processMessage(msg, context);
        }
        context.getLogger().log("done");
        return null;
    }

    private void processMessage(SQSMessage msg, Context context) {
        try {
            context.getLogger().log("Processed message " + msg.getBody());

            // TODO: Do interesting work based on the new message

        } catch (Exception e) {
            context.getLogger().log("An error occurred");
            throw e;
        }

    }
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
 SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {
         List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<SQSBatchResponse.BatchItemFailure>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

Beispiele für Step Functions unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Step Functions

Das folgende Codebeispiel zeigt, wie Sie mit Step Functions beginnen.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Java-Version von Hello.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListStateMachines in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 1. Creates an activity.
```

```
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <roleARN> <activityName> <stateMachineName>

            Where:
                roleName - The name of the IAM role to create for this state
 machine.
                activityName - The name of an activity to create.
                stateMachineName - The name of the state machine to create.
                jsonFile - The location of the chat_sfn_state_machine.json file. You
 can located it in resources/sample_files.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "states.amazonaws.com"
                        },
```

```
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
            """;


        Scanner sc = new Scanner(System.in);
        boolean action = false;

        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
```

```
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
 stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
 with it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " + runArn);
        List<String> myList;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
```

```
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();
```

```
            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
```

```
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());
```

```
        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();
```

```
            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt die Verwendung`CreateActivity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateActivity in der AWS SDK for Java 2.x API-Referenz.

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`CreateStateMachine`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
```

```
                .build();

            CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie CreateStateMachine in der AWS SDK for Java 2.x API-Referenz.

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteActivity`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

- API-Details finden Sie DeleteActivity in der AWS SDK for Java 2.x API-Referenz.

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStateMachine`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
```

```
        System.out.println(stateMachineArn + " was successfully deleted.");
    }
```

- API-Details finden Sie DeleteStateMachine in der AWS SDK for Java 2.x API-Referenz.

`DescribeExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribeExecution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
```

```
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie DescribeExecution in der AWS SDK for Java 2.x API-Referenz.

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStateMachine`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
       try {
           DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
               .stateMachineArn(stateMachineArn)
               .build();

           DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
           System.out.println("The name of the State machine is " +
response.name());
           System.out.println("The status of the State machine is " +
response.status());
           System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
           System.out.println("The role ARN value is " + response.roleArn());
```

```
        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }
```

- API-Details finden Sie DescribeStateMachine in der AWS SDK for Java 2.x API-Referenz.

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung`GetActivityTask`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
    }
```

- API-Details finden Sie GetActivityTask in der AWS SDK for Java 2.x API-Referenz.

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung`ListActivities`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListActivities in der AWS SDK for Java 2.x API-Referenz.

`ListExecutions`

Das folgende Codebeispiel zeigt die Verwendung`ListExecutions`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void getExeHistory(SfnClient sfnClient, String exeARN) {
        try {
            GetExecutionHistoryRequest historyRequest =
 GetExecutionHistoryRequest.builder()
                    .executionArn(exeARN)
                    .maxResults(10)
                    .build();

            GetExecutionHistoryResponse historyResponse =
 sfnClient.getExecutionHistory(historyRequest);
            List<HistoryEvent> events = historyResponse.events();
```

```
            for (HistoryEvent event : events) {
                System.out.println("The event type is " + event.type().toString());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ListExecutions in der AWS SDK for Java 2.x API-Referenz.

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung`ListStateMachines`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
```

```
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
 machine.name());
                System.out.println("The ARN value is : " +
 machine.stateMachineArn());
            }

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie ListStateMachines in der AWS SDK for Java 2.x API-Referenz.

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung`SendTaskSuccess`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
 json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie SendTaskSuccess in der AWS SDK for Java 2.x API-Referenz.

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
 String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();
```

```
            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie StartExecution in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Verwenden von Step Functions, um Lambda-Funktionen aufzurufen

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Zustandsmaschine erstellt wird, die AWS Lambda Funktionen nacheinander aufruft.

SDK für Java 2.x

Zeigt, wie ein AWS serverloser Workflow mithilfe von AWS Step Functions und erstellt wird. AWS SDK for Java 2.x Jeder Workflow-Schritt wird mithilfe einer AWS Lambda Funktion implementiert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SES
- Step-Funktionen

AWS STS Beispiele mit SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
```

```
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to assume
 (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
```

```
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
 roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " + exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie AssumeRole in der AWS SDK for Java 2.x API-Referenz.

Support Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren Support.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für Java 2.x

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.Service;
```

```
import software.amazon.awssdk.services.support.model.SupportException;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
                .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();
```

```
            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                }
                index++;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeServices in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.

- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie verschiedene Support Operationen aus.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
```

```
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS
 * Support Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and
 * severity level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """
```

```
               Usage:
                   <fileAttachment>Where:
                   fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
               """;

     //  if (args.length != 1) {
     //      System.out.println(usage);
     //      System.exit(1);
     //  }

       String fileAttachment = "C:\\AWS\\test.txt" ; //args[0];
       Region region = Region.US_WEST_2;
       SupportClient supportClient = SupportClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("***** Welcome to the AWS Support case example
scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Get and display available services.");
       List<String> sevCatList = displayServices(supportClient);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("2. Get and display Support severity levels.");
       String sevLevel = displaySevLevels(supportClient);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("3. Create a support case using the selected service,
category, and severity level.");
       String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
       if (caseId.compareTo("") == 0) {
           System.out.println("A support case was not successfully created!");
           System.exit(1);
       } else
           System.out.println("Support case " + caseId + " was successfully
created!");
       System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
 to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
 case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
 communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
```

```
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .includeResolvedCases(true)
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
```

```
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to
an AWS Support case");
            else
                System.out.println("There was an error adding the communication to
an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                    .fileName(myFile.getName())
```

```
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
```

```
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
```

```
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt die Verwendung`AddAttachmentsToSet`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
       try {
           File myFile = new File(fileAttachment);
           InputStream sourceStream = new FileInputStream(myFile);
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

           Attachment attachment = Attachment.builder()
                   .fileName(myFile.getName())
                   .data(sourceBytes)
                   .build();

           AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                   .attachments(attachment)
                   .build();

           AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
           return response.attachmentSetId();

       } catch (SupportException | FileNotFoundException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- API-Details finden Sie AddAttachmentsToSet in der AWS SDK for Java 2.x API-Referenz.

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`AddCommunicationToCase`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
       try {
           AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                   .caseId(caseId)
                   .attachmentSetId(attachmentSetId)
                   .communicationBody("Please refer to attachment for details.")
                   .build();

           AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
           if (response.result())
               System.out.println("You have successfully added a communication to
an AWS Support case");
           else
               System.out.println("There was an error adding the communication to
an AWS Support case");

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie AddCommunicationToCase in der AWS SDK for Java 2.x API-Referenz.

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung`CreateCase`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
       try {
           String serviceCode = sevCatList.get(0);
           String caseCat = sevCatList.get(1);
           CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                   .categoryCode(caseCat.toLowerCase())
                   .serviceCode(serviceCode.toLowerCase())
                   .severityCode(sevLevel.toLowerCase())
                   .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                   .subject("Test case, please ignore")
                   .language("en")
                   .issueType("technical")
                   .build();

           CreateCaseResponse response = supportClient.createCase(caseRequest);
           return response.caseId();

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- API-Details finden Sie CreateCase in der AWS SDK for Java 2.x API-Referenz.

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAttachment`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void describeAttachment(SupportClient supportClient, String
attachId) {
       try {
           DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                   .attachmentId(attachId)
                   .build();

           DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
           System.out.println("The name of the file is " +
response.attachment().fileName());

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeAttachment in der AWS SDK for Java 2.x API-Referenz.

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCases`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static void getOpenCase(SupportClient supportClient) {
       try {
           // Specify the start and end time.
           Instant now = Instant.now();
           java.time.LocalDate.now();
           Instant yesterday = now.minus(1, ChronoUnit.DAYS);

           DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                   .maxResults(20)
                   .afterTime(yesterday.toString())
                   .beforeTime(now.toString())
                   .build();

           DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
           List<CaseDetails> cases = response.cases();
           for (CaseDetails sinCase : cases) {
               System.out.println("The case status is " + sinCase.status());
               System.out.println("The case Id is " + sinCase.caseId());
               System.out.println("The case subject is " + sinCase.subject());
           }

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- API-Details finden Sie DescribeCases in der AWS SDK for Java 2.x API-Referenz.

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCommunications`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public static String listCommunications(SupportClient supportClient, String
caseId) {
       try {
           String attachId = null;
           DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                   .caseId(caseId)
                   .maxResults(10)
                   .build();

           DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
           List<Communication> communications = response.communications();
           for (Communication comm : communications) {
               System.out.println("the body is: " + comm.body());

               // Get the attachment id value.
               List<AttachmentDetails> attachments = comm.attachmentSet();
               for (AttachmentDetails detail : attachments) {
                   attachId = detail.attachmentId();
               }
           }
           return attachId;

       } catch (SupportException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
   }
```

- API-Details finden Sie DescribeCommunications in der AWS SDK for Java 2.x API-Referenz.

`DescribeServices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeServices`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
```

```
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
```

- API-Details finden Sie DescribeServices in der AWS SDK for Java 2.x API-Referenz.

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSeverityLevels`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();
```

```
            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- API-Details finden Sie DescribeSeverityLevels in der AWS SDK for Java 2.x API-Referenz.

`ResolveCase`

Das folgende Codebeispiel zeigt die Verwendung`ResolveCase`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();
```

```
            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- API-Details finden Sie ResolveCase in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Systems Manager unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Systems Manager

Das folgende Codebeispiel zeigt die ersten Schritte mit Systems Manager.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.DocumentFilter;
import software.amazon.awssdk.services.ssm.model.ListDocumentsRequest;
import software.amazon.awssdk.services.ssm.model.ListDocumentsResponse;

public class HelloSSM {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <awsAccount>

                Where:
                    awsAccount - Your AWS Account number.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String awsAccount = args[0] ;
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
            .region(region)
            .build();

        listDocuments(ssmClient, awsAccount);
    }

    /*
```

```
    This code automatically fetches the next set of results using the `nextToken`
 and
    stops once the desired maxResults (20 in this case) have been reached.
    */
    public static void listDocuments(SsmClient ssmClient, String awsAccount) {
        String nextToken = null;
        int totalDocumentsReturned = 0;
        int maxResults = 20;
        do {
            ListDocumentsRequest request = ListDocumentsRequest.builder()
                .documentFilterList(
                    DocumentFilter.builder()
                        .key("Owner")
                        .value(awsAccount)
                        .build()
                    )
                .maxResults(maxResults)
                .nextToken(nextToken)
                .build();

            ListDocumentsResponse response = ssmClient.listDocuments(request);
            response.documentIdentifiers().forEach(identifier ->
 System.out.println("Document Name: " + identifier.name()));
            nextToken = response.nextToken();
            totalDocumentsReturned += response.documentIdentifiers().size();
        } while (nextToken != null && totalDocumentsReturned < maxResults);
    }
}
```

- API-Details finden Sie ListDocuments in der AWS SDK for Java 2.x API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Wartungsfenster.
- Ändern Sie den Zeitplan des Wartungsfensters.
- Erstellen Sie ein Dokument.
- Senden Sie einen Befehl an eine bestimmte EC2-Instance.

- Erstellen Sie eine OpsItem.
- Aktualisieren und lösen Sie das OpsItem.
- Löschen Sie das Wartungsfenster OpsItem und das Dokument.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "SsmStack3`1";

    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
                category - The category of the parameter. Valid values are
 'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
 Performance).
                severity - The severity of the parameter. Severity should be a
 number from 1 to 4 (default is 2).
        """;

        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;
```

```
        System.out.println("Use AWS CloudFormation to create the EC2 instance that
is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS applications
and resources and a secure end-to-end management solution.
                The program's primary functionalities include creating a maintenance
window, creating a document, sending a command to a document,
                listing documents, listing commands, creating an OpsItem, modifying
an OpsItem, and deleting AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
```

```
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions that
Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
```

```
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: " +
commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was sent
to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

```
            6. Now we will create an SSM OpsItem.
            A SSM OpsItem is a feature provided by Amazon's Systems Manager (SSM)
service.
            It is a type of operational data item that allows you to manage and
track various operational issues,
            events, or tasks within your AWS environment.

            You can create OpsItems to track and manage operational issues as they
arise.
            For example, you could create an OpsItem whenever your application
detects a critical error
            or an anomaly in your infrastructure.
           """);

       waitForInputToContinue(scanner);
       String opsItemId;
       try {
           opsItemId = actions.createSSMOpsItem(title, source, category, severity);
           System.out.println(opsItemId + " was created");
       } catch (SsmException e) {
           System.err.println("SSM error: " + e.getMessage());
           return;
       } catch (RuntimeException e) {
           System.err.println("Unexpected error: " + e.getMessage());
           return;
       }
       waitForInputToContinue(scanner);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
       waitForInputToContinue(scanner);
       String description = "An update to "+opsItemId ;
       try {
           actions.updateOpsItem(opsItemId, title, description);
       } catch (SsmException e) {
           System.err.println("SSM error: " + e.getMessage());
           return;
       } catch (RuntimeException e) {
           System.err.println("Unexpected error: " + e.getMessage());
           return;
       }

       System.out.println(DASHES);
```

```
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
```

```
            System.out.println("The AWS Systems Manager resources will not be
 deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
 scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Eine Wrapper-Klasse für Systems-Manager-SDK-Methoden.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();
```

```
            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
```

```
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
```

```
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
```

```
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                        for (OpsItemSummary item : items) {
```

```
                            System.out.println("The item title is " + item.title() +
" and the status is " + item.status().toString());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description) {
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
        operationalData.put("key1",
OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
OpsItemDataValue.builder().value("value2").build());
```

```
        CompletableFuture<Void> future = getOpsItem(opsItemId).thenCompose(opsItem -
> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response -> {
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
 ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
 GetOpsItemRequest.builder().opsItemId(opsItemId).build();
        return
 getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }
```

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
     * Displays the date and time when the specific command was invoked.
     *
     * @param commandId The ID of the command to describe.
     * <p>
```

```
    * This method initiates an asynchronous request to list command invocations and
prints the date and time of each command invocation.
    * If an exception occurs, it handles the error appropriately.
    */
   public void displayCommands(String commandId) {
       ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
               .commandId(commandId)
               .build();

       CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
       future.thenAccept(response -> {
           List<CommandInvocation> commandList = response.commandInvocations();
           DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
           for (CommandInvocation invocation : commandList) {
               System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
           }
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw (SsmException) cause;
           } else {
               throw new RuntimeException(cause);
           }
       }).join();
   }

   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
```

```
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
       CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
       final String[] commandId = {null};

       commandFuture.whenComplete((commandResponse, ex) -> {
```

```
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
```

```
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();
```

```
        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
```

```
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
                .name(winName)
                .description("This is my maintenance window")
                .allowUnassociatedTargets(true)
                .duration(2)
                .cutoff(1)
                .schedule("cron(0 10 ? * MON-FRI *)")
                .build();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
```

```
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
```

```
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Aktionen

`CreateDocument`

Das folgende Codebeispiel zeigt die Verwendung`CreateDocument`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an AWS SSM document asynchronously.
     *
```

```
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
```

```
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- API-Details finden Sie CreateDocument in der AWS SDK for Java 2.x API-Referenz.

`CreateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`CreateMaintenanceWindow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Creates an SSM maintenance window asynchronously.
    *
    * @param winName The name of the maintenance window.
    * @return The ID of the created or existing maintenance window.
    * <p>
    * This method initiates an asynchronous request to create an SSM maintenance
window.
    * If the request is successful, it prints the maintenance window ID.
    * If an exception occurs, it handles the error appropriately.
    */
   public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
       CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
               .name(winName)
               .description("This is my maintenance window")
               .allowUnassociatedTargets(true)
               .duration(2)
               .cutoff(1)
               .schedule("cron(0 10 ? * MON-FRI *)")
               .build();
```

```
        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
```

```
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- API-Details finden Sie CreateMaintenanceWindow in der AWS SDK for Java 2.x API-Referenz.

`CreateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`CreateOpsItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
```

```
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- API-Details finden Sie CreateOpsItem in der AWS SDK for Java 2.x API-Referenz.

`DeleteDocument`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDocument`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API-Details finden Sie DeleteDocument in der AWS SDK for Java 2.x API-Referenz.

`DeleteMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMaintenanceWindow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * Deletes an AWS SSM Maintenance Window asynchronously.
    *
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was successfully
deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
```

```
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API-Details finden Sie DeleteMaintenanceWindow in der AWS SDK for Java 2.x API-Referenz.

`DescribeOpsItems`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOpsItems`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
```

```
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                        for (OpsItemSummary item : items) {
                            System.out.println("The item title is " + item.title() +
" and the status is " + item.status().toString());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API-Details finden Sie DescribeOpsItems in der AWS SDK for Java 2.x API-Referenz.

`DescribeParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeParameters`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.GetParameterRequest;
import software.amazon.awssdk.services.ssm.model.GetParameterResponse;
import software.amazon.awssdk.services.ssm.model.SsmException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetParameter {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String paraName = args[0];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        getParaValue(ssmClient, paraName);
        ssmClient.close();
    }

    public static void getParaValue(SsmClient ssmClient, String paraName) {
        try {
            GetParameterRequest parameterRequest = GetParameterRequest.builder()
                    .name(paraName)
                    .build();

            GetParameterResponse parameterResponse =
 ssmClient.getParameter(parameterRequest);
            System.out.println("The parameter value is " +
 parameterResponse.parameter().value());

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DescribeParameters in der AWS SDK for Java 2.x API-Referenz.

`PutParameter`

Das folgende Codebeispiel zeigt die Verwendung`PutParameter`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ssm.SsmClient;
import software.amazon.awssdk.services.ssm.model.ParameterType;
import software.amazon.awssdk.services.ssm.model.PutParameterRequest;
import software.amazon.awssdk.services.ssm.model.SsmException;

public class PutParameter {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <paraName>

                Where:
                    paraName - The name of the parameter.
                    paraValue - The value of the parameter.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String paraName = args[0];
        String paraValue = args[1];
        Region region = Region.US_EAST_1;
        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        putParaValue(ssmClient, paraName, paraValue);
        ssmClient.close();
    }
```

```
    public static void putParaValue(SsmClient ssmClient, String paraName, String
 value) {
        try {
            PutParameterRequest parameterRequest = PutParameterRequest.builder()
                    .name(paraName)
                    .type(ParameterType.STRING)
                    .value(value)
                    .build();

            ssmClient.putParameter(parameterRequest);
            System.out.println("The parameter was successfully added.");

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie PutParameter in der AWS SDK for Java 2.x API-Referenz.

`SendCommand`

Das folgende Codebeispiel zeigt die Verwendung`SendCommand`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Sends a SSM command to a managed node asynchronously.
     *
     * @param documentName The name of the document to use.
     * @param instanceId The ID of the instance to send the command to.
     * @return The command ID.
     * <p>
```

```
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
```

```
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
```

```
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- API-Details finden Sie SendCommand in der AWS SDK for Java 2.x API-Referenz.

`UpdateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateMaintenanceWindow`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
```

```
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- API-Details finden Sie UpdateMaintenanceWindow in der AWS SDK for Java 2.x API-Referenz.

`UpdateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateOpsItem`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });
```

```
        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API-Details finden Sie UpdateOpsItem in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Textract unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Textract Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AnalyzeDocument`

Das folgende Codebeispiel zeigt die Verwendung`AnalyzeDocument`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentRequest;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.FeatureType;
import software.amazon.awssdk.services.textract.model.AnalyzeDocumentResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AnalyzeDocument {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
```

```
                    sourceDoc - The path where the document is located (must be an
image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        analyzeDoc(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            List<FeatureType> featureTypes = new ArrayList<FeatureType>();
            featureTypes.add(FeatureType.FORMS);
            featureTypes.add(FeatureType.TABLES);

            AnalyzeDocumentRequest analyzeDocumentRequest =
AnalyzeDocumentRequest.builder()
                    .featureTypes(featureTypes)
                    .document(myDoc)
                    .build();

            AnalyzeDocumentResponse analyzeDocument =
textractClient.analyzeDocument(analyzeDocumentRequest);
            List<Block> docInfo = analyzeDocument.blocks();
            Iterator<Block> blockIterator = docInfo.iterator();
```

```
            while (blockIterator.hasNext()) {
                Block block = blockIterator.next();
                System.out.println("The block type is " +
 block.blockType().toString());
            }

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie AnalyzeDocument in der AWS SDK for Java 2.x API-Referenz.

`DetectDocumentText`

Das folgende Codebeispiel zeigt die Verwendung`DetectDocumentText`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen Sie Text aus einem Eingabedokument.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
import software.amazon.awssdk.services.textract.model.TextractException;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentText {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sourceDoc>\s

                Where:
                    sourceDoc - The path where the document is located (must be an
 image, for example, C:/AWS/book.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceDoc = args[0];
        Region region = Region.US_EAST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocText(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Erkennen von Text aus einem Dokument, das sich in einem Amazon-S3-Bucket befindet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.Document;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextRequest;
import software.amazon.awssdk.services.textract.model.DetectDocumentTextResponse;
import software.amazon.awssdk.services.textract.model.Block;
import software.amazon.awssdk.services.textract.model.DocumentMetadata;
```

```
import software.amazon.awssdk.services.textract.model.TextractException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectDocumentTextS3 {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s

                    docName - The document name (must be an image, i.e., book.png).
\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        detectDocTextS3(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
 bucketName, String docName) {
```

```
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DetectDocumentText in der AWS SDK for Java 2.x API-Referenz.

StartDocumentAnalysis

Das folgende Codebeispiel zeigt die Verwendung`StartDocumentAnalysis`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
 document.\s
                    docName - The document name (must be an image, for example,
 book.png).\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();

            StartDocumentAnalysisRequest documentAnalysisRequest =
 StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
 textractClient.startDocumentAnalysis(documentAnalysisRequest);
```

```
            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String jobId)
{
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

- API-Details finden Sie StartDocumentAnalysis in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract

- Amazon Translate

Beispiele für Amazon Transcribe unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`ListTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListTranscriptionJobs`

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class ListTranscriptionJobs {
    public static void main(String[] args) {
        TranscribeClient transcribeClient = TranscribeClient.builder()
            .region(Region.US_EAST_1)
```

```
            .build();

            listTranscriptionJobs(transcribeClient);
        }

        public static void listTranscriptionJobs(TranscribeClient transcribeClient)
{
            ListTranscriptionJobsRequest listJobsRequest =
ListTranscriptionJobsRequest.builder()
                .build();


transcribeClient.listTranscriptionJobsPaginator(listJobsRequest).stream()
                .flatMap(response -> response.transcriptionJobSummaries().stream())
                .forEach(jobSummary -> {
                    System.out.println("Job Name: " +
jobSummary.transcriptionJobName());
                    System.out.println("Job Status: " +
jobSummary.transcriptionJobStatus());
                    System.out.println("Output Location: " +
jobSummary.outputLocationType());
                    // Add more information as needed

                    // Retrieve additional details for the job if necessary
                    GetTranscriptionJobResponse jobDetails =
transcribeClient.getTranscriptionJob(
                        GetTranscriptionJobRequest.builder()
                            .transcriptionJobName(jobSummary.transcriptionJobName())
                            .build());

                    // Display additional details
                    System.out.println("Language Code: " +
jobDetails.transcriptionJob().languageCode());
                    System.out.println("Media Format: " +
jobDetails.transcriptionJob().mediaFormat());
                    // Add more details as needed

                    System.out.println("--------------");
                });
        }
    }
```

- API-Details finden Sie ListTranscriptionJobs in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Audio transkribieren und Auftragsdaten abrufen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen Transkriptionsauftrag mit Amazon Transcribe.
- Warten Sie, bis der -Auftrag abgeschlossen wurde.
- Ermitteln Sie die URI, unter der das Transkript gespeichert ist.

Weitere Informationen finden Sie unter Erste Schritte mit Amazon Transcribe.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Transkribiert eine PCM-Datei.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
```

```
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
```

```
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
```

```
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }
```

```
        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

Transkribiert Streaming-Audio vom Mikrofon Ihres Computers.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
```

```
            throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
```

```
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
```

```
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
```

```
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
  - GetTranscriptionJob
  - StartTranscriptionJob

Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Streaming Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK for Java 2.x

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`StartMedicalStreamTranscription`

Das folgende Codebeispiel zeigt die Verwendung. `StartMedicalStreamTranscription`

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
To run this AWS code example, ensure that you have set up your development
 environment, including your AWS credentials.

 For information, see this documentation topic:

 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

This code demonstrates the process of starting a medical transcription job using the
 AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
```

```
            throw new LineUnavailableException("The audio system microphone line is
not supported.");
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartMedicalStreamTranscriptionRequest getMedicalRequest(Integer
mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
```

```
                List<MedicalResult> results = ((MedicalTranscriptEvent)
 event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
 {
            this.subscriber = s;
```

```
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
```

```
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                .audioChunk(SdkBytes.fromByteBuffer(bb))
                .build();
        }
    }
}
```

- API-Details finden Sie StartMedicalStreamTranscription in der AWS SDK for Java 2.x API-Referenz.

`StartStreamTranscription`

Das folgende Codebeispiel zeigt die Verwendung`StartStreamTranscription`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
```

```
            throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
```

```
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
```

```
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
```

```
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK for Java 2.x API-Referenz.

Szenarien

Transkribieren einer Audiodatei

Das folgende Code-Beispiel zeigt, wie mit Amazon Transcribe-Streaming eine Transkription einer Quell-Audiodatei generiert wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * To run this AWS code example, ensure that you have set up your development
 * environment, including your AWS credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class TranscribeStreamingDemoFile {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[]) throws ExecutionException,
 InterruptedException {

        final String USAGE = "\n" +
                "Usage:\n" +
                "    <file> \n\n" +
                "Where:\n" +
                "    file - the location of a PCM file to transcribe. In this
 example, ensure the PCM file is 16 hertz (Hz). \n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }
```

```
        String file = args[0];
        client = TranscribeStreamingAsyncClient.builder()
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromFile(file)),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromFile(String file) {
        try {
            File inputFile = new File(file);
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;

        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US)
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
```

```
                    System.out.println("Error Occurred: " + sw.toString());
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
```

```
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
```

```
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK for Java 2.x API-Referenz.

Transkribieren von Audio von einem Mikrofon

Das folgende Code-Beispiel zeigt, wie mit Amazon-Transcribe-Streaming eine Transkription von einem Mikrofon generiert wird.

SDK für Java 2.x

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
 client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
```

```
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;
```

```
        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
```

```
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
```

```
        }
    }
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK for Java 2.x API-Referenz.

Beispiele für Amazon Translate unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Translate Aktionen ausführen und gängige Szenarien implementieren. AWS SDK for Java 2.x

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Amazon-Lex-Chatbots

Das folgende Codebeispiele zeigt, wie Sie einen Chatbot erstellen können, um die Besucher Ihrer Website einzubinden.

SDK für Java 2.x

Zeigt, wie Sie die Amazon-Lex-API verwenden können, um einen Chatbot innerhalb einer Webanwendung zu erstellen, um die Besucher Ihrer Website einzubinden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Java 2.x

Zeigt, wie man die Java-API für Amazon Simple Notification Service verwendet, um eine Webanwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Einrichten und Ausführen des Beispiels, das die Java Async-API verwendet, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Java 2.x

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHub Projekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

X-Ray Beispiele zur Verwendung des SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK for Java 2.x with Aktionen ausführen und gängige Szenarien implementieren X-Ray.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateGroupRequest;
import software.amazon.awssdk.services.xray.model.CreateGroupResponse;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to create\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createNewGroup(xRayClient, groupName);
    }

    public static void createNewGroup(XRayClient xRayClient, String groupName) {
        try {
            CreateGroupRequest groupRequest = CreateGroupRequest.builder()
```

```
                    .filterExpression("fault = true AND http.url CONTAINS \"example/
game\" AND responsetime >= 5")
                    .groupName(groupName)
                    .build();

            CreateGroupResponse groupResponse =
 xRayClient.createGroup(groupRequest);
            System.out.println("The Group ARN is " +
 groupResponse.group().groupARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateGroup in der AWS SDK for Java 2.x API-Referenz.

`CreateSamplingRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateSamplingRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.SamplingRule;
import software.amazon.awssdk.services.xray.model.XRayException;
import software.amazon.awssdk.services.xray.model.CreateSamplingRuleResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        createRule(xRayClient, ruleName);
    }

    public static void createRule(XRayClient xRayClient, String ruleName) {
        try {

            SamplingRule rule = SamplingRule.builder()
                    .ruleName(ruleName)
                    .priority(1)
                    .httpMethod("*")
                    .serviceType("*")
                    .serviceName("*")
                    .urlPath("*")
                    .version(1)
                    .host("*")
                    .resourceARN("*")
                    .build();
```

```
            CreateSamplingRuleRequest ruleRequest =
 CreateSamplingRuleRequest.builder()
                    .samplingRule(rule)
                    .build();

            CreateSamplingRuleResponse ruleResponse =
 xRayClient.createSamplingRule(ruleRequest);
            System.out.println(
                    "The ARN of the new rule is " +
 ruleResponse.samplingRuleRecord().samplingRule().ruleARN());

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie CreateSamplingRule in der AWS SDK for Java 2.x API-Referenz.

`DeleteGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGroup`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteGroupRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of the group to delete\s

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteSpecificGroup(xRayClient, groupName);
    }

    public static void deleteSpecificGroup(XRayClient xRayClient, String groupName)
 {
        try {
            DeleteGroupRequest groupRequest = DeleteGroupRequest.builder()
                    .groupName(groupName)
                    .build();

            xRayClient.deleteGroup(groupRequest);
            System.out.println(groupName + " was deleted!");

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- API-Details finden Sie DeleteGroup in der AWS SDK for Java 2.x API-Referenz.

`DeleteSamplingRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSamplingRule`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.DeleteSamplingRuleRequest;
import software.amazon.awssdk.services.xray.model.XRayException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteSamplingRule {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <ruleName>

                Where:
                   ruleName - The name of the rule to delete\s

                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String ruleName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        deleteRule(xRayClient, ruleName);
    }

    public static void deleteRule(XRayClient xRayClient, String ruleName) {
        try {
            DeleteSamplingRuleRequest ruleRequest =
 DeleteSamplingRuleRequest.builder()
                    .ruleName(ruleName)
                    .build();

            xRayClient.deleteSamplingRule(ruleRequest);
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie DeleteSamplingRule in der AWS SDK for Java 2.x API-Referenz.

`GetGroups`

Das folgende Codebeispiel zeigt die Verwendung`GetGroups`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetGroupsResponse;
import software.amazon.awssdk.services.xray.model.GroupSummary;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetGroups {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();

        getAllGroups(xRayClient);
    }

    public static void getAllGroups(XRayClient xRayClient) {
        try {
            GetGroupsResponse groupsResponse = xRayClient.getGroups();
            List<GroupSummary> groups = groupsResponse.groups();
            for (GroupSummary group : groups) {
                System.out.println("The AWS XRay group name is " +
 group.groupName());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetGroups in der AWS SDK for Java 2.x API-Referenz.

GetSamplingRules

Das folgende Codebeispiel zeigt die VerwendungGetSamplingRules.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetSamplingRulesResponse;
import software.amazon.awssdk.services.xray.model.SamplingRuleRecord;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSamplingRules {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        getRules(xRayClient);
    }

    public static void getRules(XRayClient xRayClient) {
        try {
            GetSamplingRulesResponse response = xRayClient.getSamplingRules(r ->
 r.build());
```

```
            List<SamplingRuleRecord> records = response.samplingRuleRecords();

            for (SamplingRuleRecord record : records) {
                System.out.println("The rule name is: " +
 record.samplingRule().ruleName());
                System.out.println("The related service is: " +
 record.samplingRule().serviceName());
            }
        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetSamplingRules in der AWS SDK for Java 2.x API-Referenz.

`GetServiceGraph`

Das folgende Codebeispiel zeigt die Verwendung`GetServiceGraph`.

SDK für Java 2.x

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.xray.XRayClient;
import software.amazon.awssdk.services.xray.model.GetServiceGraphRequest;
import software.amazon.awssdk.services.xray.model.GetServiceGraphResponse;
import software.amazon.awssdk.services.xray.model.Service;
import software.amazon.awssdk.services.xray.model.XRayException;
import java.time.LocalDateTime;
import java.time.Instant;
import java.time.ZoneId;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetServiceGraph {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <groupName>

                Where:
                   groupName - The name of a group based on which you want to
 generate a graph.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        Region region = Region.US_EAST_1;
        XRayClient xRayClient = XRayClient.builder()
                .region(region)
                .build();
        getGraph(xRayClient, groupName);
    }

    public static void getGraph(XRayClient xRayClient, String groupName) {
        try {
            // The Instant values have to be 6 hours apart.
            LocalDateTime localDateTime =
 LocalDateTime.parse("2022-03-09T06:00:00");
            Instant start = localDateTime.atZone(ZoneId.of("America/
New_York")).toInstant();

            LocalDateTime localDateTime2 =
 LocalDateTime.parse("2022-03-09T12:00:00");
            Instant end = localDateTime2.atZone(ZoneId.of("America/
New_York")).toInstant();
```

```
            GetServiceGraphRequest getServiceGraphRequest =
 GetServiceGraphRequest.builder()
                    .groupName(groupName)
                    .startTime(start)
                    .endTime(end)
                    .build();

            GetServiceGraphResponse graphResponse =
 xRayClient.getServiceGraph(getServiceGraphRequest);
            List<Service> services = graphResponse.services();
            for (Service service : services) {
                System.out.println("The name of the service is  " + service.name());
            }

        } catch (XRayException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API-Details finden Sie GetServiceGraph in der AWS SDK for Java 2.x API-Referenz.

Codebeispiele für SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie das AWS SDK für Kotlin mit verwenden. AWS

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für Kotlin-Entwicklerhandbuch — Mehr über die Verwendung von Kotlin mit. AWS
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- API-Gateway-Beispiele unter Verwendung von SDK für Kotlin
- Aurora-Beispiele unter Verwendung von SDK für Kotlin
- Auto-Scaling-Beispiele unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Bedrock unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Kotlin
- CloudWatch Beispiele zur Verwendung des SDK für Kotlin
- CloudWatch Loggt Beispiele mit SDK für Kotlin
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Kotlin
- DynamoDB-Beispiele unter Verwendung von SDK für Kotlin
- Beispiele für Amazon EC2 unter Verwendung von SDK für Kotlin
- Beispiele für Amazon ECR unter Verwendung von SDK für Kotlin
- OpenSearch Servicebeispiele, die SDK für Kotlin verwenden
- EventBridge Beispiele zur Verwendung des SDK für Kotlin
- AWS Glue Beispiele zur Verwendung des SDK für Kotlin
- IAM-Beispiele unter Verwendung von SDK für Kotlin
- AWS IoT Beispiele zur Verwendung des SDK für Kotlin
- AWS IoT data Beispiele zur Verwendung des SDK für Kotlin
- Beispiele für Amazon Keyspaces unter Verwendung von SDK für Kotlin
- AWS KMS Beispiele zur Verwendung des SDK für Kotlin
- Lambda-Beispiele unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Location unter Verwendung von SDK für Kotlin
- MediaConvert Beispiele zur Verwendung des SDK für Kotlin

- Beispiele für Amazon Pinpoint unter Verwendung von SDK für Kotlin
- Beispiele für Amazon RDS unter Verwendung von SDK für Kotlin
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Redshift unter Verwendung von SDK für Kotlin
- Beispiele für Amazon Rekognition unter Verwendung von SDK für Kotlin
- Beispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Kotlin
- Amazon-S3-Beispiele unter Verwendung von SDK für Kotlin
- SageMaker KI-Beispiele, die SDK für Kotlin verwenden
- Beispiele für Secrets Manager unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SES unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SNS unter Verwendung von SDK für Kotlin
- Beispiele für Amazon SQS unter Verwendung von SDK für Kotlin
- Beispiele für Step Functions unter Verwendung von SDK für Kotlin
- Support Beispiele zur Verwendung des SDK für Kotlin
- Beispiele für Amazon Translate unter Verwendung von SDKs für Kotlin
- X-Ray Beispiele zur Verwendung des SDK für Kotlin

API-Gateway-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Aurora-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:
```

```
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
```

```
            secretName - The name of the AWS Secrets Manager secret that contains
the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)
```

```
    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
 dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)

    println("10. Get a list of instance classes available for the selected engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
```

```
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
```

```
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
```

```
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
```

```
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
```

```
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
```

```
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
```

```
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
```

```
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Kotlin.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK für die Kotlin-API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK für die Kotlin-API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Kotlin.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
```

```
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Kotlin.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
```

```
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK für die Kotlin-API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Kotlin.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
```

```
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK für die Kotlin-API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
```

```
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK für die Kotlin-API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS SDK für die Kotlin-API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
```

```
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Kotlin.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
```

```
        }
    }
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK für die Kotlin-API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Kotlin.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
```

```
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
 role that the Auto Scaling group uses.
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
 the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
 vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)
```

```
    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
 groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
```

```
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }
```

```
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
```

```
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
```

```
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
```

```
        println("You successfully deleted $groupName")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
```

```
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK für die Kotlin-API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
```

```
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
```

```
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK für die Kotlin-API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK für die Kotlin-API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK für die Kotlin-API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK für die Kotlin-API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK für die Kotlin-API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK für die Kotlin-API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Bedrock unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung. `ListFoundationModels`

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
suspend fun listFoundationModels(): List<FoundationModelSummary>? {
    BedrockClient.fromEnvironment { region = "us-east-1" }.use { bedrockClient ->
        val response =
 bedrockClient.listFoundationModels(ListFoundationModelsRequest {})
        response.modelSummaries?.forEach { model ->
            println("==========================================")
            println(" Model ID: ${model.modelId}")
            println("------------------------------------------")
            println(" Name: ${model.modelName}")
```

```
            println(" Provider: ${model.providerName}")
            println(" Input modalities: ${model.inputModalities}")
            println(" Output modalities: ${model.outputModalities}")
            println(" Supported customizations: ${model.customizationsSupported}")
            println(" Supported inference types: ${model.inferenceTypesSupported}")
            println("----------------------------------------\n")
        }
        return response.modelSummaries
    }
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Amazon Nova

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models to
 generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
suspend fun main() {
    converse().also { println(it) }
}

suspend fun converse(): String {
    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in one line."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
```

```
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
            }
        }

        // Send the request and process the model's response
        runCatching {
            val response = client.converse(request)
            return response.output!!.asMessage().content.first().asText()
        }.getOrElse { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId",
 error)
        }
    }
}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Kotlin.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
import aws.sdk.kotlin.services.bedrockruntime.BedrockRuntimeClient
import aws.sdk.kotlin.services.bedrockruntime.model.ContentBlock
import aws.sdk.kotlin.services.bedrockruntime.model.ConversationRole
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamOutput
```

```
import aws.sdk.kotlin.services.bedrockruntime.model.ConverseStreamRequest
import aws.sdk.kotlin.services.bedrockruntime.model.Message

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message with a prompt
 * - Configure a streaming request with parameters
 * - Process the response stream in real time
 */
suspend fun main() {
    converseStream()
}

suspend fun converseStream(): String {
    // A buffer to collect the complete response
    val completeResponseBuffer = StringBuilder()

    // Create and configure the Bedrock runtime client
    BedrockRuntimeClient { region = "us-east-1" }.use { client ->

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        val modelId = "amazon.nova-lite-v1:0"

        // Create the message with the user's prompt
        val prompt = "Describe the purpose of a 'hello world' program in a
 paragraph."
        val message = Message {
            role = ConversationRole.User
            content = listOf(ContentBlock.Text(prompt))
        }

        // Configure the request with optional model parameters
        val request = ConverseStreamRequest {
            this.modelId = modelId
            messages = listOf(message)
            inferenceConfig {
                maxTokens = 500 // Maximum response length
                temperature = 0.5F // Lower values: more focused output
                // topP = 0.8F // Alternative to temperature
```

```
            }
        }

        // Process the streaming response
        runCatching {
            client.converseStream(request) { response ->
                response.stream?.collect { chunk ->
                    when (chunk) {
                        is ConverseStreamOutput.ContentBlockDelta -> {
                            // Process each text chunk as it arrives
                            chunk.value.delta?.asText()?.let { text ->
                                print(text)
                                System.out.flush() // Ensure immediate output
                                completeResponseBuffer.append(text)
                            }
                        }
                        else -> {} // Other output block types can be handled as
 needed
                    }
                }
            }
        }.onFailure { error ->
            error.message?.let { e -> System.err.println("ERROR: Can't invoke
 '$modelId'. Reason: $e") }
            throw RuntimeException("Failed to generate text with model $modelId:
 $error", error)
        }
    }

    return completeResponseBuffer.toString()
}
```

- API-Details finden Sie ConverseStream in der AWS SDK für die Kotlin-API-Referenz.

CloudWatch Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. CloudWatch

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo CloudWatch

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von CloudWatch beginnen.

SDK für Kotlin

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
```

```
            <namespace>
        Where:
            namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- API-Details finden Sie ListMetrics in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listet CloudWatch Namespaces und Metriken auf.
- Rufen Sie Statistiken für eine Metrik und die geschätzte Fakturierung ab.

- Erstellen und aktualisieren Sie ein Dashboard.
- Erstellen Sie eine Metrik und fügen Sie ihr Daten hinzu.
- Erstellen und lösen Sie einen Alarm aus und zeigen Sie dann den Alarmverlauf an.
- Fügen Sie einen Anomaliedetektor hinzu.
- Ermitteln Sie ein Metrik-Image, dann bereinigen Sie die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die CloudWatch Funktionen demonstriert.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing alerts
 are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
```

```
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a dashboard.
 (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
```

```
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select one
from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
```

```
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics, metricOption,
myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
```

```
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)
```

```
    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)

    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
```

```
        }

    val request =
        DeleteAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
```

```
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
```

```
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
```

```
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
```

```
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }
```

```
    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
```

```
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}

suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
```

```
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest =
        PutMetricAlarmRequest {
            alarmActions = alarmActionObs
            alarmDescription = "Example metric alarm"
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
            threshold = 100.00
            metricName = customMetricName
            namespace = customMetricNamespace
            evaluationPeriods = 1
            period = 10
            statistic = Statistic.Maximum
            datapointsToAlarm = 1
            treatMissingData = "ignore"
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}
```

```
suspend fun addMetricToDashboard(
    fileNameVal: String,
    dashboardNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension =
        Dimension {
            name = "UNIQUE_PAGES"
            value = "URLS"
        }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = "PAGES_VISITED"
            unit = StandardUnit.None
            value = dataPoint
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
            dimensions = listOf(dimension)
        }

    val request =
        PutMetricDataRequest {
            namespace = "SITE/TRAFFIC"
            metricData = listOf(datum)
        }
```

```
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

```
fun readFileAsString(file: String): String =
 String(Files.readAllBytes(Paths.get(file)))

suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension =
        Dimension {
            name = "Currency"
            value = "USD"
        }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest =
        GetMetricStatisticsRequest {
            metricName = "EstimatedCharges"
            namespace = "AWS/Billing"
            dimensions = dimensionList
            statistics = listOf(Statistic.Maximum)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            period = 86400
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
```

```
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteAlarm(alarmNameVal: String) {
```

```
    val request =
        DeleteAlarmsRequest {
            alarmNames = listOf(alarmNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- API-Details finden Sie DeleteAlarms in der AWS SDK für die Kotlin-API-Referenz.

`DeleteAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAnomalyDetector`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val request =
        DeleteAnomalyDetectorRequest {
```

```
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- API-Details finden Sie DeleteAnomalyDetector in der AWS SDK für die Kotlin-API-Referenz.

`DeleteDashboards`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDashboards`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest =
        DeleteDashboardsRequest {
            dashboardNames = listOf(dashboardName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteDashboards in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAlarmHistory`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmHistory`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAlarmHistory(
    fileName: String,
    date: String,
) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest =
        DescribeAlarmHistoryRequest {
            startDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            endDate =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDateVal)
            alarmName = alarmNameVal
            historyItemType = HistoryItemType.Action
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
```

```
            }
        }
    }
}
```

- API-Details finden Sie DescribeAlarmHistory in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest =
        DescribeAlarmsRequest {
            alarmTypes = typeList
            maxRecords = 10
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- API-Details finden Sie DescribeAlarms in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest =
        DescribeAlarmsForMetricRequest {
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) {
            println("No Alarm state found for $customMetricName after 10 retries.")
        } else {
            println("Alarm state found for $customMetricName.")
        }
    }
}
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK for Kotlin API-Referenz.

`DescribeAnomalyDetectors`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAnomalyDetectors`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest =
        DescribeAnomalyDetectorsRequest {
            maxResults = 10
            metricName = customMetricName
            namespace = customMetricNamespace
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- API-Details finden Sie DescribeAnomalyDetectors in der AWS SDK für die Kotlin-API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableActions(alarmName: String) {
    val request =
        DisableAlarmActionsRequest {
            alarmNames = listOf(alarmName)
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK für die Kotlin-API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableActions(alarm: String) {
    val request =
        EnableAlarmActionsRequest {
            alarmNames = listOf(alarm)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- API-Details finden Sie EnableAlarmActions in der AWS SDK für die Kotlin-API-Referenz.

`GetMetricData`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricData`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 =
        nowDate.plus(hours, ChronoUnit.HOURS).plus(
            minutes,
```

```
            ChronoUnit.MINUTES,
        )

    val met =
        Metric {
            metricName = customMetricName
            namespace = customMetricNamespace
        }

    val metStat =
        MetricStat {
            stat = "Maximum"
            period = 1
            metric = met
        }

    val dataQUery =
        MetricDataQuery {
            metricStat = metStat
            id = "foo2"
            returnData = true
        }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq =
        GetMetricDataRequest {
            maxDatapoints = 10
            scanBy = ScanBy.TimestampDescending
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(nowDate)
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(date2)
            metricDataQueries = dq
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
```

```
    }
}
```

- API-Details finden Sie GetMetricData in der AWS SDK für die Kotlin-API-Referenz.

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricStatistics`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAndDisplayMetricStatistics(
    nameSpaceVal: String,
    metVal: String,
    metricOption: String,
    date: String,
    myDimension: Dimension,
) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest =
        GetMetricStatisticsRequest {
            endTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(endDate)
            startTime =
                aws.smithy.kotlin.runtime.time
                    .Instant(start)
            dimensions = listOf(myDimension)
            metricName = metVal
            namespace = nameSpaceVal
            period = 86400
            statistics = listOf(Statistic.fromValue(metricOption))
        }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- API-Details finden Sie GetMetricStatistics in der AWS SDK für die Kotlin-API-Referenz.

`GetMetricWidgetImage`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricWidgetImage`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
```

```
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest =
        GetMetricWidgetImageRequest {
            metricWidget = myJSON
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- API-Details finden Sie GetMetricWidgetImage in der AWS SDK für die Kotlin-API-Referenz.

`ListDashboards`

Das folgende Codebeispiel zeigt die Verwendung`ListDashboards`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDashboards() {
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient
            .listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- API-Details finden Sie ListDashboards in der AWS SDK für die Kotlin-API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request =
        ListMetricsRequest {
            namespace = namespaceVal
        }
    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
```

```
    return metList
}
```

- API-Details finden Sie ListMetrics in der AWS SDK für die Kotlin-API-Referenz.

`PutAnomalyDetector`

Das folgende Codebeispiel zeigt die Verwendung`PutAnomalyDetector`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal =
        SingleMetricAnomalyDetector {
            metricName = customMetricName
            namespace = customMetricNamespace
            stat = "Maximum"
        }

    val anomalyDetectorRequest =
        PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
```

```
}
```

- API-Details finden Sie PutAnomalyDetector in der AWS SDK für die Kotlin-API-Referenz.

`PutDashboard`

Das folgende Codebeispiel zeigt die Verwendung`PutDashboard`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDashboardWithMetrics(
    dashboardNameVal: String,
    fileNameVal: String,
) {
    val dashboardRequest =
        PutDashboardRequest {
            dashboardName = dashboardNameVal
            dashboardBody = readFileAsString(fileNameVal)
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
```

```
}
```

- API-Details finden Sie PutDashboard in der AWS SDK für die Kotlin-API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putMetricAlarm(
    alarmNameVal: String,
    instanceIdVal: String,
) {
    val dimensionOb =
        Dimension {
            name = "InstanceId"
            value = instanceIdVal
        }

    val request =
        PutMetricAlarmRequest {
            alarmName = alarmNameVal
            comparisonOperator = ComparisonOperator.GreaterThanThreshold
            evaluationPeriods = 1
            metricName = "CPUUtilization"
            namespace = "AWS/EC2"
            period = 60
            statistic = Statistic.fromValue("Average")
            threshold = 70.0
            actionsEnabled = false
            alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
            unit = StandardUnit.fromValue("Seconds")
            dimensions = listOf(dimensionOb)
```

```
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK für die Kotlin-API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace = rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1001.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
```

```
        }

    val datum2 =
        MetricDatum {
            metricName = customMetricName
            unit = StandardUnit.None
            value = 1002.00
            timestamp =
                aws.smithy.kotlin.runtime.time
                    .Instant(instant)
        }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request =
        PutMetricDataRequest {
            namespace = customMetricNamespace
            metricData = metricDataList
        }

    CloudWatchClient.fromEnvironment { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- API-Details finden Sie PutMetricData in der AWS SDK für die Kotlin-API-Referenz.

CloudWatch Loggt Beispiele mit SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin with CloudWatch Logs Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSubscriptionFilter`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSubFilter(
    filter: String?,
    logGroup: String?,
) {
    val request =
        DeleteSubscriptionFilterRequest {
            filterName = filter
            logGroupName = logGroup
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
 $filter")
    }
}
```

- API-Details finden Sie DeleteSubscriptionFilter in der AWS SDK für die Kotlin-API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeFilters(logGroup: String) {
    val request =
        DescribeSubscriptionFiltersRequest {
            logGroupName = logGroup
            limit = 1
        }

    CloudWatchLogsClient.fromEnvironment { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name  ${filter.filterName} pattern
 ${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- API-Details finden Sie DescribeSubscriptionFilters in der AWS SDK für die Kotlin-API-Referenz.

`StartLiveTail`

Das folgende Codebeispiel zeigt die Verwendung`StartLiveTail`.

SDK für Kotlin

Binden Sie die erforderlichen Dateien ein.

```
import aws.sdk.kotlin.services.cloudwatchlogs.CloudWatchLogsClient
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailRequest
import aws.sdk.kotlin.services.cloudwatchlogs.model.StartLiveTailResponseStream
import kotlinx.coroutines.flow.takeWhile
```

Starten Sie die Live-Tail-Sitzung.

```
    val client = CloudWatchLogsClient.fromEnvironment()

    val request = StartLiveTailRequest {
        logGroupIdentifiers = logGroupIdentifiersVal
        logStreamNames = logStreamNamesVal
        logEventFilterPattern = logEventFilterPatternVal
    }

    val startTime = System.currentTimeMillis()

    try {
        client.startLiveTail(request) { response ->
            val stream = response.responseStream
            if (stream != null) {
                /* Set a timeout to unsubcribe from the flow. This will:
                * 1). Close the stream
                * 2). Stop the Live Tail session
                */
                stream.takeWhile { System.currentTimeMillis() - startTime <
10000 }.collect { value ->
                    if (value is StartLiveTailResponseStream.SessionStart) {
                        println(value.asSessionStart())
                    } else if (value is StartLiveTailResponseStream.SessionUpdate) {
                        for (e in value.asSessionUpdate().sessionResults!!) {
                            println(e)
                        }
                    } else {
                        throw IllegalArgumentException("Unknown event type")
                    }
                }
            } else {
                throw IllegalArgumentException("No response stream")
            }
        }
    } catch (e: Exception) {
        println("Exception occurred during StartLiveTail: $e")
        System.exit(1)
    }
```

- API-Details finden Sie StartLiveTail in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung. `AdminGetUser`

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
```

```
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- API-Details finden Sie AdminGetUser in der AWS SDK für die Kotlin-API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
```

```
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK für die Kotlin-API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
```

```
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK für die Kotlin-API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
```

```
        println(secretCode)
        return tokenResponse.session
    }
}
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK für die Kotlin-API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK für die Kotlin-API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- API-Details finden Sie ListUsers in der AWS SDK für die Kotlin-API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK für die Kotlin-API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
```

```
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- API-Details finden Sie SignUp in der AWS SDK für die Kotlin-API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
```

```
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.
```

```
 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted
 to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
 script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
```

```
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
```

```
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
```

```
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient.fromEnvironment { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Beispiele für Amazon Comprehend unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Kotlin

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

DynamoDB-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen

- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine DynamoDB-Tabelle.

```
suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
```

```
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
```

```
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

Rufen Sie ein Element aus einer Tabelle ab.

```
suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Vollständiges Beispiel.

```
suspend fun main() {
    val tableName = "Movies"
    val fileName = "../../../resources/sample_files/movies.json"
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and
 a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
```

```
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deletIssuesTable(tableName)
}

suspend fun createScenarioTable(
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
```

```
            // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
 ${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(
    tableName: String,
    fileName: String,
) {
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
```

```
    itemValues["info"] = AttributeValue.S(info)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deletIssuesTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }
```

```
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
```

```
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewTable(
    tableNameVal: String,
    key: String,
```

```
): String? {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef)
            keySchema = listOf(keySchemaVal)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists {
            // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}
```

- API-Details finden Sie CreateTable in der AWS SDK für die Kotlin-API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDynamoDBItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        DeleteItemRequest {
            tableName = tableNameVal
            key = keyToGet
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- API-Details finden Sie DeleteItem in der AWS SDK für die Kotlin-API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API-Details finden Sie DeleteTable in der AWS SDK für die Kotlin-API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
) {
```

```
    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request =
        GetItemRequest {
            key = keyToGet
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

- API-Details finden Sie GetItem in der AWS SDK für die Kotlin-API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllTables() {
    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK für die Kotlin-API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
    awardVal: String,
    songTitle: String,
    songTitleVal: String,
) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request =
        PutItemRequest {
            tableName = tableNameVal
            item = itemValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
```

```
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- API-Details finden Sie PutItem in der AWS SDK für die Kotlin-API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String,
): Int {
    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request =
        QueryRequest {
            tableName = tableNameVal
            keyConditionExpression = "$partitionAlias = :$partitionKeyName"
            expressionAttributeNames = attrNameAlias
            this.expressionAttributeValues = attrValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
```

```
        val response = ddb.query(request)
        return response.count
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der API–Referenz zum AWS SDK für Kotlin.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun scanItems(tableNameVal: String) {
    val request =
        ScanRequest {
            tableName = tableNameVal
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Kotlin.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateTableItem(
    tableNameVal: String,
    keyName: String,
    keyVal: String,
    name: String,
    updateVal: String,
) {
    val itemKey = mutableMapOf<String, AttributeValue>()
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] =
        AttributeValueUpdate {
            value = AttributeValue.S(updateVal)
            action = AttributeAction.Put
        }

    val request =
        UpdateItemRequest {
            tableName = tableNameVal
            key = itemKey
            attributeUpdates = updatedValues
        }

    DynamoDbClient.fromEnvironment { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- API-Details finden Sie UpdateItem in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id
 and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(
```

```
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}
```

```
suspend fun putRecordBatch(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,
 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie1
        }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie2
        }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersMovie3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
```

```
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }
```

```
    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec1
        }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec2
        }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
```

```
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 =
        BatchStatementRequest {
            statement = sqlStatement
            parameters = parametersRec3
        }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest =
        BatchExecuteStatementRequest {
            statements = myBatchStatementList
        }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für die Kotlin-API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.

- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main() {
    val ddb = DynamoDbClient.fromEnvironment { region = "us-east-1" }
    val tableName = "MoviesPartiQ"
    val fileName = "../../../resources/sample_files/movies.json"
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
 id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("******* Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("******* Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("******* Updating a record.")
    updateTableItemPartiQL(ddb)

    println("******* Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("******* Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}

suspend fun createTablePartiQL(
    ddb: DynamoDbClient,
    tableNameVal: String,
    key: String,
```

```
) {
    val attDef =
        AttributeDefinition {
            attributeName = key
            attributeType = ScalarAttributeType.N
        }

    val attDef1 =
        AttributeDefinition {
            attributeName = "title"
            attributeType = ScalarAttributeType.S
        }

    val keySchemaVal =
        KeySchemaElement {
            attributeName = key
            keyType = KeyType.Hash
        }

    val keySchemaVal1 =
        KeySchemaElement {
            attributeName = "title"
            keyType = KeyType.Range
        }

    val request =
        CreateTableRequest {
            attributeDefinitions = listOf(attDef, attDef1)
            keySchema = listOf(keySchemaVal, keySchemaVal1)
            billingMode = BillingMode.PayPerRequest
            tableName = tableNameVal
        }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists {
        // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
 ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(
    ddb: DynamoDbClient,
```

```
    fileName: String,
) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {
        if (t == 200) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?,
 'info' : ?}"
```

```
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C.
 Cooper\",\"Ernest B. Schoedsack\' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {
    val request =
        DeleteTableRequest {
            tableName = tableNameVal
        }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>,
): ExecuteStatementResponse {
```

```
    val request =
        ExecuteStatementRequest {
            statement = statementVal
            parameters = parametersVal
        }

    return ddb.executeStatement(request)
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon EC2 unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon EC2.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks:

 1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI’s
 architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance
 type.
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
```

```
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
 Console.
            myIpAddress - The IP address of your development machine.

"""

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }

    val keyName = args[0]
    val fileName = args[1]
    val groupName = args[2]
    val groupDesc = args[3]
    val vpcId = args[4]
    val myIpAddress = args[5]
    var newInstanceId: String? = ""

    println(DASHES)
    println("Welcome to the Amazon EC2 example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an RSA key pair and save the private key material as a .pem
 file.")
    createKeyPairSc(keyName, fileName)
    println(DASHES)

    println(DASHES)
    println("2. List key pairs.")
    describeEC2KeysSc()
    println(DASHES)

    println(DASHES)
```

```
    println("3. Create a security group.")
    val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId, myIpAddress)
    println(DASHES)

    println(DASHES)
    println("4. Display security group info for the newly created security group.")
    describeSecurityGroupsSc(groupId.toString())
    println(DASHES)

    println(DASHES)
    println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in the
name.")
    val instanceId = getParaValuesSc()
    if (instanceId == "") {
        println("The instance Id value isn't valid.")
        exitProcess(0)
    }
    println("The instance Id is $instanceId.")
    println(DASHES)

    println(DASHES)
    println("6. Get more information about an amzn2 image and return the AMI
value.")
    val amiValue = instanceId?.let { describeImageSc(it) }
    if (instanceId == "") {
        println("The instance Id value is invalid.")
        exitProcess(0)
    }
    println("The AMI value is $amiValue.")
    println(DASHES)

    println(DASHES)
    println("7. Get a list of instance types.")
    val instanceType = getInstanceTypesSc()
    println(DASHES)

    println(DASHES)
    println("8. Create an instance.")
    if (amiValue != null) {
        newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
        println("The instance Id is $newInstanceId")
    }
    println(DASHES)
```

```
    println(DASHES)
    println("9. Display information about the running instance. ")
    var ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("10.  Stop the instance.")
    if (newInstanceId != null) {
        stopInstanceSc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("11.  Start the instance.")
    if (newInstanceId != null) {
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
 instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
```

```
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
    println("17. Delete the key pair.")
    deleteKeysSc(keyName)
    println(DASHES)

    println(DASHES)
    println("You successfully completed the Amazon EC2 scenario.")
    println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
```

```
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitUntilInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
```

```
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?. value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
 ${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
 ${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
```

```
                    println("Instance state is
 ${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}

suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
 $amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
```

```
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
 ${response.images?.get(0)?.description}")
        println("The name of the first image is  ${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
```

```
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient.fromEnvironment { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
    return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() + "
 and group VPC " + group.vpcId)
        }
    }
}
```

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
```

```
    }
}

suspend fun describeEC2KeysSc() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        val content = response.keyMaterial
        if (content != null) {
            File(fileNameVal).writeText(content)
        }
        println("Successfully created key pair named $keyNameVal")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair

- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
```

```
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- API-Details finden Sie AllocateAddress in der AWS SDK für die Kotlin-API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
```

```
}
```

- API-Details finden Sie AssociateAddress in der AWS SDK für die Kotlin-API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
```

```
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK für die Kotlin-API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
```

```
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.createKeyPair(request)
        println("The key ID is ${response.keyPairId}")
    }
}
```

- API-Details finden Sie CreateKeyPair in der AWS SDK für die Kotlin-API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "0.0.0.0/0"
```

```
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
                fromPort = 22
                ipRanges = listOf(ipRange)
            }

        val authRequest =
            AuthorizeSecurityGroupIngressRequest {
                groupName = groupNameVal
                ipPermissions = listOf(ipPerm, ipPerm2)
            }
        ec2.authorizeSecurityGroupIngress(authRequest)
        println("Successfully added ingress policy to Security Group $groupNameVal")
        return resp.groupId
    }
}
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK für die Kotlin-API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeys(keyPair: String?) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK für die Kotlin-API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK für die Kotlin-API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
 ${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
 ${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
        return instanceType
    }
}
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK für die Kotlin-API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstances(request)
        response.reservations?.forEach { reservation ->
            reservation.instances?.forEach { instance ->
                println("Instance Id is ${instance.instanceId}")
                println("Image id is ${instance.imageId}")
                println("Instance type is ${instance.instanceType}")
                println("Instance state name is ${instance.state?.name}")
                println("monitoring information is ${instance.monitoring?.state}")
            }
        }
    }
}
```

- API-Details finden Sie DescribeInstances in der AWS SDK für die Kotlin-API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2Keys() {
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
 ${ keyPair.keyFingerprint}")
        }
    }
}
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK für die Kotlin-API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
```

```
        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
 ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für die Kotlin-API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- API-Details finden Sie DisassociateAddress in der AWS SDK für die Kotlin-API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- API-Details finden Sie ReleaseAddress in der AWS SDK für die Kotlin-API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
```

```
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }

        val requestTags =
            CreateTagsRequest {
                resources = listOf(instanceId.toString())
                tags = listOf(tag)
            }
        ec2.createTags(requestTags)
        println("Successfully started EC2 Instance $instanceId based on AMI $amiId")
        return instanceId
    }
}
```

- API-Details finden Sie RunInstances in der AWS SDK für die Kotlin-API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
 minutes.")
        ec2.waitUntilInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- API-Details finden Sie StartInstances in der AWS SDK für die Kotlin-API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
```

```
        println("Waiting until instance $instanceId stops. This will take a few
 minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

- API-Details finden Sie StopInstances in der AWS SDK für die Kotlin-API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is ${instance.instanceId}")
        }
    }
}
```

- API-Details finden Sie TerminateInstances in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon ECR unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon ECR

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon ECR.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess
```

```
suspend fun main(args: Array<String>) {
    val usage = """
            Usage: <repositoryName>

            Where:
               repositoryName - The name of the Amazon ECR repository.

    """.trimIndent()

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val repoName = args[0]
    listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
    val listImages =
        ListImagesRequest {
            repositoryName = repoName
        }

    EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
        val imageResponse = ecrClient.listImages(listImages)
        imageResponse.imageIds?.forEach { imageId ->
            println("Image tag: ${imageId.imageTag}")
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS -SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-ECR-Repository.

- Legen Sie Repository-Richtlinien fest.
- Rufen Sie Repository-URIs ab.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.
- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.
- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Amazon-ECR-Funktionen demonstriert.

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with the
 Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a local
 image,
```

```
 * this program will not successfully run. For more information including how to
 create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage: <iamRoleARN> <accountId>

        Where:
           iamRoleARN - The IAM role ARN that has the necessary permissions to
 access and manage the Amazon ECR repository.
           accountId - Your AWS account number.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        return
    }

    var iamRole = args[0]
    var localImageName: String
    var accountId = args[1]
    val ecrActions = ECRActions()
    val scanner = Scanner(System.`in`)

    println(
        """
        The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
 container registry
        service provided by AWS. It allows developers and organizations to securely
        store, manage, and deploy Docker container images.
        ECR provides a simple and scalable way to manage container images throughout
 their lifecycle,
        from building and testing to production deployment.
```

```
        The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
        programmatically interact with the Amazon ECR service. This allows
developers to
        automate the storage, retrieval, and management of container images as part
of their application
        deployment pipelines. With ECR, teams can focus on building and deploying
their
        applications without having to worry about the underlying infrastructure
required to
        host and manage a container registry.

        This scenario walks you through how to perform key operations for this
service.
        Let's get started...

         You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
            this program that did not complete).

        """.trimIndent(),
    )

    while (true) {
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
            val repoName = "echo-text"
            ecrActions.deleteECRRepository(repoName)
            return
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }

    waitForInputToContinue(scanner)
    println(DASHES)
    println(
```

```
        """
        1. Create an ECR repository.

        The first task is to ensure we have a local Docker image named echo-text.
        If this image exists, then an Amazon ECR repository is created.

        An ECR repository is a private Docker container repository provided
        by Amazon Web Services (AWS). It is a managed service that makes it easy
        to store, manage, and deploy Docker container images.

        """.trimIndent(),
    )

    // Ensure that a local docker image named echo-text exists.
    val doesExist = ecrActions.listLocalImages()
    val repoName: String
    if (!doesExist) {
        println("The local image named echo-text does not exist")
        return
    } else {
        localImageName = "echo-text"
        repoName = "echo-text"
    }

    val repoArn = ecrActions.createECRRepository(repoName).toString()
    println("The ARN of the ECR repository is $repoArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
```

```
    ecrActions.setRepoPolicy(repoName, iamRole)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val policyText = ecrActions.getRepoPolicy(repoName)
    println("Policy Text:")
    println(policyText)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
        ECR repository, such as pushing, pulling, or managing your Docker images.


        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    ecrActions.getAuthToken()
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
```

```
        """
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
    println("The repository URI is $repositoryURI")
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker images
from your repositories,
        freeing up storage space and reducing costs.

        """.trimIndent(),
    )
    waitForInputToContinue(scanner)
    val pol = ecrActions.setLifeCyclePolicy(repoName)
    println(pol)
    waitForInputToContinue(scanner)

    println(DASHES)
    println(
        """
        7. Push a docker image to the Amazon ECR Repository.
```

```
        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    ecrActions.pushDockerImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("8. Verify if the image is in the ECR Repository.")
    waitForInputToContinue(scanner)
    ecrActions.verifyImage(repoName, localImageName)
    waitForInputToContinue(scanner)

    println(DASHES)
    println("9. As an optional step, you can interact with the image in Amazon ECR
by using the CLI.")
    println("Would you like to view instructions on how to use the CLI to run the
image? (y/n)")
    val ans = scanner.nextLine().trim()
    if (ans.equals("y", true)) {
        val instructions = """
        1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
you need to authenticate with the registry. You can do this using the AWS CLI:

            aws ecr get-login-password --region us-east-1 | docker login --username
AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

        2. Describe the image using this command:
```

```
           aws ecr describe-images --repository-name $repoName --image-ids imageTag=
$localImageName

        3. Run the Docker container and view the output using this command:

           docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
$localImageName
        """
        println(instructions)
    }
    waitForInputToContinue(scanner)

    println(DASHES)
    println("10. Delete the ECR Repository.")
    println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.

        """.trimIndent(),
    )
    println("Would you like to delete the Amazon ECR Repository? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        println("You selected to delete the AWS ECR resources.")
        waitForInputToContinue(scanner)
        ecrActions.deleteECRRepository(repoName)
    }

    println(DASHES)
    println("This concludes the Amazon ECR SDK scenario")
    println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
```

```
            println("Continuing with the program...")
            println("")
            break
        } else {
            // Handle invalid input.
            println("Invalid input. Please try again.")
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon ECR SDK.

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
    private var dockerClient: DockerClient? = null

    private fun getDockerClient(): DockerClient? {
        val osName = System.getProperty("os.name")
        if (osName.startsWith("Windows")) {
            // Make sure Docker Desktop is running.
            val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
            val dockerCmdExecFactory: DockerCmdExecFactory =
```

```
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
           dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory).
       } else {
           dockerClient = DockerClientBuilder.getInstance().build()
       }
       return dockerClient
   }


   /**
    * Sets the lifecycle policy for the specified repository.
    *
    * @param repoName the name of the repository for which to set the lifecycle
policy.
    */
   suspend fun setLifeCyclePolicy(repoName: String): String? {
       val polText =
           """
            {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14
                    },
                    "action": {
                        "type": "expire"
                    }
                }
           ]
           }

           """.trimIndent()
       val lifecyclePolicyPreviewRequest =
           StartLifecyclePolicyPreviewRequest {
               lifecyclePolicyText = polText
               repositoryName = repoName
           }
```

```
        // Execute the request asynchronously.
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response =
ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
            return response.lifecyclePolicyText
        }
    }


    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }


    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
```

```
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }


    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }


    /**
     * Sets the repository policy for the specified ECR repository.
     *
     * @param repoName the name of the ECR repository.
     * @param iamRole the IAM role to be granted access to the repository.
     */
    suspend fun setRepoPolicy(
        repoName: String?,
        iamRole: String?,
    ) {
```

```
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }


    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
```

```
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }

    suspend fun getRepoARN(repoName: String): String? {
        // Fetch the existing repository's ARN.
        val describeRequest =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeResponse = ecrClient.describeRepositories(describeRequest)
            return describeResponse.repositories?.get(0)?.repositoryArn
        }
    }

    fun listLocalImages(): Boolean = try {
        val images = getDockerClient()?.listImagesCmd()?.exec()
        images?.any { image ->
            image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
        } ?: false
    } catch (ex: Exception) {
        println("ERROR: ${ex.message}")
        false
    }


    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
```

```
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }


    /**
     * Deletes an ECR (Elastic Container Registry) repository.
     *
     * @param repoName the name of the repository to delete.
     */
```

```
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }

    // Return an AuthConfig.
    private suspend fun getAuthConfig(repoName: String): AuthConfig {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            val decodedToken = String(Base64.getDecoder().decode(token))
            val password = decodedToken.substring(4)

            val request =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val descrRepoResponse = ecrClient.describeRepositories(request)
            val repoData = descrRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
            val registryURL: String = repoData?.repositoryUri?.split("/")?.get(0) ?:
""

            return AuthConfig()
                .withUsername("AWS")
                .withPassword(password)
                .withRegistryAddress(registryURL)
        }
```

```
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Aktionen

`CreateRepository`

Das folgende Codebeispiel zeigt die Verwendung`CreateRepository`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
     *
     * @param repoName the name of the repository to create.
     * @return the Amazon Resource Name (ARN) of the created repository, or an empty
string if the operation failed.
     * @throws RepositoryAlreadyExistsException if the repository exists.
```

```
     * @throws EcrException         if an error occurs while creating the
repository.
     */
    suspend fun createECRRepository(repoName: String?): String? {
        val request =
            CreateRepositoryRequest {
                repositoryName = repoName
            }

        return try {
            EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
                val response = ecrClient.createRepository(request)
                response.repository?.repositoryArn
            }
        } catch (e: RepositoryAlreadyExistsException) {
            println("Repository already exists: $repoName")
            repoName?.let { getRepoARN(it) }
        } catch (e: EcrException) {
            println("An error occurred: ${e.message}")
            null
        }
    }
```

- API-Details finden Sie CreateRepository in der AWS SDK für die Kotlin-API-Referenz.

`DeleteRepository`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRepository`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Deletes an ECR (Elastic Container Registry) repository.
```

```
     *
     * @param repoName the name of the repository to delete.
     */
    suspend fun deleteECRRepository(repoName: String) {
        if (repoName.isNullOrEmpty()) {
            throw IllegalArgumentException("Repository name cannot be null or
empty")
        }

        val repositoryRequest =
            DeleteRepositoryRequest {
                force = true
                repositoryName = repoName
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            ecrClient.deleteRepository(repositoryRequest)
            println("You have successfully deleted the $repoName repository")
        }
    }
```

- API-Details finden Sie DeleteRepository in der AWS SDK für die Kotlin-API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
     *
```

```
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- API-Details finden Sie DescribeImages in der AWS SDK für die Kotlin-API-Referenz.

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRepositories`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the repository URI for the specified repository name.
     *
     * @param repoName the name of the repository to retrieve the URI for.
     * @return the repository URI for the specified repository name.
     */
    suspend fun getRepositoryURI(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }
        val request =
            DescribeRepositoriesRequest {
                repositoryNames = listOf(repoName)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
            if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
                return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
            } else {
                println("No repositories found for the given name.")
                return ""
            }
        }
    }
```

- API-Details finden Sie DescribeRepositories in der AWS SDK für die Kotlin-API-Referenz.

`GetAuthorizationToken`

Das folgende Codebeispiel zeigt die Verwendung`GetAuthorizationToken`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
     *
     */
    suspend fun getAuthToken() {
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            // Retrieve the authorization token for ECR.
            val response = ecrClient.getAuthorizationToken()
            val authorizationData = response.authorizationData?.get(0)
            val token = authorizationData?.authorizationToken
            if (token != null) {
                println("The token was successfully retrieved.")
            }
        }
    }
```

- API-Details finden Sie GetAuthorizationToken in der AWS SDK für die Kotlin-API-Referenz.

`GetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetRepositoryPolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Gets the repository policy for the specified repository.
     *
     * @param repoName the name of the repository.
     */
    suspend fun getRepoPolicy(repoName: String?): String? {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
be null or empty" }

        // Create the request
        val getRepositoryPolicyRequest =
            GetRepositoryPolicyRequest {
                repositoryName = repoName
            }
        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
            val responseText = response.policyText
            return responseText
        }
    }
```

- API-Details finden Sie GetRepositoryPolicy in der AWS SDK für die Kotlin-API-Referenz.

`PushImageCmd`

Das folgende Codebeispiel zeigt die Verwendung`PushImageCmd`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
repository.
     *
     * @param repoName the name of the ECR repository to push the image to.
     * @param imageName the name of the Docker image.
     */
    suspend fun pushDockerImage(
        repoName: String,
        imageName: String,
    ) {
        println("Pushing $imageName to $repoName will take a few seconds")
        val authConfig = getAuthConfig(repoName)

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val desRequest =
                DescribeRepositoriesRequest {
                    repositoryNames = listOf(repoName)
                }

            val describeRepoResponse = ecrClient.describeRepositories(desRequest)
            val repoData =
                describeRepoResponse.repositories?.firstOrNull { it.repositoryName
== repoName }
                    ?: throw RuntimeException("Repository not found: $repoName")

            val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
            if (tagImageCmd != null) {
                tagImageCmd.exec()
            }
            val pushImageCmd =
                repoData.repositoryUri?.let {
```

```
                    dockerClient?.pushImageCmd(it)
                        // ?.withTag("latest")
                        ?.withAuthConfig(authConfig)
                }

            try {
                if (pushImageCmd != null) {
                    pushImageCmd.start().awaitCompletion()
                }
                println("The $imageName was pushed to Amazon ECR")
            } catch (e: IOException) {
                throw RuntimeException(e)
            }
        }
    }
```

- API-Details finden Sie PushImageCmd in der AWS SDK für die Kotlin-API-Referenz.

`SetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`SetRepositoryPolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
    repoName: String?,
    iamRole: String?,
) {
```

```
        val policyDocumentTemplate =
            """
             {
              "Version":"2012-10-17",
              "Statement" : [ {
                "Sid" : "new statement",
                "Effect" : "Allow",
                "Principal" : {
                  "AWS" : "$iamRole"
                },
                "Action" : "ecr:BatchGetImage"
              } ]
            }

            """.trimIndent()
        val setRepositoryPolicyRequest =
            SetRepositoryPolicyRequest {
                repositoryName = repoName
                policyText = policyDocumentTemplate
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val response = ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
            if (response != null) {
                println("Repository policy set successfully.")
            }
        }
    }
```

- API-Details finden Sie SetRepositoryPolicy in der AWS SDK für die Kotlin-API-Referenz.

`StartLifecyclePolicyPreview`

Das folgende Codebeispiel zeigt die Verwendung`StartLifecyclePolicyPreview`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
     *
     * @param repositoryName The name of the Amazon ECR repository.
     * @param imageTag       The tag of the image to verify.
     */
    suspend fun verifyImage(
        repoName: String?,
        imageTagVal: String?,
    ) {
        require(!(repoName == null || repoName.isEmpty())) { "Repository name cannot
 be null or empty" }
        require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag cannot
 be null or empty" }

        val imageId =
            ImageIdentifier {
                imageTag = imageTagVal
            }
        val request =
            DescribeImagesRequest {
                repositoryName = repoName
                imageIds = listOf(imageId)
            }

        EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
            val describeImagesResponse = ecrClient.describeImages(request)
            if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
                println("Image is present in the repository.")
            } else {
                println("Image is not present in the repository.")
            }
        }
    }
```

- API-Details finden Sie StartLifecyclePolicyPreview in der AWS SDK für die Kotlin-API-Referenz.

OpenSearch Servicebeispiele, die SDK für Kotlin verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin with OpenSearch Service Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDomain`

Das folgende Codebeispiel zeigt die Verwendung`CreateDomain`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            dedicatedMasterEnabled = true
            dedicatedMasterCount = 3
            dedicatedMasterType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceType =
 OpenSearchPartitionInstanceType.fromValue("t2.small.search")
            instanceCount = 5
        }
```

```
    val ebsOptionsOb =
        EbsOptions {
            ebsEnabled = true
            volumeSize = 10
            volumeType = VolumeType.Gp2
        }

    val encryptionOptionsOb =
        NodeToNodeEncryptionOptions {
            enabled = true
        }

    val request =
        CreateDomainRequest {
            domainName = domainNameVal
            engineVersion = "OpenSearch_1.0"
            clusterConfig = clusterConfigOb
            ebsOptions = ebsOptionsOb
            nodeToNodeEncryptionOptions = encryptionOptionsOb
        }

    println("Sending domain creation request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val createResponse = searchClient.createDomain(request)
        println("Domain status is ${createResponse.domainStatus}")
        println("Domain Id is ${createResponse.domainStatus?.domainId}")
    }
}
```

- API-Details finden Sie CreateDomain in der AWS SDK für die Kotlin-API-Referenz.

`DeleteDomain`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDomain`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificDomain(domainNameVal: String) {
    val request =
        DeleteDomainRequest {
            domainName = domainNameVal
        }
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        searchClient.deleteDomain(request)
        println("$domainNameVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteDomain in der AWS SDK für die Kotlin-API-Referenz.

`ListDomainNames`

Das folgende Codebeispiel zeigt die Verwendung`ListDomainNames`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllDomains() {
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val response: ListDomainNamesResponse =
 searchClient.listDomainNames(ListDomainNamesRequest {})
        response.domainNames?.forEach { domain ->
            println("Domain name is " + domain.domainName)
        }
    }
}
```

- API-Details finden Sie ListDomainNames in der AWS SDK für die Kotlin-API-Referenz.

`UpdateDomainConfig`

Das folgende Codebeispiel zeigt die Verwendung`UpdateDomainConfig`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateSpecificDomain(domainNameVal: String?) {
    val clusterConfigOb =
        ClusterConfig {
            instanceCount = 3
        }

    val request =
        UpdateDomainConfigRequest {
            domainName = domainNameVal
            clusterConfig = clusterConfigOb
        }

    println("Sending domain update request...")
    OpenSearchClient.fromEnvironment { region = "us-east-1" }.use { searchClient ->
        val updateResponse = searchClient.updateDomainConfig(request)
        println("Domain update response from Amazon OpenSearch Service:")
        println(updateResponse.toString())
    }
}
```

- API-Details finden Sie UpdateDomainConfig in der AWS SDK für die Kotlin-API-Referenz.

EventBridge Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. EventBridge

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo EventBridge

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von EventBridge beginnen.

SDK für Kotlin

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.eventbridge.EventBridgeClient
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesRequest
import aws.sdk.kotlin.services.eventbridge.model.ListEventBusesResponse

suspend fun main() {
    listBusesHello()
}

suspend fun listBusesHello() {
    val request =
        ListEventBusesRequest {
```

```
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val response: ListEventBusesResponse = eventBrClient.listEventBuses(request)
        response.eventBuses?.forEach { bus ->
            println("The name of the event bus is ${bus.name}")
            println("The ARN of the event bus is ${bus.arn}")
        }
    }
}
```

- API-Details finden Sie ListEventBuses in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Regel und fügen Sie ihr ein Ziel hinzu.
- Aktivieren und deaktivieren Sie Regeln.
- Listen Sie Regeln und Ziele auf und aktualisieren Sie sie.
- Senden Sie Ereignisse und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/*
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
```

```
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin example performs the following tasks with Amazon EventBridge:

 1. Creates an AWS Identity and Access Management (IAM) role to use with Amazon
 EventBridge.
 2. Creates an Amazon Simple Storage Service (Amazon S3) bucket with EventBridge
 events enabled.
 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 4. Lists rules on the event bus.
 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and lets the
 user subscribe to it.
 6. Adds a target to the rule that sends an email to the specified topic.
 7. Creates an EventBridge event that sends an email when an Amazon S3 object is
 created.
 8. Lists targets.
 9. Lists the rules for the same target.
 10. Triggers the rule by uploading a file to the S3 bucket.
 11. Disables a specific rule.
 12. Checks and prints the state of the rule.
 13. Adds a transform to the rule to change the text of the email.
 14. Enables a specific rule.
 15. Triggers the updated rule by uploading a file to the S3 bucket.
 16. Updates the rule to a custom rule pattern.
 17. Sends an event to trigger the rule.
 18. Cleans up resources.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <roleName> <bucketName> <topicName> <eventRuleName>

    Where:
        roleName - The name of the role to create.
        bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name to
 create.
        topicName - The name of the Amazon Simple Notification Service (Amazon SNS)
 topic to create.
        eventRuleName - The Amazon EventBridge rule name to create.
    """
    val polJSON =
        "{" +
```

```
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": \"events.amazonaws.com\"" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}"

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val sc = Scanner(System.`in`)
    val roleName = args[0]
    val bucketName = args[1]
    val topicName = args[2]
    val eventRuleName = args[3]

    println(DASHES)
    println("Welcome to the Amazon EventBridge example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS Identity and Access Management (IAM) role to use with
 Amazon EventBridge.")
    val roleArn = createIAMRole(roleName, polJSON)
    println(DASHES)

    println(DASHES)
    println("2. Create an S3 bucket with EventBridge events enabled.")
    if (checkBucket(bucketName)) {
        println("$bucketName already exists. Ending this scenario.")
        exitProcess(1)
    }

    createBucket(bucketName)
    delay(3000)
    setBucketNotification(bucketName)
    println(DASHES)

    println(DASHES)
```

```
   println("3. Create a rule that triggers when an object is uploaded to Amazon
S3.")
   delay(10000)
   addEventRule(roleArn, bucketName, eventRuleName)
   println(DASHES)

   println(DASHES)
   println("4. List rules on the event bus.")
   listRules()
   println(DASHES)

   println(DASHES)
   println("5. Create a new SNS topic for testing and let the user subscribe to the
topic.")
   val topicArn = createSnsTopic(topicName)
   println(DASHES)

   println(DASHES)
   println("6. Add a target to the rule that sends an email to the specified
topic.")
   println("Enter your email to subscribe to the Amazon SNS topic:")
   val email = sc.nextLine()
   subEmail(topicArn, email)
   println("Use the link in the email you received to confirm your subscription.
Then press Enter to continue.")
   sc.nextLine()
   println(DASHES)

   println(DASHES)
   println("7. Create an EventBridge event that sends an email when an Amazon S3
object is created.")
   addSnsEventRule(eventRuleName, topicArn, topicName, eventRuleName, bucketName)
   println(DASHES)

   println(DASHES)
   println("8. List targets.")
   listTargets(eventRuleName)
   println(DASHES)

   println(DASHES)
   println(" 9. List the rules for the same target.")
   listTargetRules(topicArn)
   println(DASHES)
```

```
    println(DASHES)
    println("10. Trigger the rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("11. Disable a specific rule.")
    changeRuleState(eventRuleName, false)
    println(DASHES)

    println(DASHES)
    println("12. Check and print the state of the rule.")
    checkRule(eventRuleName)
    println(DASHES)

    println(DASHES)
    println("13. Add a transform to the rule to change the text of the email.")
    updateSnsEventRule(topicArn, eventRuleName)
    println(DASHES)

    println(DASHES)
    println("14. Enable a specific rule.")
    changeRuleState(eventRuleName, true)
    println(DASHES)

    println(DASHES)
    println("15. Trigger the updated rule by uploading a file to the S3 bucket.")
    println("Press Enter to continue.")
    sc.nextLine()
    uploadTextFiletoS3(bucketName)
    println(DASHES)

    println(DASHES)
    println("16. Update the rule to a custom rule pattern.")
    updateToCustomRule(eventRuleName)
    println("Updated event rule $eventRuleName to use a custom pattern.")
    updateCustomRuleTargetWithTransform(topicArn, eventRuleName)
    println("Updated event target $topicArn.")
    println(DASHES)

    println(DASHES)
```

```
    println("17. Send an event to trigger the rule. This will trigger a subscription
 email.")
    triggerCustomRule(email)
    println("Events have been sent. Press Enter to continue.")
    sc.nextLine()
    println(DASHES)

    println(DASHES)
    println("18. Clean up resources.")
    println("Do you want to clean up resources (y/n)")
    val ans = sc.nextLine()
    if (ans.compareTo("y") == 0) {
        cleanupResources(topicArn, eventRuleName, bucketName, roleName)
    } else {
        println("The resources will not be cleaned up. ")
    }
    println(DASHES)

    println(DASHES)
    println("The Amazon EventBridge example scenario has successfully completed.")
    println(DASHES)
}

suspend fun cleanupResources(
    topicArn: String?,
    eventRuleName: String?,
    bucketName: String?,
    roleName: String?,
) {
    println("Removing all targets from the event rule.")
    deleteTargetsFromRule(eventRuleName)
    deleteRuleByName(eventRuleName)
    deleteSNSTopic(topicArn)
    deleteS3Bucket(bucketName)
    deleteRole(roleName)
}

suspend fun deleteRole(roleNameVal: String?) {
    val policyArnVal = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
    val policyRequest =
        DetachRolePolicyRequest {
            policyArn = policyArnVal
            roleName = roleNameVal
        }
```

```
    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        iam.detachRolePolicy(policyRequest)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")

        // Delete the role.
        val roleRequest =
            DeleteRoleRequest {
                roleName = roleNameVal
            }

        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteS3Bucket(bucketName: String?) {
    // Remove all the objects from the S3 bucket.
    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val myObjects = res.contents
        val toDelete = mutableListOf<ObjectIdentifier>()

        if (myObjects != null) {
            for (myValue in myObjects) {
                toDelete.add(
                    ObjectIdentifier {
                        key = myValue.key
                    },
                )
            }
        }

        val delOb =
            Delete {
                objects = toDelete
            }

        val dor =
            DeleteObjectsRequest {
                bucket = bucketName
```

```
                delete = delOb
            }
        s3Client.deleteObjects(dor)

        // Delete the S3 bucket.
        val deleteBucketRequest =
            DeleteBucketRequest {
                bucket = bucketName
            }
        s3Client.deleteBucket(deleteBucketRequest)
        println("You have deleted the bucket and the objects")
    }
}

// Delete the SNS topic.
suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println(" $topicArnVal was deleted.")
    }
}

suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}

suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
```

```
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}

suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
```

```
        eventBrClient.putEvents(eventsRequest)
    }
}

suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}

suspend fun updateToCustomRule(ruleName: String?) {
    val customEventsPattern =
        "{" +
            "\"source\": [\"ExampleSource\"]," +
            "\"detail-type\": [\"ExampleType\"]" +
            "}"
    val request =
        PutRuleRequest {
            name = ruleName
```

```
            description = "Custom test rule"
            eventPattern = customEventsPattern
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putRule(request)
    }
}

// Update an Amazon S3 object created rule with a transform on the target.
suspend fun updateSnsEventRule(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()
    val myMap = mutableMapOf<String, String>()
    myMap["bucket"] = "$.detail.bucket.name"
    myMap["time"] = "$.time"

    val inputTransOb =
        InputTransformer {
            inputTemplate = "\"Notification: an object was uploaded to bucket
 <bucket> at <time>.\""
            inputPathsMap = myMap
        }
    val targetOb =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(targetOb)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
```

```
}

suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}

suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}

// Create and upload a file to an S3 bucket to trigger an event.
@Throws(IOException::class)
suspend fun uploadTextFiletoS3(bucketName: String?) {
```

```
    val fileSuffix = SimpleDateFormat("yyyyMMddHHmmss").format(Date())
    val fileName = "TextFile$fileSuffix.txt"
    val myFile = File(fileName)
    val fw = FileWriter(myFile.absoluteFile)
    val bw = BufferedWriter(fw)
    bw.write("This is a sample file for testing uploads.")
    bw.close()

    val putOb =
        PutObjectRequest {
            bucket = bucketName
            key = fileName
            body = myFile.asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putObject(putOb)
    }
}

suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}

suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
```

```
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
        }
    }
}

// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
}

suspend fun subEmail(
    topicArnVal: String?,
    email: String?,
) {
    val request =
```

```
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" Subscription ARN: ${result.subscriptionArn}")
    }
}

suspend fun createSnsTopic(topicName: String): String? {
    val topicPolicy = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Sid": "EventBridgePublishTopic",
                "Effect": "Allow",
                "Principal": {
                    "Service": "events.amazonaws.com"
                },
                "Resource": "*",
                "Action": "sns:Publish"
            }
        ]
    }
    """.trimIndent()

    val topicAttributes = mutableMapOf<String, String>()
    topicAttributes["Policy"] = topicPolicy

    val topicRequest =
        CreateTopicRequest {
            name = topicName
            attributes = topicAttributes
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        println("Added topic $topicName for email subscriptions.")
        return response.topicArn
```

```
    }
}

suspend fun listRules() {
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}

// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
```

```
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}

// Set the Amazon S3 bucket notification configuration.
suspend fun setBucketNotification(bucketName: String) {
    val eventBridgeConfig =
        EventBridgeConfiguration {
        }

    val configuration =
        NotificationConfiguration {
            eventBridgeConfiguration = eventBridgeConfig
        }

    val configurationRequest =
        PutBucketNotificationConfigurationRequest {
            bucket = bucketName
            notificationConfiguration = configuration
            skipDestinationValidation = true
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
        s3Client.putBucketNotificationConfiguration(configurationRequest)
        println("Added bucket $bucketName with EventBridge events enabled.")
    }
}

// Create an S3 bucket using a waiter.
suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        s3.waitUntilBucketExists {
```

```
            bucket = bucketName
        }
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String?): Boolean {
    try {
        // Determine if the S3 bucket exists.
        val headBucketRequest =
            HeadBucketRequest {
                bucket = bucketName
            }

        S3Client.fromEnvironment { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            return true
        }
    } catch (e: S3Exception) {
        System.err.println(e.message)
    }
    return false
}

suspend fun createIAMRole(
    rolenameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    val rolePolicyRequest =
        AttachRolePolicyRequest {
            roleName = rolenameVal
            policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess"
        }

    IamClient.fromEnvironment { region = "us-east-1" }.use { iam ->
        val response = iam.createRole(request)
        iam.attachRolePolicy(rolePolicyRequest)
```

```
        return response.role?.arn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule
  - PutTargets

Aktionen

`DeleteRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteRuleByName(ruleName: String?) {
    val ruleRequest =
        DeleteRuleRequest {
            name = ruleName
        }
```

```
    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.deleteRule(ruleRequest)
        println("Successfully deleted the rule")
    }
}
```

- API-Details finden Sie DeleteRule in der AWS SDK für die Kotlin-API-Referenz.

`DescribeRule`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkRule(eventRuleName: String?) {
    val ruleRequest =
        DescribeRuleRequest {
            name = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.describeRule(ruleRequest)
        println("The state of the rule is $response")
    }
}
```

- API-Details finden Sie DescribeRule in der AWS SDK für die Kotlin-API-Referenz.

`DisableRule`

Das folgende Codebeispiel zeigt die Verwendung`DisableRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API-Details finden Sie DisableRule in der AWS SDK für die Kotlin-API-Referenz.

`EnableRule`

Das folgende Codebeispiel zeigt die Verwendung`EnableRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun changeRuleState(
    eventRuleName: String,
    isEnabled: Boolean?,
) {
    if (!isEnabled!!) {
        println("Disabling the rule: $eventRuleName")
        val ruleRequest =
            DisableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient { region = "us-east-1" }.use { eventBrClient ->
            eventBrClient.disableRule(ruleRequest)
        }
    } else {
        println("Enabling the rule: $eventRuleName")
        val ruleRequest =
            EnableRuleRequest {
                name = eventRuleName
            }
        EventBridgeClient.fromEnvironment { region = "us-east-1" }.use
 { eventBrClient ->
            eventBrClient.enableRule(ruleRequest)
        }
    }
}
```

- API-Details finden Sie EnableRule in der AWS SDK für die Kotlin-API-Referenz.

`ListRuleNamesByTarget`

Das folgende Codebeispiel zeigt die Verwendung`ListRuleNamesByTarget`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargetRules(topicArnVal: String?) {
    val ruleNamesByTargetRequest =
        ListRuleNamesByTargetRequest {
            targetArn = topicArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest)
        response.ruleNames?.forEach { rule ->
            println("The rule name is $rule")
        }
    }
}
```

- API-Details finden Sie ListRuleNamesByTarget in der AWS SDK für die Kotlin-API-Referenz.

`ListRules`

Das folgende Codebeispiel zeigt die Verwendung`ListRules`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listRules() {
```

```
    val rulesRequest =
        ListRulesRequest {
            eventBusName = "default"
            limit = 10
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listRules(rulesRequest)
        response.rules?.forEach { rule ->
            println("The rule name is ${rule.name}")
            println("The rule ARN is ${rule.arn}")
        }
    }
}
```

- API-Details finden Sie ListRules in der AWS SDK für die Kotlin-API-Referenz.

`ListTargetsByRule`

Das folgende Codebeispiel zeigt die Verwendung`ListTargetsByRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTargets(ruleName: String?) {
    val ruleRequest =
        ListTargetsByRuleRequest {
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(ruleRequest)
        response.targets?.forEach { target ->
            println("Target ARN: ${target.arn}")
```

```
        }
    }
}
```

- API-Details finden Sie ListTargetsByRule in der AWS SDK für die Kotlin-API-Referenz.

`PutEvents`

Das folgende Codebeispiel zeigt die Verwendung`PutEvents`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun triggerCustomRule(email: String) {
    val json =
        "{" +
            "\"UserEmail\": \"" + email + "\"," +
            "\"Message\": \"This event was generated by example code.\"" +
            "\"UtcTime\": \"Now.\"" +
            "}"

    val entry =
        PutEventsRequestEntry {
            source = "ExampleSource"
            detail = json
            detailType = "ExampleType"
        }

    val eventsRequest =
        PutEventsRequest {
            this.entries = listOf(entry)
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putEvents(eventsRequest)
```

```
    }
}
```

- API-Details finden Sie PutEvents in der AWS SDK für die Kotlin-API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine geplante Regel.

```
suspend fun createScRule(
    ruleName: String?,
    cronExpression: String?,
) {
    val ruleRequest =
        PutRuleRequest {
            name = ruleName
            eventBusName = "default"
            scheduleExpression = cronExpression
            state = RuleState.Enabled
            description = "A test rule that runs on a schedule created by the Kotlin
 API"
        }

    EventBridgeClient.fromEnvironment { region = "us-west-2" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

Erstellen Sie eine Regel, die ausgelöst wird, wenn ein Objekt zu einem Amazon-Simple-Storage-Service-Bucket hinzugefügt wird.

```
// Create a new event rule that triggers when an Amazon S3 object is created in a
 bucket.
suspend fun addEventRule(
    roleArnVal: String?,
    bucketName: String,
    eventRuleName: String?,
) {
    val pattern = """
    {
        "source": ["aws.s3"],
        "detail-type": ["Object Created"],
        "detail": {
            "bucket": {
                "name": ["$bucketName"]
            }
        }
    }
    """.trimIndent()

    val ruleRequest =
        PutRuleRequest {
            description = "Created by using the AWS SDK for Kotlin"
            name = eventRuleName
            eventPattern = pattern
            roleArn = roleArnVal
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- API-Details finden Sie PutRule in der AWS SDK für die Kotlin-API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Add a rule that triggers an SNS target when a file is uploaded to an S3 bucket.
suspend fun addSnsEventRule(
    ruleName: String?,
    topicArn: String?,
    topicName: String,
    eventRuleName: String,
    bucketName: String,
) {
    val targetID = UUID.randomUUID().toString()
    val myTarget =
        Target {
            id = targetID
            arn = topicArn
        }

    val targetsOb = mutableListOf<Target>()
    targetsOb.add(myTarget)

    val request =
        PutTargetsRequest {
            eventBusName = null
            targets = targetsOb
            rule = ruleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(request)
        println("Added event rule $eventRuleName with Amazon SNS target $topicName
 for bucket $bucketName.")
    }
```

```
}
```

Fügen Sie einen Eingabe-Transformator als Ziel für eine Regel hinzu.

```
suspend fun updateCustomRuleTargetWithTransform(
    topicArn: String?,
    ruleName: String?,
) {
    val targetId = UUID.randomUUID().toString()

    val inputTransformerOb =
        InputTransformer {
            inputTemplate = "\"Notification: sample event was received.\""
        }

    val target =
        Target {
            id = targetId
            arn = topicArn
            inputTransformer = inputTransformerOb
        }

    val targetsRequest =
        PutTargetsRequest {
            rule = ruleName
            targets = listOf(target)
            eventBusName = null
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        eventBrClient.putTargets(targetsRequest)
    }
}
```

- API-Details finden Sie PutTargets in der AWS SDK für die Kotlin-API-Referenz.

`RemoveTargets`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTargets`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTargetsFromRule(eventRuleName: String?) {
    // First, get all targets that will be deleted.
    val request =
        ListTargetsByRuleRequest {
            rule = eventRuleName
        }

    EventBridgeClient.fromEnvironment { region = "us-east-1" }.use { eventBrClient -
>
        val response = eventBrClient.listTargetsByRule(request)
        val allTargets = response.targets

        // Get all targets and delete them.
        if (allTargets != null) {
            for (myTarget in allTargets) {
                val removeTargetsRequest =
                    RemoveTargetsRequest {
                        rule = eventRuleName
                        ids = listOf(myTarget.id.toString())
                    }
                eventBrClient.removeTargets(removeTargetsRequest)
                println("Successfully removed the target")
            }
        }
    }
}
```

- API-Details finden Sie RemoveTargets in der AWS SDK für die Kotlin-API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. AWS Glue

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
 <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon
 S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
```

```
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
```

```
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
```

```
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
```

```
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
```

```
            maxResults = 10
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
```

```
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- API-Details finden Sie CreateCrawler in der AWS SDK für die Kotlin-API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- API-Details finden Sie GetCrawler in der AWS SDK für die Kotlin-API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient.fromEnvironment { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- API-Details finden Sie GetDatabase in der AWS SDK für die Kotlin-API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
```

```
            name = crawlerName
        }

    GlueClient.fromEnvironment { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- API-Details finden Sie StartCrawler in der AWS SDK für die Kotlin-API-Referenz.

IAM-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
```

```
        bucketName - The name of the Amazon S3 bucket from which objects are read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
```

```
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }
```

```
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()
```

```
        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
```

```
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}
```

```
suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
```

```
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für die Kotlin-API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- API-Details finden Sie CreateAccessKey in der AWS SDK für die Kotlin-API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API-Details finden Sie CreateAccountAlias in der AWS SDK für die Kotlin-API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "dynamodb:DeleteItem",
                    "dynamodb:GetItem",
                    "dynamodb:PutItem",
                    "dynamodb:Scan",
                    "dynamodb:UpdateItem"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- API-Details finden Sie CreatePolicy in der AWS SDK für die Kotlin-API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API-Details finden Sie CreateUser in der AWS SDK für die Kotlin-API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKey(
```

```
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK für die Kotlin-API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- API-Details finden Sie DeleteAccountAlias in der AWS SDK für die Kotlin-API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API-Details finden Sie DeletePolicy in der AWS SDK für die Kotlin-API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API-Details finden Sie DeleteUser in der AWS SDK für die Kotlin-API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
```

```
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK für die Kotlin-API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- API-Details finden Sie GetPolicy in der AWS SDK für die Kotlin-API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- API-Details finden Sie ListAccessKeys in der AWS SDK für die Kotlin-API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API-Details finden Sie ListAccountAliases in der AWS SDK für die Kotlin-API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- API-Details finden Sie ListUsers in der AWS SDK für die Kotlin-API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- API-Details finden Sie UpdateUser in der AWS SDK für die Kotlin-API-Referenz.

AWS IoT Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. AWS IoT

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für Kotlin

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }
```

```
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie unter listThings in der API-Referenz zum AWS -SDK für Kotlin.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Shadow.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche eine AWS IoT Sache.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
```

```
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-getting-
started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS Internet
 of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an IoT
 thing, generating a device certificate,
        updating the thing with attributes, and so on.
```

```
        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that can
be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
        A device certificate performs a role in securing the communication between
devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
```

```
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)
```

```
    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
```

```
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName? (y/
n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
 delete.")
    }
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase = true))
 {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)
```

```
    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
```

```
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse = iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
```

```
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
```

```
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
```

```
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
```

```
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
```

```
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- API-Details finden Sie AttachThingPrincipal in der AWS SDK für die Kotlin-API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS SDK für die Kotlin-API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIoTThing(thingNameVal: String) {
```

```
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- API-Details finden Sie CreateThing in der AWS SDK für die Kotlin-API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
```

```
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- API-Details finden Sie CreateTopicRule in der AWS SDK für die Kotlin-API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteCertificate in der AWS SDK für die Kotlin-API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- API-Details finden Sie DeleteThing in der AWS SDK für die Kotlin-API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK für die Kotlin-API-Referenz.

`DescribeThing`

Das folgende Codebeispiel zeigt die Verwendung`DescribeThing`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
```

```
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- API-Details finden Sie DescribeThing in der AWS SDK für die Kotlin-API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- API-Details finden Sie DetachThingPrincipal in der AWS SDK für die Kotlin-API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- API-Details finden Sie ListCertificates in der AWS SDK für die Kotlin-API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- API-Details finden Sie SearchIndex in der AWS SDK für die Kotlin-API-Referenz.

`UpdateThing`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThing`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
```

```
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- API-Details finden Sie UpdateThing in der AWS SDK für die Kotlin-API-Referenz.

AWS IoT data Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. AWS IoT data

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}
```

- API-Details finden Sie GetThingShadow in der AWS SDK für die Kotlin-API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateShawdowThing(thingNameVal: String?) {
```

```
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use { iotPlaneClient
 ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}
```

- API-Details finden Sie UpdateThingShadow in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Keyspaces unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Keyspaces Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Keyspaces

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von Amazon Keyspaces beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- API-Details finden Sie ListKeyspaces in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die zeitpunktbezogene Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.
- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
```

```
 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this file
 from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
```

```
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-in-
time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
```

```
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)

    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
```

```
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
```

```
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null
```

```
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }

        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }
```

```
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\"
 WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0 AND
 year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
```

```
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE title
 = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin
\";")
    resultSet.forEach { item: Row ->
```

```
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )
```

```
        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
```

```
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
```

```
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace

- GetTable
- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyspace`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- API-Details finden Sie CreateKeyspace in der AWS SDK für die Kotlin-API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)
```

```
    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- API-Details finden Sie CreateTable in der AWS SDK für die Kotlin-API-Referenz.

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- API-Details finden Sie DeleteKeyspace in der AWS SDK für die Kotlin-API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteTable(
```

```
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- API-Details finden Sie DeleteTable in der AWS SDK für die Kotlin-API-Referenz.

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyspace`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse = keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- API-Details finden Sie GetKeyspace in der AWS SDK für die Kotlin-API-Referenz.

`GetTable`

Das folgende Codebeispiel zeigt die Verwendung`GetTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
```

```
            }
        }
    }
}
```

- API-Details finden Sie GetTable in der AWS SDK für die Kotlin-API-Referenz.

`ListKeyspaces`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyspaces`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- API-Details finden Sie ListKeyspaces in der AWS SDK für die Kotlin-API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK für die Kotlin-API-Referenz.

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung`RestoreTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun restoreTable(
```

```
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- API-Details finden Sie RestoreTable in der AWS SDK für die Kotlin-API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
```

```
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- API-Details finden Sie UpdateTable in der AWS SDK für die Kotlin-API-Referenz.

AWS KMS Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. AWS KMS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createCustomAlias(
    targetKeyIdVal: String?,
    aliasNameVal: String?,
) {
    val request =
        CreateAliasRequest {
            aliasName = aliasNameVal
            targetKeyId = targetKeyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.createAlias(request)
        println("$aliasNameVal was successfully created")
    }
}
```

- API-Details finden Sie CreateAlias in der AWS SDK für die Kotlin-API-Referenz.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGrant(
```

```
    keyIdVal: String?,
    granteePrincipalVal: String?,
    operation: String,
): String? {
    val operationOb = GrantOperation.fromValue(operation)
    val grantOperationList = ArrayList<GrantOperation>()
    grantOperationList.add(operationOb)

    val request =
        CreateGrantRequest {
            keyId = keyIdVal
            granteePrincipal = granteePrincipalVal
            operations = grantOperationList
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.createGrant(request)
        return response.grantId
    }
}
```

- API-Details finden Sie CreateGrant in der AWS SDK für die Kotlin-API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
```

```
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- API-Details finden Sie CreateKey in der AWS SDK für die Kotlin-API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
```

```
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS -SDK für Kotlin.

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }
```

```
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- API-Details finden Sie DescribeKey in der AWS SDK für die Kotlin-API-Referenz.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API-Details finden Sie DisableKey in der AWS SDK für die Kotlin-API-Referenz.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API-Details finden Sie EnableKey in der AWS SDK für die Kotlin-API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()
```

```
    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS -SDK für Kotlin.

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAliases`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllAliases() {
    val request =
        ListAliasesRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- API-Details finden Sie ListAliases in der AWS SDK für die Kotlin-API-Referenz.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
```

```
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- API-Details finden Sie ListGrants in der AWS SDK für die Kotlin-API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllKeys() {
    val request =
        ListKeysRequest {
            limit = 15
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- API-Details finden Sie ListKeys in der AWS SDK für die Kotlin-API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.

- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
 has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
 example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
 that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
 or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
 example, LambdaHello-1.0-SNAPSHOT.jar).
            """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val functionName = args[0]
    val role = args[1]
    val handler = args[2]
    val bucketName = args[3]
```

```
    val updatedBucketName = args[4]
    val key = args[5]

    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("*** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("*** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("*** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    updateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String {
    val functionCode =
        FunctionCode {
```

```
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
    val functionRequest =
        GetFunctionRequest {
            functionName = functionNameVal
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
 ${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {
    val request =
        ListFunctionsRequest {
            maxItems = 10
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
```

```
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            payload = byteArray
            logType = LogType.Tail
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(
    functionNameVal: String?,
    bucketName: String?,
    key: String?,
) {
    val functionCodeRequest =
        UpdateFunctionCodeRequest {
            functionName = functionNameVal
            publish = true
            s3Bucket = bucketName
            s3Key = key
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}
```

```
suspend fun updateFunctionConfiguration(
    functionNameVal: String?,
    handlerVal: String?,
) {
    val configurationRequest =
        UpdateFunctionConfigurationRequest {
            functionName = functionNameVal
            handler = handlerVal
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String,
): String? {
    val functionCode =
        FunctionCode {
            s3Bucket = s3BucketName
            s3Key = myS3Key
        }

    val request =
        CreateFunctionRequest {
            functionName = myFunctionName
            code = functionCode
            description = "Created by the Lambda Kotlin API"
            handler = myHandler
            role = myRole
            runtime = Runtime.Java17
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
```

```
        return functionResponse.functionArn
    }
}
```

- API-Details finden Sie CreateFunction in der AWS SDK für die Kotlin-API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request =
        DeleteFunctionRequest {
            functionName = myFunctionName
        }

    LambdaClient { region = "us-east-1" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- API-Details finden Sie DeleteFunction in der AWS SDK für die Kotlin-API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun invokeFunction(functionNameVal: String) {
    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request =
        InvokeRequest {
            functionName = functionNameVal
            logType = LogType.Tail
            payload = byteArray
        }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Kotlin.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Beispiele für Amazon Location unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Location Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Location

Das folgende Codebeispiel zeigt, wie Sie mit der Nutzung von Amazon Location Service beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you need to create a collection using the AWS Management
console. For information, see the following documentation.

https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:
            <colletionName>

        Where:
            colletionName - The Amazon location collection name.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
    val colletionName = args[0]
```

```
    listGeofences(colletionName)
}

/**
 * Lists the geofences for the specified collection name.
 *
 * @param collectionName the name of the geofence collection
 */
suspend fun listGeofences(collectionName: String) {
    val request = ListGeofencesRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.listGeofences(request)
        val geofences = response.entries
        if (geofences.isNullOrEmpty()) {
            println("No Geofences found")
        } else {
            geofences.forEach { geofence ->
                println("Geofence ID: ${geofence.geofenceId}")
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - ListGeofenceCollections
  - ListGeofences

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Location-Karte.
- Erstellen Sie einen API-Schlüssel für Amazon Location.
- Zeigen Sie die Karten-URL an.

- Erstellen Sie eine Geofence-Sammlung.
- Speichern einer Geofence-Geometrie
- Erstellen einer Tracker-Ressource
- Aktualisieren der Position eines Geräts
- Rufen Sie die neueste Positionsaktualisierung für ein bestimmtes Gerät ab.
- Erstellen Sie einen Routenrechner.
- Ermitteln der Entfernung zwischen Seattle und Vancouver
- Verwenden Sie Amazon-Location-APIs höherer Ebene.
- Löschen Sie die Amazon-Location-Komponenten.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val scanner = Scanner(System.`in`)
val DASHES = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """

        Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

        Where:
            mapName - The name of the map to create (e.g., "AWSMap").
            keyName - The name of the API key to create (e.g., "AWSApiKey").
```

```
            collectionName - The name of the geofence collection (e.g.,
"AWSLocationCollection").
            geoId - The geographic identifier used for the geofence or map (e.g.,
"geoId").
            trackerName - The name of the tracker (e.g., "geoTracker").
            calculatorName - The name of the route calculator (e.g.,
"AWSRouteCalc").
            deviceId - The ID of the device (e.g., "iPhone-112356").
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(0)
    }

    val mapName = args[0]
    val keyName = args[1]
    val collectionName = args[2]
    val geoId = args[3]
    val trackerName = args[4]
    val calculatorName = args[5]
    val deviceId = args[6]

    println(
        """
    AWS Location Service is a fully managed service offered by Amazon Web Services
(AWS) that
    provides location-based services for developers. This service simplifies
    the integration of location-based features into applications, making it
    easier to build and deploy location-aware applications.

    The AWS Location Service offers a range of location-based services,
    including:

    - Maps: The service provides access to high-quality maps, satellite imagery,
      and geospatial data from various providers, allowing developers to
      easily embed maps into their applications.

    - Tracking: The Location Service enables real-time tracking of mobile devices,
      assets, or other entities, allowing developers to build applications
      that can monitor the location of people, vehicles, or other objects.

    - Geocoding: The service provides the ability to convert addresses or
      location names into geographic coordinates (latitude and longitude),
```

```
      and vice versa, enabling developers to integrate location-based search
      and routing functionality into their applications.
        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println(DASHES)
    println("1. Create an AWS Location Service map")
    println(
        """
        An AWS Location map can enhance the user experience of your
        application by providing accurate and personalized location-based
        features. For example, you could use the geocoding capabilities to
        allow users to search for and locate businesses, landmarks, or
        other points of interest within a specific region.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    val mapArn = createMap(mapName)
    println("The Map ARN is: $mapArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    waitForInputToContinue(scanner)
    println("2. Create an AWS Location API key")
    println(
        """
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.

        """.trimIndent(),
    )
    val keyArn = createKey(keyName, mapArn)
    println("The Key ARN is: $keyArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("3. Display Map URL")
```

```
    println(
        """
        In order to get the MAP URL, you need to get the API Key value.
        You can get the key value using the AWS Management Console under
        Location Services. This operation cannot be completed using the
        AWS SDK. For more information about getting the key value, see
        the AWS Location Documentation.
        """.trimIndent(),
    )
    val mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/$mapName/tiles/{z}/
{x}/{y}?key={KeyValue}"
    println("Embed this URL in your Web app: $mapUrl")
    println("")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("4. Create a geofence collection, which manages and stores geofences.")
    waitForInputToContinue(scanner)
    val collectionArn: String =
        createGeofenceCollection(collectionName)
    println("The geofence collection was successfully created: $collectionArn")
    waitForInputToContinue(scanner)

    println(DASHES)
    println("5. Store a geofence geometry in a given geofence collection.")
    println(
        """
        An AWS Location geofence is a virtual boundary that defines a geographic
 area
        on a map. It is a useful feature for tracking the location of
        assets or monitoring the movement of objects within a specific region.

        To define a geofence, you need to specify the coordinates of a
        polygon that represents the area of interest. The polygon must be
        defined in a counter-clockwise direction, meaning that the points of
        the polygon must be listed in a counter-clockwise order.

        This is a requirement for the AWS Location service to correctly
        interpret the geofence and ensure that the location data is
        accurately processed within the defined area.
        """.trimIndent(),
    )
```

```
    waitForInputToContinue(scanner)
    putGeofence(collectionName, geoId)
    println("Successfully created geofence: $geoId")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("6. Create a tracker resource which lets you retrieve current and
historical location of devices.")
    waitForInputToContinue(scanner)
    val trackerArn: String = createTracker(trackerName)
    println("Successfully created tracker. ARN: $trackerArn")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("7. Update the position of a device in the location tracking system.")
    println(
        """
        The AWS location service does not enforce a strict format for deviceId, but
it must:
            - Be a string (case-sensitive).
            - Be 1-100 characters long.
            - Contain only:
            - Alphanumeric characters (A-Z, a-z, 0-9)
            - Underscores (_)
            - Hyphens (-)
            - Be the same ID used when sending and retrieving positions.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    updateDevicePosition(trackerName, deviceId)
    println("$deviceId was successfully updated in the location tracking system.")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("8. Retrieve the most recent position update for a specified device.")
    waitForInputToContinue(scanner)
    val response = getDevicePosition(trackerName, deviceId)
    println("Successfully fetched device position: ${response.position}")
    waitForInputToContinue(scanner)
```

```
    println(DASHES)

    println(DASHES)
    println("9. Create a route calculator.")
    waitForInputToContinue(scanner)
    val routeResponse = createRouteCalculator(calculatorName)
    println("Route calculator created successfully: ${routeResponse.calculatorArn}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("10. Determine the distance in kilometers between Seattle and Vancouver
using the route calculator.")
    waitForInputToContinue(scanner)
    val responseDis = calcDistance(calculatorName)
    println("Successfully calculated route. The distance in kilometers is
${responseDis.summary?.distance}")
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("11. Use the GeoPlacesClient to perform additional operations.")
    println(
        """
        This scenario will show use of the GeoPlacesClient that enables
        location search and geocoding capabilities for your applications.

        We are going to use this client to perform these AWS Location tasks:
            - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
            - Place Search (searchText): Finds places based on search queries.
            - Nearby Search (searchNearby): Finds places near a specific location.

        """.trimIndent(),
    )

    waitForInputToContinue(scanner)
    println("First we will perform a Reverse Geocoding operation")
    waitForInputToContinue(scanner)
    reverseGeocode()

    println("Now we are going to perform a text search using coffee shop.")
    waitForInputToContinue(scanner)
    searchText("coffee shop")
```

```
    waitForInputToContinue(scanner)

    println("Now we are going to perform a nearby Search.")
    waitForInputToContinue(scanner)
    searchNearby()
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS Location Services resources.")
    println("Would you like to delete the AWS Location Services resources? (y/n)")
    val delAns = scanner.nextLine().trim { it <= ' ' }
    if (delAns.equals("y", ignoreCase = true)) {
        deleteMap(mapName)
        deleteKey(keyName)
        deleteGeofenceCollection(collectionName)
        deleteTracker(trackerName)
        deleteRouteCalculator(calculatorName)
    } else {
        println("The AWS resources will not be deleted.")
    }
    waitForInputToContinue(scanner)
    println(DASHES)

    println(DASHES)
    println(" This concludes the AWS Location Service scenario.")
    println(DASHES)
}

/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
    }
}
```

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}


/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}

/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}


/**
 * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
 * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
 * The results are processed and printed once the search completes successfully.
 */
suspend fun searchNearby() {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    // Set up the request for searching nearby places.
    val request = SearchNearbyRequest {
        this.queryPosition = queryPosition
        this.queryRadius = 1000L
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchNearby(request)
```

```
        // Process the response and print the results.
        response.resultItems?.forEach { result ->
            println("Title: ${result.title}")
            println("Address: ${result.address?.label}")
            println("Distance: ${result.distance} meters")
            println("-------------------------")
        }
    }
}


/**
 * Searches for a place using the provided search query and prints the detailed
 information of the first result.
 *
 * @param searchQuery the search query to be used for the place search (ex, coffee
 shop)
 */
suspend fun searchText(searchQuery: String) {
    val latitude = 37.7749
    val longitude = -122.4194
    val queryPosition = listOf(longitude, latitude)

    val request = SearchTextRequest {
        this.queryText = searchQuery
        this.biasPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.searchText(request)

        response.resultItems?.firstOrNull()?.let { result ->
            val placeId = result.placeId // Get Place ID
            println("Found Place with id: $placeId")

            // Fetch detailed info using getPlace.
            val getPlaceRequest = GetPlaceRequest {
                this.placeId = placeId
            }

            val placeResponse = client.getPlace(getPlaceRequest)

            // Print detailed place information.
```

```
            println("Detailed Place Information:")
            println("Title: ${placeResponse.title}")
            println("Address: ${placeResponse.address?.label}")

            // Print each food type (if any).
            placeResponse.foodTypes?.takeIf { it.isNotEmpty() }?.let {
                println("Food Types:")
                it.forEach { foodType ->
                    println("  - $foodType")
                }
            } ?: run {
                println("No food types available.")
            }

            println("-------------------------")
        }
    }
}

/**
 * Performs reverse geocoding using the AWS Geo Places API.
 * Reverse geocoding is the process of converting geographic coordinates (latitude
 and longitude) to a human-readable address.
 * This method uses the latitude and longitude of San Francisco as the input, and
 prints the resulting address.
 */
suspend fun reverseGeocode() {
    val latitude = 37.7749
    val longitude = -122.4194
    println("Use latitude 37.7749 and longitude -122.4194")

    // AWS expects [longitude, latitude].
    val queryPosition = listOf(longitude, latitude)
    val request = ReverseGeocodeRequest {
        this.queryPosition = queryPosition
    }

    GeoPlacesClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.reverseGeocode(request)
        response.resultItems?.forEach { result ->
            println("The address is: ${result.address?.label}")
        }
    }
}
```

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}

/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
```

```
    }
}

/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}

/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
```

```
        client.batchUpdateDevicePosition(request)
    }
}

/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}

/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }
```

```
    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}

/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}

/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
```

```
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}

/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}

fun waitForInputToContinue(scanner: Scanner) {
    while (true) {
        println("")
        println("Enter 'c' followed by <ENTER> to continue:")
        val input = scanner.nextLine()
        if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
            println("Continuing with the program...")
            println("")
            break
        } else {
```

```
            println("Invalid input. Please try again.")
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - BatchUpdateDevicePosition
  - CalculateRoute
  - CreateGeofenceCollection
  - CreateKey
  - CreateMap
  - CreateRouteCalculator
  - CreateTracker
  - DeleteGeofenceCollection
  - DeleteKey
  - DeleteMap
  - DeleteRouteCalculator
  - DeleteTracker
  - GetDevicePosition
  - PutGeofence

Aktionen

`BatchUpdateDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`BatchUpdateDevicePosition`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Updates the position of a device in the location tracking system.
 *
 * @param trackerName the name of the tracker associated with the device
 * @param deviceId    the unique identifier of the device
 */
suspend fun updateDevicePosition(trackerName: String, deviceId: String) {
    val latitude = 37.7749
    val longitude = -122.4194

    val positionUpdate = DevicePositionUpdate {
        this.deviceId = deviceId
        sampleTime = aws.smithy.kotlin.runtime.time.Instant.now() // Timestamp of
 position update.
        position = listOf(longitude, latitude) // AWS requires [longitude, latitude]
    }

    val request = BatchUpdateDevicePositionRequest {
        this.trackerName = trackerName
        updates = listOf(positionUpdate)
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.batchUpdateDevicePosition(request)
    }
}
```

- API-Details finden Sie BatchUpdateDevicePosition in der AWS SDK für die Kotlin-API-Referenz.

`CalculateRoute`

Das folgende Codebeispiel zeigt die Verwendung`CalculateRoute`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Calculates the distance between two locations.
 *
 * @param routeCalcName the name of the route calculator to use
 * @return a {@link CompletableFuture} that will complete with a {@link
 CalculateRouteResponse} containing the distance and estimated duration of the route
 */
suspend fun calcDistance(routeCalcName: String): CalculateRouteResponse {
    // Define coordinates for Seattle, WA and Vancouver, BC.
    val departurePosition = listOf(-122.3321, 47.6062)
    val arrivePosition = listOf(-123.1216, 49.2827)

    val request = CalculateRouteRequest {
        this.calculatorName = routeCalcName
        this.departurePosition = departurePosition
        this.destinationPosition = arrivePosition
        this.travelMode = TravelMode.Car // Options: Car, Truck, Walking, Bicycle
        this.distanceUnit = DistanceUnit.Kilometers // Options: Meters, Kilometers,
 Miles
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.calculateRoute(request)
    }
}
```

- API-Details finden Sie CalculateRoute in der AWS SDK für die Kotlin-API-Referenz.

`CreateGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateGeofenceCollection`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new geofence collection.
 *
 * @param collectionName the name of the geofence collection to be created
 */
suspend fun createGeofenceCollection(collectionName: String): String {
    val collectionRequest = CreateGeofenceCollectionRequest {
        this.collectionName = collectionName
        description = "Created by using the AWS SDK for Kotlin"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createGeofenceCollection(collectionRequest)
        return response.collectionArn
    }
}
```

- API-Details finden Sie CreateGeofenceCollection in der AWS SDK für die Kotlin-API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new API key with the specified name and restrictions.
 *
 * @param keyName the name of the API key to be created
 * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which the
 API key will be associated
 * @return the Amazon Resource Name (ARN) of the created API key
 */
suspend fun createKey(keyName: String, mapArn: String): String {
```

```
    val keyRestrictions = ApiKeyRestrictions {
        allowActions = listOf("geo:GetMap*")
        allowResources = listOf(mapArn)
    }

    val request = CreateKeyRequest {
        this.keyName = keyName
        this.restrictions = keyRestrictions
        noExpiry = true
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createKey(request)
        return response.keyArn
    }
}
```

- API-Details finden Sie CreateKey in der AWS SDK für die Kotlin-API-Referenz.

`CreateMap`

Das folgende Codebeispiel zeigt die Verwendung`CreateMap`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new map with the specified name and configuration.
 *
 * @param mapName the name of the map to be created
 * @return he Amazon Resource Name (ARN) of the created map
 */
suspend fun createMap(mapName: String): String {
    val configuration = MapConfiguration {
        style = "VectorEsriNavigation"
```

```
    }

    val mapRequest = CreateMapRequest {
        this.mapName = mapName
        this.configuration = configuration
        description = "A map created using the Kotlin SDK"
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createMap(mapRequest)
        return response.mapArn
    }
}
```

- API-Details finden Sie CreateMap in der AWS SDK für die Kotlin-API-Referenz.

`CreateRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteCalculator`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new route calculator with the specified name and data source.
 *
 * @param routeCalcName the name of the route calculator to be created
 */
suspend fun createRouteCalculator(routeCalcName: String):
 CreateRouteCalculatorResponse {
    val dataSource = "Esri"

    val request = CreateRouteCalculatorRequest {
        this.calculatorName = routeCalcName
        this.dataSource = dataSource
    }
```

```
    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.createRouteCalculator(request)
    }
}
```

- API-Details finden Sie CreateRouteCalculator in der AWS SDK für die Kotlin-API-Referenz.

`CreateTracker`

Das folgende Codebeispiel zeigt die Verwendung`CreateTracker`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Creates a new tracker resource in your AWS account, which you can use to track
 the location of devices.
 *
 * @param trackerName the name of the tracker to be created
 * @return a {@link CompletableFuture} that, when completed, will contain the Amazon
 Resource Name (ARN) of the created tracker
 */
suspend fun createTracker(trackerName: String): String {
    val trackerRequest = CreateTrackerRequest {
        description = "Created using the Kotlin SDK"
        this.trackerName = trackerName
        positionFiltering = PositionFiltering.TimeBased // Options: TimeBased,
 DistanceBased, AccuracyBased
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        val response = client.createTracker(trackerRequest)
        return response.trackerArn
    }
}
```

- API-Details finden Sie CreateTracker in der AWS SDK für die Kotlin-API-Referenz.

`DeleteGeofenceCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGeofenceCollection`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a geofence collection.
 *
 * @param collectionName the name of the geofence collection to be deleted
 * @return a {@link CompletableFuture} that completes when the geofence collection
 has been deleted
 */
suspend fun deleteGeofenceCollection(collectionName: String) {
    val collectionRequest = DeleteGeofenceCollectionRequest {
        this.collectionName = collectionName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteGeofenceCollection(collectionRequest)
        println("The geofence collection $collectionName was deleted.")
    }
}
```

- API-Details finden Sie DeleteGeofenceCollection in der AWS SDK für die Kotlin-API-Referenz.

`DeleteKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKey`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteKey(keyName: String) {
    val keyRequest = DeleteKeyRequest {
        this.keyName = keyName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteKey(keyRequest)
        println("The key $keyName was deleted.")
    }
}
```

- API-Details finden Sie DeleteKey in der AWS SDK for Kotlin API-Referenz.

`DeleteMap`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMap`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * Deletes the specified key from the key-value store.
 *
 * @param keyName the name of the key to be deleted
 */
suspend fun deleteMap(mapName: String) {
    val mapRequest = DeleteMapRequest {
        this.mapName = mapName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteMap(mapRequest)
        println("The map $mapName was deleted.")
    }
}
```

- API-Details finden Sie DeleteMap in der AWS SDK für die Kotlin-API-Referenz.

`DeleteRouteCalculator`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRouteCalculator`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a route calculator from the system.
 * @param calcName the name of the route calculator to delete
 */
suspend fun deleteRouteCalculator(calcName: String) {
    val calculatorRequest = DeleteRouteCalculatorRequest {
        this.calculatorName = calcName
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.deleteRouteCalculator(calculatorRequest)
        println("The route calculator $calcName was deleted.")
```

```
    }
}
```

- API-Details finden Sie DeleteRouteCalculator in der AWS SDK für die Kotlin-API-Referenz.

`DeleteTracker`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTracker`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Deletes a tracker with the specified name.
 * @param trackerName the name of the tracker to be deleted
 */
suspend fun deleteTracker(trackerName: String) {
    val trackerRequest = DeleteTrackerRequest {
        this.trackerName = trackerName
    }

    LocationClient { region = "us-east-1" }.use { client ->
        client.deleteTracker(trackerRequest)
        println("The tracker $trackerName was deleted.")
    }
}
```

- API-Details finden Sie DeleteTracker in der AWS SDK für die Kotlin-API-Referenz.

`GetDevicePosition`

Das folgende Codebeispiel zeigt die Verwendung`GetDevicePosition`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Retrieves the position of a device using the provided LocationClient.
 *
 * @param trackerName The name of the tracker associated with the device.
 * @param deviceId    The ID of the device to retrieve the position for.
 */
suspend fun getDevicePosition(trackerName: String, deviceId: String):
 GetDevicePositionResponse {
    val request = GetDevicePositionRequest {
        this.trackerName = trackerName
        this.deviceId = deviceId
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        return client.getDevicePosition(request)
    }
}
```

- API-Details finden Sie GetDevicePosition in der AWS SDK für die Kotlin-API-Referenz.

`PutGeofence`

Das folgende Codebeispiel zeigt die Verwendung`PutGeofence`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * Adds a new geofence to the specified collection.
 *
 * @param collectionName the name of the geofence collection to add the geofence to
 * @param geoId          the unique identifier for the geofence
 */
suspend fun putGeofence(collectionName: String, geoId: String) {
    val geofenceGeometry = GeofenceGeometry {
        polygon = listOf(
            listOf(
                listOf(-122.3381, 47.6101),
                listOf(-122.3281, 47.6101),
                listOf(-122.3281, 47.6201),
                listOf(-122.3381, 47.6201),
                listOf(-122.3381, 47.6101),
            ),
        )
    }

    val geofenceRequest = PutGeofenceRequest {
        this.collectionName = collectionName
        this.geofenceId = geoId
        this.geometry = geofenceGeometry
    }

    LocationClient.fromEnvironment { region = "us-east-1" }.use { client ->
        client.putGeofence(geofenceRequest)
    }
}
```

- API-Details finden Sie PutGeofence in der AWS SDK für die Kotlin-API-Referenz.

MediaConvert Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. MediaConvert

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMediaJob(
    mcClient: MediaConvertClient,
    mcRoleARN: String,
    fileInput1: String,
): String? {
    // Step 1: Describe endpoints to get the MediaConvert endpoint URL
    val describeResponse = mcClient.describeEndpoints(
        DescribeEndpointsRequest {
            maxResults = 1
        },
    )

    val endpointUrl = describeResponse.endpoints?.firstOrNull()?.url
        ?: error("No MediaConvert endpoint found")

    // Step 2: Create MediaConvert client with resolved endpoint
    val mediaConvert = MediaConvertClient.fromEnvironment {
        region = "us-west-2"
        endpointProvider = MediaConvertEndpointProvider {
            Endpoint(endpointUrl)
        }
```

```
    }

    // Output destination folder in S3 - put in 'output/' folder beside input
    val outputDestination = fileInput1.substringBeforeLast('/') + "/output/"

    // Step 3: Create the job request with minimal valid video codec settings
    val jobRequest = CreateJobRequest {
        role = mcRoleARN
        settings = JobSettings {
            inputs = listOf(
                Input {
                    fileInput = fileInput1
                },
            )
            outputGroups = listOf(
                OutputGroup {
                    outputGroupSettings = OutputGroupSettings {
                        type = OutputGroupType.FileGroupSettings
                        fileGroupSettings = FileGroupSettings {
                            destination = outputDestination
                        }
                    }
                    outputs = listOf(
                        Output {
                            containerSettings = ContainerSettings {
                                container = ContainerType.Mp4
                            }
                            videoDescription = VideoDescription {
                                width = 1280
                                height = 720
                                codecSettings = VideoCodecSettings {
                                    codec = VideoCodec.H264
                                    h264Settings = H264Settings {
                                        rateControlMode = H264RateControlMode.Qvbr
                                        qvbrSettings = H264QvbrSettings {
                                            qvbrQualityLevel = 7
                                        }
                                        maxBitrate = 5_000_000
                                        codecLevel = H264CodecLevel.Auto
                                        codecProfile = H264CodecProfile.Main
                                        framerateControl =
H264FramerateControl.InitializeFromSource
                                    }
                                }
```

```
                            }
                        },
                    )
                },
            )
        }
    }

    // Step 4: Call MediaConvert to create the job
    val response = mediaConvert.createJob(jobRequest)

    // Return the job ID or null if not found
    return response.job?.id
}
```

- API-Details finden Sie CreateJob in der AWS SDK für die Kotlin-API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSpecificJob(mcClient: MediaConvertClient, jobId: String) {
    // 1. Discover the correct endpoint
    val res = mcClient.describeEndpoints(DescribeEndpointsRequest { maxResults =
 1 })
    var endpointUrl = res.endpoints?.firstOrNull()?.url
        ?: error(" No MediaConvert endpoint found")

    // 2. Create a new client using the endpoint
    val clientWithEndpoint = MediaConvertClient {
        region = "us-west-2"
        endpointUrl = endpointUrl
```

```
    }

    // 3. Get the job details
    val jobResponse = clientWithEndpoint.getJob(GetJobRequest { id = jobId })
    val job = jobResponse.job

    println("Job status: ${job?.status}")
    println("Job ARN: ${job?.arn}")
    println("Output group count: ${job?.settings?.outputGroups?.size}")
}
```

- API-Details finden Sie GetJob in der AWS SDK für die Kotlin-API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCompleteJobs(mcClient: MediaConvertClient) {
    val describeEndpoints =
        DescribeEndpointsRequest {
            maxResults = 20
        }

    val res = mcClient.describeEndpoints(describeEndpoints)
    if (res.endpoints?.size!! <= 0) {
        println("Cannot find MediaConvert service endpoint URL!")
        exitProcess(0)
    }
    val endpointURL = res.endpoints!![0].url!!
    val mediaConvert =
        MediaConvertClient.fromEnvironment {
            region = "us-west-2"
```

```
            endpointProvider =
                MediaConvertEndpointProvider {
                    Endpoint(endpointURL)
                }
        }

    val jobsRequest =
        ListJobsRequest {
            maxResults = 10
            status = JobStatus.fromValue("COMPLETE")
        }

    val jobsResponse = mediaConvert.listJobs(jobsRequest)
    val jobs = jobsResponse.jobs
    if (jobs != null) {
        for (job in jobs) {
            println("The JOB ARN is ${job.arn}")
        }
    }
}
```

- API-Details finden Sie ListJobs in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Pinpoint unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateApp`

Das folgende Codebeispiel zeigt die Verwendung. `CreateApp`

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createApplication(applicationName: String?): String? {
    val createApplicationRequestOb =
        CreateApplicationRequest {
            name = applicationName
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.createApp(
                CreateAppRequest {
                    createApplicationRequest = createApplicationRequestOb
                },
            )
        return result.applicationResponse?.id
    }
}
```

- API-Details finden Sie CreateApp in der AWS SDK für die Kotlin-API-Referenz.

`CreateCampaign`

Das folgende Codebeispiel zeigt die Verwendung`CreateCampaign`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createPinCampaign(
    appId: String,
    segmentIdVal: String,
) {
    val scheduleOb =
        Schedule {
            startTime = "IMMEDIATE"
        }

    val defaultMessageOb =
        Message {
            action = Action.OpenApp
            body = "My message body"
            title = "My message title"
        }

    val messageConfigurationOb =
        MessageConfiguration {
            defaultMessage = defaultMessageOb
        }

    val writeCampaign =
        WriteCampaignRequest {
            description = "My description"
            schedule = scheduleOb
            name = "MyCampaign"
            segmentId = segmentIdVal
            messageConfiguration = messageConfigurationOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse =
            pinpoint.createCampaign(
                CreateCampaignRequest {
```

```
                    applicationId = appId
                    writeCampaignRequest = writeCampaign
                },
            )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- API-Details finden Sie CreateCampaign in der AWS SDK für die Kotlin-API-Referenz.

`CreateSegment`

Das folgende Codebeispiel zeigt die Verwendung`CreateSegment`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {
    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts =
        AttributeDimension {
            attributeType = AttributeType.Inclusive
            values = myList
        }

    segmentAttributes["Team"] = atts
    val recencyDimension =
        RecencyDimension {
            duration = Duration.fromValue("DAY_30")
            recencyType = RecencyType.fromValue("ACTIVE")
        }
```

```
    val segmentBehaviors =
        SegmentBehaviors {
            recency = recencyDimension
        }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb =
        SegmentDimensions {
            attributes = segmentAttributes
            behavior = segmentBehaviors
            demographic = SegmentDemographics {}
            location = segmentLocation
        }

    val writeSegmentRequestOb =
        WriteSegmentRequest {
            name = "MySegment101"
            dimensions = dimensionsOb
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse =
            pinpoint.createSegment(
                CreateSegmentRequest {
                    applicationId = applicationIdVal
                    writeSegmentRequest = writeSegmentRequestOb
                },
            )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- API-Details finden Sie CreateSegment in der AWS SDK für die Kotlin-API-Referenz.

`DeleteApp`

Das folgende Codebeispiel zeigt die Verwendung`DeleteApp`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinApp(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.deleteApp(
                DeleteAppRequest {
                    applicationId = appId
                },
            )
        val appName = result.applicationResponse?.name
        println("Application $appName has been deleted.")
    }
}
```

- API-Details finden Sie DeleteApp in der AWS SDK für die Kotlin-API-Referenz.

`DeleteEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEndpoint`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deletePinEncpoint(
    appIdVal: String?,
    endpointIdVal: String?,
) {
```

```
    val deleteEndpointRequest =
        DeleteEndpointRequest {
            applicationId = appIdVal
            endpointId = endpointIdVal
        }

    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is  $id")
    }
}
```

- API-Details finden Sie DeleteEndpoint in der AWS SDK für die Kotlin-API-Referenz.

`GetEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`GetEndpoint`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun lookupPinpointEndpoint(
    appId: String?,
    endpoint: String?,
) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val result =
            pinpoint.getEndpoint(
                GetEndpointRequest {
                    applicationId = appId
                    endpointId = endpoint
                },
            )
        val endResponse = result.endpointResponse
```

```
        // Uses the Google Gson library to pretty print the endpoint JSON.
        val gson: com.google.gson.Gson =
            GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting()
                .create()

        val endpointJson: String = gson.toJson(endResponse)
        println(endpointJson)
    }
}
```

- API-Details finden Sie GetEndpoint in der AWS SDK für die Kotlin-API-Referenz.

`GetSegments`

Das folgende Codebeispiel zeigt die Verwendung`GetSegments`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSegs(appId: String?) {
    PinpointClient.fromEnvironment { region = "us-west-2" }.use { pinpoint ->
        val response =
            pinpoint.getSegments(
                GetSegmentsRequest {
                    applicationId = appId
                },
            )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

- API-Details finden Sie GetSegments in der AWS SDK für die Kotlin-API-Referenz.

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

val body: String =
    """
    Amazon Pinpoint test (AWS SDK for Kotlin)

    This email was sent through the Amazon Pinpoint Email API using the AWS SDK for
 Kotlin.

    """.trimIndent()

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <subject> <appId> <senderAddress> <toAddress>

    Where:
        subject - The email subject to use.
        senderAddress - The from address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
```

```
        toAddress - The to address. This address has to be verified in Amazon
 Pinpoint in the region you're using to send email
    """

    if (args.size != 3) {
        println(usage)
        exitProcess(0)
    }

    val subject = args[0]
    val senderAddress = args[1]
    val toAddress = args[2]
    sendEmail(subject, senderAddress, toAddress)
}

suspend fun sendEmail(
    subjectVal: String?,
    senderAddress: String,
    toAddressVal: String,
) {
    var content =
        Content {
            data = body
        }

    val messageBody =
        Body {
            text = content
        }

    val subContent =
        Content {
            data = subjectVal
        }

    val message =
        Message {
            body = messageBody
            subject = subContent
        }

    val destinationOb =
        Destination {
            toAddresses = listOf(toAddressVal)
```

```
        }

    val emailContent =
        EmailContent {
            simple = message
        }

    val sendEmailRequest =
        SendEmailRequest {
            fromEmailAddress = senderAddress
            destination = destinationOb
            this.content = emailContent
        }

    PinpointEmailClient.fromEnvironment { region = "us-east-1" }.use { pinpointemail
 ->
        pinpointemail.sendEmail(sendEmailRequest)
        println("Message Sent")
    }
}
```

- API-Details finden Sie SendMessages in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon RDS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this code example, set up your development environment, including
 your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:
```

```
https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html

This example performs the following tasks:

1. Returns a list of the available DB engines by invoking the
 DescribeDbEngineVersions method.
2. Selects an engine family and create a custom DB parameter group by invoking the
 createDBParameterGroup method.
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.
4. Gets parameters in the group by invoking the DescribeDbParameters method.
5. Modifies both the auto_increment_offset and auto_increment_increment parameters
 by invoking the modifyDbParameterGroup method.
6. Gets and displays the updated parameters.
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions
 method.
8. Gets a list of micro instance classes available for the selected engine.
9. Creates an Amazon Relational Database Service (Amazon RDS) database instance that
 contains a MySQL database and uses the parameter group.
10. Waits for DB instance to be ready and prints out the connection endpoint value.
11. Creates a snapshot of the DB instance.
12. Waits for the DB snapshot to be ready.
13. Deletes the DB instance.
14. Deletes the parameter group.
 */

var sleepTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
 <dbSnapshotIdentifier><secretName>

        Where:
            dbGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """
```

```
    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val dbSnapshotIdentifier = args[4]
    val secretName = args[5]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
engine")
    getMicroInstances()
```

```
    println("9. Create an RDS database instance that contains a MySql database and
 uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
 username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(
    dbGroupName: String,
    dbARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
```

```
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest =
            DeleteDbParameterGroupRequest {
                dbParameterGroupName = dbGroupName
            }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(
    dbInstanceIdentifierVal: String?,
```

```
    dbSnapshotIdentifierVal: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbSnapshotsRequest {
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    while (!snapshotReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(
    dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?,
) {
    val snapshotRequest =
        CreateDbSnapshotRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbSnapshotIdentifier = dbSnapshotIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
```

```
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(
    dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?,
    dbNameVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
```

```
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNameVal
            dbParameterGroupName = dbGroupNameVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
```

```
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "mysql"
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.Immediate
            parameterValue = "5"
        }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
 successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(
    dbGroupName: String?,
    flag: Int,
```

```
) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
            }
        } else {
            DescribeDbParametersRequest {
                dbParameterGroupName = dbGroupName
                source = "user"
            }
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
 auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    System.out.println("*** The parameter value is
 ${para.parameterValue}")
                    System.out.println("*** The parameter data type is
 ${para.dataType}")
                    System.out.println("*** The parameter description is
 ${para.description}")
                    System.out.println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest =
        DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
```

```
        }
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(
    dbGroupName: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is
 ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions
```

```
        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
 ${engineOb.engineVersion}")
            }
        }
    }
}

suspend fun getSecretValues(secretName: String?): String? {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-west-2" }.use
 { secretsClient ->
        val valueResponse = secretsClient.getSecretValue(valueRequest)
        return valueResponse.secretString
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters

- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            allocatedStorage = 100
            dbName = dbNamedbVal
            engine = "mysql"
            dbInstanceClass = "db.t3.micro"
            engineVersion = "8.4"
            storageType = "gp2"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
```

```
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true
                    } else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Kotlin.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Kotlin.

`DescribeAccountAttributes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAccountAttributes`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAccountAttributes() {
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})
        response.accountQuotas?.forEach { quotas ->
            val response = response.accountQuotas
            println("Name is: ${quotas.accountQuotaName}")
            println("Max value is ${quotas.max}")
        }
    }
}
```

- API-Details finden Sie DescribeAccountAttributes in der AWS SDK für die Kotlin-API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeInstances() {
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})
        response.dbInstances?.forEach { instance ->
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")
            println("The Engine is ${instance.engine}")
            println("Connection endpoint is ${instance.endpoint?.address}")
        }
    }
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Kotlin.

`ModifyDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBInstance`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun updateIntance(
    dbInstanceIdentifierVal: String?,
    masterUserPasswordVal: String?,
) {
    val request =
        ModifyDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            publiclyAccessible = true
            masterUserPassword = masterUserPasswordVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val instanceResponse = rdsClient.modifyDbInstance(request)
        println("The ARN of the modified database is
 ${instanceResponse.dbInstance?.dbInstanceArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter ModifyDBInstance in der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Redshift unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Redshift Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCluster`

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
suspend fun createCluster(
    clusterId: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?,
) {
    val clusterRequest =
        CreateClusterRequest {
            clusterIdentifier = clusterId
            availabilityZone = "us-east-1a"
            masterUsername = masterUsernameVal
            masterUserPassword = masterUserPasswordVal
            nodeType = "ra3.4xlarge"
            publiclyAccessible = true
            numberOfNodes = 2
        }

    RedshiftClient.fromEnvironment { region = "us-east-1" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}
```

- API-Details finden Sie CreateCluster in der AWS SDK für die Kotlin-API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
    val request =
        DeleteClusterRequest {
            clusterIdentifier = clusterId
            skipFinalClusterSnapshot = true
        }

    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- API-Details finden Sie DeleteCluster in der AWS SDK für die Kotlin-API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
suspend fun describeRedshiftClusters() {
```

```
    RedshiftClient.fromEnvironment { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
 redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK für die Kotlin-API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest =
        ModifyClusterRequest {
            clusterIdentifier = clusterId
            preferredMaintenanceWindow = "wed:07:30-wed:08:00"
        }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println(
```

```
            "The modified cluster was successfully modified and has
 ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window",
        )
    }
}
```

- API-Details finden Sie ModifyCluster in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Redshift-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Beispiele für Amazon Rekognition unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
```

```
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches with
 ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
 ${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- API-Details finden Sie CompareFaces in der AWS SDK für die Kotlin-API-Referenz.

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateCollection`.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- API-Details finden Sie CreateCollection in der AWS SDK für die Kotlin-API-Referenz.

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCollection`.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- API-Details finden Sie DeleteCollection in der AWS SDK für die Kotlin-API-Referenz.

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFaces`.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
```

```
        println("$faceIdVal was deleted from the collection")
    }
}
```

- API-Details finden Sie DeleteFaces in der AWS SDK for Kotlin API-Referenz.

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCollection`.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- API-Details finden Sie DescribeCollection in der AWS SDK für die Kotlin-API-Referenz.

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung`DetectFaces`.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low}
 and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- API-Details finden Sie DetectFaces in der AWS SDK für die Kotlin-API-Referenz.

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- API-Details finden Sie DetectLabels in der AWS SDK für die Kotlin-API-Referenz.

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectModerationLabels`.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- API-Details finden Sie DetectModerationLabels in der AWS SDK für die Kotlin-API-Referenz.

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung`DetectText`.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- API-Details finden Sie DetectText in der AWS SDK für die Kotlin-API-Referenz.

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung`IndexFaces`.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")
```

```
            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- API-Details finden Sie IndexFaces in der AWS SDK für die Kotlin-API-Referenz.

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung`ListCollections`.

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- API-Details finden Sie ListCollections in der AWS SDK für die Kotlin-API-Referenz.

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung`ListFaces`.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- API-Details finden Sie ListFaces in der AWS SDK für die Kotlin-API-Referenz.

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung`RecognizeCelebrities`.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- API-Details finden Sie RecognizeCelebrities in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Informationen in Videos erkennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie Amazon-Rekognition-Aufträge, um Elemente wie Personen, Objekte und Text in Videos zu erkennen.
- Überprüfen Sie den Auftragsstatus, bis die Aufträge abgeschlossen sind.
- Gibt die Liste der von jedem Auftrag erkannten Elemente aus.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erkennen von Gesichtern in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startFaceDetection(
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
```

```
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Erkennen von unangemessenen oder anstößigen Inhalten in einem Video, das in einem Amazon-S3-Bucket gespeichert ist.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
```

```
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection

- StartTextDetection

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für die Route-53-Domainregistrierung unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Route 53-Domainregistrierung Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Route-53-Domainregistrierung

Das folgende Codebeispiel zeigt, wie Sie mit der Route 53-Domainregistrierung beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <domainType>

       Where:
           domainType - The domain type (for example, com).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }
```

```
    val domainType = args[0]
    println("Invokes ListPrices using a Paginated method.")
    listPricesPaginated(domainType)
}

suspend fun listPricesPaginated(domainType: String) {
    val pricesRequest =
        ListPricesRequest {
            maxItems = 10
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API-Details finden Sie ListPrices in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Auflisten der aktuellen Domains und der Vorgänge des letzten Jahres
- Anzeigen der Abrechnung für das vergangene Jahr und der Preise für Domaintypen

- Abrufen von Domainvorschlägen
- Überprüfen der Verfügbarkeit und Übertragbarkeit von Domains
- Optional: Anfordern einer Domainregistrierung
- Abrufen eines Vorgangsdetails
- Optional: Abrufen eines Domaindetails

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin code example performs the following operations:

1. List current domains.
2. List operations in the past year.
3. View billing for the account in the past year.
4. View prices for domain types.
5. Get domain suggestions.
6. Check domain availability.
7. Check domain transferability.
8. Request a domain registration.
9. Get operation details.
10. Optionally, get domain details.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
```

```
        Usage:
            <domainType> <phoneNumber> <email> <domainSuggestion> <firstName>
<lastName> <city>
        Where:
           domainType - The domain type (for example, com).
           phoneNumber - The phone number to use (for example, +1.2065550100)
           email - The email address to use.
           domainSuggestion - The domain suggestion (for example, findmy.example).
           firstName - The first name to use to register a domain.
           lastName -  The last name to use to register a domain.
           city - The city to use to register a domain.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val domainType = args[0]
    val phoneNumber = args[1]
    val email = args[2]
    val domainSuggestion = args[3]
    val firstName = args[4]
    val lastName = args[5]
    val city = args[6]

    println(DASHES)
    println("Welcome to the Amazon Route 53 domains example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List current domains.")
    listDomains()
    println(DASHES)

    println(DASHES)
    println("2. List operations in the past year.")
    listOperations()
    println(DASHES)

    println(DASHES)
    println("3. View billing for the account in the past year.")
    listBillingRecords()
    println(DASHES)
```

```
    println(DASHES)
    println("4. View prices for domain types.")
    listAllPrices(domainType)
    println(DASHES)

    println(DASHES)
    println("5. Get domain suggestions.")
    listDomainSuggestions(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("6. Check domain availability.")
    checkDomainAvailability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("7. Check domain transferability.")
    checkDomainTransferability(domainSuggestion)
    println(DASHES)

    println(DASHES)
    println("8. Request a domain registration.")
    val opId = requestDomainRegistration(domainSuggestion, phoneNumber, email,
 firstName, lastName, city)
    println(DASHES)

    println(DASHES)
    println("9. Get operation details.")
    getOperationalDetail(opId)
    println(DASHES)

    println(DASHES)
    println("10. Get domain details.")
    println("Note: You must have a registered domain to get details.")
    println("Otherwise an exception is thrown that states ")
    println("Domain xxxxxxx not found in xxxxxxx account.")
    getDomainDetails(domainSuggestion)
    println(DASHES)
}

suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
```

```
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}

suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}

suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
```

```
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}

suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}

suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
```

```
        println("$domainSuggestion is ${response.availability}")
    }
}

suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}

suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
```

```
}

suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}

suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}

suspend fun listDomains() {
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

Aktionen

`CheckDomainAvailability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainAvailability`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainAvailability(domainSuggestion: String) {
    val availabilityRequest =
        CheckDomainAvailabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainAvailability(availabilityRequest)
        println("$domainSuggestion is ${response.availability}")
    }
}
```

- API-Details finden Sie CheckDomainAvailability in der AWS SDK für die Kotlin-API-Referenz.

`CheckDomainTransferability`

Das folgende Codebeispiel zeigt die Verwendung`CheckDomainTransferability`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun checkDomainTransferability(domainSuggestion: String?) {
    val transferabilityRequest =
        CheckDomainTransferabilityRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response =
 route53DomainsClient.checkDomainTransferability(transferabilityRequest)
        println("Transferability: ${response.transferability?.transferable}")
    }
}
```

- API-Details finden Sie CheckDomainTransferability in der AWS SDK für die Kotlin-API-Referenz.

`GetDomainDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainDetail`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getDomainDetails(domainSuggestion: String?) {
    val detailRequest =
        GetDomainDetailRequest {
            domainName = domainSuggestion
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainDetail(detailRequest)
        println("The contact first name is
 ${response.registrantContact?.firstName}")
        println("The contact last name is ${response.registrantContact?.lastName}")
```

```
        println("The contact org name is
 ${response.registrantContact?.organizationName}")
    }
}
```

- API-Details finden Sie GetDomainDetail in der AWS SDK für die Kotlin-API-Referenz.

`GetDomainSuggestions`

Das folgende Codebeispiel zeigt die Verwendung`GetDomainSuggestions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomainSuggestions(domainSuggestion: String?) {
    val suggestionsRequest =
        GetDomainSuggestionsRequest {
            domainName = domainSuggestion
            suggestionCount = 5
            onlyAvailable = true
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getDomainSuggestions(suggestionsRequest)
        response.suggestionsList?.forEach { suggestion ->
            println("Suggestion Name: ${suggestion.domainName}")
            println("Availability: ${suggestion.availability}")
            println(" ")
        }
    }
}
```

- API-Details finden Sie GetDomainSuggestions in der AWS SDK für die Kotlin-API-Referenz.

`GetOperationDetail`

Das folgende Codebeispiel zeigt die Verwendung`GetOperationDetail`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getOperationalDetail(opId: String?) {
    val detailRequest =
        GetOperationDetailRequest {
            operationId = opId
        }
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.getOperationDetail(detailRequest)
        println("Operation detail message is ${response.message}")
    }
}
```

- API-Details finden Sie GetOperationDetail in der AWS SDK für die Kotlin-API-Referenz.

`ListDomains`

Das folgende Codebeispiel zeigt die Verwendung`ListDomains`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listDomains() {
```

```
    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listDomainsPaginated(ListDomainsRequest {})
            .transform { it.domains?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("The domain name is ${content.domainName}")
            }
    }
}
```

- API-Details finden Sie ListDomains in der AWS SDK für die Kotlin-API-Referenz.

`ListOperations`

Das folgende Codebeispiel zeigt die Verwendung`ListOperations`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listOperations() {
    val currentDate = Date()
    var localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
    localDateTime = localDateTime.minusYears(1)
    val myTime: java.time.Instant? = localDateTime.toInstant(zoneOffset)
    val time2: Instant? = myTime?.let { Instant(it) }
    val operationsRequest =
        ListOperationsRequest {
            submittedSince = time2
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
```

```
        route53DomainsClient
            .listOperationsPaginated(operationsRequest)
            .transform { it.operations?.forEach { obj -> emit(obj) } }
            .collect { content ->
                println("Operation Id: ${content.operationId}")
                println("Status: ${content.status}")
                println("Date: ${content.submittedDate}")
            }
    }
}
```

- API-Details finden Sie ListOperations in der AWS SDK für die Kotlin-API-Referenz.

`ListPrices`

Das folgende Codebeispiel zeigt die Verwendung`ListPrices`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllPrices(domainType: String?) {
    val pricesRequest =
        ListPricesRequest {
            tld = domainType
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .listPricesPaginated(pricesRequest)
            .transform { it.prices?.forEach { obj -> emit(obj) } }
            .collect { pr ->
                println("Registration: ${pr.registrationPrice}
 ${pr.registrationPrice?.currency}")
                println("Renewal: ${pr.renewalPrice?.price}
 ${pr.renewalPrice?.currency}")
```

```
                println("Transfer: ${pr.transferPrice?.price}
 ${pr.transferPrice?.currency}")
                println("Restoration: ${pr.restorationPrice?.price}
 ${pr.restorationPrice?.currency}")
            }
    }
}
```

- API-Details finden Sie ListPrices in der AWS SDK für die Kotlin-API-Referenz.

`RegisterDomain`

Das folgende Codebeispiel zeigt die Verwendung`RegisterDomain`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun requestDomainRegistration(
    domainSuggestion: String?,
    phoneNumberVal: String?,
    emailVal: String?,
    firstNameVal: String?,
    lastNameVal: String?,
    cityVal: String?,
): String? {
    val contactDetail =
        ContactDetail {
            contactType = ContactType.Company
            state = "LA"
            countryCode = CountryCode.In
            email = emailVal
            firstName = firstNameVal
            lastName = lastNameVal
            city = cityVal
            phoneNumber = phoneNumberVal
            organizationName = "My Org"
```

```
            addressLine1 = "My Address"
            zipCode = "123 123"
        }

    val domainRequest =
        RegisterDomainRequest {
            adminContact = contactDetail
            registrantContact = contactDetail
            techContact = contactDetail
            domainName = domainSuggestion
            autoRenew = true
            durationInYears = 1
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        val response = route53DomainsClient.registerDomain(domainRequest)
        println("Registration requested. Operation Id: ${response.operationId}")
        return response.operationId
    }
}
```

- API-Details finden Sie RegisterDomain in der AWS SDK für die Kotlin-API-Referenz.

`ViewBilling`

Das folgende Codebeispiel zeigt die Verwendung`ViewBilling`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBillingRecords() {
    val currentDate = Date()
    val localDateTime =
 currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime()
    val zoneOffset = ZoneOffset.of("+01:00")
```

```
    val localDateTime2 = localDateTime.minusYears(1)
    val myStartTime = localDateTime2.toInstant(zoneOffset)
    val myEndTime = localDateTime.toInstant(zoneOffset)
    val timeStart: Instant? = myStartTime?.let { Instant(it) }
    val timeEnd: Instant? = myEndTime?.let { Instant(it) }

    val viewBillingRequest =
        ViewBillingRequest {
            start = timeStart
            end = timeEnd
        }

    Route53DomainsClient.fromEnvironment { region = "us-east-1" }.use
 { route53DomainsClient ->
        route53DomainsClient
            .viewBillingPaginated(viewBillingRequest)
            .transform { it.billingRecords?.forEach { obj -> emit(obj) } }
            .collect { billing ->
                println("Bill Date: ${billing.billDate}")
                println("Operation: ${billing.operation}")
                println("Price: ${billing.price}")
            }
    }
}
```

- API-Details finden Sie ViewBilling in der AWS SDK für die Kotlin-API-Referenz.

Amazon-S3-Beispiele unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
```

```
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for example,
 C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
```

```
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
```

```
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
```

```
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
```

```
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- API-Details finden Sie CopyObject in der AWS SDK für die Kotlin-API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- API-Details finden Sie CreateBucket in der AWS SDK für die Kotlin-API-Referenz.

`CreateMultiRegionAccessPoint`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultiRegionAccessPoint`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Steuerungs-Client so, dass er eine Anfrage an die Region us-west-2 sendet.

```
    suspend fun createS3ControlClient(): S3ControlClient {
        // Configure your S3ControlClient to send requests to US West (Oregon).
        val s3Control = S3ControlClient.fromEnvironment {
            region = "us-west-2"
        }
        return s3Control
    }
```

Erstellen Sie den Multi-Region Access Point.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                    Region {
                        bucket = bucketName2
```

```
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

Warten Sie, bis der Multi-Region Access Point verfügbar ist.

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )

            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
```

```
                delay(timeBetweenChecks)
                describeResponse =
 s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- API-Details finden Sie CreateMultiRegionAccessPoint in der AWS SDK für die Kotlin-API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK für die Kotlin-API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- API-Details finden Sie DeleteObjects in der AWS SDK für die Kotlin-API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK für die Kotlin-API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- API-Details finden Sie GetObject in der AWS SDK für die Kotlin-API-Referenz.

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAcl`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- API-Details finden Sie GetObjectAcl in der AWS SDK für die Kotlin-API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für die Kotlin-API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
```

```
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- API-Details finden Sie PutBucketAcl in der AWS SDK für die Kotlin-API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- API-Details finden Sie PutObject in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine vorsignierte `GetObject`-Anfrage und verwenden Sie die URL, um ein Objekt herunterzuladen.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET request
 to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Erstellen Sie eine `GetObject`-vorsignierte Anfrage mit erweiterten Optionen.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
```

```
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be used
 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Erstellen Sie eine vorsignierte `PutObject`-Anfrage und verwenden Sie sie, um ein Objekt hochzuladen.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
```

```
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Ruft ein Objekt von einem Multi-Region Access Point ab

Das folgende Codebeispiel zeigt, wie ein Objekt von einem Multi-Region Access Point abgerufen wird.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konfigurieren Sie den S3-Client so, dass er den Signaturalgorithmus Asymmetric Sigv4 (SigV4a) verwendet.

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric SigV4 (SigV4a) signing
algorithm.
```

```
            val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4aScheme)
            }
            return s3
        }
```

Verwenden Sie den Multi-Region Access Point-ARN anstelle eines Bucket-Namens, um das Objekt abzurufen.

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS SDK für Kotlin.
- API-Details finden Sie GetObject in der AWS SDK für die Kotlin-API-Referenz.

SageMaker KI-Beispiele, die SDK für Kotlin verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit SageMaker KI Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo KI SageMaker

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von SageMaker KI beginnen können.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listBooks() {
    SageMakerClient.fromEnvironment { region = "us-west-2" }.use { sageMakerClient -
>
        val response =
 sageMakerClient.listNotebookInstances(ListNotebookInstancesRequest {})
        response.notebookInstances?.forEach { item ->
            println("The notebook name is: ${item.notebookInstanceName}")
        }
    }
}
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK für die Kotlin-API-Referenz.

Aktionen

`CreatePipeline`

Das folgende Codebeispiel zeigt die Verwendung`CreatePipeline`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
```

```
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}
```

- API-Details finden Sie CreatePipeline in der AWS SDK für die Kotlin-API-Referenz.

`DeletePipeline`

Das folgende Codebeispiel zeigt die Verwendung`DeletePipeline`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
}
```

- API-Details finden Sie DeletePipeline in der AWS SDK für die Kotlin-API-Referenz.

`DescribePipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribePipelineExecution`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}
```

- API-Details finden Sie DescribePipelineExecution in der AWS SDK für die Kotlin-API-Referenz.

`StartPipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartPipelineExecution`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
```

```
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
```

```
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}
```

- API-Details finden Sie StartPipelineExecution in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erste Schritte mit Geodatenaufträgen und Pipelines

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie Ressourcen für eine Pipeline ein.
- Richten Sie eine Pipeline ein, die einen Geodatenauftrag ausführt.
- Starten Sie eine Pipeline-Ausführung.
- Überwachen Sie den Status der Ausführung.
- Zeigen Sie die Ausgabe der Pipeline an.
- Bereinigen Sie die Ressourcen.

Weitere Informationen finden Sie unter Erstellen und Ausführen von SageMaker Pipelines mithilfe von AWS SDKs on. Community.aws

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
private var eventSourceMapping = ""

suspend fun main(args: Array<String>) {
    val usage = """
```

```
    Usage:
          <sageMakerRoleName> <lambdaRoleName> <functionName> <functionKey>
 <queueName> <bucketName> <bucketFunction> <lnglatData> <spatialPipelinePath>
 <pipelineName>

    Where:
        sageMakerRoleName - The name of the Amazon SageMaker role.
        lambdaRoleName - The name of the AWS Lambda role.
        functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).
        functionKey - The name of the Amazon S3 key name that represents the Lambda
 function (for example, SageMakerLambda.zip).
        queueName - The name of the Amazon Simple Queue Service (Amazon SQS) queue.
        bucketName - The name of the Amazon Simple Storage Service (Amazon S3)
 bucket.
        bucketFunction - The name of the Amazon S3 bucket that contains the Lambda
 ZIP file.
        lnglatData - The file location of the latlongtest.csv file required for this
 use case.
        spatialPipelinePath - The file location of the GeoSpatialPipeline.json file
 required for this use case.
        pipelineName - The name of the pipeline to create (for example, sagemaker-
sdk-example-pipeline).
    """

    if (args.size != 10) {
        println(usage)
        exitProcess(1)
    }

    val sageMakerRoleName = args[0]
    val lambdaRoleName = args[1]
    val functionKey = args[2]
    val functionName = args[3]
    val queueName = args[4]
    val bucketName = args[5]
    val bucketFunction = args[6]
    val lnglatData = args[7]
    val spatialPipelinePath = args[8]
    val pipelineName = args[9]
    val handlerName = "org.example.SageMakerLambdaFunction::handleRequest"

    println(DASHES)
    println("Welcome to the Amazon SageMaker pipeline example scenario.")
```

```
    println(
        """
         This example workflow will guide you through setting up and running an
         Amazon SageMaker pipeline. The pipeline uses an AWS Lambda function and an
         Amazon SQS Queue. It runs a vector enrichment reverse geocode job to
         reverse geocode addresses in an input file and store the results in an
export file.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("First, we will set up the roles, functions, and queue needed by the
SageMaker pipeline.")
    val lambdaRoleArn: String = checkLambdaRole(lambdaRoleName)
    val sageMakerRoleArn: String = checkSageMakerRole(sageMakerRoleName)
    val functionArn = checkFunction(functionName, bucketFunction, functionKey,
handlerName, lambdaRoleArn)
    val queueUrl = checkQueue(queueName, functionName)
    println(DASHES)

    println(DASHES)
    println("Setting up bucket $bucketName")
    if (!checkBucket(bucketName)) {
        setupBucket(bucketName)
        println("Put $lnglatData into $bucketName")
        val objectKey = "samplefiles/latlongtest.csv"
        putS3Object(bucketName, objectKey, lnglatData)
    }
    println(DASHES)

    println(DASHES)
    println("Now we can create and run our pipeline.")
    setupPipeline(spatialPipelinePath, sageMakerRoleArn, functionArn, pipelineName)
    val pipelineExecutionARN = executePipeline(bucketName, queueUrl,
sageMakerRoleArn, pipelineName)
    println("The pipeline execution ARN value is $pipelineExecutionARN")
    waitForPipelineExecution(pipelineExecutionARN)
    println("Wait 30 secs to get output results $bucketName")
    TimeUnit.SECONDS.sleep(30)
    getOutputResults(bucketName)
    println(DASHES)

    println(DASHES)
```

```
    println(
        """
             The pipeline has completed. To view the pipeline and runs in SageMaker
 Studio, follow these instructions:
             https://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-studio.html
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println("Do you want to delete the AWS resources used in this Workflow? (y/n)")
    val `in` = Scanner(System.`in`)
    val delResources = `in`.nextLine()
    if (delResources.compareTo("y") == 0) {
        println("Lets clean up the AWS resources. Wait 30 seconds")
        TimeUnit.SECONDS.sleep(30)
        deleteEventSourceMapping(functionName)
        deleteSQSQueue(queueName)
        listBucketObjects(bucketName)
        deleteBucket(bucketName)
        delLambdaFunction(functionName)
        deleteLambdaRole(lambdaRoleName)
        deleteSagemakerRole(sageMakerRoleName)
        deletePipeline(pipelineName)
    } else {
        println("The AWS Resources were not deleted!")
    }
    println(DASHES)

    println(DASHES)
    println("SageMaker pipeline scenario is complete.")
    println(DASHES)
}

// Delete a SageMaker pipeline by name.
suspend fun deletePipeline(pipelineNameVal: String) {
    val pipelineRequest = DeletePipelineRequest {
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        sageMakerClient.deletePipeline(pipelineRequest)
        println("*** Successfully deleted $pipelineNameVal")
    }
```

```
}

suspend fun deleteSagemakerRole(roleNameVal: String) {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in sageMakerRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}

suspend fun deleteLambdaRole(roleNameVal: String) {
    val lambdaRolePolicies = getLambdaRolePolicies()
    IamClient { region = "us-west-2" }.use { iam ->
        for (policy in lambdaRolePolicies) {
            // First the policy needs to be detached.
            val rolePolicyRequest = DetachRolePolicyRequest {
                policyArn = policy
                roleName = roleNameVal
            }
            iam.detachRolePolicy(rolePolicyRequest)
        }

        // Delete the role.
        val roleRequest = DeleteRoleRequest {
            roleName = roleNameVal
        }
        iam.deleteRole(roleRequest)
        println("*** Successfully deleted $roleNameVal")
    }
}
```

```
suspend fun delLambdaFunction(myFunctionName: String) {
    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

suspend fun deleteBucketObjects(bucketName: String, objectName: String?) {
    val toDelete = ArrayList<ObjectIdentifier>()
    val obId = ObjectIdentifier {
        key = objectName
    }
    toDelete.add(obId)
    val delOb = Delete {
        objects = toDelete
    }
    val dor = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.deleteObjects(dor)
        println("*** $bucketName objects were deleted.")
    }
}

suspend fun listBucketObjects(bucketNameVal: String) {
    val listObjects = ListObjectsRequest {
        bucket = bucketNameVal
```

```
    }

    S3Client { region = "us-east-1" }.use { s3Client ->
        val res = s3Client.listObjects(listObjects)
        val objects = res.contents
        if (objects != null) {
            for (myValue in objects) {
                println("The name of the key is ${myValue.key}")
                deleteBucketObjects(bucketNameVal, myValue.key)
            }
        }
    }
}

// Delete the specific Amazon SQS queue.
suspend fun deleteSQSQueue(queueNameVal: String?) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val urlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = urlVal
        }
        sqsClient.deleteQueue(deleteQueueRequest)
    }
}

// Delete the queue event mapping.
suspend fun deleteEventSourceMapping(functionNameVal: String) {
    if (eventSourceMapping.compareTo("") == 0) {
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val request = ListEventSourceMappingsRequest {
                functionName = functionNameVal
            }
            val response = lambdaClient.listEventSourceMappings(request)
            val eventList = response.eventSourceMappings
            if (eventList != null) {
                for (event in eventList) {
                    eventSourceMapping = event.uuid.toString()
                }
            }
        }
```

```
    }

    val eventSourceMappingRequest = DeleteEventSourceMappingRequest {
        uuid = eventSourceMapping
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest)
        println("The event mapping is deleted!")
    }
}

// Reads the objects in the S3 bucket and displays the values.
private suspend fun readObject(bucketName: String, keyVal: String?) {
    println("Output file contents: \n")
    val objectRequest = GetObjectRequest {
        bucket = bucketName
        key = keyVal
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        s3Client.getObject(objectRequest) { resp ->
            val byteArray = resp.body?.toByteArray()
            val text = byteArray?.let { String(it, StandardCharsets.UTF_8) }
            println("Text output: $text")
        }
    }
}

// Display the results from the output directory.
suspend fun getOutputResults(bucketName: String?) {
    println("Getting output results $bucketName.")
    val listObjectsRequest = ListObjectsRequest {
        bucket = bucketName
        prefix = "outputfiles/"
    }
    S3Client { region = "us-east-1" }.use { s3Client ->
        val response = s3Client.listObjects(listObjectsRequest)
        val s3Objects: List<Object>? = response.contents
        if (s3Objects != null) {
            for (`object` in s3Objects) {
                if (bucketName != null) {
                    readObject(bucketName, (`object`.key))
                }
            }
        }
```

```
    }
}

suspend fun waitForPipelineExecution(executionArn: String?) {
    var status: String
    var index = 0
    do {
        val pipelineExecutionRequest = DescribePipelineExecutionRequest {
            pipelineExecutionArn = executionArn
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            val response =
 sageMakerClient.describePipelineExecution(pipelineExecutionRequest)
            status = response.pipelineExecutionStatus.toString()
            println("$index. The status of the pipeline is $status")
            TimeUnit.SECONDS.sleep(4)
            index++
        }
    } while ("Executing" == status)
    println("Pipeline finished with status $status")
}

// Start a pipeline run with job configurations.
suspend fun executePipeline(bucketName: String, queueUrl: String?, roleArn: String?,
 pipelineNameVal: String): String? {
    println("Starting pipeline execution.")
    val inputBucketLocation = "s3://$bucketName/samplefiles/latlongtest.csv"
    val output = "s3://$bucketName/outputfiles/"

    val gson = GsonBuilder()
        .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
        .setPrettyPrinting()
        .create()

    // Set up all parameters required to start the pipeline.
    val parameters: MutableList<Parameter> = java.util.ArrayList<Parameter>()

    val para1 = Parameter {
        name = "parameter_execution_role"
        value = roleArn
    }
    val para2 = Parameter {
        name = "parameter_queue_url"
```

```
        value = queueUrl
    }

    val inputJSON = """{
        "DataSourceConfig": {
        "S3Data": {
            "S3Uri": "s3://$bucketName/samplefiles/latlongtest.csv"
        },
        "Type": "S3_DATA"
        },
        "DocumentType": "CSV"
    }"""
    println(inputJSON)
    val para3 = Parameter {
        name = "parameter_vej_input_config"
        value = inputJSON
    }

    // Create an ExportVectorEnrichmentJobOutputConfig object.
    val jobS3Data = VectorEnrichmentJobS3Data {
        s3Uri = output
    }

    val outputConfig = ExportVectorEnrichmentJobOutputConfig {
        s3Data = jobS3Data
    }

    val gson4: String = gson.toJson(outputConfig)
    val para4: Parameter = Parameter {
        name = "parameter_vej_export_config"
        value = gson4
    }
    println("parameter_vej_export_config:" + gson.toJson(outputConfig))

    val para5JSON =
        "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":{\"XAttributeName\":
\"Longitude\",\"YAttributeName\":\"Latitude\"}}"

    val para5: Parameter = Parameter {
        name = "parameter_step_1_vej_config"
        value = para5JSON
    }

    parameters.add(para1)
```

```
    parameters.add(para2)
    parameters.add(para3)
    parameters.add(para4)
    parameters.add(para5)

    val pipelineExecutionRequest = StartPipelineExecutionRequest {
        pipelineExecutionDescription = "Created using Kotlin SDK"
        pipelineExecutionDisplayName = "$pipelineName-example-execution"
        pipelineParameters = parameters
        pipelineName = pipelineNameVal
    }

    SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
        val response =
 sageMakerClient.startPipelineExecution(pipelineExecutionRequest)
        return response.pipelineExecutionArn
    }
}

// Create a pipeline from the example pipeline JSON.
suspend fun setupPipeline(filePath: String?, roleArnVal: String?, functionArnVal:
 String?, pipelineNameVal: String?) {
    println("Setting up the pipeline.")
    val parser = JSONParser()

    // Read JSON and get pipeline definition.
    FileReader(filePath).use { reader ->
        val obj: Any = parser.parse(reader)
        val jsonObject: JSONObject = obj as JSONObject
        val stepsArray: JSONArray = jsonObject.get("Steps") as JSONArray
        for (stepObj in stepsArray) {
            val step: JSONObject = stepObj as JSONObject
            if (step.containsKey("FunctionArn")) {
                step.put("FunctionArn", functionArnVal)
            }
        }
        println(jsonObject)

        // Create the pipeline.
        val pipelineRequest = CreatePipelineRequest {
            pipelineDescription = "Kotlin SDK example pipeline"
            roleArn = roleArnVal
            pipelineName = pipelineNameVal
            pipelineDefinition = jsonObject.toString()
```

```
        }

        SageMakerClient { region = "us-west-2" }.use { sageMakerClient ->
            sageMakerClient.createPipeline(pipelineRequest)
        }
    }
}

suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {
    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        body = File(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putObject(request)
        println("Successfully placed $objectKey into bucket $bucketName")
    }
}

suspend fun setupBucket(bucketName: String) {
    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun checkBucket(bucketName: String): Boolean {
    try {
        val headBucketRequest = HeadBucketRequest {
            bucket = bucketName
        }
        S3Client { region = "us-east-1" }.use { s3Client ->
            s3Client.headBucket(headBucketRequest)
            println("$bucketName exists")
            return true
        }
    } catch (e: S3Exception) {
        println("Bucket does not exist")
```

```
    }
    return false
}

// Connect the queue to the Lambda function as an event source.
suspend fun connectLambda(queueUrlVal: String?, lambdaNameVal: String?) {
    println("Connecting the Lambda function and queue for the pipeline.")
    var queueArn = ""

    // Specify the attributes to retrieve.
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)
    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val queueAtts = response.attributes
        if (queueAtts != null) {
            for ((key, value) in queueAtts) {
                println("Key = $key, Value = $value")
                queueArn = value
            }
        }
    }
    val eventSourceMappingRequest = CreateEventSourceMappingRequest {
        eventSourceArn = queueArn
        functionName = lambdaNameVal
    }
    LambdaClient { region = "us-west-2" }.use { lambdaClient ->
        val response1 =
 lambdaClient.createEventSourceMapping(eventSourceMappingRequest)
        eventSourceMapping = response1.uuid.toString()
        println("The mapping between the event source and Lambda function was
 successful")
    }
}

// Set up the SQS queue to use with the pipeline.
suspend fun setupQueue(queueNameVal: String, lambdaNameVal: String): String {
    println("Setting up queue named $queueNameVal")
    val queueAtt: MutableMap<String, String> = HashMap()
```

```
    queueAtt.put("DelaySeconds", "5")
    queueAtt.put("ReceiveMessageWaitTimeSeconds", "5")
    queueAtt.put("VisibilityTimeout", "300")

    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
        attributes = queueAtt
    }

    SqsClient { region = "us-west-2" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("\nGet queue url")
        val getQueueUrlResponse = sqsClient.getQueueUrl(GetQueueUrlRequest
 { queueName = queueNameVal })
        TimeUnit.SECONDS.sleep(15)
        connectLambda(getQueueUrlResponse.queueUrl, lambdaNameVal)
        println("Queue ready with Url " + getQueueUrlResponse.queueUrl)
        return getQueueUrlResponse.queueUrl.toString()
    }
}

// Checks to see if the Amazon SQS queue exists. If not, this method creates a new
 queue
// and returns the ARN value.
suspend fun checkQueue(queueNameVal: String, lambdaNameVal: String): String? {
    println("Checking to see if the queue exists. If not, a new queue will be
 created for use in this workflow.")
    var queueUrl: String
    try {
        val request = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-west-2" }.use { sqsClient ->
            val response = sqsClient.getQueueUrl(request)
            queueUrl = response.queueUrl.toString()
            println(queueUrl)
        }
    } catch (e: SqsException) {
        println(e.message + " A new queue will be created")
        queueUrl = setupQueue(queueNameVal, lambdaNameVal)
    }
    return queueUrl
}
```

```
suspend fun createNewFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java11
        memorySize = 1024
        timeout = 200
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        println("${functionResponse.functionArn} was created")
        return functionResponse.functionArn.toString()
    }
}

suspend fun checkFunction(myFunctionName: String, s3BucketName: String, myS3Key:
 String, myHandler: String, myRole: String): String {
    println("Checking to see if the function exists. If not, a new AWS Lambda
 function will be created for use in this workflow.")
    var functionArn: String
    try {
        // Does this function already exist.
        val functionRequest = GetFunctionRequest {
            functionName = myFunctionName
        }
        LambdaClient { region = "us-west-2" }.use { lambdaClient ->
            val response = lambdaClient.getFunction(functionRequest)
            functionArn = response.configuration?.functionArn.toString()
            println("$functionArn exists")
        }
```

```
    } catch (e: LambdaException) {
        println(e.message + " A new function will be created")
        functionArn = createNewFunction(myFunctionName, s3BucketName, myS3Key,
 myHandler, myRole)
    }
    return functionArn
}

// Checks to see if the SageMaker role exists. If not, this method creates it.
suspend fun checkSageMakerRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS SageMaker to use.")
    var roleArn: String
    try {
        val roleRequest = GetRoleRequest {
            roleName = roleNameVal
        }
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
        println(e.message + " A new role will be created")
        roleArn = createSageMakerRole(roleNameVal)
    }
    return roleArn
}

suspend fun createSageMakerRole(roleNameVal: String): String {
    val sageMakerRolePolicies = getSageMakerRolePolicies()
    println("Creating a role to use with SageMaker.")
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
```

```
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in sageMakerRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
        TimeUnit.SECONDS.sleep(15)
        System.out.println("Role ready with ARN ${roleResult.role?.arn}")
        return roleResult.role?.arn.toString()
    }
}

// Checks to see if the Lambda role exists. If not, this method creates it.
suspend fun checkLambdaRole(roleNameVal: String): String {
    println("Checking to see if the role exists. If not, a new role will be created
 for AWS Lambda to use.")
    var roleArn: String
    val roleRequest = GetRoleRequest {
        roleName = roleNameVal
    }

    try {
        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.getRole(roleRequest)
            roleArn = response.role?.arn.toString()
            println(roleArn)
        }
    } catch (e: IamException) {
```

```
        println(e.message + " A new role will be created")
        roleArn = createLambdaRole(roleNameVal)
    }

    return roleArn
}

private suspend fun createLambdaRole(roleNameVal: String): String {
    val lambdaRolePolicies = getLambdaRolePolicies()
    val assumeRolePolicy = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "\"Service\": [" +
        "\"sagemaker.amazonaws.com\"," +
        "\"sagemaker-geospatial.amazonaws.com\"," +
        "\"lambda.amazonaws.com\"," +
        "\"s3.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}"

    val request = CreateRoleRequest {
        roleName = roleNameVal
        assumeRolePolicyDocument = assumeRolePolicy
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val roleResult = iamClient.createRole(request)

        // Attach the policies to the role.
        for (policy in lambdaRolePolicies) {
            val attachRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policy
            }
            iamClient.attachRolePolicy(attachRequest)
        }

        // Allow time for the role to be ready.
```

```
        TimeUnit.SECONDS.sleep(15)
        println("Role ready with ARN " + roleResult.role?.arn)
        return roleResult.role?.arn.toString()
    }
}

fun getLambdaRolePolicies(): Array<String?> {
    val lambdaRolePolicies = arrayOfNulls<String>(5)
    lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy"
    lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole"
    return lambdaRolePolicies
}

fun getSageMakerRolePolicies(): Array<String?> {
    val sageMakerRolePolicies = arrayOfNulls<String>(3)
    sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess"
    sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess"
    sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess"
    return sageMakerRolePolicies
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Beispiele für Secrets Manager unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Secrets Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getValue(secretName: String?) {
    val valueRequest =
        GetSecretValueRequest {
            secretId = secretName
        }

    SecretsManagerClient.fromEnvironment { region = "us-east-1" }.use
 { secretsClient ->
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
```

```
    }
}
```

- API-Details finden Sie GetSecretValue in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon SES unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie man die Amazon-DynamoDB-API verwendet, um eine dynamische Webanwendung zu erstellen, die DynamoDB-Arbeitsdaten verfolgt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB

- Amazon SES

Erstellen einer Webanwendung zur Verfolgung von Amazon-Redshift-Daten

Das folgende Codebeispiel zeigt, wie Sie eine Webanwendung erstellen, die Arbeitselemente mit einer Amazon-Redshift-Datenbank verfolgt und darüber berichtet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die in einer Amazon-Redshift-Datenbank gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Redshift-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Redshift
- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Kotlin

Zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente, die in einer Amazon RDS-Datenbank gespeichert sind, verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer Spring-REST-API, die Amazon Aurora Serverless-Daten abfragt, und zur Verwendung durch eine React-Anwendung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Beispiele für Amazon SNS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SNS beginnen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
```

```
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK für die Kotlin-API-Referenz.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API-Details finden Sie CreateTopic in der AWS SDK für die Kotlin-API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteTopic in der AWS SDK für die Kotlin-API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK für die Kotlin-API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSSubscriptions() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API-Details finden Sie ListSubscriptions in der AWS SDK für die Kotlin-API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listSNSTopics() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK für die Kotlin-API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
```

```
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
```

```
}
```

- API-Details finden Sie SetTopicAttributes in der AWS SDK für die Kotlin-API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Abonnieren Sie eine Lambda-Funktion für ein Thema.

```
suspend fun subLambda(
```

```
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für Kotlin.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

    val tag2 =
```

```
        Tag {
            key = "Environment"
            value = "Gamma"
        }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request =
        TagResourceRequest {
            resourceArn = topicArn
            tags = tagList
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API-Details finden Sie TagResource in der AWS SDK für die Kotlin-API-Referenz.

`Abonnement beenden`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun unSub(subscriptionArnVal: String) {
    val request =
        UnsubscribeRequest {
            subscriptionArn = subscriptionArnVal
        }
```

```
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS SDK für Kotlin.

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS -Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
```

```
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Kotlin.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
```

```
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
```

```
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
queue to the topic.
                You can select from several options for configuring the topic and
the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
published and determined to have the same deduplication ID,
```

```
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
```

```
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
```

```
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
```

```
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)

    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)
```

```
    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
```

```
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }
```

```
            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
 aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
```

```
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
```

```
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
```

```
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
```

```
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic

- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Beispiele für Amazon SQS unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.kotlin.sqs

import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.paginators.listQueuesPaginated
import kotlinx.coroutines.flow.transform

suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient
            .listQueuesPaginated { }
            .transform { it.queueUrls?.forEach { queue -> emit(queue) } }
            .collect { queue ->
                println("The Queue URL is $queue")
            }
    }
}
```

- API-Details finden Sie ListQueues in der AWS SDK für die Kotlin-API-Referenz.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest =
        CreateQueueRequest {
            queueName = queueNameVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest =
            GetQueueUrlRequest {
                queueName = queueNameVal
            }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- API-Details finden Sie CreateQueue in der AWS SDK für die Kotlin-API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API-Details finden Sie DeleteMessage in der AWS SDK für die Kotlin-API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest =
        PurgeQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request =
        DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- API-Details finden Sie DeleteQueue in der AWS SDK für die Kotlin-API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest =
        ListQueuesRequest {
            queueNamePrefix = prefix
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- API-Details finden Sie ListQueues in der AWS SDK für die Kotlin-API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun receiveMessages(queueUrlVal: String?) {
    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest =
        ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für die Kotlin-API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendMessages(
    queueUrlVal: String,
    message: String,
) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest =
        SendMessageRequest {
            queueUrl = queueUrlVal
```

```
            messageBody = message
            delaySeconds = 10
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 =
        SendMessageBatchRequestEntry {
            id = "id1"
            messageBody = "Hello from msg 1"
        }

    val msg2 =
        SendMessageBatchRequestEntry {
            id = "id2"
            messageBody = "Hello from msg 2"
        }

    val sendMessageBatchRequest =
        SendMessageBatchRequest {
            queueUrl = queueUrlVal
            entries = listOf(msg1, msg2)
        }

    SqsClient.fromEnvironment { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- API-Details finden Sie SendMessage in der AWS SDK für die Kotlin-API-Referenz.

Szenarien

Erstellen einer Nachrichtenanwendung

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS eine Nachrichtenanwendung erstellen können.

SDK für Kotlin

Zeigt, wie Sie die Amazon-SQS-API verwenden können, um eine Spring-REST-API zu entwickeln, die Nachrichten sendet und abruft.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon SQS

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
```

```
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and queues.")
    println(
        """
                In this scenario, you will create an SNS topic and subscribe an SQS
 queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication and
 message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
        selectFIFO = true
```

```
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically generated
 from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://docs.aws.amazon.com/
sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended to
 the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
    }
```

```
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
    println("6. Subscribe to the SQS queue.")
```

```
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/latest/
dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's subscription
 to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the following
 \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following \"tone
\" attributes.")
            println("1. cheerful")
```

```
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key to
 continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was deleted")
    }
}

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
    }
```

```
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
```

```
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
```

```
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
```

```
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic " +
 topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
            val jsonString = "{\"tone\": []}"
```

```
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
    SqsClient { region = "us-east-1" }.use { sqsClient ->
```

```
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
```

```
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

suspend fun createSNSTopic(topicName: String?): String? {
    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Beispiele für Step Functions unter Verwendung von SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Step Functions

Das folgende Codebeispiel zeigt, wie Sie mit Step Functions beginnen.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API-Details finden Sie ListStateMachines in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.iam.IamClient
import aws.sdk.kotlin.services.iam.model.CreateRoleRequest
import aws.sdk.kotlin.services.sfn.SfnClient
import aws.sdk.kotlin.services.sfn.model.CreateActivityRequest
import aws.sdk.kotlin.services.sfn.model.CreateStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DeleteActivityRequest
import aws.sdk.kotlin.services.sfn.model.DeleteStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.DescribeExecutionRequest
import aws.sdk.kotlin.services.sfn.model.DescribeStateMachineRequest
import aws.sdk.kotlin.services.sfn.model.GetActivityTaskRequest
import aws.sdk.kotlin.services.sfn.model.ListActivitiesRequest
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest
import aws.sdk.kotlin.services.sfn.model.SendTaskSuccessRequest
import aws.sdk.kotlin.services.sfn.model.StartExecutionRequest
```

```
import aws.sdk.kotlin.services.sfn.model.StateMachineType
import aws.sdk.kotlin.services.sfn.paginators.listActivitiesPaginated
import aws.sdk.kotlin.services.sfn.paginators.listStateMachinesPaginated
import com.fasterxml.jackson.databind.JsonNode
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.databind.node.ObjectNode
import kotlinx.coroutines.flow.transform
import java.util.Scanner
import java.util.UUID
import kotlin.collections.ArrayList
import kotlin.system.exitProcess

/**
 To run this code example, place the chat_sfn_state_machine.json file into your
 project's resources folder.

 You can obtain the JSON file to create a state machine in the following GitHub
 location:

 https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files

 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This Kotlin code example performs the following tasks:

 1. List activities using a paginator.
 2. List state machines using a paginator.
 3. Creates an activity.
 4. Creates a state machine.
 5. Describes the state machine.
 6. Starts execution of the state machine and interacts with it.
 7. Describes the execution.
 8. Deletes the activity.
 9. Deletes the state machine.
 */

val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
    val usage = """
```

```
    Usage:
        <roleARN> <activityName> <stateMachineName>

    Where:
        roleName - The name of the IAM role to create for this state machine.
        activityName - The name of an activity to create.
        stateMachineName - The name of the state machine to create.
        jsonFile - The location of the chat_sfn_state_machine.json file. You can
located it in resources/sample_files.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(0)
    }

    val roleName = args[0]
    val activityName = args[1]
    val stateMachineName = args[2]
    val jsonFile = args[3]
    val sc = Scanner(System.`in`)
    var action = false

    val polJSON = """{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "states.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
    }"""

    println(DASHES)
    println("Welcome to the AWS Step Functions example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List activities using a Paginator.")
    listActivitesPagnator()
```

```
    println(DASHES)

    println(DASHES)
    println("2. List state machines using a paginator.")
    listStatemachinesPagnator()
    println(DASHES)

    println(DASHES)
    println("3. Create a new activity.")
    val activityArn = createActivity(activityName)
    println("The ARN of the Activity is $activityArn")
    println(DASHES)

    // Get JSON to use for the state machine and place the activityArn value into
it.
    val stream = GetStream()
    val jsonString = stream.getStream(jsonFile)

    // Modify the Resource node.
    val objectMapper = ObjectMapper()
    val root: JsonNode = objectMapper.readTree(jsonString)
    (root.path("States").path("GetInput") as ObjectNode).put("Resource",
activityArn)

    // Convert the modified Java object back to a JSON string.
    val stateDefinition = objectMapper.writeValueAsString(root)
    println(stateDefinition)

    println(DASHES)
    println("4. Create a state machine.")
    val roleARN = createIAMRole(roleName, polJSON)
    val stateMachineArn = createMachine(roleARN, stateMachineName, stateDefinition)
    println("The ARN of the state machine is $stateMachineArn")
    println(DASHES)

    println(DASHES)
    println("5. Describe the state machine.")
    describeStateMachine(stateMachineArn)
    println("What should ChatSFN call you?")
    val userName = sc.nextLine()
    println("Hello $userName")
    println(DASHES)

    println(DASHES)
```

```
    // The JSON to pass to the StartExecution call.
    val executionJson = "{ \"name\" : \"$userName\" }"
    println(executionJson)
    println("6. Start execution of the state machine and interact with it.")
    val runArn = startWorkflow(stateMachineArn, executionJson)
    println("The ARN of the state machine execution is $runArn")
    var myList: List<String>
    while (!action) {
        myList = getActivityTask(activityArn)
        println("ChatSFN: " + myList[1])
        println("$userName please specify a value.")
        val myAction = sc.nextLine()
        if (myAction.compareTo("done") == 0) {
            action = true
        }
        println("You have selected $myAction")
        val taskJson = "{ \"action\" : \"$myAction\" }"
        println(taskJson)
        sendTaskSuccess(myList[0], taskJson)
    }
    println(DASHES)

    println(DASHES)
    println("7. Describe the execution.")
    describeExe(runArn)
    println(DASHES)

    println(DASHES)
    println("8. Delete the activity.")
    deleteActivity(activityArn)
    println(DASHES)

    println(DASHES)
    println("9. Delete the state machines.")
    deleteMachine(stateMachineArn)
    println(DASHES)

    println(DASHES)
    println("The AWS Step Functions example scenario is complete.")
    println(DASHES)
}

suspend fun listStatemachinesPagnator() {
    val machineRequest =
```

```
        ListStateMachinesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listStateMachinesPaginated(machineRequest)
            .transform { it.stateMachines?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The state machine ARN is ${obj.stateMachineArn}")
            }
    }
}

suspend fun listActivitesPagnator() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient
            .listActivitiesPaginated(activitiesRequest)
            .transform { it.activities?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" The activity ARN is ${obj.activityArn}")
            }
    }
}

suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}

suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
```

```
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}

suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}

suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
```

```
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}

suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
}

suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}

suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
```

```
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}

suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}

suspend fun createIAMRole(
    roleNameVal: String?,
    polJSON: String?,
): String? {
    val request =
        CreateRoleRequest {
            roleName = roleNameVal
            assumeRolePolicyDocument = polJSON
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}
```

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Kotlin.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt die Verwendung`CreateActivity`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createActivity(activityName: String): String? {
    val activityRequest =
        CreateActivityRequest {
            name = activityName
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createActivity(activityRequest)
        return response.activityArn
    }
}
```

- API-Details finden Sie CreateActivity in der AWS SDK für die Kotlin-API-Referenz.

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`CreateStateMachine`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createMachine(
    roleARNVal: String?,
    stateMachineName: String?,
    jsonVal: String?,
): String? {
```

```
    val machineRequest =
        CreateStateMachineRequest {
            definition = jsonVal
            name = stateMachineName
            roleArn = roleARNVal
            type = StateMachineType.Standard
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.createStateMachine(machineRequest)
        return response.stateMachineArn
    }
}
```

- API-Details finden Sie CreateStateMachine in der AWS SDK für die Kotlin-API-Referenz.

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteActivity`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteActivity(actArn: String?) {
    val activityRequest =
        DeleteActivityRequest {
            activityArn = actArn
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteActivity(activityRequest)
        println("You have deleted $actArn")
    }
}
```

- API-Details finden Sie DeleteActivity in der AWS SDK für die Kotlin-API-Referenz.

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStateMachine`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteMachine(stateMachineArnVal: String?) {
    val deleteStateMachineRequest =
        DeleteStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- API-Details finden Sie DeleteStateMachine in der AWS SDK für die Kotlin-API-Referenz.

`DescribeExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribeExecution`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeExe(executionArnVal: String?) {
    val executionRequest =
        DescribeExecutionRequest {
            executionArn = executionArnVal
        }

    var status = ""
    var hasSucceeded = false
    while (!hasSucceeded) {
        SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.describeExecution(executionRequest)
            status = response.status.toString()
            if (status.compareTo("Running") == 0) {
                println("The state machine is still running, let's wait for it to
 finish.")
                Thread.sleep(2000)
            } else if (status.compareTo("Succeeded") == 0) {
                println("The Step Function workflow has succeeded")
                hasSucceeded = true
            } else {
                println("The Status is $status")
            }
        }
    }
    println("The Status is $status")
}
```

- API-Details finden Sie DescribeExecution in der AWS SDK für die Kotlin-API-Referenz.

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStateMachine`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeStateMachine(stateMachineArnVal: String?) {
    val stateMachineRequest =
        DescribeStateMachineRequest {
            stateMachineArn = stateMachineArnVal
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.describeStateMachine(stateMachineRequest)
        println("The name of the State machine is ${response.name}")
        println("The status of the State machine is ${response.status}")
        println("The ARN value of the State machine is ${response.stateMachineArn}")
        println("The role ARN value is ${response.roleArn}")
    }
}
```

- API-Details finden Sie DescribeStateMachine in der AWS SDK für die Kotlin-API-Referenz.

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung`GetActivityTask`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getActivityTask(actArn: String?): List<String> {
    val myList: MutableList<String> = ArrayList()
    val getActivityTaskRequest =
        GetActivityTaskRequest {
            activityArn = actArn
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getActivityTask(getActivityTaskRequest)
        myList.add(response.taskToken.toString())
        myList.add(response.input.toString())
        return myList
    }
```

```
}
```

- API-Details finden Sie GetActivityTask in der AWS SDK für die Kotlin-API-Referenz.

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung`ListActivities`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listAllActivites() {
    val activitiesRequest =
        ListActivitiesRequest {
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listActivities(activitiesRequest)
        response.activities?.forEach { item ->
            println("The activity ARN is ${item.activityArn}")
            println("The activity name is ${item.name}")
        }
    }
}
```

- API-Details finden Sie ListActivities in der AWS SDK für die Kotlin-API-Referenz.

`ListExecutions`

Das folgende Codebeispiel zeigt die Verwendung`ListExecutions`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getExeHistory(exeARN: String?) {
    val historyRequest =
        GetExecutionHistoryRequest {
            executionArn = exeARN
            maxResults = 10
        }

    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- API-Details finden Sie ListExecutions in der AWS SDK für die Kotlin-API-Referenz.

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung`ListStateMachines`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import aws.sdk.kotlin.services.sfn.SfnClient
```

```
import aws.sdk.kotlin.services.sfn.model.ListStateMachinesRequest

/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */

suspend fun main() {
    println(DASHES)
    println("Welcome to the AWS Step Functions Hello example.")
    println("Lets list up to ten of your state machines:")
    println(DASHES)

    listMachines()
}

suspend fun listMachines() {
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- API-Details finden Sie ListStateMachines in der AWS SDK für die Kotlin-API-Referenz.

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung`SendTaskSuccess`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun sendTaskSuccess(
    token: String?,
    json: String?,
) {
    val successRequest =
        SendTaskSuccessRequest {
            taskToken = token
            output = json
        }
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        sfnClient.sendTaskSuccess(successRequest)
    }
}
```

- API-Details finden Sie SendTaskSuccess in der AWS SDK für die Kotlin-API-Referenz.

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun startWorkflow(
    stateMachineArnVal: String?,
    jsonEx: String?,
): String? {
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val executionRequest =
        StartExecutionRequest {
            input = jsonEx
            stateMachineArn = stateMachineArnVal
            name = uuidValue
        }
```

```
    SfnClient.fromEnvironment { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.startExecution(executionRequest)
        return response.executionArn
    }
}
```

- API-Details finden Sie StartExecution in der AWS SDK für die Kotlin-API-Referenz.

Support Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. Support

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für Kotlin

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
suspend fun displaySomeServices() {
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1
```

```
        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- API-Details finden Sie DescribeServices in der AWS SDK für die Kotlin-API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für Kotlin

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
In addition, you must have the AWS Business Support Plan to use the AWS Support Java
 API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following tasks:
1. Gets and displays available services.
2. Gets and displays severity levels.
3. Creates a support case by using the selected service, category, and severity
 level.
4. Gets a list of open cases for the current day.
5. Creates an attachment set with a generated file.
6. Adds a communication with the attachment to the support case.
7. Lists the communications of the support case.
8. Describes the attachment set included with the communication.
9. Resolves the support case.
10. Gets a list of resolved cases for the current day.
*/

suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <fileAttachment>
    Where:
         fileAttachment - The file can be a simple saved .txt file to use as an
 email attachment.
```

```
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val fileAttachment = args[0]
    println("***** Welcome to the AWS Support case example scenario.")
    println("***** Step 1. Get and display available services.")
    val sevCatList = displayServices()

    println("***** Step 2. Get and display Support severity levels.")
    val sevLevel = displaySevLevels()

    println("***** Step 3. Create a support case using the selected service,
category, and severity level.")
    val caseIdVal = createSupportCase(sevCatList, sevLevel)
    if (caseIdVal != null) {
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
    getOpenCase()

    println("***** Step 5. Create an attachment set with a generated file to add to
the case.")
    val attachmentSetId = addAttachment(fileAttachment)
    println("The Attachment Set id value is $attachmentSetId")

    println("***** Step 6. Add communication with the attachment to the support
case.")
    addAttachSupportCase(caseIdVal, attachmentSetId)

    println("***** Step 7. List the communications of the support case.")
    val attachId = listCommunications(caseIdVal)
    println("The Attachment id value is $attachId")

    println("***** Step 8. Describe the attachment set included with the
communication.")
    describeAttachment(attachId)
```

```
    println("***** Step 9. Resolve the support case.")
    resolveSupportCase(caseIdVal)

    println("***** Step 10. Get a list of resolved cases for the current day.")
    getResolvedCase()
    println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 30
            afterTime = yesterday.toString()
            beforeTime = now.toString()
            includeResolvedCases = true
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
```

```
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
```

```
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
```

```
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
```

```
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
```

```
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Kotlin.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt die Verwendung`AddAttachmentsToSet`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal =
        Attachment {
            fileName = myFile.name
```

```
            data = sourceBytes
        }

    val setRequest =
        AddAttachmentsToSetRequest {
            attachments = listOf(attachmentVal)
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- API-Details finden Sie AddAttachmentsToSet in der AWS SDK für die Kotlin-API-Referenz.

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`AddCommunicationToCase`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun addAttachSupportCase(
    caseIdVal: String?,
    attachmentSetIdVal: String?,
) {
    val caseRequest =
        AddCommunicationToCaseRequest {
            caseId = caseIdVal
            attachmentSetId = attachmentSetIdVal
            communicationBody = "Please refer to attachment for details."
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
```

```
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
 case")
        } else {
            println("There was an error adding the communication to an AWS Support
 case")
        }
    }
}
```

- API-Details finden Sie AddCommunicationToCase in der AWS SDK für die Kotlin-API-Referenz.

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung`CreateCase`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createSupportCase(
    sevCatListVal: List<String>,
    sevLevelVal: String,
): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest =
        CreateCaseRequest {
            categoryCode = caseCategory.lowercase(Locale.getDefault())
            serviceCode = serCode.lowercase(Locale.getDefault())
            severityCode = sevLevelVal.lowercase(Locale.getDefault())
            communicationBody = "Test issue with
 ${serCode.lowercase(Locale.getDefault())}"
            subject = "Test case, please ignore"
            language = "en"
```

```
            issueType = "technical"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- API-Details finden Sie CreateCase in der AWS SDK für die Kotlin-API-Referenz.

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAttachment`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest =
        DescribeAttachmentRequest {
            attachmentId = attachId
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}
```

- API-Details finden Sie DescribeAttachment in der AWS SDK für die Kotlin-API-Referenz.

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCases`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest =
        DescribeCasesRequest {
            maxResults = 20
            afterTime = yesterday.toString()
            beforeTime = now.toString()
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- API-Details finden Sie DescribeCases in der AWS SDK für die Kotlin-API-Referenz.

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCommunications`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest =
        DescribeCommunicationsRequest {
            caseId = caseIdVal
            maxResults = 10
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- API-Details finden Sie DescribeCommunications in der AWS SDK für die Kotlin-API-Referenz.

`DescribeServices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeServices`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest =
        DescribeServicesRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- API-Details finden Sie DescribeServices in der AWS SDK für die Kotlin-API-Referenz.

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSeverityLevels`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest =
        DescribeSeverityLevelsRequest {
            language = "en"
        }

    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}
```

- API-Details finden Sie DescribeSeverityLevels in der AWS SDK für die Kotlin-API-Referenz.

`ResolveCase`

Das folgende Codebeispiel zeigt die Verwendung`ResolveCase`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest =
        ResolveCaseRequest {
            caseId = caseIdVal
        }
    SupportClient.fromEnvironment { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- API-Details finden Sie ResolveCase in der AWS SDK für die Kotlin-API-Referenz.

Beispiele für Amazon Translate unter Verwendung von SDKs für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Amazon Translate Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Amazon-SNS-Anwendung

Das folgende Codebeispiel zeigt, wie eine Anwendung erstellt wird, die über Abonnements- und Veröffentlichungsfunktionen verfügt und Nachrichten übersetzt.

SDK für Kotlin

Zeigt, wie man die Kotlin-API für Amazon SNS verwendet, um eine Anwendung zu erstellen, die über Abonnement- und Veröffentlichungsfunktionen verfügt. Darüber hinaus übersetzt diese Beispielanwendung auch Nachrichten.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer Web-App finden Sie im vollständigen Beispiel unter GitHub.

Den vollständigen Quellcode und Anweisungen zum Erstellen einer nativen Android-App finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon SNS
- Amazon Translate

X-Ray Beispiele zur Verwendung des SDK für Kotlin

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Kotlin mit Aktionen ausführen und allgemeine Szenarien implementieren. X-Ray

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createNewGroup(groupNameVal: String?) {
    val groupRequest =
        CreateGroupRequest {
            filterExpression = "fault = true AND http.url CONTAINS \"example/game\"
 AND responsetime >= 5"
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val groupResponse = xRayClient.createGroup(groupRequest)
        println("The Group ARN is " + (groupResponse.group?.groupArn))
    }
}
```

- API-Details finden Sie CreateGroup in der AWS SDK für die Kotlin-API-Referenz.

`CreateSamplingRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateSamplingRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun createRule(ruleNameVal: String?) {
    val rule =
        SamplingRule {
            ruleName = ruleNameVal
            priority = 1
            httpMethod = "*"
            serviceType = "*"
            serviceName = "*"
            urlPath = "*"
            version = 1
            host = "*"
            resourceArn = "*"
        }

    val ruleRequest =
        CreateSamplingRuleRequest {
            samplingRule = rule
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val ruleResponse: CreateSamplingRuleResponse =
 xRayClient.createSamplingRule(ruleRequest)
        println("The ARN of the new rule is
 ${ruleResponse.samplingRuleRecord?.samplingRule?.ruleArn}")
    }
}
```

- API-Details finden Sie CreateSamplingRule in der AWS SDK für die Kotlin-API-Referenz.

`DeleteGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGroup`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteSpecificGroup(groupNameVal: String) {
    val groupRequest =
        DeleteGroupRequest {
            groupName = groupNameVal
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteGroup(groupRequest)
        println("$groupNameVal was deleted!")
    }
}
```

- API-Details finden Sie DeleteGroup in der AWS SDK für die Kotlin-API-Referenz.

`DeleteSamplingRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSamplingRule`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun deleteRule(ruleNameVal: String?) {
    val ruleRequest =
        DeleteSamplingRuleRequest {
            ruleName = ruleNameVal
```

```
        }

    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        xRayClient.deleteSamplingRule(ruleRequest)
        println("$ruleNameVal was deleted")
    }
}
```

- API-Details finden Sie DeleteSamplingRule in der AWS SDK für die Kotlin-API-Referenz.

`GetGroups`

Das folgende Codebeispiel zeigt die Verwendung`GetGroups`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getAllGroups() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getGroups(GetGroupsRequest {})
        response.groups?.forEach { group ->
            println("The AWS X-Ray group name is ${group.groupName}")
        }
    }
}
```

- API-Details finden Sie GetGroups in der AWS SDK für die Kotlin-API-Referenz.

`GetSamplingRules`

Das folgende Codebeispiel zeigt die Verwendung`GetSamplingRules`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getRules() {
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getSamplingRules(GetSamplingRulesRequest {})
        response.samplingRuleRecords?.forEach { record ->
            println("The rule name is ${record.samplingRule?.ruleName}")
            println("The related service is: ${record.samplingRule?.serviceName}")
        }
    }
}
```

- API-Details finden Sie GetSamplingRules in der AWS SDK für die Kotlin-API-Referenz.

`GetServiceGraph`

Das folgende Codebeispiel zeigt die Verwendung`GetServiceGraph`.

SDK für Kotlin

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
suspend fun getGraph(groupNameVal: String?) {
    val time = aws.smithy.kotlin.runtime.time.Instant
    val getServiceGraphRequest =
        GetServiceGraphRequest {
            groupName = groupNameVal
            this.startTime = time.now()
            endTime = time.now()
```

```
        }
    XRayClient.fromEnvironment { region = "us-east-1" }.use { xRayClient ->
        val response = xRayClient.getServiceGraph(getServiceGraphRequest)
        response.services?.forEach { service ->
            println("The name of the service is  ${service.name}")
        }
    }
}
```

- API-Details finden Sie GetServiceGraph in der AWS SDK für die Kotlin-API-Referenz.

Codebeispiele für SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für PHP with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für PHP-Entwicklerhandbuch — Mehr über die Verwendung von PHP mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- API-Gateway-Beispiele unter Verwendung von SDK für PHP
- Aurora-Beispiele unter Verwendung von SDK für PHP

- Auto-Scaling-Beispiele unter Verwendung von SDK für PHP
- Beispiele für Amazon Bedrock unter Verwendung von SDK für PHP
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für PHP
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für PHP
- DynamoDB-Beispiele unter Verwendung von SDK für PHP
- Beispiele für Amazon EC2 unter Verwendung von SDK für PHP
- AWS Glue Beispiele zur Verwendung des SDK für PHP
- IAM-Beispiele unter Verwendung von SDK für PHP
- Kinesis-Beispiele unter Verwendung von SDK für PHP
- AWS KMS Beispiele mit SDK für PHP
- Lambda-Beispiele unter Verwendung von SDK für PHP
- Beispiele für Amazon MSK unter Verwendung von SDK für PHP
- Beispiele für Amazon RDS unter Verwendung von SDK für PHP
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für PHP
- Beispiele für Amazon Rekognition unter Verwendung von SDK für PHP
- Beispiele für Amazon S3 unter Verwendung von SDK für PHP
- Beispiele für S3 Directory Buckets unter Verwendung von SDK für PHP
- Beispiele für Amazon SES unter Verwendung von SDK für PHP
- Beispiele für Amazon SNS unter Verwendung von SDK für PHP
- Beispiele für Amazon SQS unter Verwendung von SDK für PHP

API-Gateway-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von API Gateway Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`GetBasePathMapping`

Das folgende Codebeispiel zeigt die Verwendung`GetBasePathMapping`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
```

```
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- API-Details finden Sie GetBasePathMapping in der AWS SDK für PHP API-Referenz.

`ListBasePathMappings`

Das folgende Codebeispiel zeigt die Verwendung`ListBasePathMappings`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 * /////////////////////////////////////////////////////////////////////////// */

function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}
```

```
function listTheBasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listTheBasePathMappings();
```

- API-Details finden Sie ListBasePathMappings in der AWS SDK für PHP API-Referenz.

`UpdateBasePathMapping`

Das folgende Codebeispiel zeigt die Verwendung`UpdateBasePathMapping`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;


/* ///////////////////////////////////////////////////////////////////////////
 *
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
```

```
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 * - $patchOperations: The base path update operations to apply.
 *
 * Returns: Information about the updated base path mapping, if available;
 * otherwise, the error message.
 * ////////////////////////////////////////////////////////////////////////// */

function updateBasePathMapping(
    $apiGatewayClient,
    $basePath,
    $domainName,
    $patchOperations
) {
    try {
        $result = $apiGatewayClient->updateBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
            'patchOperations' => $patchOperations
        ]);
        return 'The updated base path\'s URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function updateTheBasePathMapping()
{
    $patchOperations = array([
        'op' => 'replace',
        'path' => '/stage',
        'value' => 'stage2'
    ]);

    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);
```

```
    echo updateBasePathMapping(
        $apiGatewayClient,
        '(none)',
        'example.com',
        $patchOperations
    );
}

// Uncomment the following line to run this code in an AWS account.
// updateTheBasePathMapping();
```

- API-Details finden Sie UpdateBasePathMapping in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Aurora-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PPHP-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
```

```
         * Step 0: Create an EC2 launch template that you'll use to create an Auto
 Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you created
 in step 0.
         */
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
```

```
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups: {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
```

```
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this->autoScalingGroupName,
 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
```

```
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
         */
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
```

```
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']} recorded:
\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
```

```
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }

    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection

- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function createAutoScalingGroup(
    $autoScalingGroupName,
    $availabilityZones,
    $minSize,
    $maxSize,
    $launchTemplateId
) {
    return $this->autoScalingClient->createAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'AvailabilityZones' => $availabilityZones,
        'MinSize' => $minSize,
        'MaxSize' => $maxSize,
        'LaunchTemplate' => [
            'LaunchTemplateId' => $launchTemplateId,
        ],
    ]);
}
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK für PHP API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function deleteAutoScalingGroup($autoScalingGroupName)
{
    return $this->autoScalingClient->deleteAutoScalingGroup([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'ForceDelete' => true,
    ]);
}
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK für PHP API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function describeAutoScalingGroups($autoScalingGroupNames)
{
    return $this->autoScalingClient->describeAutoScalingGroups([
        'AutoScalingGroupNames' => $autoScalingGroupNames
```

```
        ]);
    }
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für PHP API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function describeAutoScalingInstances($instanceIds)
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK für PHP API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function describeScalingActivities($autoScalingGroupName)
    {
        return $this->autoScalingClient->describeScalingActivities([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK für PHP API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK für PHP API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function enableMetricsCollection($autoScalingGroupName, $granularity)
{
    return $this->autoScalingClient->enableMetricsCollection([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'Granularity' => $granularity,
    ]);
}
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK für PHP API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
{
    return $this->autoScalingClient->setDesiredCapacity([
        'AutoScalingGroupName' => $autoScalingGroupName,
        'DesiredCapacity' => $desiredCapacity,
    ]);
}
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK für PHP API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient->terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' => $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
$attempts < 5) {
                error_log("Cannot terminate an instance while it is still pending.
Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK für PHP API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function updateAutoScalingGroup($autoScalingGroupName, $args)
{
    if (array_key_exists('MaxSize', $args)) {
        $maxSize = ['MaxSize' => $args['MaxSize']];
    } else {
        $maxSize = [];
    }
    if (array_key_exists('MinSize', $args)) {
        $minSize = ['MinSize' => $args['MinSize']];
    } else {
        $minSize = [];
    }
    $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
    $parameters = array_merge($parameters, $minSize, $maxSize);
    return $this->autoScalingClient->updateAutoScalingGroup($parameters);
}
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon Bedrock unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
public function listFoundationModels()
{
    $bedrockClient = new BedrockClient([
        'region' => 'us-west-2',
        'profile' => 'default'
    ]);
    $response = $bedrockClient->listFoundationModels();
    return $response['modelSummaries'];
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Amazon Nova
- Stable Image Core

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für PHP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// Use the Conversation API to send a text message to Amazon Nova.

use Aws\BedrockRuntime\BedrockRuntimeClient;
use Aws\Exception\AwsException;
use RuntimeException;

class Converse
{
    public function converse(): string
    {
```

```
        // Create a Bedrock Runtime client in the AWS Region you want to use.
        $client = new BedrockRuntimeClient([
            'region' => 'us-east-1',
            'profile' => 'default'
        ]);

        // Set the model ID, e.g., Amazon Nova Lite.
        $modelId = 'amazon.nova-lite-v1:0';

        // Start a conversation with the user message.
        $userMessage = "Describe the purpose of a 'hello world' program in one
 line.";
        $conversation = [
            [
                "role" => "user",
                "content" => [["text" => $userMessage]]
            ]
        ];

        try {
            // Send the message to the model, using a basic inference configuration.
            $response = $client->converse([
                'modelId' => $modelId,
                'messages' => $conversation,
                'inferenceConfig' => [
                    'maxTokens' => 512,
                    'temperature' => 0.5
                ]
            ]);

            // Extract and return the response text.
            $responseText = $response['output']['message']['content'][0]['text'];
            return $responseText;
        } catch (AwsException $e) {
            echo "ERROR: Can't invoke {$modelId}. Reason: {$e-
>getAwsErrorMessage()}";
            throw new RuntimeException("Failed to invoke model: " . $e-
>getAwsErrorMessage(), 0, $e);
        }
    }
}

$demo = new Converse();
echo $demo->converse();
```

- Details zur API finden Sie unter Converse in der AWS SDK für PHP -API-Referenz.

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für PHP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
    public function invokeStableDiffusion(string $prompt, int $seed = 0, string
 $aspect_ratio = '1:1')
    {
        // The different model providers have individual request and response
 formats.
        // For the format, ranges, and available parameters of Stable Diffusion
 models refer to:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
stability-diffusion.html

        $base64_image_data = "";
        try {
            $modelId = 'stability.stable-image-core-v1:1';
            $body = [
                'prompt' => $prompt,
                'aspect_ratio' => $aspect_ratio,
                'seed' => $seed,
                'output_format' => 'png',
            ];
```

```
            $result = $this->bedrockRuntimeClient->invokeModel([
                'contentType' => 'application/json',
                'body' => json_encode($body),
                'modelId' => $modelId,
            ]);
            $response_body = json_decode($result['body']);
            $base64_image_data = $response_body->images[0];
        } catch (Exception $e) {
            echo "Error: ({$e->getCode()}) - {$e->getMessage()}\n";
        }

        return $base64_image_data;
    }
```

- API-Details finden Sie InvokeModel in der AWS SDK für PHP API-Referenz.

Beispiele für Amazon DocumentDB unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
```

```
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

DynamoDB-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.

- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Da in diesem Beispiel unterstützende Dateien verwendet werden, lesen Sie unbedingt die Anleitung in der README.md PHP-Beispieldatei.

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
```

```
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
            'TableName' => $tableName,
        ]);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
```

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];
        $attributes = ["rating" =>
            [
                'AttributeName' => 'rating',
                'AttributeType' => 'N',
                'Value' => $rating,
            ],
            'plot' => [
                'AttributeName' => 'plot',
                'AttributeType' => 'S',
                'Value' => $plot,
            ]
        ];
        $service->updateItemAttributesByKey($tableName, $key, $attributes);
        echo "Movie added and updated.";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);


        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

        $movie = $service->getItemByKey($tableName, $key);
```

```
        echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']
['rating']['N']}\n";

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
```

```
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
```

```
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für PHP API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function writeBatch(string $TableName, array $Batch, int $depth = 2)
    {
        if (--$depth <= 0) {
            throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
        }
```

```
        $marshal = new Marshaler();
        $total = 0;
        foreach (array_chunk($Batch, 25) as $Items) {
            foreach ($Items as $Item) {
                $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item'
=> $marshal->marshalItem($Item)]];
            }
            try {
                echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
                $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
                $BatchWrite = [];
            } catch (Exception $e) {
                echo "uh oh...";
                echo $e->getMessage();
                die();
            }
            if ($total >= 250) {
                echo "250 movies is probably enough. Right? We can stop there.\n";
                break;
            }
        }
    }
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für PHP API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine -Tabelle.

```
        $tableName = "ddb_demo_table_$uuid";
```

```
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

    public function createTable(string $tableName, array $attributes)
    {
        $keySchema = [];
        $attributeDefinitions = [];
        foreach ($attributes as $attribute) {
            if (is_a($attribute, DynamoDBAttribute::class)) {
                $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
                $attributeDefinitions[] =
                    ['AttributeName' => $attribute->AttributeName, 'AttributeType'
=> $attribute->AttributeType];
            }
        }

        $this->dynamoDbClient->createTable([
            'TableName' => $tableName,
            'KeySchema' => $keySchema,
            'AttributeDefinitions' => $attributeDefinitions,
            'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,
'WriteCapacityUnits' => 10],
        ]);
    }
```

- API-Details finden Sie CreateTable in der AWS SDK für PHP API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $key = [
            'Item' => [
                'title' => [
                    'S' => $movieName,
                ],
                'year' => [
                    'N' => $movieYear,
                ],
            ]
        ];

        $service->deleteItemByKey($tableName, $key);
        echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

    public function deleteItemByKey(string $tableName, array $key)
    {
        $this->dynamoDbClient->deleteItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API-Details finden Sie DeleteItem in der AWS SDK für PHP API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- API-Details finden Sie DeleteTable in der AWS SDK für PHP API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => "$statement",
        'Parameters' => $parameters,
    ]);
}
```

```
    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }

    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für PHP API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $movie = $service->getItemByKey($tableName, $key);
        echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

    public function getItemByKey(string $tableName, array $key)
    {
        return $this->dynamoDbClient->getItem([
            'Key' => $key['Item'],
            'TableName' => $tableName,
        ]);
    }
```

- API-Details finden Sie GetItem in der AWS SDK für PHP API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function listTables($exclusiveStartTableName = "", $limit = 100)
    {
        $this->dynamoDbClient->listTables([
            'ExclusiveStartTableName' => $exclusiveStartTableName,
            'Limit' => $limit,
        ]);
    }
```

- API-Details finden Sie ListTables in der AWS SDK für PHP API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    echo "What's the name of the last movie you watched?\n";
    while (empty($movieName)) {
        $movieName = testable_readline("Movie name: ");
    }
    echo "And what year was it released?\n";
    $movieYear = "year";
    while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
        $movieYear = testable_readline("Year released: ");
    }

    $service->putItem([
        'Item' => [
            'year' => [
                'N' => "$movieYear",
            ],
            'title' => [
                'S' => $movieName,
            ],
        ],
        'TableName' => $tableName,
    ]);

public function putItem(array $array)
{
    $this->dynamoDbClient->putItem($array);
}
```

- API-Details finden Sie PutItem in der AWS SDK für PHP API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);

    public function query(string $tableName, $key)
    {
        $expressionAttributeValues = [];
        $expressionAttributeNames = [];
        $keyConditionExpression = "";
        $index = 1;
        foreach ($key as $name => $value) {
            $keyConditionExpression .= "#" . array_key_first($value) . " = :v
$index,";
            $expressionAttributeNames["#" . array_key_first($value)] =
 array_key_first($value);
            $hold = array_pop($value);
            $expressionAttributeValues[":v$index"] = [
                array_key_first($hold) => array_pop($hold),
            ];
        }
        $keyConditionExpression = substr($keyConditionExpression, 0, -1);
        $query = [
            'ExpressionAttributeValues' => $expressionAttributeValues,
            'ExpressionAttributeNames' => $expressionAttributeNames,
            'KeyConditionExpression' => $keyConditionExpression,
            'TableName' => $tableName,
```

```
        ];
        return $this->dynamoDbClient->query($query);
    }
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für PHP -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

    public function scan(string $tableName, array $key, string $filters)
    {
        $query = [
            'ExpressionAttributeNames' => ['#year' => 'year'],
            'ExpressionAttributeValues' => [
```

```
                ":min" => ['N' => '1990'],
                ":max" => ['N' => '1999'],
            ],
            'FilterExpression' => "#year between :min and :max",
            'TableName' => $tableName,
        ];
        return $this->dynamoDbClient->scan($query);
    }
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für PHP -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $service->updateItemAttributeByKey($tableName, $key, 'rating', 'N',
$rating);

    public function updateItemAttributeByKey(
        string $tableName,
        array $key,
        string $attributeName,
        string $attributeType,
        string $newValue
    ) {
        $this->dynamoDbClient->updateItem([
            'Key' => $key['Item'],
```

```
            'TableName' => $tableName,
            'UpdateExpression' => "set #NV=:NV",
            'ExpressionAttributeNames' => [
                '#NV' => $attributeName,
            ],
            'ExpressionAttributeValues' => [
                ':NV' => [
                    $attributeType => $newValue
                ]
            ],
        ]);
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\loadMovieData;
use function AwsUtilities\testable_readline;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");
```

```
        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQLBatch($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
```

```
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);
        echo "Movie added and updated.\n";

        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']}
        was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']['S']}?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQLBatch($statement, $parameters);

        $movie = $service->getItemByPartiQLBatch($tableName, [$key]);
        echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']}
        as a {$movie['Responses'][0]['Item']['rating']['N']}\n";

        $service->deleteItemByPartiQLBatch($statement, $parameters);
```

```
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
```

```
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function getItemByPartiQLBatch(string $tableName, array $keys): Result
    {
        $statements = [];
        foreach ($keys as $key) {
            list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
            $statements[] = [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ];
        }

        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
```

```
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für PHP API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
```

```
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

use function AwsUtilities\testable_readline;
use function AwsUtilities\loadMovieData;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
 $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
```

```
                'year' => [
                    'N' => "$movieYear",
                ],
                'title' => [
                    'S' => $movieName,
                ],
            ],
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
        $service->insertItemByPartiQL($statement, $parameters);

        echo "How would you rate the movie from 1-10?\n";
        $rating = 0;
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        echo "What was the movie about?\n";
        while (empty($plot)) {
            $plot = testable_readline("Plot summary: ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
        ];

        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);
        echo "Movie added and updated.\n";



        $batch = json_decode(loadMovieData());

        $service->writeBatch($tableName, $batch);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
 {$movie['Items'][0]['year']['N']}.\n";
        echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?
\n";
        $rating = 0;
```

```
        while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
 $rating > 10) {
            $rating = testable_readline("Rating (1-10): ");
        }
        $attributes = [
            new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
            new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
        ];
        list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
        $service->updateItemByPartiQL($statement, $parameters);

        $movie = $service->getItemByPartiQL($tableName, $key);
        echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
 {$movie['Items'][0]['rating']['N']}\n";

        $service->deleteItemByPartiQL($statement, $parameters);
        echo "But, bad news, this was a trap. That movie has now been deleted
 because of your rating...harsh.\n";

        echo "That's okay though. The book was better. Now, for something lighter,
 in what year were you born?\n";
        $birthYear = "not a number";
        while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
            $birthYear = testable_readline("Birth year: ");
        }
        $birthKey = [
            'Key' => [
                'year' => [
                    'N' => "$birthYear",
                ],
            ],
        ];
        $result = $service->query($tableName, $birthKey);
        $marshal = new Marshaler();
        echo "Here are the movies in our collection released the year you were born:
\n";
        $oops = "Oops! There were no movies released in that year (that we know of).
\n";
        $display = "";
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            $display .= $movie['title'] . "\n";
        }
```

```
        echo ($display) ?: $oops;

        $yearsKey = [
            'Key' => [
                'year' => [
                    'N' => [
                        'minRange' => 1990,
                        'maxRange' => 1999,
                    ],
                ],
            ],
        ];
        $filter = "year between 1990 and 1999";
        echo "\nHere's a list of all the movies released in the 90s:\n";
        $result = $service->scan($tableName, $yearsKey, $filter);
        foreach ($result['Items'] as $movie) {
            $movie = $marshal->unmarshalItem($movie);
            echo $movie['title'] . "\n";
        }

        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}

    public function insertItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ]);
    }

    public function getItemByPartiQL(string $tableName, array $key): Result
    {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
 $tableName, $key['Item']);

        return $this->dynamoDbClient->executeStatement([
            'Parameters' => $parameters,
            'Statement' => $statement,
        ]);
    }
```

```
    public function updateItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }

    public function deleteItemByPartiQL(string $statement, array $parameters)
    {
        $this->dynamoDbClient->executeStatement([
            'Statement' => $statement,
            'Parameters' => $parameters,
        ]);
    }
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für PHP API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
```

```
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
```

```
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Beispiele für Amazon EC2 unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVpc`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpc`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
 * @param string $cidr
 * @return array
 */
public function createVpc(string $cidr): array
{
    try {
        $result = $this->ec2Client->createVpc([
            "CidrBlock" => $cidr,
```

```
            ]);
            return $result['Vpc'];
        }catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie CreateVpc in der AWS SDK für PHP API-Referenz.

`CreateVpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpcEndpoint`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $serviceName
     * @param string $vpcId
     * @param array $routeTableIds
     * @return array
     */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
 $routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);
```

```
            return $result["VpcEndpoint"];
        } catch(Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie CreateVpcEndpoint in der AWS SDK für PHP API-Referenz.

`DeleteVpc`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVpc`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $vpcId
     * @return void
     */
    public function deleteVpc(string $vpcId)
    {
        try {
            $this->ec2Client->deleteVpc([
                "VpcId" => $vpcId,
            ]);
        }catch(Ec2Exception $caught){
            echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie DeleteVpc in der AWS SDK für PHP API-Referenz.

`DeleteVpcEndpoints`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVpcEndpoints`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * @param string $vpcEndpointId
     * @return void
     */
    public function deleteVpcEndpoint(string $vpcEndpointId)
    {
        try {
            $this->ec2Client->deleteVpcEndpoints([
                "VpcEndpointIds" => [$vpcEndpointId],
            ]);
        }catch (Ec2Exception $caught){
            echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie DeleteVpcEndpoints in der AWS SDK für PHP API-Referenz.

`DescribeRouteTables`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRouteTables`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /**
    * @param array $routeTableIds
    * @param array $filters
    * @return array
    */
   public function describeRouteTables(array $routeTableIds = [], array $filters =
[]): array
   {
       $parameters = [];
       if($routeTableIds){
           $parameters['RouteTableIds'] = $routeTableIds;
       }
       if($filters){
           $parameters['Filters'] = $filters;
       }
       try {
           $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
           $contents = [];
           foreach ($paginator as $result) {
               foreach ($result['RouteTables'] as $object) {
                   $contents[] = $object['RouteTableId'];
               }
           }
       }catch (Ec2Exception $caught){
           echo "There was a problem paginating the results of DescribeRouteTables:
{$caught->getAwsErrorMessage()}\n";
           throw $caught;
       }
       return $contents;
   }
```

- API-Details finden Sie DescribeRouteTables in der AWS SDK für PHP API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für PHP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für PHP with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");
```

```
        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);

        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
```

```
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
```

```
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
```

```
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

```
    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
```

```
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
```

```
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler

- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- API-Details finden Sie CreateCrawler in der AWS SDK für PHP API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

   public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
   {
       return $this->glueClient->createJob([
           'Name' => $jobName,
           'Role' => $role,
           'Command' => [
               'Name' => 'glueetl',
               'ScriptLocation' => $scriptLocation,
               'PythonVersion' => $pythonVersion,
           ],
           'GlueVersion' => $glueVersion,
       ]);
   }
```

- API-Details finden Sie CreateJob in der AWS SDK für PHP API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- API-Details finden Sie DeleteCrawler in der AWS SDK für PHP API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the databases.\n";
```

```
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- API-Details finden Sie DeleteDatabase in der AWS SDK für PHP API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- API-Details finden Sie DeleteJob in der AWS SDK für PHP API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK für PHP API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- API-Details finden Sie GetCrawler in der AWS SDK für PHP API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $databaseName = "doc-example-database-$uniqid";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
```

```
                'Name' => $databaseName,
            ]);
        });
    }
```

- API-Details finden Sie GetDatabase in der AWS SDK für PHP API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
```

```
        ]);
    }
```

- API-Details finden Sie GetJobRun in der AWS SDK für PHP API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- API-Details finden Sie GetJobRuns in der AWS SDK für PHP API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    $databaseName = "doc-example-database-$uniqid";

    $tables = $glueService->getTables($databaseName);

public function getTables($databaseName): Result
{
    return $this->glueClient->getTables([
        'DatabaseName' => $databaseName,
    ]);
}
```

- API-Details finden Sie GetTables in der AWS SDK für PHP API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    $jobs = $glueService->listJobs();
    echo "Current jobs:\n";
    foreach ($jobs['JobNames'] as $jobsName) {
        echo "{$jobsName}\n";
    }
```

```
    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- API-Details finden Sie ListJobs in der AWS SDK für PHP API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
```

```
    }
```

- API-Details finden Sie StartCrawler in der AWS SDK für PHP API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):
Result
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- API-Details finden Sie StartJobRun in der AWS SDK für PHP API-Referenz.

IAM-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.

- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
```

```
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}
```

```
$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
```

```
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für PHP API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
```

```
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName, $policyDocument)
 {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- API-Details finden Sie CreatePolicy in der AWS SDK für PHP API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
```

```
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- API-Details finden Sie CreateRole in der AWS SDK für PHP API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK für PHP API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- API-Details finden Sie CreateUser in der AWS SDK für PHP API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
```

```
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK für PHP API-Referenz.

GetPolicy

Das folgende Codebeispiel zeigt die VerwendungGetPolicy.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- API-Details finden Sie GetPolicy in der AWS SDK für PHP API-Referenz.

GetRole

Das folgende Codebeispiel zeigt die VerwendungGetRole.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- API-Details finden Sie GetRole in der AWS SDK für PHP API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
 "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
```

```
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK für PHP API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
```

```
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- API-Details finden Sie ListGroups in der AWS SDK für PHP API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- API-Details finden Sie ListPolicies in der AWS SDK für PHP API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- API-Details finden Sie ListRolePolicies in der AWS SDK für PHP API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- API-Details finden Sie ListRoles in der AWS SDK für PHP API-Referenz.

`Listet SAML-Anbieter auf`

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für PHP -API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
```

```
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- API-Details finden Sie ListUsers in der AWS SDK für PHP API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Kinesis Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}
```

```
$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }
```

```
    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

AWS KMS Beispiele mit SDK für PHP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für PHP with Aktionen ausführen und gängige Szenarien implementieren AWS KMS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS KMS

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

SDK für PHP

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;
```

```
echo "This file shows how to connect to the KmsClient, uses a paginator to get the
 keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or to
 break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
    echo "End of page one of results. Alter the \$pageLength variable to see more
 results.\n";
    break;
}
```

- API-Details finden Sie ListKeys in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines KMS-Schlüssels.
- Listen Sie die KMS-Schlüssel für Ihr Konto auf und rufen Sie Details über sie ab.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.
- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Taggen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo "\n";
        echo "--------------------------------------\n";
        echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using the
 AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;
        echo "--------------------------------------\n";
        $this->pressEnter();

        $this->kmsClient = new KmsClient([]);
        // Initialize the KmsService class with the client. This allows you to
 override any defaults in the client before giving it to the service class.
        $this->kmsService = new KmsService($this->kmsClient);

        // 1. Create a symmetric KMS key.
        echo "\n";
        echo "1. Create a symmetric KMS key.\n";
```

```
        echo "First, we will create a symmetric KMS key that is used to encrypt and
 decrypt data by invoking createKey().\n";
        $this->pressEnter();

        $key = $this->kmsService->createKey();
        $this->resources['symmetricKey'] = $key['KeyId'];
        echo "Created a customer key with ARN {$key['Arn']}.\n";
        $this->pressEnter();

        // 2. Enable a KMS key.
        echo "\n";
        echo "2. Enable a KMS key.\n";
        echo "By default when you create an AWS key, it is enabled. The code checks
 to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
        $this->pressEnter();

        $keyInfo = $this->kmsService->describeKey($key['KeyId']);
        if(!$keyInfo['Enabled']){
            echo "The key was not enabled, so we will enable it.\n";
            $this->pressEnter();
            $this->kmsService->enableKey($key['KeyId']);
            echo "The key was successfully enabled.\n";
        }else{
            echo "The key was already enabled, so there was no need to enable it.
\n";
        }
        $this->pressEnter();

        // 3. Encrypt data using the symmetric KMS key.
        echo "\n";
        echo "3. Encrypt data using the symmetric KMS key.\n";
        echo "One of the main uses of symmetric keys is to encrypt and decrypt data.
\n";
        echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
 SYMMETRIC_DEFAULT encryption algorithm.\n";
        $this->pressEnter();
        $text = "Hello, AWS KMS!";
        $encryption = $this->kmsService->encrypt($key['KeyId'], $text);
        echo "The plaintext data was successfully encrypted with the algorithm:
 {$encryption['EncryptionAlgorithm']}.\n";
        $this->pressEnter();

        // 4. Create an alias.
```

```
        echo "\n";
        echo "4. Create an alias.\n";
        $aliasInput = testable_readline("Please enter an alias prefixed with
 \"alias/\" or press enter to use a default value: ");
        if($aliasInput == ""){
            $aliasInput = "alias/dev-encryption-key";
        }
        $this->kmsService->createAlias($key['KeyId'], $aliasInput);
        $this->resources['alias'] = $aliasInput;
        echo "The alias \"$aliasInput\" was successfully created.\n";
        $this->pressEnter();

        // 5. List all of your aliases.
        $aliasPageSize = 10;
        echo "\n";
        echo "5. List all of your aliases, up to $aliasPageSize.\n";
        $this->pressEnter();
        $aliasPaginator = $this->kmsService->listAliases();
        foreach($aliasPaginator as $pages){
            foreach($pages['Aliases'] as $alias){
                echo $alias['AliasName'] . "\n";
            }
            break;
        }
        $this->pressEnter();

        // 6. Enable automatic rotation of the KMS key.
        echo "\n";
        echo "6. Enable automatic rotation of the KMS key.\n";
        echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.";
        $this->pressEnter();
        $this->kmsService->enableKeyRotation($key['KeyId']);
        echo "The key's rotation was successfully set for key: {$key['KeyId']}\n";
        $this->pressEnter();

        // 7. Create a grant.
        echo "7. Create a grant.\n";
        echo "\n";
        echo "A grant is a policy instrument that allows Amazon Web Services
 principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
```

```
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.\n";
        $granteeARN = testable_readline("Please enter the Amazon Resource Name
 (ARN) of an Amazon Web Services principal. Valid principals include Amazon Web
 Services accounts, IAM users, IAM roles, federated users, and assumed role users.
 For help with the ARN syntax for a principal, see IAM ARNs in the Identity and
 Access Management User Guide. \nTo skip this step, press enter without any other
 values: ");
        if($granteeARN){
            $operations = [
                "ENCRYPT",
                "DECRYPT",
                "DESCRIBE_KEY",
            ];
            $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
 $operations);
            echo "The grant Id is: {$grant['GrantId']}\n";
        }else{
            echo "Steps 7, 8, and 9 will be skipped.\n";
        }
        $this->pressEnter();

        // 8. List grants for the KMS key.
        if($granteeARN){
            echo "8. List grants for the KMS key.\n\n";
            $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
            foreach($grantsPaginator as $page){
                foreach($page['Grants'] as $grant){
                    echo $grant['GrantId'] . "\n";
                }
            }
        }else{
            echo "Skipping step 8...\n";
        }
        $this->pressEnter();

        // 9. Revoke the grant.
        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
```

```
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted earlier
 in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
 $encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are the
 primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in the
 key policy determine who has permission to use the KMS key and how they can use it.
\n";
        echo " You can also use IAM policies and grants to control access to the KMS
 key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
}
KEYPOLICY;
        echo $keyPolicy;
        $this->pressEnter();
        $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
```

```
        echo "The Key Policy was successfully replaced!\n";
        $this->pressEnter();

        // 12. Retrieve the key policy.
        echo "\n";
        echo "12. Retrieve the key policy.\n";
        echo "Let's get some information about the new policy and print it to the
screen.\n";
        $this->pressEnter();
        $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
        echo "We got the info! Here is the policy: \n";
        echo $policyInfo['Policy'] . "\n";
        $this->pressEnter();

        // 13. Create an asymmetric KMS key and sign data.
        echo "\n";
        echo "13. Create an asymmetric KMS key and sign data.\n";
        echo "Signing your data with an AWS key can provide several benefits that
make it an attractive option for your data signing needs.\n";
        echo "By using an AWS KMS key, you can leverage the security controls and
compliance features provided by AWS, which can help you meet various regulatory
requirements and enhance the overall security posture of your organization.\n";
        echo "First we'll create the asymmetric key.\n";
        $this->pressEnter();
        $keySpec = "RSA_2048";
        $keyUsage = "SIGN_VERIFY";
        $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
        $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
        echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
        echo "Next, we'll sign the data.\n";
        $this->pressEnter();
        $algorithm = "RSASSA_PSS_SHA_256";
        $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text, $algorithm);
        $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
        echo "Signature verification result: {$sign['signature']}\n";
        $this->pressEnter();

        // 14. Tag the symmetric KMS key.
        echo "\n";
        echo "14. Tag the symmetric KMS key.\n";
        echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and control
access to your encrypted data within your AWS environment.\n";
```

```
        echo "Let's tag our symmetric key as Environment->Production\n";
        $this->pressEnter();
        $this->kmsService->tagResource($key['KeyId'], [
            [
                'TagKey' => "Environment",
                'TagValue' => "Production",
            ],
        ]);
        echo "The key was successfully tagged!\n";
        $this->pressEnter();

        // 15. Schedule the deletion of the KMS key
        echo "\n";
        echo "15. Schedule the deletion of the KMS key.\n";
        echo "By default, KMS applies a waiting period of 30 days, but you can
 specify a waiting period of 7-30 days.\n";
        echo "When this operation is successful, the key state of the KMS key
 changes to PendingDeletion and the key can't be used in any cryptographic
 operations.\n";
        echo "It remains in this state for the duration of the waiting period.\n\n";

        echo "Deleting a KMS key is a destructive and potentially dangerous
 operation. When a KMS key is deleted, all data that was encrypted under the KMS key
 is unrecoverable.\n\n";

        $cleanUp = testable_readline("Would you like to delete the resources created
 during this scenario, including the keys? (y/n): ");
        if($cleanUp == "Y" || $cleanUp == "y"){
            $this->cleanUp();
        }

        echo
 "--------------------------------------------------------------------------------
\n";
        echo "This concludes the AWS Key Management SDK Basics scenario\n";
        echo
 "--------------------------------------------------------------------------------
\n";



namespace Kms;

use Aws\Kms\Exception\KmsException;
```

```
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{

    protected KmsClient $client;
    protected bool $verbose;

    /***
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }


    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
 $description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
```

```
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
 more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
```

```
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $alias
```

```
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
not valid.\n";
```

```
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
```

```
    }



    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }



    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
```

```
    }



    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }


    /***
     * @param string $keyId
     * @return Result
     */
    public function getKeyPolicy(string $keyId)
    {
        try {
            return $this->client->getKeyPolicy([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


    /***
     * @param string $grantId
```

```
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param array $tags
     * @return void
```

```
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }



    /***
     * @param string $keyId
     * @param int $rotationPeriodInDays
     * @return void
```

```
     */
    public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
 365)
    {
        try{
            $this->client->enableKeyRotation([
                'KeyId' => $keyId,
                'RotationPeriodInDays' => $rotationPeriodInDays,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
 resource could not be found.\n";
            }
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $aliasName
     * @return void
```

```
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }




    /***
     * @param string $keyId
     * @param string $message
     * @param string $signature
     * @param string $signingAlgorithm
     * @return bool
     */
    public function verify(string $keyId, string $message, string $signature, string
 $signingAlgorithm)
    {
        try {
            $result = $this->client->verify([
                'KeyId' => $keyId,
                'Message' => $message,
                'Signature' => $signature,
                'SigningAlgorithm' => $signingAlgorithm,
            ]);
            return $result['SignatureValid'];
        }catch(KmsException $caught){
            echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }


}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion
  - Sign
  - TagResource

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $alias
     * @return void
     */
    public function createAlias(string $keyId, string $alias)
    {
        try{
            $this->client->createAlias([
                'TargetKeyId' => $keyId,
                'AliasName' => $alias,
            ]);
        }catch (KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
                echo "The request was rejected because the specified alias name is
not valid.";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie CreateAlias in der AWS SDK für PHP API-Referenz.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
```

```
     * @param string $granteePrincipal
     * @param array $operations
     * @param array $grantTokens
     * @return Result
     */
    public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
    {
        $args = [
            'KeyId' => $keyId,
            'GranteePrincipal' => $granteePrincipal,
            'Operations' => $operations,
        ];
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token is
not valid.\n";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie CreateGrant in der AWS SDK für PHP API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
    {
        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie CreateKey in der AWS SDK für PHP API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm =
 "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK für PHP -API-Referenz.

`DeleteAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlias`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $aliasName
     * @return void
     */
    public function deleteAlias(string $aliasName)
    {
        try {
            $this->client->deleteAlias([
                'AliasName' => $aliasName,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie DeleteAlias in der AWS SDK für PHP API-Referenz.

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKey`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie DescribeKey in der AWS SDK für PHP API-Referenz.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
```

```
     */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie DisableKey in der AWS SDK für PHP API-Referenz.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return void
     */
    public function enableKey(string $keyId)
    {
        try {
            $this->client->enableKey([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
```

```
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie EnableKey in der AWS SDK für PHP API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $text
     * @return Result
     */
    public function encrypt(string $keyId, string $text)
    {
        try {
            return $this->client->encrypt([
                'KeyId' => $keyId,
                'Plaintext' => $text,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "DisabledException"){
                echo "The request was rejected because the specified KMS key is not
enabled.\n";
            }
```

```
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK für PHP -API-Referenz.

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAliases`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $limit
     * @return ResultPaginator
     */
    public function listAliases(string $keyId = "", int $limit = 0)
    {
        $args = [];
        if($keyId){
            $args['KeyId'] = $keyId;
        }
        if($limit){
            $args['Limit'] = $limit;
        }
        try{
            return $this->client->getPaginator("ListAliases", $args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
                echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
```

```
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie ListAliases in der AWS SDK für PHP API-Referenz.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @return Result
     */
    public function listGrants(string $keyId)
    {
        try{
            return $this->client->listGrants([
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "    The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }
```

- API-Details finden Sie ListGrants in der AWS SDK für PHP API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @return array
     */
    public function listKeys()
    {
        try {
            $contents = [];
            $paginator = $this->client->getPaginator("ListKeys");
            foreach($paginator as $result){
                foreach ($result['Content'] as $object) {
                    $contents[] = $object;
                }
            }
            return $contents;
        }catch(KmsException $caught){
            echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie ListKeys in der AWS SDK für PHP API-Referenz.

`PutKeyPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutKeyPolicy`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $policy
     * @return void
     */
    public function putKeyPolicy(string $keyId, string $policy)
    {
        try {
            $this->client->putKeyPolicy([
                'KeyId' => $keyId,
                'Policy' => $policy,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie PutKeyPolicy in der AWS SDK für PHP API-Referenz.

`RevokeGrant`

Das folgende Codebeispiel zeigt die Verwendung`RevokeGrant`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $grantId
     * @param string $keyId
     * @return void
     */
    public function revokeGrant(string $grantId, string $keyId)
    {
        try{
            $this->client->revokeGrant([
                'GrantId' => $grantId,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie RevokeGrant in der AWS SDK für PHP API-Referenz.

`ScheduleKeyDeletion`

Das folgende Codebeispiel zeigt die Verwendung`ScheduleKeyDeletion`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param int $pendingWindowInDays
     * @return void
     */
    public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays = 7)
    {
        try {
            $this->client->scheduleKeyDeletion([
                'KeyId' => $keyId,
                'PendingWindowInDays' => $pendingWindowInDays,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie ScheduleKeyDeletion in der AWS SDK für PHP API-Referenz.

`Sign`

Das folgende Codebeispiel zeigt die Verwendung`Sign`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param string $message
     * @param string $algorithm
     * @return Result
     */
    public function sign(string $keyId, string $message, string $algorithm)
    {
        try {
            return $this->client->sign([
                'KeyId' => $keyId,
                'Message' => $message,
                'SigningAlgorithm' => $algorithm,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- Weitere API-Informationen finden Sie unter Sign in der AWS SDK für PHP -API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /***
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
```

- API-Details finden Sie TagResource in der AWS SDK für PHP API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $iamService = new IAMService();
        $s3client = new S3Client($clientArgs);
        $lambdaService = new LambdaService();

        echo "First, let's create a role to run our Lambda code.\n";
        $roleName = "test-lambda-role-$uniqid";
        $rolePolicyDocument = "{
            \"Version\": \"2012-10-17\",
            \"Statement\": [
                {
                    \"Effect\": \"Allow\",
                    \"Principal\": {
```

```
                        \"Service\": \"lambda.amazonaws.com\"
                    },
                    \"Action\": \"sts:AssumeRole\"
                }
            ]
        }";
        $role = $iamService->createRole($roleName, $rolePolicyDocument);
        echo "Created role {$role['RoleName']}.\n";

        $iamService->attachRolePolicy(
            $role['RoleName'],
            "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        );
        echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";

        echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
        $bucketName = "amzn-s3-demo-bucket-$uniqid";
        $s3client->createBucket([
            'Bucket' => $bucketName,
        ]);
        echo "Created bucket $bucketName.\n";

        $functionName = "doc_example_lambda_$uniqid";
        $codeBasic = __DIR__ . "/lambda_handler_basic.zip";
        $handler = "lambda_handler_basic";
        $file = file_get_contents($codeBasic);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "Uploaded the Lambda code.\n";

        $createLambdaFunction = $lambdaService->createFunction($functionName, $role,
 $bucketName, $handler);
        // Wait until the function has finished being created.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['State'] == "Pending");
        echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";

        sleep(1);
```

```
        echo "\nOk, let's invoke that Lambda code.\n";
        $basicParams = [
            'action' => 'increment',
            'number' => 3,
        ];
        /** @var Stream $invokeFunction */
        $invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
        $result = json_decode($invokeFunction->getContents())->result;
        echo "After invoking the Lambda code with the input of
 {$basicParams['number']} we received $result.\n";

        echo "\nSince that's working, let's update the Lambda code.\n";
        $codeCalculator = "lambda_handler_calculator.zip";
        $handlerCalculator = "lambda_handler_calculator";
        echo "First, put the new code into the S3 bucket.\n";
        $file = file_get_contents($codeCalculator);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => $functionName,
            'Body' => $file,
        ]);
        echo "New code uploaded.\n";

        $lambdaService->updateFunctionCode($functionName, $bucketName,
 $functionName);
        // Wait for the Lambda code to finish updating.
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
 "Successful");
        echo "New Lambda code uploaded.\n";

        $environment = [
            'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
        ];
        $lambdaService->updateFunctionConfiguration($functionName,
 $handlerCalculator, $environment);
        do {
            $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
```

```
        } while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
        echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
information.\n";

        echo "Invoke the new code with some new data.\n";
        $calculatorParams = [
            'action' => 'plus',
            'x' => 5,
            'y' => 4,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
$result.\n";
        echo "Here's the extra debug info: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nBut what happens if you try to divide by zero?\n";
        $divZeroParams = [
            'action' => 'divide',
            'x' => 5,
            'y' => 0,
        ];
        $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
        $result = json_decode($invokeFunction['Payload']->getContents())->result;
        echo "You get a |$result| result.\n";
        echo "And an error message: ";
        echo base64_decode($invokeFunction['LogResult']) . "\n";

        echo "\nHere's all the Lambda functions you have in this Region:\n";
        $listLambdaFunctions = $lambdaService->listFunctions(5);
        $allLambdaFunctions = $listLambdaFunctions['Functions'];
        $next = $listLambdaFunctions->get('NextMarker');
        while ($next != false) {
            $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
            $next = $listLambdaFunctions->get('NextMarker');
            $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
        }
        foreach ($allLambdaFunctions as $function) {
            echo "{$function['FunctionName']}\n";
```

```
        }

        echo "\n\nAnd don't forget to clean up your data!\n";

        $lambdaService->deleteFunction($functionName);
        echo "Deleted Lambda function.\n";
        $iamService->deleteRole($role['RoleName']);
        echo "Deleted Role.\n";
        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Deleted all objects from the S3 bucket.\n";
        $s3client->deleteBucket(['Bucket' => $bucketName]);
        echo "Deleted the bucket.\n";
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function createFunction($functionName, $role, $bucketName, $handler)
    {
        //This assumes the Lambda function is in an S3 bucket.
        return $this->customWaiter(function () use ($functionName, $role,
$bucketName, $handler) {
            return $this->lambdaClient->createFunction([
                'Code' => [
                    'S3Bucket' => $bucketName,
                    'S3Key' => $functionName,
                ],
                'FunctionName' => $functionName,
                'Role' => $role['Arn'],
                'Runtime' => 'python3.9',
                'Handler' => "$handler.lambda_handler",
            ]);
        });
    }
```

- API-Details finden Sie CreateFunction in der AWS SDK für PHP API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteFunction($functionName)
    {
        return $this->lambdaClient->deleteFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- API-Details finden Sie DeleteFunction in der AWS SDK für PHP API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function getFunction($functionName)
    {
        return $this->lambdaClient->getFunction([
            'FunctionName' => $functionName,
        ]);
    }
```

- API-Details finden Sie GetFunction in der AWS SDK für PHP API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für PHP -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
public function listFunctions($maxItems = 50, $marker = null)
{
    if (is_null($marker)) {
        return $this->lambdaClient->listFunctions([
            'MaxItems' => $maxItems,
        ]);
    }
```

```
        return $this->lambdaClient->listFunctions([
            'Marker' => $marker,
            'MaxItems' => $maxItems,
        ]);
    }
```

- API-Details finden Sie ListFunctions in der AWS SDK für PHP API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
    {
        return $this->lambdaClient->updateFunctionCode([
            'FunctionName' => $functionName,
            'S3Bucket' => $s3Bucket,
            'S3Key' => $s3Key,
        ]);
    }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für PHP API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
   {
       return $this->lambdaClient->updateFunctionConfiguration([
           'FunctionName' => $functionName,
           'Handler' => "$handler.lambda_handler",
           'Environment' => $environment,
       ]);
   }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für PHP API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS -Community.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
```

```
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
```

```
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Kinesis\KinesisHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends KinesisHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleKinesis(KinesisEvent $event, Context $context): void
    {
        $this->logger->info("Processing records");
        $records = $event->getRecords();
        foreach ($records as $record) {
            $data = $record->getData();
            $this->logger->info(json_encode($data));
            // TODO: Do interesting work based on the new data
```

```
            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\DynamoDb\DynamoDbHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends DynamoDbHandler
```

```
{
    private StderrLogger $logger;

    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleDynamoDb(DynamoDbEvent $event, Context $context): void
    {
        $this->logger->info("Processing DynamoDb table items");
        $records = $event->getRecords();

        foreach ($records as $record) {
            $eventName = $record->getEventName();
            $keys = $record->getKeys();
            $old = $record->getOldImage();
            $new = $record->getNewImage();

            $this->logger->info("Event Name:".$eventName."\n");
            $this->logger->info("Keys:". json_encode($keys)."\n");
            $this->logger->info("Old Image:". json_encode($old)."\n");
            $this->logger->info("New Image:". json_encode($new));

            // TODO: Do interesting work based on the new data

            // Any exception thrown will be logged and the invocation will be marked
 as failed
        }

        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords items");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

require __DIR__.'/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Handler;

class DocumentDBEventHandler implements Handler
{
    public function handle($event, Context $context): string
    {

        $events = $event['events'] ?? [];
        foreach ($events as $record) {
            $this->logDocumentDBEvent($record['event']);
        }
        return 'OK';
    }

    private function logDocumentDBEvent($event): void
    {
        // Extract information from the event record

        $operationType = $event['operationType'] ?? 'Unknown';
        $db = $event['ns']['db'] ?? 'Unknown';
```

```
        $collection = $event['ns']['coll'] ?? 'Unknown';
        $fullDocument = $event['fullDocument'] ?? [];

        // Log the event details

        echo "Operation type: $operationType\n";
        echo "Database: $db\n";
        echo "Collection: $collection\n";
        echo "Full document: " . json_encode($fullDocument, JSON_PRETTY_PRINT) .
 "\n";
    }
}
return new DocumentDBEventHandler();
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
```

```
require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");
```

```
        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
```

```
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kinesis\KinesisEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
```

```
require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $kinesisEvent = new KinesisEvent($event);
        $this->logger->info("Processing records");
        $records = $kinesisEvent->getRecords();

        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\DynamoDb\DynamoDbEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
```

```
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handle(mixed $event, Context $context): array
    {
        $dynamoDbEvent = new DynamoDbEvent($event);
        $this->logger->info("Processing records");

        $records = $dynamoDbEvent->getRecords();
        $failedRecords = [];
        foreach ($records as $record) {
            try {
                $data = $record->getData();
                $this->logger->info(json_encode($data));
                // TODO: Do interesting work based on the new data
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $failedRecords[] = $record->getSequenceNumber();
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");

        // change format for the response
        $failures = array_map(
            fn(string $sequenceNumber) => ['itemIdentifier' => $sequenceNumber],
            $failedRecords
        );

        return [
            'batchItemFailures' => $failures
        ];
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
```

```
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Beispiele für Amazon MSK unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\Kafka\KafkaEvent;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
```

```
     */
    public function handle(mixed $event, Context $context): void
    {
        $kafkaEvent = new KafkaEvent($event);
        $this->logger->info("Processing records");
        $records = $kafkaEvent->getRecords();

        foreach ($records as $record) {
            try {
                $key = $record->getKey();
                $this->logger->info("Key: $key");

                $values = $record->getValue();
                $this->logger->info(json_encode($values));

                foreach ($values as $value) {
                    $this->logger->info("Value: $value");
                }

            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords records");
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Beispiele für Amazon RDS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;



$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$dbClass = 'db.t2.micro';
$storage = 5;
$engine = 'MySQL';
```

```
$username = 'MyUser';
$password = 'MyPassword';

try {
    $result = $rdsClient->createDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBInstanceClass' => $dbClass,
        'AllocatedStorage' => $storage,
        'Engine' => $engine,
        'MasterUsername' => $username,
        'MasterUserPassword' => $password,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der AWS SDK für PHP -API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

$dbIdentifier = '<<{{db-identifier}}>>';
$snapshotName = '<<{{backup_2018_12_25}}>>';

try {
    $result = $rdsClient->createDBSnapshot([
        'DBInstanceIdentifier' => $dbIdentifier,
        'DBSnapshotIdentifier' => $snapshotName,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für PHP -API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-1'
]);

$dbIdentifier = '<<{{db-identifier}}>>';

try {
    $result = $rdsClient->deleteDBInstance([
        'DBInstanceIdentifier' => $dbIdentifier,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der AWS SDK für PHP -API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require __DIR__ . '/vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
//Create an RDSClient
$rdsClient = new Aws\Rds\RdsClient([
    'region' => 'us-east-2'
]);

try {
    $result = $rdsClient->describeDBInstances();
    foreach ($result['DBInstances'] as $instance) {
        print('<p>DB Identifier: ' . $instance['DBInstanceIdentifier']);
        print('<br />Endpoint: ' . $instance['Endpoint']["Address"]
            . ':' . $instance['Endpoint']["Port"]);
        print('<br />Current Status: ' . $instance["DBInstanceStatus"]);
        print('</p>');
    }
    print(" Raw Result ");
    var_dump($result);
} catch (AwsException $e) {
    echo $e->getMessage();
    echo "\n";
}
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für PHP - API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PPHP-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS

- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit PHP.

```
<?php
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

# using bref/bref and bref/logger for simplicity
```

```
use Bref\Context\Context;
use Bref\Event\Handler as StdHandler;
use Bref\Logger\StderrLogger;
use Aws\Rds\AuthTokenGenerator;
use Aws\Credentials\CredentialProvider;

require __DIR__ . '/vendor/autoload.php';

class Handler implements StdHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }


    private function getAuthToken(): string {
        // Define connection authentication parameters
        $dbConnection = [
            'hostname' => getenv('DB_HOSTNAME'),
            'port' => getenv('DB_PORT'),
            'username' => getenv('DB_USERNAME'),
            'region' => getenv('AWS_REGION'),
        ];

        // Create RDS AuthTokenGenerator object
        $generator = new AuthTokenGenerator(CredentialProvider::defaultProvider());

        // Request authorization token from RDS, specifying the username
        return $generator->createToken(
            $dbConnection['hostname'] . ':' . $dbConnection['port'],
            $dbConnection['region'],
            $dbConnection['username']
        );
    }

    private function getQueryResults() {
        // Obtain auth token
        $token = $this->getAuthToken();

        // Define connection configuration
        $connectionConfig = [
            'host' => getenv('DB_HOSTNAME'),
```

```
            'user' => getenv('DB_USERNAME'),
            'password' => $token,
            'database' => getenv('DB_NAME'),
        ];

        // Create the connection to the DB
        $conn = new PDO(

 "mysql:host={$connectionConfig['host']};dbname={$connectionConfig['database']}",
            $connectionConfig['user'],
            $connectionConfig['password'],
            [
                PDO::MYSQL_ATTR_SSL_CA => '/path/to/rds-ca-2019-root.pem',
                PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
            ]
        );

        // Obtain the result of the query
        $stmt = $conn->prepare('SELECT ?+? AS sum');
        $stmt->execute([3, 2]);

        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * @param mixed $event
     * @param Context $context
     * @return array
     */
    public function handle(mixed $event, Context $context): array
    {
        $this->logger->info("Processing query");

        // Execute database flow
        $result = $this->getQueryResults();

        return [
            'sum' => $result['sum']
        ];
    }
}

$logger = new StderrLogger();
```

```
return new Handler($logger);
```

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für PHP Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PPHP-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Rekognition unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Beispiele für Amazon S3 unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- API-Details finden Sie ListBuckets in der AWS SDK für PHP API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("--------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
```

```
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
```

```
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Einfache Kopie eines Objekts.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
```

```
            echo "Failed to copy $fileName with error: " . $exception->getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- API-Details finden Sie CopyObject in der AWS SDK für PHP API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- API-Details finden Sie CreateBucket in der AWS SDK für PHP API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie einen leeren Bucket.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- API-Details finden Sie DeleteBucket in der AWS SDK für PHP API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    public function deleteObject(string $bucketName, string $fileName, array $args =
[])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
```

- API-Details finden Sie DeleteObject in der AWS SDK für PHP API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Reihe von Objekten aus einer Schlüsselliste.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
```

```
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (isset($check['Contents']) && count($check['Contents']) > 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- API-Details finden Sie DeleteObjects in der AWS SDK für PHP API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt ab.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- API-Details finden Sie GetObject in der AWS SDK für PHP API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Objekte in einem Bucket auf.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
```

```
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für PHP API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie ein Objekt in einen Bucket hoch.

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- API-Details finden Sie PutObject in der AWS SDK für PHP API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für PHP

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the S3
 bucket to use: ");
```

```
        $key = $this->testable_readline("Next, provide the key of an object in the
 given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
```

```
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
```

```
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
```

```
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
```

```
    }

}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
```

```
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' . $bucket .
 '. Make sure they exist and your bucket is in the same region as this function.' .
 "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Beispiele für S3 Directory Buckets unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von S3 Directory Buckets Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.

- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.
- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
        echo "\n";
        echo "--------------------------------------\n";
        echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
        echo "--------------------------------------\n";

        // Change these both of these values to use a different region/availability
 zone.
        $region = "us-west-2";
        $az = "usw2-az1";

        $this->s3Service = new S3Service(new S3Client(['region' => $region]));
        $this->iamService = new IAMService(new IamClient(['region' => $region]));

        $uuid = uniqid();

        echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute resources
 in the same AZ as your Directory
bucket.\n
```

```
INTRO;
        pressEnter();
        // 1. Configure a gateway VPC endpoint. This is the recommended method to
 allow S3 Express One Zone traffic without
        // the need to pass through an internet gateway or NAT device.
        echo "\n";
        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program is
 running in an EC2 instance in the same AZ as your Directory buckets will be.\n";
        $ec2Choice = testable_readline("Are you running this in an EC2 instance
 located in the same AZ as your intended Directory buckets? Enter Y/y to setup a VPC
 Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
 $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
 ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
 $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
```

```
        echo "Now, we'll set up some policies, roles, and a user. This user will
 only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();
        $file = file_get_contents(__DIR__ . "/../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
        $result = $this->cloudFormationClient->createStack([
            'StackName' => $stackName,
            'TemplateBody' => $file,
            'Capabilities' => ['CAPABILITY_IAM'],
        ]);
        $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
 ['StackName' => $stackName]);
        try {
            $waiter->promise()->wait();
        }catch(CloudFormationException $caught){
            echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
        $this->resources['stackName'] = $stackName;
        $stackInfo = $this->cloudFormationClient->describeStacks([
            'StackName' => $result['StackId'],
        ]);

        $expressUserName = "";
        $regularUserName = "";
        foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
            if ($output['OutputKey'] == "RegularUser") {
                $regularUserName = $output['OutputValue'];
            }
            if ($output['OutputKey'] == "ExpressUser") {
                $expressUserName = $output['OutputValue'];
            }
        }
        $regularKey = $this->iamService->createAccessKey($regularUserName);
        $regularCredentials = new Credentials($regularKey['AccessKeyId'],
 $regularKey['SecretAccessKey']);
        $expressKey = $this->iamService->createAccessKey($expressUserName);
        $expressCredentials = new Credentials($expressKey['AccessKeyId'],
 $expressKey['SecretAccessKey']);
```

```
        // 3. Create an additional client using the credentials with S3 Express
permissions.
        echo "\n";
        echo "3. Create an additional client using the credentials with S3 Express
permissions.\n";
        echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
        pressEnter();
        $s3RegularClient = new S3Client([
            'Region' => $region,
            'Credentials' => $regularCredentials,
        ]);
        $s3RegularService = new S3Service($s3RegularClient);
        $s3ExpressClient = new S3Client([
            'Region' => $region,
            'Credentials' => $expressCredentials,
        ]);
        $s3ExpressService = new S3Service($s3ExpressClient);
        echo "All the roles and policies were created an attached to the user. Then,
a new S3 Client and Service were created using that user's credentials.\n";
        echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
        pressEnter();

        // 4. Create two buckets.
        echo "\n";
        echo "3. Create two buckets.\n";
        echo "Now we will create a Directory bucket, which is the linchpin of the S3
Express One Zone service.\n";
        echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
        echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
        pressEnter();

        // Create a directory bucket. These are different from normal S3 buckets in
subtle ways.
        $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-s3";
        echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
        pressEnter();
        $s3ExpressService->createBucket($directoryBucketName, [
```

```
            'CreateBucketConfiguration' => [
                'Bucket' => [
                    'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                    'DataRedundancy' => "SingleAvailabilityZone",
                ],
                'Location' => [
                    'Name' => $az,
                    'Type' => "AvailabilityZone",
                ],
            ],
        ]);
        $this->resources['directoryBucketName'] = $directoryBucketName;

        // Create a normal bucket.
        $normalBucketName = "normal-bucket-$uuid";
        $s3RegularService->createBucket($normalBucketName);
        $this->resources['normalBucketName'] = $normalBucketName;
        echo "Great! Both buckets were created.\n";
        pressEnter();

        // 5. Create an object and copy it over.
        echo "\n";
        echo "5. Create an object and copy it over.\n";
        echo "We'll create a basic object consisting of some text and upload it to
the normal bucket.\n";
        echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
        echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
        pressEnter();

        $objectKey = "basic-text-object";
        $s3RegularService->putObject($normalBucketName, $objectKey, $args = ['Body'
=> "Look Ma, I'm a bucket!"]);
        $this->resources['objectKey'] = $objectKey;

        // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
        $s3ExpressService->createSession($directoryBucketName);
        $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");
```

```
        echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
        echo "Instead of validating permissions on every call as normal buckets do,
Directory buckets utilize the user credentials and session token to validate.\n";
        echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot of
time saved.\n";
        pressEnter();

        // 6. Demonstrate performance difference.
        echo "\n";
        echo "6. Demonstrate performance difference.\n";
        $downloads = 1000;
        echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
        $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
        if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
            $downloads = $downloadChoice;
        }

        // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
        $directoryStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3ExpressService->getObject($directoryBucketName, $objectKey);
        }
        $directoryEndTime = hrtime(true);
        $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

        $normalStartTime = hrtime(true);
        for($i = 0; $i < $downloads; ++$i){
            $s3RegularService->getObject($normalBucketName, $objectKey);
        }
        $normalEndTime = hrtime(true);
        $normalTimeDiff = $normalEndTime - $normalStartTime;

        echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
```

```
        echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
 " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
 seconds.\n";
        pressEnter();

        // 7. Populate the buckets to show the lexicographical difference.
        echo "\n";
        echo "7. Populate the buckets to show the lexicographical difference.\n";
        echo "Now let's explore how Directory buckets store objects in a different
 manner to regular buckets.\n";
        echo "The key is in the name \"Directory!\"\n";
        echo "Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual directories/folders.\n";
        echo "This allows for more rapid indexing, traversing, and therefore
 retrieval times!\n";
        echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
        echo "This structural difference also causes ListObjects to behave
 differently, which can cause unexpected results.\n";
        echo "Let's add a few more objects with layered directories as see how the
 output of ListObjects changes.\n";
        pressEnter();

        // Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        $otherObject = "other/$objectKey";
        $altObject = "alt/$objectKey";
        $otherAltObject = "other/alt/$objectKey";
        $s3ExpressService->putObject($directoryBucketName, $otherObject);
        $s3RegularService->putObject($normalBucketName, $otherObject);
        $this->resources['otherObject'] = $otherObject;
        $s3ExpressService->putObject($directoryBucketName, $altObject);
        $s3RegularService->putObject($normalBucketName, $altObject);
        $this->resources['altObject'] = $altObject;
        $s3ExpressService->putObject($directoryBucketName, $otherAltObject);
        $s3RegularService->putObject($normalBucketName, $otherAltObject);
        $this->resources['otherAltObject'] = $otherAltObject;

        $listDirectoryBucket = $s3ExpressService->listObjects($directoryBucketName);
        $listNormalBucket = $s3RegularService->listObjects($normalBucketName);

        // Directory bucket content
        echo "Directory bucket content\n";
        foreach($listDirectoryBucket['Contents'] as $result){
```

```
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical order,
 while the directory bucket does not. This is because the normal bucket considers
 the whole \"key\" to be the object identifies, while the directory bucket actually
 creates directories and uses the object \"key\" as a path to the object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
 Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
 created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }



namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
```

```
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }


    public function emptyAndDeleteBucket($bucketName, array $args = [])
    {
        try {
            $objects = $this->listAllObjects($bucketName, $args);
            $this->deleteObjects($bucketName, $objects, $args);
            if ($this->verbose) {
                echo "Deleted all objects and folders from $bucketName.\n";
            }
            $this->deleteBucket($bucketName, $args);
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
```

```
                echo "\nPlease fix error with bucket deletion before continuing.\n";
            }
            throw $exception;
        }
    }



    public function createBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->createBucket($parameters);
            if ($this->verbose) {
                echo "Created the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket creation before continuing.";
            }
            throw $exception;
        }
    }



    public function putObject(string $bucketName, string $key, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $this->client->putObject($parameters);
            if ($this->verbose) {
                echo "Uploaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create $key in $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with object uploading before continuing.";
            }
            throw $exception;
```

```
        }
    }



    public function getObject(string $bucketName, string $key, array $args = []):
 Result
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key], $args);
        try {
            $object = $this->client->getObject($parameters);
            if ($this->verbose) {
                echo "Downloaded the object named: $key to the bucket named:
 $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to download $key from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object downloading before continuing.";
            }
            throw $exception;
        }
        return $object;
    }



    public function copyObject($bucketName, $key, $copySource, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
 "CopySource" => $copySource], $args);
        try {
            $this->client->copyObject($parameters);
            if ($this->verbose) {
                echo "Copied the object from: $copySource in $bucketName to: $key.
\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to copy $copySource in $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object copying before continuing.";
            }
```

```
            throw $exception;
        }
    }



    public function listObjects(string $bucketName, $start = 0, $max = 1000, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }



    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
```

```
        return $contents;
    }



    public function deleteObjects(string $bucketName, array $objects, array $args =
[])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' => ['Objects'
=> $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
```

```
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function deleteObject(string $bucketName, string $fileName, array $args =
 [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
 $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
 {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }



    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }



    public function preSignedUrl(CommandInterface $command, DateTimeInterface|int|
string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
 $options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }



    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
```

```
    }

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Beispiele für Amazon SES unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für PHP

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für PHP

Zeigt, wie Sie mithilfe von Amazon Simple Email Service (Amazon SES) eine Webanwendung erstellen, die Arbeitselemente in einer Amazon RDS-Datenbank nachverfolgt und Berichte per E-Mail versendet. AWS SDK für PHP In diesem Beispiel wird ein Frontend verwendet, das für React.js die Interaktion mit einem RESTful-PPHP-Backend erstellt wurde.

- Integrieren Sie eine React.js Webanwendung in Dienste. AWS
- In einer Amazon-RDS-Tabelle können Sie Elemente auflisten, aktualisieren und löschen.
- Senden Sie einen E-Mail-Bericht über gefilterte Arbeitselemente mit Amazon SES.
- Stellen Sie Beispielressourcen mit dem mitgelieferten AWS CloudFormation Skript bereit und verwalten Sie sie.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon SNS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

- Szenarien
- Serverless-Beispiele

Aktionen

`CheckIfPhoneNumberIsOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`CheckIfPhoneNumberIsOptedOut`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
```

```
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- API-Details finden Sie CheckIfPhoneNumberIsOptedOut in der AWS SDK für PHP API-Referenz.

`ConfirmSubscription`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSubscription`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie ConfirmSubscription in der AWS SDK für PHP API-Referenz.

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Create a Simple Notification Service topics in your AWS account at the requested
 region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- API-Details finden Sie CreateTopic in der AWS SDK für PHP API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie DeleteTopic in der AWS SDK für PHP API-Referenz.

`GetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetSMSAttributes`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter GetSMSAttributes in der AWS SDK für PHP -API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK für PHP API-Referenz.

`ListPhoneNumbersOptedOut`

Das folgende Codebeispiel zeigt die Verwendung`ListPhoneNumbersOptedOut`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- API-Details finden Sie ListPhoneNumbersOptedOut in der AWS SDK für PHP API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie ListSubscriptions in der AWS SDK für PHP API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie ListTopics in der AWS SDK für PHP API-Referenz.

Veröffentlichen

Das folgende Codebeispiel zeigt die VerwendungPublish.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
```

```
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für PHP -API-Referenz.

`SetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSMSAttributes`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter SetSMSAttributes in der AWS SDK für PHP -API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
```

```
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API-Details finden Sie SetTopicAttributes in der AWS SDK für PHP API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Abonnieren Sie einen HTTP-Endpunkt für ein Thema.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für PHP -API-Referenz.

`Abonnement beenden`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für PHP -API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS -Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für PHP

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Weitere Informationen finden Sie im AWS SDK für PHP -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für PHP -API-Referenz.

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von PHP

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-
custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be marked
 as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Beispiele für Amazon SQS unter Verwendung von SDK für PHP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für PHP

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für PHP

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

# using bref/bref and bref/logger for simplicity

use Bref\Context\Context;
use Bref\Event\InvalidLambdaEvent;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
```

```
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $body = $record->getBody();
            // TODO: Do interesting work based on the new message
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für PHP

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von SQS-Batchelementfehlern mit Lambda unter Verwendung von PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\Sqs\SqsEvent;
use Bref\Event\Sqs\SqsHandler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends SqsHandler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    /**
     * @throws JsonException
     * @throws \Bref\Event\InvalidLambdaEvent
     */
    public function handleSqs(SqsEvent $event, Context $context): void
    {
        $this->logger->info("Processing SQS records");
        $records = $event->getRecords();

        foreach ($records as $record) {
            try {
                // Assuming the SQS message is in JSON format
                $message = json_decode($record->getBody(), true);
                $this->logger->info(json_encode($message));
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                $this->logger->error($e->getMessage());
                // failed processing the record
                $this->markAsFailed($record);
            }
        }
        $totalRecords = count($records);
        $this->logger->info("Successfully processed $totalRecords SQS records");
```

```
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Codebeispiele für Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie den AWS -Tools für PowerShell V5 mit verwenden. AWS

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- Tools for PowerShell V5 Developer Guide — Weitere Informationen zur Verwendung mit PowerShell . AWS
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele mit Tools für V5 PowerShell
- Beispiele für Auto Scaling von Anwendungen mithilfe von Tools für V5 PowerShell
- WorkSpaces Anwendungsbeispiele, die Tools für V5 verwenden PowerShell

- Aurora-Beispiele zur Verwendung von Tools für V5 PowerShell
- Beispiele für Auto Scaling mithilfe von Tools für V5 PowerShell
- AWS Budgets Beispiele zur Verwendung von Tools für V5 PowerShell
- AWS Cloud9 Beispiele zur Verwendung von Tools für V5 PowerShell
- CloudFormation Beispiele zur Verwendung von Tools für V5 PowerShell
- CloudFront Beispiele zur Verwendung von Tools für V5 PowerShell
- CloudTrail Beispiele zur Verwendung von Tools für V5 PowerShell
- CloudWatch Beispiele zur Verwendung von Tools für V5 PowerShell
- CodeCommit Beispiele zur Verwendung von Tools für V5 PowerShell
- CodeDeploy Beispiele zur Verwendung von Tools für V5 PowerShell
- CodePipeline Beispiele zur Verwendung von Tools für V5 PowerShell
- Beispiele für Amazon Cognito Identity mit Tools für V5 PowerShell
- AWS Config Beispiele mit Tools for PowerShell V5
- Beispiele für Gerätefarmen, die Tools für V5 verwenden PowerShell
- Directory Service Beispiele zur Verwendung von Tools für V5 PowerShell
- AWS DMS Beispiele zur Verwendung von Tools für V5 PowerShell
- DynamoDB-Beispiele mit Tools für V5 PowerShell
- Amazon EC2-Beispiele zur Verwendung von Tools für V5 PowerShell
- Amazon ECR-Beispiele mit Tools für V5 PowerShell
- Amazon ECS-Beispiele zur Verwendung von Tools für V5 PowerShell
- Amazon EFS-Beispiele zur Verwendung von Tools für V5 PowerShell
- Amazon EKS-Beispiele zur Verwendung von Tools für V5 PowerShell
- Elastic Load Balancing — Beispiele für Version 1 unter Verwendung von Tools für V5 PowerShell
- Elastic Load Balancing — Beispiele für Version 2 unter Verwendung von Tools für V5 PowerShell
- Amazon FSx-Beispiele zur Verwendung von Tools für V5 PowerShell
- Amazon Glacier-Beispiele mit Tools für V5 PowerShell
- AWS Glue Beispiele zur Verwendung von Tools für V5 PowerShell
- AWS Health Beispiele zur Verwendung von Tools für V5 PowerShell
- IAM-Beispiele mit Tools für V5 PowerShell
- Kinesis-Beispiele mit Tools für V5 PowerShell

- Lambda-Beispiele mit Tools für V5 PowerShell
- Amazon ML-Beispiele zur Verwendung von Tools für V5 PowerShell
- Macie-Beispiele zur Verwendung von Tools für V5 PowerShell
- AWS-Preisliste Beispiele zur Verwendung von Tools für V5 PowerShell
- Beispiele für Ressourcengruppen, die Tools für V5 verwenden PowerShell
- API-Beispiele für das Taggen von Ressourcengruppen mithilfe von Tools für V5 PowerShell
- Route-53-Beispiele mithilfe von Tools für V5 PowerShell
- Amazon S3-Beispiele mit Tools für V5 PowerShell
- Security Hub CSPM-Beispiele unter Verwendung von Tools für V5 PowerShell
- Amazon SES-Beispiele zur Verwendung von Tools für V5 PowerShell
- Beispiele für Amazon SES API v2 unter Verwendung von Tools für V5 PowerShell
- Amazon SNS-Beispiele zur Verwendung von Tools für V5 PowerShell
- Amazon SQS-Beispiele mit Tools für V5 PowerShell
- AWS STS Beispiele zur Verwendung von Tools für V5 PowerShell
- Support Beispiele zur Verwendung von Tools für V5 PowerShell
- Beispiele für Systems Manager zur Verwendung von Tools für V5 PowerShell
- Amazon Translate-Beispiele mit Tools für V5 PowerShell
- AWS WAFV2 Beispiele zur Verwendung von Tools für V5 PowerShell
- WorkSpaces Beispiele zur Verwendung von Tools für V5 PowerShell

ACM-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit ACM verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-ACMCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ACMCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie Sie ein Zertifikat und dessen Zertifikatskette mithilfe des ARN des Zertifikats zurückgeben.

```
Get-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- API-Details finden Sie GetCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ACMCertificateDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ACMCertificateDetail`

Tools für PowerShell V5

Beispiel 1: Gibt Details zum angegebenen Zertifikat zurück.

```
Get-ACMCertificateDetail -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

Ausgabe:

```
CertificateArn          : arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
CreatedAt               : 1/21/2016 5:55:59 PM
DomainName              : www.example.com
DomainValidationOptions : {www.example.com}
InUseBy                 : {}
IssuedAt                : 1/1/0001 12:00:00 AM
Issuer                  :
KeyAlgorithm            : RSA-2048
NotAfter                : 1/1/0001 12:00:00 AM
NotBefore               : 1/1/0001 12:00:00 AM
RevocationReason        :
RevokedAt               : 1/1/0001 12:00:00 AM
```

```
Serial                  :
SignatureAlgorithm      : SHA256WITHRSA
Status                  : PENDING_VALIDATION
Subject                 : CN=www.example.com
SubjectAlternativeNames : {www.example.net}
```

- API-Details finden Sie DescribeCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ACMCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ACMCertificateList`

Tools für PowerShell V5

Beispiel 1: Ruft eine Liste aller Zertifikat-ARNs und den Domainnamen für jeden ARN ab. Das Cmdlet paginiert automatisch, um alle ARNs abzurufen. Um die Seitennummerierung manuell zu steuern, verwenden Sie den MaxItem Parameter -, um zu steuern, wie viele Zertifikat-ARNs für jeden Serviceaufruf zurückgegeben werden, und den NextToken Parameter -, um den Startpunkt für jeden Aufruf anzugeben.

```
Get-ACMCertificateList
```

Ausgabe:

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
 www.example.com
```

Beispiel 2: Ruft eine Liste all Ihrer Zertifikat-ARNs ab, bei denen der Zertifikatsstatus mit den angegebenen Status übereinstimmt.

```
Get-ACMCertificateList -CertificateStatus "VALIDATION_TIMED_OUT","FAILED"
```

Beispiel 3: Dieses Beispiel gibt eine Liste aller Zertifikate in der Region us-east-1 zurück, die den Schlüsseltyp RSA_2048 und die erweiterte Schlüsselverwendung oder den Zweck CODE_SIGNING aufweisen. Die Werte für diese Filterparameter finden Sie im Referenzthema

ListCertificates Filters API: https://docs.aws.amazon.com/acm/latest/APIReference/API_Filters.html.

```
Get-ACMCertificateList -Region us-east-1 -Includes_KeyType RSA_2048 -
Includes_ExtendedKeyUsage CODE_SIGNING
```

Ausgabe:

```
CertificateArn
 DomainName
--------------
 ----------
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-d7c0-48c1-af8d-2133d8f30zzz
 *.route53docs.com
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-98a5-443d-a734-800430c80zzz
 nerdzizm.net
arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-2be6-4376-8fa7-bad559525zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-e7ca-44c5-803e-24d9f2f36zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-1241-4b71-80b1-090305a62zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-8709-4568-8c64-f94617c99zzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-a8fa-4a61-98cf-e08ccc0eezzz

arn:aws:acm:us-east-1:8xxxxxxxxxxx:certificate/xxxxxxxx-fa47-40fe-a714-2d277d3eezzz
 *.route53docs.com
```

- API-Details finden Sie ListCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ACMCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `New-ACMCertificate`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com"
```

Ausgabe:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

Beispiel 2: Erstellt ein neues Zertifikat. Der Service gibt den ARN des neuen Zertifikats zurück.

```
New-ACMCertificate -DomainName "www.example.com" -SubjectAlternativeName
 "example.com","www.example.net"
```

Ausgabe:

```
arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

- API-Details finden Sie RequestCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ACMCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ACMCertificate`

Tools für PowerShell V5

Beispiel 1: Löscht das Zertifikat, das durch den angegebenen ARN und den zugehörigen privaten Schlüssel identifiziert wurde. Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, fügen Sie den Schalter -Force hinzu, um die Prompts zu unterdrücken.

```
Remove-ACMCertificate -CertificateArn "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
```

- API-Details finden Sie DeleteCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Send-ACMValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung`Send-ACMValidationEmail`.

Tools für PowerShell V5

Beispiel 1: Fordert an, dass die E-Mail zur Validierung der Eigentümerschaft der Domain „www.example.com“ gesendet wird. Wenn $ in Ihrer Shell auf 'Medium' oder niedriger gesetzt

ConfirmPreference ist, fordert das Cmdlet vor dem Fortfahren zur Bestätigung auf. Fügen Sie den Schalter -Force hinzu, um Bestätigungsaufforderungen zu unterdrücken.

```
$params = @{
    CertificateArn="arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
    Domain="www.example.com"
    ValidationDomain="example.com"
}
Send-ACMValidationEmail @params
```

- API-Details finden Sie ResendValidationEmail in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Auto Scaling von Anwendungen mithilfe von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Application Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-AASScalableTarget`

Das folgende Codebeispiel zeigt die Verwendung`Add-AASScalableTarget`.

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet registriert oder aktualisiert ein skalierbares Ziel. Ein skalierbares Ziel ist eine Ressource, die dank Application Auto Scaling auf- bzw. abskaliert werden kann.

```
Add-AASScalableTarget -ServiceNamespace AppStream -ResourceId fleet/MyFleet -
ScalableDimension appstream:fleet:DesiredCapacity -MinCapacity 2 -MaxCapacity 10
```

- API-Details finden Sie RegisterScalableTarget in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-AASScalableTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-AASScalableTarget`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den skalierbaren Zielen von Auto Scaling für Anwendungen im angegebenen Namespace bereitgestellt.

```
Get-AASScalableTarget -ServiceNamespace "AppStream"
```

Ausgabe:

```
CreationTime      : 11/7/2019 2:30:03 AM
MaxCapacity       : 5
MinCapacity       : 1
ResourceId        : fleet/Test
RoleARN           : arn:aws:iam::012345678912:role/aws-
service-role/appstream.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_AppStreamFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
SuspendedState    : Amazon.ApplicationAutoScaling.Model.SuspendedState
```

- API-Details finden Sie DescribeScalableTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-AASScalingActivity`

Das folgende Codebeispiel zeigt die Verwendung. `Get-AASScalingActivity`

Tools für PowerShell V5

Beispiel 1: Liefert beschreibende Informationen zu den Skalierungsaktivitäten im angegebenen Service-Namespace aus den letzten sechs Wochen.

```
Get-AASScalingActivity -ServiceNamespace AppStream
```

Ausgabe:

```
ActivityId        : 2827409f-b639-4cdb-a957-8055d5d07434
Cause             : monitor alarm Appstream2-MyFleet-default-scale-in-Alarm in state
 ALARM triggered policy default-scale-in
Description       : Setting desired capacity to 2.
Details           :
EndTime           : 12/14/2019 11:32:49 AM
ResourceId        : fleet/MyFleet
ScalableDimension : appstream:fleet:DesiredCapacity
ServiceNamespace  : appstream
StartTime         : 12/14/2019 11:32:14 AM
StatusCode        : Successful
StatusMessage     : Successfully set desired capacity to 2. Change successfully
 fulfilled by appstream.
```

- API-Details finden Sie DescribeScalingActivities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-AASScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-AASScalingPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet beschreibt die Richtlinien für Application Auto Scaling für den angegebenen Service-Namespace.

```
Get-AASScalingPolicy -ServiceNamespace AppStream
```

Ausgabe:

```
Alarms                                  : {Appstream2-LabFleet-default-scale-out-
Alarm}
```

```
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-out
PolicyName                                : default-scale-out
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :

Alarms                                    : {Appstream2-LabFleet-default-scale-in-
Alarm}
CreationTime                              : 9/3/2019 2:48:15 AM
PolicyARN                                 : arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:5659b069-b5cd-4af1-9f7f-3e956d36233e:resource/
appstream/fleet/LabFleet:
                                            policyName/default-scale-in
PolicyName                                : default-scale-in
PolicyType                                : StepScaling
ResourceId                                : fleet/LabFleet
ScalableDimension                         : appstream:fleet:DesiredCapacity
ServiceNamespace                          : appstream
StepScalingPolicyConfiguration            :
 Amazon.ApplicationAutoScaling.Model.StepScalingPolicyConfiguration
TargetTrackingScalingPolicyConfiguration :
```

- API-Details finden Sie DescribeScalingPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-AASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-AASScheduledAction`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die für die Auto-Scaling-Gruppe geplanten Aktionen auf, die noch nicht ausgeführt wurden oder deren Endzeit noch nicht erreicht ist.

```
Get-AASScheduledAction -ServiceNamespace AppStream
```

Ausgabe:

```
CreationTime         : 12/22/2019 9:25:52 AM
EndTime              : 1/1/0001 12:00:00 AM
ResourceId           : fleet/MyFleet
ScalableDimension    : appstream:fleet:DesiredCapacity
ScalableTargetAction : Amazon.ApplicationAutoScaling.Model.ScalableTargetAction
Schedule             : cron(0 0 8 ? * MON-FRI *)
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:012345678912:scheduledAction:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:scheduledActionName
                       /WeekDaysFleetScaling
ScheduledActionName  : WeekDaysFleetScaling
ServiceNamespace     : appstream
StartTime            : 1/1/0001 12:00:00 AM
```

- API-Details finden Sie DescribeScheduledActions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-AASScalableTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-AASScalableTarget`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet hebt die Registrierung eines skalierbaren Ziels von Application Auto Scaling auf. Durch das Aufheben der Registrierung eines skalierbaren Ziels werden die zugehörigen Skalierungsrichtlinien gelöscht.

```
Remove-AASScalableTarget -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -ServiceNamespace AppStream
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScalableTarget (DeregisterScalableTarget)" on
 target "fleet/MyFleet".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeregisterScalableTarget in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-AASScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-AASScalingPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht die angegebene Skalierungsrichtlinie für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScalingPolicy -ServiceNamespace AppStream -PolicyName "default-scale-out"
 -ResourceId fleet/Test -ScalableDimension appstream:fleet:DesiredCapacity
```

- API-Details finden Sie DeleteScalingPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-AASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-AASScheduledAction`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht die angegebene geplante Aktion für ein skalierbares Ziel von Application Auto Scaling.

```
Remove-AASScheduledAction -ServiceNamespace AppStream -ScheduledActionName
 WeekDaysFleetScaling -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-AASScheduledAction (DeleteScheduledAction)" on
 target "WeekDaysFleetScaling".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteScheduledAction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-AASScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Set-AASScalingPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine Richtlinie für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScalingPolicy -ServiceNamespace AppStream -PolicyName ASFleetScaleInPolicy
 -PolicyType StepScaling  -ResourceId fleet/MyFleet -ScalableDimension
 appstream:fleet:DesiredCapacity -StepScalingPolicyConfiguration_AdjustmentType
 ChangeInCapacity -StepScalingPolicyConfiguration_Cooldown 360
 -StepScalingPolicyConfiguration_MetricAggregationType Average -
StepScalingPolicyConfiguration_StepAdjustments @{ScalingAdjustment = -1;
 MetricIntervalUpperBound = 0}
```

Ausgabe:

```
Alarms    PolicyARN
------    ---------
{}        arn:aws:autoscaling:us-
west-2:012345678912:scalingPolicy:4897ca24-3caa-4bf1-8484-851a089b243c:resource/
appstream/fleet/MyFleet:policyName/ASFleetScaleInPolicy
```

- API-Details finden Sie PutScalingPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-AASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Set-AASScheduledAction`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet erstellt oder aktualisiert eine geplante Aktion für ein skalierbares Ziel von Application Auto Scaling. Jedes skalierbare Ziel wird durch einen Service-Namespace, eine Ressourcen-ID und eine skalierbare Dimension identifiziert.

```
Set-AASScheduledAction -ServiceNamespace AppStream -ResourceId fleet/
MyFleet -Schedule "cron(0 0 8 ? * MON-FRI *)" -ScalableDimension
 appstream:fleet:DesiredCapacity -ScheduledActionName WeekDaysFleetScaling -
ScalableTargetAction_MinCapacity 5 -ScalableTargetAction_MaxCapacity 10
```

- API-Details finden Sie PutScheduledAction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

WorkSpaces Anwendungsbeispiele, die Tools für V5 verwenden PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit WorkSpaces Anwendungen verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-APSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-APSResourceTag`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel fügt der Ressource ein Ressourcen-Tag hinzu AppStream

```
Add-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -Tag @{StackState='Test'} -Select ^Tag
```

Ausgabe:

```
Name                           Value
----                           -----
StackState                     Test
```

- API-Details finden Sie TagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Copy-APSImage`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-APSImage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Image in eine andere Region kopiert

```
Copy-APSImage -DestinationImageName TestImageCopy -DestinationRegion us-west-2 -
SourceImageName Powershell
```

Ausgabe:

```
TestImageCopy
```

- API-Details finden Sie CopyImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-APSUSer`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-APSUSer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Benutzer in USERPOOL deaktiviert

```
Disable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- API-Details finden Sie DisableUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-APSUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein deaktivierter Benutzer in USERPOOL aktiviert

```
Enable-APSUser -AuthenticationType USERPOOL -UserName TestUser@lab.com
```

- API-Details finden Sie EnableUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSAssociatedFleetList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSAssociatedFleetList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die einem Stack zugeordnete Flotte angezeigt

```
Get-APSAssociatedFleetList -StackName PowershellStack
```

Ausgabe:

```
PowershellFleet
```

- API-Details finden Sie ListAssociatedFleets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSAssociatedStackList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSAssociatedStackList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der einer Flotte zugeordnete Stack angezeigt

```
Get-APSAssociatedStackList -FleetName PowershellFleet
```

Ausgabe:

```
PowershellStack
```

- API-Details finden Sie ListAssociatedStacks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSDirectoryConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSDirectoryConfigList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Verzeichniskonfigurationen angezeigt, die in erstellt wurden AppStream

```
Get-APSDirectoryConfigList | Select DirectoryName,
 OrganizationalUnitDistinguishedNames, CreatedTime
```

Ausgabe:

```
DirectoryName OrganizationalUnitDistinguishedNames CreatedTime
------------- ------------------------------------ -----------
Test.com      {OU=AppStream,DC=Test,DC=com}    9/6/2019 10:56:40 AM
contoso.com   {OU=AppStream,OU=contoso,DC=contoso,DC=com}  8/9/2019 9:08:50 AM
```

- API-Details finden Sie DescribeDirectoryConfigs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSFleetList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSFleetList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details einer Flotte angezeigt

```
Get-APSFleetList -Name Test
```

Ausgabe:

```
Arn                            : arn:aws:appstream:us-east-1:1234567890:fleet/Test
ComputeCapacityStatus          : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                    : 9/12/2019 5:00:45 PM
Description                    : Test
DisconnectTimeoutInSeconds     : 900
DisplayName                    : Test
DomainJoinInfo                 :
EnableDefaultInternetAccess    : False
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                       : arn:aws:appstream:us-east-1:1234567890:image/Test
ImageName                      : Test
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
Name                           : Test
State                          : STOPPED
VpcConfig                      : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie DescribeFleets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSImageBuilderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImageBuilderList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt Details eines ImageBuilder

```
Get-APSImageBuilderList -Name TestImage
```

Ausgabe:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:1234567890:image-builder/
TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
```

```
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : STOPPED
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie DescribeImageBuilders in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSImageList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImageList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden private AppStream Bilder angezeigt

```
Get-APSImageList -Type PRIVATE | select DisplayName, ImageBuilderName, Visibility,
 arn
```

Ausgabe:

```
DisplayName          ImageBuilderName      Visibility Arn
-----------          ----------------      ---------- ---
OfficeApps           OfficeApps            PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/OfficeApps
SessionScriptV2      SessionScriptTest     PRIVATE    arn:aws:appstream:us-
east-1:123456789012:image/SessionScriptV2
```

- API-Details finden Sie DescribeImages in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSImagePermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Bildberechtigungen für ein geteiltes AppStream Bild angezeigt

```
Get-APSImagePermission -Name Powershell  | select SharedAccountId,
 @{n="AllowFleet";e={$_.ImagePermissions.AllowFleet}},
 @{n="AllowImageBuilder";e={$_.ImagePermissions.AllowImageBuilder}}
```

Ausgabe:

```
SharedAccountId AllowFleet AllowImageBuilder
--------------- ---------- -----------------
123456789012          True              True
```

- API-Details finden Sie DescribeImagePermissions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSSessionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSSessionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Sitzungen für eine Flotte angezeigt

```
Get-APSSessionList -FleetName PowershellFleet -StackName PowershellStack
```

Ausgabe:

```
AuthenticationType         : API
ConnectionState            : CONNECTED
FleetName                  : PowershellFleet
Id                         : d8987c70-4394-4324-a396-2d485c26f2a2
MaxExpirationTime          : 12/27/2019 4:54:07 AM
NetworkAccessConfiguration : Amazon.AppStream.Model.NetworkAccessConfiguration
StackName                  : PowershellStack
```

```
StartTime                   : 12/26/2019 12:54:12 PM
State                       : ACTIVE
UserId                      : Test
```

- API-Details finden Sie DescribeSessions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSStackList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSStackList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von AppStream Stacks angezeigt

```
Get-APSStackList | Select DisplayName, Arn, CreatedTime
```

Ausgabe:

```
DisplayName                   Arn
                        CreatedTime
-----------                   ---
                        -----------
PowershellStack               arn:aws:appstream:us-east-1:123456789012:stack/
PowershellStack               4/24/2019 8:49:29 AM
SessionScriptTest             arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest             9/12/2019 3:23:12 PM
```

- API-Details finden Sie DescribeStacks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSTagsForResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSTagsForResourceList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Tags auf einer AppStream Ressource angezeigt

```
Get-APSTagsForResourceList -ResourceArn arn:aws:appstream:us-
east-1:123456789012:stack/SessionScriptTest
```

Ausgabe:

```
Key        Value
---        -----
StackState Test
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUsageReportSubscription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden AppStreamUsageReport Konfigurationsdetails angezeigt

```
Get-APSUsageReportSubscription
```

Ausgabe:

```
LastGeneratedReportDate S3BucketName                                Schedule
 SubscriptionErrors
----------------------- ------------                                --------
 ------------------
1/1/0001 12:00:00 AM    appstream-logs-us-east-1-123456789012-sik1hnxe DAILY    {}
```

- API-Details finden Sie DescribeUsageReportSubscriptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Benutzern mit dem Status „Aktiviert“ angezeigt

```
Get-APSUser -AuthenticationType USERPOOL | Select-Object UserName,
 AuthenticationType, Enabled
```

Ausgabe:

```
UserName                   AuthenticationType Enabled
--------                   ------------------ -------
foo1@contoso.com USERPOOL              True
foo2@contoso.com        USERPOOL              True
foo3@contoso.com    USERPOOL              True
foo4@contoso.com   USERPOOL              True
foo5@contoso.com         USERPOOL              True
```

- API-Details finden Sie DescribeUsers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-APSUserStackAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-APSUserStackAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der Benutzer angezeigt, die einem Stack zugewiesen sind

```
Get-APSUserStackAssociation -StackName PowershellStack
```

Ausgabe:

```
AuthenticationType SendEmailNotification StackName       UserName
------------------ --------------------- ---------       --------
USERPOOL           False                 PowershellStack TestUser1@lab.com
USERPOOL           False                 PowershellStack TestUser2@lab.com
```

- API-Details finden Sie DescribeUserStackAssociations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung`New-APSDirectoryConfig`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt eine Verzeichniskonfiguration in AppStream

```
New-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso\ServiceAccount
 -ServiceAccountCredentials_AccountPassword MyPass -DirectoryName contoso.com -
OrganizationalUnitDistinguishedName "OU=AppStream,OU=Contoso,DC=Contoso,DC=com"
```

Ausgabe:

```
CreatedTime            DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------            ------------- -----------------------------------
 -------------------------
12/27/2019 11:00:30 AM contoso.com   {OU=AppStream,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- API-Details finden Sie CreateDirectoryConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSFleet`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt eine neue Flotte AppStream

```
New-APSFleet -ComputeCapacity_DesiredInstance 1 -InstanceType stream.standard.medium
 -Name TestFleet -DisplayName TestFleet -FleetType ON_DEMAND -
EnableDefaultInternetAccess $True -VpcConfig_SubnetIds "subnet-123ce32","subnet-
a1234cfd" -VpcConfig_SecurityGroupIds sg-4d012a34 -ImageName SessionScriptTest -
Region us-west-2
```

Ausgabe:

```
Arn                           : arn:aws:appstream:us-west-2:123456789012:fleet/
TestFleet
ComputeCapacityStatus         : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                   : 12/27/2019 11:24:42 AM
Description                   :
DisconnectTimeoutInSeconds    : 900
DisplayName                   : TestFleet
DomainJoinInfo                :
EnableDefaultInternetAccess   : True
```

```
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 0
ImageArn                       : arn:aws:appstream:us-west-2:123456789012:image/
SessionScriptTest
ImageName                      : SessionScriptTest
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
Name                           : TestFleet
State                          : STOPPED
VpcConfig                      : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie CreateFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-APSImageBuilder

Das folgende Codebeispiel zeigt die Verwendung. `New-APSImageBuilder`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt einen Image Builder in AppStream

```
New-APSImageBuilder -InstanceType stream.standard.medium -Name TestIB -DisplayName
 TestIB -ImageName AppStream-WinServer2012R2-12-12-2019 -EnableDefaultInternetAccess
 $True -VpcConfig_SubnetId subnet-a1234cfd -VpcConfig_SecurityGroupIds sg-2d012a34 -
Region us-west-2
```

Ausgabe:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
DomainJoinInfo              :
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
```

```
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  :
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie CreateImageBuilder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSImageBuilderStreamingURL`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSImageBuilderStreamingURL`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt eine ImageBuilder Streaming-URL mit einer Gültigkeit von 2 Stunden

```
New-APSImageBuilderStreamingURL -Name TestIB -Validity 7200 -Region us-west-2
```

Ausgabe:

```
Expires               StreamingURL
-------               ------------
12/27/2019 1:49:13 PM https://appstream2.us-west-2.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiQURNSU4iLCJleHBpcmVzIjoiMTU3NzQ1NDU1MyIsImF3c0FjY291bnRJZCI6IjM5MzQwM
```

- API-Details finden Sie CreateImageBuilderStreamingURL in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSStack`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt einen neuen Stack AppStream

```
New-APSStack -Name TestStack -DisplayName TestStack -ApplicationSettings_Enabled
 $True -ApplicationSettings_SettingsGroup TestStack -Region us-west-2
```

Ausgabe:

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-west-2:123456789012:stack/TestStack
CreatedTime         : 12/27/2019 12:34:19 PM
Description         :
DisplayName         : TestStack
EmbedHostDomains    : {}
FeedbackURL         :
Name                : TestStack
RedirectURL         :
StackErrors         : {}
StorageConnectors   : {}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- API-Details finden Sie CreateStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSStreamingURL`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSStreamingURL`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Streaming-URL eines Stacks erstellt

```
New-APSStreamingURL -StackName SessionScriptTest -FleetName SessionScriptNew -UserId
 TestUser
```

Ausgabe:

```
Expires                StreamingURL
-------                ------------
12/27/2019 12:43:37 PM https://appstream2.us-east-1.aws.amazon.com/authenticate?
parameters=eyJ0eXBlIjoiRU5EX1VTRVIiLCJleHBpcmVzIjoiMTU3NzQ1MDYxNyIsImF3c0FjY291bnRJZCI6IjM5M
```

- API-Details finden Sie unter CreateStreaming URL in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSUsageReportSubscription`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktiviert AppStream Nutzungsberichte

```
New-APSUsageReportSubscription
```

Ausgabe:

```
S3BucketName                                    Schedule
------------                                    --------
appstream-logs-us-east-1-123456789012-sik2hnxe DAILY
```

- API-Details finden Sie CreateUsageReportSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `New-APSUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Benutzer in USERPOOL erstellt

```
New-APSUser -UserName Test@lab.com -AuthenticationType USERPOOL -FirstName 'kt' -
LastName 'aws' -Select ^UserName
```

Ausgabe:

```
Test@lab.com
```

- API-Details finden Sie CreateUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Register-APSFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Flotte bei einem Stack registriert

```
Register-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- API-Details finden Sie AssociateFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-APSUserStackBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Register-APSUserStackBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird einem Benutzer in USERPOOL ein Stack zugewiesen

```
Register-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- API-Details finden Sie BatchAssociateUserStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSDirectoryConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die AppStream Verzeichniskonfiguration entfernt

```
Remove-APSDirectoryConfig -DirectoryName contoso.com
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSDirectoryConfig (DeleteDirectoryConfig)" on
 target "contoso.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteDirectoryConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSFleet`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel entfernt, löscht eine Flotte AppStream

```
Remove-APSFleet -Name TestFleet -Region us-west-2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSFleet (DeleteFleet)" on target "TestFleet".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSImage`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Image gelöscht

```
Remove-APSImage -Name TestImage -Region us-west-2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSImage (DeleteImage)" on target "TestImage".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

Applications                : {}
AppstreamAgentVersion       : LATEST
Arn                         : arn:aws:appstream:us-west-2:123456789012:image/
TestImage
BaseImageArn                :
CreatedTime                 : 12/27/2019 1:34:10 PM
Description                 :
DisplayName                 : TestImage
ImageBuilderName            :
ImageBuilderSupported       : True
ImagePermissions            :
Name                        : TestImage
Platform                    : WINDOWS
PublicBaseImageReleasedDate : 6/12/2018 12:00:00 AM
State                       : AVAILABLE
StateChangeReason           :
Visibility                  : PRIVATE
```

- API-Details finden Sie DeleteImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImageBuilder`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel löscht eine ImageBuilder

```
Remove-APSImageBuilder -Name TestIB -Region us-west-2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-APSImageBuilder (DeleteImageBuilder)" on target
 "TestIB".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A

AccessEndpoints             : {}
AppstreamAgentVersion       : 12-16-2019
Arn                         : arn:aws:appstream:us-west-2:123456789012:image-
builder/TestIB
CreatedTime                 : 12/27/2019 11:39:24 AM
Description                 :
DisplayName                 : TestIB
DomainJoinInfo              :
EnableDefaultInternetAccess : True
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-west-2::image/AppStream-
WinServer2012R2-12-12-2019
ImageBuilderErrors          : {}
InstanceType                : stream.standard.medium
Name                        : TestIB
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : DELETING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie DeleteImageBuilder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSImagePermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Berechtigungen eines Images entfernt

```
Remove-APSImagePermission -Name Powershell -SharedAccountId 123456789012
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?
Performing the operation "Remove-APSImagePermission (DeleteImagePermissions)" on
 target "Powershell".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteImagePermissions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel entfernt ein Ressourcen-Tag aus der AppStream Ressource

```
Remove-APSResourceTag -ResourceArn arn:aws:appstream:us-east-1:123456789012:stack/
SessionScriptTest -TagKey StackState
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSResourceTag (UntagResource)" on target
 "arn:aws:appstream:us-east-1:123456789012:stack/SessionScriptTest".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie UntagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSStack`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Stack gelöscht

```
Remove-APSStack -Name TestStack -Region us-west-2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSStack (DeleteStack)" on target "TestStack".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSUsageReportSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSUsageReportSubscription`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel deaktiviert das Abonnement für AppStream Nutzungsberichte

```
Remove-APSUsageReportSubscription
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUsageReportSubscription
 (DeleteUsageReportSubscription)" on target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteUsageReportSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-APSUser`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-APSUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Benutzer aus USERPOOL gelöscht

```
Remove-APSUser -UserName TestUser@lab.com -AuthenticationType USERPOOL
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-APSUser (DeleteUser)" on target "TestUser@lab.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): A
```

- API-Details finden Sie DeleteUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Revoke-APSSession`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-APSSession`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel widerruft eine Sitzung mit fleet AppStream

```
Revoke-APSSession -SessionId 6cd2f9a3-f948-4aa1-8014-8a7dcde14877
```

- API-Details finden Sie ExpireSession in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Start-APSFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Flotte gestartet

```
Start-APSFleet -Name PowershellFleet
```

- API-Details finden Sie StartFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Start-APSImageBuilder`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel startet eine ImageBuilder

```
Start-APSImageBuilder -Name TestImage
```

Ausgabe:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : PENDING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie StartImageBuilder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-APSFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Flotte gestoppt

```
Stop-APSFleet -Name PowershellFleet
```

- API-Details finden Sie StopFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-APSImageBuilder`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-APSImageBuilder`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel stoppt einen ImageBuilder

```
Stop-APSImageBuilder -Name TestImage
```

Ausgabe:

```
AccessEndpoints             : {}
AppstreamAgentVersion       : 06-19-2019
Arn                         : arn:aws:appstream:us-east-1:123456789012:image-
builder/TestImage
CreatedTime                 : 1/14/2019 4:33:05 AM
Description                 :
DisplayName                 : TestImage
DomainJoinInfo              :
EnableDefaultInternetAccess : False
IamRoleArn                  :
ImageArn                    : arn:aws:appstream:us-east-1::image/Base-Image-
Builder-05-02-2018
ImageBuilderErrors          : {}
InstanceType                : stream.standard.large
Name                        : TestImage
NetworkAccessConfiguration  : Amazon.AppStream.Model.NetworkAccessConfiguration
Platform                    : WINDOWS
State                       : STOPPING
StateChangeReason           :
VpcConfig                   : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie StopImageBuilder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-APSFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung einer Flotte vom Stack aufgehoben

```
Unregister-APSFleet -StackName TestStack -FleetName TestFleet -Region us-west-2
```

- API-Details finden Sie DisassociateFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-APSUserStackBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-APSUserStackBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Benutzer aus einem zugewiesenen Stack entfernt

```
Unregister-APSUserStackBatch -UserStackAssociation
 @{AuthenticationType="USERPOOL";SendEmailNotification=
$False;StackName="PowershellStack";UserName="TestUser1@lab.com"}
```

- API-Details finden Sie BatchDisassociateUserStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-APSDirectoryConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSDirectoryConfig`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktualisiert die Verzeichniskonfiguration, die in erstellt wurde AppStream

```
Update-APSDirectoryConfig -ServiceAccountCredentials_AccountName contoso
\ServiceAccount -ServiceAccountCredentials_AccountPassword MyPass@1$@#
 -DirectoryName contoso.com -OrganizationalUnitDistinguishedName
 "OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com"
```

Ausgabe:

```
CreatedTime           DirectoryName OrganizationalUnitDistinguishedNames
 ServiceAccountCredentials
-----------           ------------- ------------------------------------
 -------------------------
12/27/2019 3:50:02 PM contoso.com   {OU=AppStreamNew,OU=Contoso,DC=Contoso,DC=com}
 Amazon.AppStream.Model.ServiceAccountCredentials
```

- API-Details finden Sie UpdateDirectoryConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-APSFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Eigenschaften einer Flotte aktualisiert

```
Update-APSFleet -Name PowershellFleet -EnableDefaultInternetAccess $True -
DisconnectTimeoutInSecond 950
```

Ausgabe:

```
Arn                            : arn:aws:appstream:us-east-1:123456789012:fleet/
PowershellFleet
ComputeCapacityStatus          : Amazon.AppStream.Model.ComputeCapacityStatus
CreatedTime                    : 4/24/2019 8:39:41 AM
Description                    : PowershellFleet
DisconnectTimeoutInSeconds     : 950
DisplayName                    : PowershellFleet
DomainJoinInfo                 :
EnableDefaultInternetAccess    : True
FleetErrors                    : {}
FleetType                      : ON_DEMAND
IamRoleArn                     :
IdleDisconnectTimeoutInSeconds : 900
ImageArn                       : arn:aws:appstream:us-east-1:123456789012:image/
Powershell
ImageName                      : Powershell
InstanceType                   : stream.standard.medium
MaxUserDurationInSeconds       : 57600
```

```
Name                            : PowershellFleet
State                           : STOPPED
VpcConfig                       : Amazon.AppStream.Model.VpcConfig
```

- API-Details finden Sie UpdateFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-APSImagePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSImagePermission`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel teilt ein AppStream Bild mit einem anderen Konto

```
Update-APSImagePermission -Name Powershell -SharedAccountId 123456789012 -
ImagePermissions_AllowFleet $True -ImagePermissions_AllowImageBuilder $True
```

- API-Details finden Sie UpdateImagePermissions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-APSStack`

Das folgende Codebeispiel zeigt die Verwendung. `Update-APSStack`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Persistenz der Anwendungseinstellungen und die Basisordner auf einem Stack aktualisiert (aktiviert)

```
Update-APSStack -Name PowershellStack -ApplicationSettings_Enabled $True
 -ApplicationSettings_SettingsGroup PowershellStack -StorageConnector
 @{ConnectorType="HOMEFOLDERS"}
```

Ausgabe:

```
AccessEndpoints     : {}
ApplicationSettings : Amazon.AppStream.Model.ApplicationSettingsResponse
Arn                 : arn:aws:appstream:us-east-1:123456789012:stack/PowershellStack
CreatedTime         : 4/24/2019 8:49:29 AM
Description         : PowershellStack
DisplayName         : PowershellStack
```

```
EmbedHostDomains    : {}
FeedbackURL         :
Name                : PowershellStack
RedirectURL         :
StackErrors         : {}
StorageConnectors   : {Amazon.AppStream.Model.StorageConnector,
 Amazon.AppStream.Model.StorageConnector}
UserSettings        : {Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting, Amazon.AppStream.Model.UserSetting,
 Amazon.AppStream.Model.UserSetting}
```

- API-Details finden Sie UpdateStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Aurora-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS -Tools für PowerShell V5 mit Aurora Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-RDSOrderableDBInstanceOption`

Das folgende Codebeispiel zeigt die Verwendung`Get-RDSOrderableDBInstanceOption`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet die DB-Engine-Versionen auf, die eine bestimmte DB-Instance-Klasse in einer AWS-Region unterstützen.

```
$params = @{
```

```
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Beispiel 2: Dieses Beispiel listet die DB-Instance-Klassen auf, die von einer bestimmten DB-Engine-Version in einer AWS-Region unterstützt werden.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Auto Scaling mithilfe von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Auto Scaling Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-ASLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung`Add-ASLoadBalancer`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer an die spezifizierte Auto-Scaling-Gruppe hinzugefügt.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API-Details finden Sie AttachLoadBalancers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Complete-ASLifecycleAction`

Das folgende Codebeispiel zeigt die Verwendung. `Complete-ASLifecycleAction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Lebenszyklusaktion abgeschlossen.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- API-Details finden Sie CompleteLifecycleAction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-ASMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-ASMetricsCollection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die spezifizierten Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric @("GroupMinSize",
 "GroupMaxSize")
```

Beispiel 2: In diesem Beispiel wird die Überwachung aller Metriken für die angegebene Auto-Scaling-Gruppe deaktiviert.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- API-Details finden Sie DisableMetricsCollection in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-ASInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-ASInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached in
 response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance von der angegebenen Auto-Scaling-Gruppe getrennt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached in
 response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- API-Details finden Sie DetachInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-ASLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-ASLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer von der spezifizierten Auto-Scaling-Gruppe getrennt.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API-Details finden Sie DetachLoadBalancers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-ASMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-ASMetricsCollection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Überwachung der angegebenen Metriken für die angegebene Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

Beispiel 2: In diesem Beispiel wird eine Überwachung aller Metriken der angegebenen Auto-Scaling-Gruppe aktiviert.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- API-Details finden Sie EnableMetricsCollection in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enter-ASStandby`

Das folgende Codebeispiel zeigt die Verwendung. `Enter-ASStandby`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt und die gewünschte Kapazität verringert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance in den Standby-Modus versetzt, ohne die gewünschte Kapazität zu verringern. Auto Scaling startet eine neue Ersatz-Instance.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- API-Details finden Sie EnterStandby in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Exit-ASStandby`

Das folgende Codebeispiel zeigt die Verwendung. `Exit-ASStandby`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance aus dem Standby-Modus verschoben.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out of
 standby in response to a user
```

```
                        request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
StatusMessage        :
```

- API-Details finden Sie ExitStandby in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAccountLimit`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die Auto Scaling-Ressourcenlimits für Ihr AWS Konto.

```
Get-ASAccountLimit
```

Ausgabe:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- API-Details finden Sie DescribeAccountLimits in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAdjustmentType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAdjustmentType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Anpassungstypen beschrieben, die von Auto Scaling unterstützt werden.

```
Get-ASAdjustmentType
```

Ausgabe:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- API-Details finden Sie DescribeAdjustmentTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Auto-Scaling-Gruppen aufgeführt.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

Ausgabe:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

Ausgabe:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
```

```
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASAutoScalingGroup
```

Beispiel 6: Dieses LaunchTemplate Beispiel beschreibt die angegebene Auto Scaling-Gruppe. In diesem Beispiel wird davon ausgegangen, dass die Option „Instance-Kaufoptionen“ auf „Startvorlage beachten“ eingestellt ist. Falls diese Option auf „Kaufoptionen und Instanztypen kombinieren“ gesetzt ist, LaunchTemplate kann über die Eigenschaft "MixedInstancesPolicy.LaunchTemplate" zugegriffen werden.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

Ausgabe:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAutoScalingInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die IDs Ihrer Auto-Scaling-Instances aufgeführt.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Instance beschrieben.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

Ausgabe:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

Beispiel 3: In diesem Beispiel werden die angegebenen zwei Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

Beispiel 4: Dieses Beispiel beschreibt die Auto Scaling-Instances für die angegebene Auto-Scaling-Gruppe.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

Beispiel 5: In diesem Beispiel werden alle Ihre Auto-Scaling-Instances beschrieben.

```
Get-ASAutoScalingInstance
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAutoScalingNotificationType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAutoScalingNotificationType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Benachrichtigungstypen aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASAutoScalingNotificationType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- API-Details finden Sie DescribeAutoScalingNotificationTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLaunchConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Startkonfigurationen aufgeführt.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

Ausgabe:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

Beispiel 2: In diesem Beispiel wird die angegebene Startkonfiguration beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

Ausgabe:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
```

```
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

Beispiel 3: In diesem Beispiel werden die beiden angegebenen Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

Beispiel 4: In diesem Beispiel werden alle Ihre Startkonfigurationen beschrieben.

```
Get-ASLaunchConfiguration
```

- API-Details finden Sie DescribeLaunchConfigurations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLifecycleHook`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName myLifecycleHook
```

Ausgabe:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

Beispiel 2: In diesem Beispiel werden alle Lebenszyklus-Hooks für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

Beispiel 3: In diesem Beispiel werden alle Lebenszyklus-Hooks für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASLifecycleHook
```

- API-Details finden Sie DescribeLifecycleHooks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASLifecycleHookType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLifecycleHookType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die von Auto Scaling unterstützten Typen von Lebenszyklus-Hooks aufgeführt.

```
Get-ASLifecycleHookType
```

Ausgabe:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- API-Details finden Sie DescribeLifecycleHookTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Load Balancers für die angegebene Auto-Scaling-Gruppebeschrieben.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

Ausgabe:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- API-Details finden Sie DescribeLoadBalancers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASMetricCollectionType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASMetricCollectionType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Arten der Metrikerfassung aufgeführt, die von Auto Scaling unterstützt werden.

```
(Get-ASMetricCollectionType).Metrics
```

Ausgabe:

```
Metric
------
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

Beispiel 2: In diesem Beispiel werden die entsprechenden Granularitäten aufgeführt.

```
(Get-ASMetricCollectionType).Granularities
```

Ausgabe:

```
Granularity
-----------
```

```
1Minute
```

- API-Details finden Sie DescribeMetricCollectionTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASNotificationConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die Benachrichtigungsaktionen, die der angegebenen Auto Scaling-Gruppe zugewiesen sind.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

Ausgabe:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

Beispiel 2: In diesem Beispiel werden die Benachrichtigungsaktionen beschrieben, die allen Ihren Auto-Scaling-Gruppen zugewiesen sind.

```
Get-ASNotificationConfiguration
```

- API-Details finden Sie DescribeNotificationConfigurations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

Ausgabe:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

Beispiel 2: In diesem Beispiel werden die angegebenen Richtlinien für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

Beispiel 3: In diesem Beispiel werden alle Richtlinien für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASPolicy
```

- API-Details finden Sie DescribePolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASScalingActivity`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScalingActivity`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

Ausgabe:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set group
 desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the capacity
 from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity from
 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
```

```
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Skalierungsaktivität beschrieben.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

Beispiel 3: In diesem Beispiel werden die Skalierungsaktivitäten der letzten sechs Wochen für all Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScalingActivity
```

- API-Details finden Sie DescribeScalingActivities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASScalingProcessType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScalingProcessType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Prozesstypen aufgelistet, die von Auto Scaling unterstützt werden.

```
Get-ASScalingProcessType
```

Ausgabe:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- API-Details finden Sie DescribeScalingProcessTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASScheduledAction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die geplanten Skalierungsaktionen für die angegebene Auto-Scaling-Gruppe beschrieben.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

Ausgabe:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-asg:scheduledActionName/
myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

Beispiel 2: In diesem Beispiel werden die angegebenen geplanten Skalierungsaktionen beschrieben.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

Beispiel 3: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit beginnen.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

Beispiel 4: In diesem Beispiel werden die geplanten Skalierungsaktionen beschrieben, die zur angegebenen Zeit enden.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

Beispiel 5: In diesem Beispiel werden die geplanten Skalierungsaktionen für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASScheduledAction
```

- API-Details finden Sie DescribeScheduledActions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tags mit dem Schlüsselwert „myTag“ oder „myTag2“ beschrieben. Die möglichen Werte für den Filternamen sind „auto-scaling-group“, „key“, „value“ und „propagate-at-launch“. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

Ausgabe:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Parameter Filter zu erstellen.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

Beispiel 3: In diesem Beispiel werden alle Tags für alle Ihre Auto-Scaling-Gruppen beschrieben.

```
Get-ASTag
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASTerminationPolicyType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASTerminationPolicyType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Beendigungsrichtlinien aufgeführt, die von Auto Scaling unterstützt werden.

```
Get-ASTerminationPolicyType
```

Ausgabe:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API-Details finden Sie DescribeTerminationPolicyTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Mount-ASInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Mount-ASInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance an die spezifizierte Auto-Scaling-Gruppe hinzugefügt. Auto Scaling erhöht automatisch die gewünschte Kapazität der Auto-Scaling-Gruppe.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- API-Details finden Sie AttachInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASAutoScalingGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Auto-Scaling-Gruppe mit dem angegebenen Namen und den angegebenen Attributen erstellt. Die standardmäßige gewünschte Kapazität entspricht der minimalen Größe. Daher startet diese Auto-Scaling-Gruppe zwei Instances – jeweils eine in jeder der angegebenen beiden Availability Zones.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-lc -
MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASLaunchConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Startkonfiguration mit dem Namen „my-lc“ erstellt. Die von Auto-Scaling-Gruppen mit dieser Startkonfiguration gestarteten EC2-Instances verwenden den angegebenen Instance-Typ, das angegebene AMI, die Sicherheitsgruppe und die IAM-Rolle.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType "m3.medium" -
ImageId "ami-12345678" -SecurityGroup "sg-12345678" -IamInstanceProfile "myIamRole"
```

- API-Details finden Sie CreateLaunchConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASAutoScalingGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht, sofern keine Instances ausgeführt werden. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on Target
 "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

Beispiel 3: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe gelöscht und alle ausgeführten Instances, die sie enthält, werden beendet.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASLaunchConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASLaunchConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Startkonfiguration gelöscht, sofern sie nicht einer Auto-Scaling-Gruppe zugewiesen ist. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)" on
 Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- API-Details finden Sie DeleteLaunchConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASLifecycleHook`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- API-Details finden Sie DeleteLifecycleHook in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASNotificationConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Benachrichtigungsaktion gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- API-Details finden Sie DeleteNotificationConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- API-Details finden Sie DeletePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASScheduledAction`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASScheduledAction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene geplante Aktion für die spezifizierte Auto-Scaling-Gruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- API-Details finden Sie DeleteScheduledAction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ASTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Tag der spezifizierten Auto-Scaling-Gruppe entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Wenn Sie den Force-Parameter angeben, werden Sie vor der Ausführung des Vorgangs nicht zur Bestätigung aufgefordert.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

Beispiel 3: Mit Powershell-Version 2 New-Object müssen Sie das Tag für den Tag-Parameter erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- API-Details finden Sie DeleteTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Resume-ASProcess`

Das folgende Codebeispiel zeigt die Verwendung. `Resume-ASProcess`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle ausgesetzten Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe fortgesetzt.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- API-Details finden Sie ResumeProcesses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ASDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASDesiredCapacity`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

Beispiel 2: In diesem Beispiel wird die Größe der angegebenen Auto-Scaling-Gruppe angegeben und gewartet, bis die Ruhephase abgeschlossen ist, bevor auf die neue Größe skaliert wird.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -HonorCooldown
 $true
```

- API-Details finden Sie SetDesiredCapacity in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ASInstanceHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASInstanceHealth`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerhaft“ gesetzt, wodurch sie außer Betrieb genommen wird. Auto Scaling beendet und ersetzt die Instance.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance auf „Fehlerfrei“ gesetzt, sodass sie weiterhin in Betrieb bleibt. Legen Sie die Wartefrist für die Zustandsprüfung einer Auto-Scaling-Gruppe fest.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- API-Details finden Sie SetInstanceHealth in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ASInstanceProtection`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASInstanceProtection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Instance-Schutz für die angegebene Instance aktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

Beispiel 2: In diesem Beispiel wird der Instance-Schutz für die angegebene Instanz deaktiviert.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- API-Details finden Sie SetInstanceProtection in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ASTag`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ASTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebenen Auto-Scaling-Gruppe ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist TagValue 'my'. Auto Scaling leitet dieses Tag an die nachfolgenden EC2-Instances weiter, die von der Auto-Scaling-Gruppe gestartet wurden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- API-Details finden Sie CreateOrUpdateTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-ASPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Start-ASPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel führt die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe aus.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

Beispiel 2: In diesem Beispiel wird die angegebene Richtlinie für die spezifizierte Auto-Scaling-Gruppe ausgeführt, nachdem die Wartezeit abgelaufen ist.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- API-Details finden Sie ExecutePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-ASInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-ASInstanceInAutoScalingGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet und die gewünschte Kapazität ihrer Auto-Scaling-Gruppe reduziert, sodass Auto Scaling keine Ersatz-Instance startet.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

Beispiel 2: In diesem Beispiel wird die angegebene Instance beendet, ohne die gewünschte Kapazität ihrer Auto Scaling-Gruppe zu reduzieren. Auto Scaling startet eine neue Ersatz-Instance.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

Ausgabe:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out of
 service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Suspend-ASProcess`

Das folgende Codebeispiel zeigt die Verwendung. `Suspend-ASProcess`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Auto-Scaling-Prozess für die spezifizierte Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

Beispiel 2: In diesem Beispiel werden alle Auto-Scaling-Prozesse für die angegebene Auto-Scaling-Gruppe ausgesetzt.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- API-Details finden Sie SuspendProcesses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-ASAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ASAutoScalingGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die minimale und maximale Größe der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

Beispiel 2: In diesem Beispiel wird die standardmäßige Ruhephase der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

Beispiel 3: In diesem Beispiel werden die Availability Zones der angegebenen Auto-Scaling-Gruppe aktualisiert.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

Beispiel 4: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe aktualisiert, um Zustandsprüfungen mit Elastic Load Balancing zu verwenden.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-ASLifecycleActionHeartbeat`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASLifecycleActionHeartbeat`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Heartbeat für die angegebene Lebenszyklusaktion aufgezeichnet. Dadurch bleibt die Instance im Status „Ausstehend“, bis die benutzerdefinierte Aktion abgeschlossen ist.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- API-Details finden Sie RecordLifecycleActionHeartbeat in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-ASLifecycleHook`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASLifecycleHook`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Lebenszyklus-Hook zur spezifizierten Auto-Scaling-Gruppe hinzugefügt.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- API-Details finden Sie PutLifecycleHook in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-ASNotificationConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASNotificationConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten von EC2-Instances eine Benachrichtigung an das spezifizierte SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-west-2:123456789012:my-
topic"
```

Beispiel 2: In diesem Beispiel wird die angegebene Auto-Scaling-Gruppe so konfiguriert, dass sie beim Starten oder Beenden von EC2-Instances eine Benachrichtigung an das angegebene SNS-Thema sendet.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- API-Details finden Sie PutNotificationConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-ASScalingPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASScalingPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie der spezifizierten Auto-Scaling-Gruppe hinzugefügt. Der angegebene Anpassungstyp bestimmt, wie der ScalingAdjustment Parameter zu interpretieren ist. Bei 'ChangeInCapacity' erhöht ein positiver Wert die Kapazität um die angegebene Anzahl von Instanzen und ein negativer Wert verringert die Kapazität um die angegebene Anzahl von Instanzen.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

Ausgabe:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
:policyName/myScaleInPolicy
```

- API-Details finden Sie PutScalingPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-ASScheduledUpdateGroupAction`

Das folgende Codebeispiel zeigt die Verwendung. `Write-ASScheduledUpdateGroupAction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine einmalig geplante Aktion erstellt oder aktualisiert, um die gewünschte Kapazität zur angegebenen Startzeit zu ändern.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -ScheduledActionName
 "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -DesiredCapacity 10
```

- API-Details finden Sie PutScheduledUpdateGroupAction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS Budgets Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS Budgets

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`New-BGTBudget`

Das folgende Codebeispiel zeigt die Verwendung`New-BGTBudget`.

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues Budget mit den angegebenen Budget- und Zeitvorgaben sowie E-Mail-Benachrichtigungen.

```
$notification = @{
    NotificationType = "ACTUAL"
    ComparisonOperator = "GREATER_THAN"
    Threshold = 80
}

$addressObject = @{
    Address = @("user@domain.com")
    SubscriptionType = "EMAIL"
}
```

```
$subscriber = New-Object Amazon.Budgets.Model.NotificationWithSubscribers
$subscriber.Notification = $notification
$subscriber.Subscribers.Add($addressObject)

$startDate = [datetime]::new(2017,09,25)
$endDate = [datetime]::new(2017,10,25)

New-BGTBudget -Budget_BudgetName "Tester" -Budget_BudgetType COST -
CostTypes_IncludeTax $true -Budget_TimeUnit MONTHLY -BudgetLimit_Unit USD -
TimePeriod_Start $startDate -TimePeriod_End $endDate -AccountId 123456789012 -
BudgetLimit_Amount 200 -NotificationsWithSubscriber $subscriber
```

- API-Details finden Sie CreateBudget in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS Cloud9 Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS Cloud9

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-C9EnvironmentData`

Das folgende Codebeispiel zeigt die Verwendung`Get-C9EnvironmentData`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentData -EnvironmentId
 685f892f431b45c2b28cb69eadcdb0EX,1980b80e5f584920801c09086667f0EX
```

Ausgabe:

```
Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:685f892f431b45c2b28cb69eadcdb0EX
Description : Created from CodeStar.
Id          : 685f892f431b45c2b28cb69eadcdb0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ec2-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ec2

Arn         : arn:aws:cloud9:us-
east-1:123456789012:environment:1980b80e5f584920801c09086667f0EX
Description :
Id          : 1980b80e5f584920801c09086667f0EX
Lifecycle   : Amazon.Cloud9.Model.EnvironmentLifecycle
Name        : my-demo-ssh-env
OwnerArn    : arn:aws:iam::123456789012:user/MyDemoUser
Type        : ssh
```

Beispiel 2: In diesem Beispiel werden Informationen über den Lebenszyklusstatus der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
(Get-C9EnvironmentData -EnvironmentId 685f892f431b45c2b28cb69eadcdb0EX).Lifecycle
```

Ausgabe:

```
FailureResource Reason Status
--------------- ------ ------
                       CREATED
```

- API-Details finden Sie DescribeEnvironments in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-C9EnvironmentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Identifikatoren der AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentList
```

Ausgabe:

```
685f892f431b45c2b28cb69eadcdb0EX
1980b80e5f584920801c09086667f0EX
```

- API-Details finden Sie ListEnvironments in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-C9EnvironmentMembershipList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentMembershipList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über Umgebungsmitglieder für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX

EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 2: In diesem Beispiel werden Informationen über den Besitzer der angegebenen AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentMembershipList -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -
Permission owner
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

Beispiel 3: In diesem Beispiel werden Informationen über das angegebene Umgebungsmitglied für mehrere AWS Cloud9-Entwicklungsumgebungen abgerufen.

```
Get-C9EnvironmentMembershipList -UserArn arn:aws:iam::123456789012:user/MyDemoUser
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/17/2018 7:48:14 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX

EnvironmentId : 1980b80e5f584920801c09086667f0EX
LastAccess    : 1/16/2018 11:21:24 PM
Permissions   : owner
UserArn       : arn:aws:iam::123456789012:user/MyDemoUser
UserId        : AIDAJ3LOROMOUXTBSU6EX
```

- API-Details finden Sie DescribeEnvironmentMemberships in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-C9EnvironmentStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-C9EnvironmentStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Statusinformationen für die angegebene AWS Cloud9-Entwicklungsumgebung abgerufen.

```
Get-C9EnvironmentStatus -EnvironmentId 349c86d4579e4e7298d500ff57a6b2EX
```

Ausgabe:

```
Message                     Status
-------                     ------
Environment is ready to use ready
```

- API-Details finden Sie DescribeEnvironmentStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-C9EnvironmentEC2`

Das folgende Codebeispiel zeigt die Verwendung. `New-C9EnvironmentEC2`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine AWS Cloud9-Entwicklungsumgebung mit den angegebenen Einstellungen erstellt, eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance gestartet und dann eine Verbindung von der Instance zur Umgebung hergestellt.

```
New-C9EnvironmentEC2 -Name my-demo-env -AutomaticStopTimeMinutes 60 -Description
 "My demonstration development environment." -InstanceType t2.micro -OwnerArn
 arn:aws:iam::123456789012:user/MyDemoUser -SubnetId subnet-d43a46EX
```

Ausgabe:

```
ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API-Details finden Sie CreateEnvironmentEc2 in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `New-C9EnvironmentMembership`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Umgebungsmitglied zur angegebenen AWS Cloud9-Entwicklungsumgebung hinzugefügt.

```
New-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/AnotherDemoUser
 -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-write
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
Permissions   : read-write
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API-Details finden Sie CreateEnvironmentMembership in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-C9Environment`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-C9Environment`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel löscht die angegebene AWS Cloud9-Entwicklungsumgebung. Wenn die Umgebung mit einer Amazon-EC2-Instance verbunden ist, wird auch die Instance beendet.

```
Remove-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API-Details finden Sie DeleteEnvironment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-C9EnvironmentMembership`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel löscht das angegebene Umgebungsmitglied aus der angegebenen AWS Cloud9-Entwicklungsumgebung.

```
Remove-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX
```

- API-Details finden Sie DeleteEnvironmentMembership in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-C9Environment`

Das folgende Codebeispiel zeigt die Verwendung. `Update-C9Environment`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ändert die angegebenen Einstellungen der angegebenen vorhandenen AWS Cloud9-Entwicklungsumgebung.

```
Update-C9Environment -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Description
 "My changed demonstration development environment." -Name my-changed-demo-env
```

- API-Details finden Sie UpdateEnvironment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-C9EnvironmentMembership`

Das folgende Codebeispiel zeigt die Verwendung. `Update-C9EnvironmentMembership`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Einstellungen des angegebenen vorhandenen Umgebungsmitglieds für die angegebene AWS Cloud9-Entwicklungsumgebung geändert.

```
Update-C9EnvironmentMembership -UserArn arn:aws:iam::123456789012:user/
AnotherDemoUser -EnvironmentId ffd88420d4824eeeaeaa8a04bfde8cEX -Permission read-
only
```

Ausgabe:

```
EnvironmentId : ffd88420d4824eeeaeaa8a04bfde8cEX
LastAccess    : 1/1/0001 12:00:00 AM
```

```
Permissions   : read-only
UserArn       : arn:aws:iam::123456789012:user/AnotherDemoUser
UserId        : AIDAJ3BA6O2FMJWCWXHEX
```

- API-Details finden Sie UpdateEnvironmentMembership in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CloudFormation Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CloudFormation

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung`Get-CFNStack`.

Tools für PowerShell V5

Beispiel 1: Gibt eine Sammlung von Stack-Instances zurück, die alle Stacks des Benutzers beschreiben.

```
Get-CFNStack
```

Beispiel 2: Gibt eine Stack-Instance zurück, die den angegebenen Stack beschreibt

```
Get-CFNStack -StackName "myStack"
```

- API-Details finden Sie DescribeStacks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNStackEvent`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackEvent`

Tools für PowerShell V5

Beispiel 1: Gibt alle stapelbezogenen Ereignisse für den angegebenen Stapel zurück.

```
Get-CFNStackEvent -StackName "myStack"
```

- API-Details finden Sie DescribeStackEvents in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNStackResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResource`

Tools für PowerShell V5

Beispiel 1: Gibt die Beschreibung einer Ressource zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "MyDBInstance" identifiziert wurde.

```
Get-CFNStackResource -StackName "myStack" -LogicalResourceId "MyDBInstance"
```

- API-Details finden Sie DescribeStackResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNStackResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResourceList`

Tools für PowerShell V5

Beispiel 1: Gibt die AWS Ressourcenbeschreibungen für bis zu 100 Ressourcen zurück, die dem angegebenen Stack zugeordnet sind. Um Details zu allen Ressourcen zu erhalten, die einem Stack zugeordnet sind, verwenden Sie den Get-CFNStackResourceSummary, der auch das manuelle Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -StackName "myStack"
```

Beispiel 2: Gibt die Beschreibung der Amazon EC2-Instance zurück, die in der Vorlage, die mit dem angegebenen Stack verbunden ist, durch die logische ID "Ec2Instance" identifiziert wurde.

```
Get-CFNStackResourceList -StackName "myStack" -LogicalResourceId "Ec2Instance"
```

Beispiel 3: Gibt die Beschreibung von bis zu 100 Ressourcen zurück, die mit dem Stack verbunden sind, der eine Amazon EC2-Instance enthält, die durch die Instance-ID "i-123456" identifiziert wird. Um Details zu allen Ressourcen zu erhalten, die einem Stack zugeordnet sind, verwenden Sie den Get-CFNStackResourceSummary, der auch das manuelle Paging der Ergebnisse unterstützt.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456"
```

Beispiel 4: Gibt die Beschreibung der Amazon EC2-Instance zurück, die durch die logische ID "Ec2Instance" in der Vorlage für einen Stack identifiziert wird. Der Stack wird anhand der physischen Ressourcen-ID einer darin enthaltenen Ressource identifiziert, in diesem Fall also einer Amazon EC2-Instance mit der Instance-ID "i-123456". Je nach Inhalt der Vorlage könnte auch eine andere physische Ressource zur Identifizierung des Stacks verwendet werden, zum Beispiel ein Amazon S3-Bucket.

```
Get-CFNStackResourceList -PhysicalResourceId "i-123456" -LogicalResourceId
 "Ec2Instance"
```

- API-Details finden Sie DescribeStackResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNStackResourceSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackResourceSummary`

Tools für PowerShell V5

Beispiel 1: Gibt Beschreibungen aller Ressourcen zurück, die mit dem angegebenen Stapel verbunden sind.

```
Get-CFNStackResourceSummary -StackName "myStack"
```

- API-Details finden Sie ListStackResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNStackSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNStackSummary`

Tools für PowerShell V5

Beispiel 1: Gibt zusammenfassende Informationen für alle Stapel zurück.

```
Get-CFNStackSummary
```

Beispiel 2: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt werden.

```
Get-CFNStackSummary -StackStatusFilter "CREATE_IN_PROGRESS"
```

Beispiel 3: Gibt zusammenfassende Informationen für alle Stapel zurück, die gerade erstellt oder aktualisiert werden.

```
Get-CFNStackSummary -StackStatusFilter @("CREATE_IN_PROGRESS", "UPDATE_IN_PROGRESS")
```

- API-Details finden Sie ListStacks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFNTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFNTemplate`

Tools für PowerShell V5

Beispiel 1: Gibt die Vorlage zurück, die mit dem angegebenen Stapel verbunden ist.

```
Get-CFNTemplate -StackName "myStack"
```

- API-Details finden Sie GetTemplate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Measure-CFNTemplateCost`

Das folgende Codebeispiel zeigt die Verwendung. `Measure-CFNTemplateCost`

Tools für PowerShell V5

Beispiel 1: Gibt eine URL für den AWS einfachen Monatsrechner mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die für die Ausführung der Vorlage erforderlich sind. Die Vorlage wird von der angegebenen Amazon-S3-URL abgerufen und der einzelne Anpassungsparameter wird angewendet. Der Parameter kann auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Measure-CFNTemplateCost -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                        -Region us-west-1 `
                        -Parameter @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" }
```

Beispiel 2: Gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die für die Ausführung der Vorlage erforderlich sind. Die Vorlage wird anhand des bereitgestellten Inhalts und der angewendeten Anpassungsparameter analysiert (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType'). Die Anpassungsparameter können auch mithilfe von 'Key' und 'Value' anstelle von '' und ParameterKey '' angegeben werden. ParameterValue

```
Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" `
                        -Parameter @( @{ ParameterKey="KeyName";
 ParameterValue="myKeyPairName" },`
                                      @{ ParameterKey="InstanceType";
 ParameterValue="m1.large" })
```

Beispiel 3: Wird verwendet New-Object , um den Satz von Vorlagenparametern zu erstellen, und gibt eine AWS Simple Monthly Calculator-URL mit einer Abfragezeichenfolge zurück, die die Ressourcen beschreibt, die für die Ausführung der Vorlage erforderlich sind. Die Vorlage wird anhand des bereitgestellten Inhalts mit Anpassungsparametern analysiert (in diesem Beispiel wird davon ausgegangen, dass der Vorlageninhalt zwei Parameter deklariert hätte, 'KeyName' und 'InstanceType').

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "KeyName"
$p1.ParameterValue = "myKeyPairName"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
```

```
$p2.ParameterKey = "InstanceType"
$p2.ParameterValue = "m1.large"

Measure-CFNTemplateCost -TemplateBody "{TEMPLATE CONTENT HERE}" -Parameter @( $p1,
 $p2 )
```

- API-Details finden Sie EstimateTemplateCost in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFNStack`

Tools für PowerShell V5

Beispiel 1: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser Parameter. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stapels fehlschlägt, wird er nicht zurückgesetzt.

```
New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" }) `
             -DisableRollback $true
```

Beispiel 2: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird aus dem gelieferten Inhalt mit Anpassungsparametern geparst ('PK1' und 'PK2' stehen für die Namen der im Vorlageninhalt deklarierten Parameter, 'PV1' und 'PV2' für die Werte dieser Parameter. Die Anpassungsparameter können auch mithilfe von 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden. Wenn die Erstellung des Stapels fehlschlägt, wird er zurückgerollt.

```
$p1 = New-Object -Type Amazon.CloudFormation.Model.Parameter
$p1.ParameterKey = "PK1"
$p1.ParameterValue = "PV1"

$p2 = New-Object -Type Amazon.CloudFormation.Model.Parameter
```

```
$p2.ParameterKey = "PK2"
$p2.ParameterValue = "PV2"

New-CFNStack -StackName "myStack" `
             -TemplateBody "{TEMPLATE CONTENT HERE}" `
             -Parameter @( $p1, $p2 ) `
             -OnFailure "ROLLBACK"
```

Beispiel 3: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Vorlageninhalt deklariert ist, 'PV1' steht für den Wert für den Parameter). Die Anpassungsparameter können auch mithilfe von 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er rückgängig gemacht (genau wie bei der Angabe von - DisableRollback $false).

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Erzeugt einen neuen Stapel mit dem angegebenen Namen. Die Vorlage wird von der Amazon S3-URL mit Anpassungsparametern abgerufen ('PK1' steht für den Namen eines Parameters, der im Vorlageninhalt deklariert ist, 'PV1' steht für den Wert für den Parameter. Die Anpassungsparameter können auch mithilfe von 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue Wenn die Erstellung des Stacks fehlschlägt, wird er rückgängig gemacht (genau wie bei der Angabe von - DisableRollback $false). Die angegebenen Benachrichtigungs-AENs empfangen veröffentlichte stapelbezogene Ereignisse.

```
New-CFNStack -StackName "myStack" `
             -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
             -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" } `
             -NotificationARN @( "arn1", "arn2" )
```

- API-Details finden Sie CreateStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CFNStack`

Tools für PowerShell V5

Beispiel 1: Löscht den angegebenen Stapel.

```
Remove-CFNStack -StackName "myStack"
```

- API-Details finden Sie DeleteStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Resume-CFNUpdateRollback`

Das folgende Codebeispiel zeigt die Verwendung. `Resume-CFNUpdateRollback`

Tools für PowerShell V5

Beispiel 1: Setzt Rollback des benannten Stapels fort, der sich im Zustand 'UPDATE_ROLLBACK_FAILED' befinden sollte. Wenn das fortgesetzte Rollback erfolgreich ist, geht der Stack in den Zustand 'UPDATE_ROLLBACK_COMPLETE' über.

```
Resume-CFNUpdateRollback -StackName "myStack"
```

- API-Details finden Sie ContinueUpdateRollback in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-CFNUpdateStack`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CFNUpdateStack`

Tools für PowerShell V5

Beispiel 1: Bricht eine Aktualisierung auf dem angegebenen Stapel ab.

```
Stop-CFNUpdateStack -StackName "myStack"
```

- API-Details finden Sie CancelUpdateStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Test-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Test-CFNStack`

Tools für PowerShell V5

Beispiel 1: Testet, ob der Stack einen der Status UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE oder UPDATE_COMPLETE erreicht hat.

```
Test-CFNStack -StackName MyStack
```

Ausgabe:

```
False
```

Beispiel 2: Testet, ob der Stack den Status UPDATE_COMPLETE oder UPDATE_ROLLBACK_COMPLETE erreicht hat.

```
Test-CFNStack -StackName MyStack -Status UPDATE_COMPLETE,UPDATE_ROLLBACK_COMPLETE
```

Ausgabe:

```
True
```

- API-Details finden Sie Test-CFNStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Test-CFNTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `Test-CFNTemplate`

Tools für PowerShell V5

Beispiel 1: Validiert den angegebenen Vorlageninhalt. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateBody "{TEMPLATE CONTENT HERE}"
```

Beispiel 2: Validiert die angegebene Vorlage, auf die über eine Amazon S3 URL zugegriffen wird. Die Ausgabe enthält die Fähigkeiten, Beschreibung und Parameter der Vorlage.

```
Test-CFNTemplate -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template
```

- API-Details finden Sie ValidateTemplate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CFNStack`

Tools für PowerShell V5

Beispiel 1: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV1' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und 'ParameterKey' angegeben werden. ParameterValue

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 2: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateBody "{Template Content Here}" `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 3: Aktualisiert den Stack 'myStack' mit der angegebenen Vorlage und den Anpassungsparametern. 'PK1' steht für den Namen eines in der Vorlage deklarierten Parameters und 'PV2' für dessen Wert. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" -TemplateBody "{Template Content Here}" -
Parameters @{ ParameterKey="PK1"; ParameterValue="PV1" }
```

Beispiel 4: Aktualisiert den Stack 'myStack' mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die

Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden.

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } )
```

Beispiel 5: Aktualisiert den Stack 'myStack', von dem in diesem Beispiel angenommen wird, dass er IAM-Ressourcen enthält, mit der angegebenen, von Amazon S3 bezogenen Vorlage und den Anpassungsparametern. 'PK1' und 'PK2' stehen für die Namen der in der Vorlage deklarierten Parameter, 'PV1' und 'PV2' für die gewünschten Werte. Die Anpassungsparameter können auch mit 'Key' und 'Value' anstelle von '' und ParameterKey 'ParameterValue' angegeben werden. Bei Stacks, die IAM-Ressourcen enthalten, müssen Sie den Parameter -Capabilities „CAPABILITY_IAM“ angeben. Andernfalls schlägt das Update mit dem Fehler '' fehl. InsufficientCapabilities

```
Update-CFNStack -StackName "myStack" `
                -TemplateURL https://s3.amazonaws.com/amzn-s3-demo-bucket/
templatefile.template `
                -Parameter @( @{ ParameterKey="PK1"; ParameterValue="PV1" },
 @{ ParameterKey="PK2"; ParameterValue="PV2" } ) `
                -Capabilities "CAPABILITY_IAM"
```

- API-Details finden Sie UpdateStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Wait-CFNStack`

Das folgende Codebeispiel zeigt die Verwendung. `Wait-CFNStack`

Tools für PowerShell V5

Beispiel 1: Testet, ob der Stack einen der Status UPDATE_ROLLBACK_COMPLETE, CREATE_COMPLETE, ROLLBACK_COMPLETE oder UPDATE_COMPLETE erreicht hat. Wenn sich der Stack nicht in einem der Status befindet, ruht der Befehl zwei Sekunden lang, bevor er den Status erneut testet. Dies wird wiederholt, bis der Stack einen der angeforderten Zustände erreicht hat oder das Standard-Timeout von 60 Sekunden abgelaufen ist. Wird der Timeout-Zeitraum überschritten, wird eine Ausnahme ausgelöst. Wenn der Stack innerhalb des Timeout-Zeitraums einen der angeforderten Zustände erreicht, wird er an die Pipeline zurückgegeben.

```
$stack = Wait-CFNStack -StackName MyStack
```

Beispiel 2: In diesem Beispiel wird insgesamt 5 Minuten (300 Sekunden) gewartet, bis der Stack einen der angegebenen Zustände erreicht. In diesem Beispiel wird der Status vor dem Timeout erreicht und das Stack-Objekt wird daher an die Pipeline zurückgegeben.

```
Wait-CFNStack -StackName MyStack -Timeout 300 -Status
 CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Ausgabe:

```
Capabilities      : {CAPABILITY_IAM}
ChangeSetId       :
CreationTime      : 6/1/2017 9:29:33 AM
Description       : AWS CloudFormation Sample Template
 ec2_instance_with_instance_profile: Create an EC2 instance with an associated
 instance profile. **WARNING** This template creates one or more Amazon EC2
                    instances and an Amazon SQS queue. You will be billed for the
 AWS resources used if you create a stack from this template.
DisableRollback   : False
LastUpdatedTime   : 1/1/0001 12:00:00 AM
NotificationARNs  : {}
Outputs           : {}
Parameters        : {}
RoleARN           :
StackId           : arn:aws:cloudformation:us-west-2:123456789012:stack/
MyStack/7ea87b50-46e7-11e7-9c9b-503a90a9c4d1
StackName         : MyStack
StackStatus       : CREATE_COMPLETE
StackStatusReason :
Tags              : {}
TimeoutInMinutes  : 0
```

Beispiel 3: Dieses Beispiel zeigt die Fehlerausgabe, wenn ein Stack innerhalb des Timeout-Zeitraums (in diesem Fall der Standardzeitraum von 60 Sekunden) keinen der angeforderten Zustände erreicht.

```
Wait-CFNStack -StackName MyStack -Status CREATE_COMPLETE,ROLLBACK_COMPLETE
```

Ausgabe:

```
Wait-CFNStack : Timed out after 60 seconds waiting for CloudFormation
 stack MyStack in region us-west-2 to reach one of state(s):
 UPDATE_ROLLBACK_COMPLETE,CREATE_COMPLETE,ROLLBACK_COMPLETE,UPDATE_COMPLETE
At line:1 char:1
+ Wait-CFNStack -StackName MyStack -State CREATE_COMPLETE,ROLLBACK_COMPLETE
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation:
 (Amazon.PowerShe...tCFNStackCmdlet:WaitCFNStackCmdlet) [Wait-CFNStack],
 InvalidOperationException
    + FullyQualifiedErrorId :
 InvalidOperationException,Amazon.PowerShell.Cmdlets.CFN.WaitCFNStackCmdlet
```

- API-Details finden Sie Wait-CFNStack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CloudFront Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CloudFront

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-CFCloudFrontOriginAccessIdentity`

Das folgende Codebeispiel zeigt die Verwendung`Get-CFCloudFrontOriginAccessIdentity`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine bestimmte CloudFront Amazon-Zugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Obwohl der -Id-Parameter nicht erforderlich ist, werden keine Ergebnisse zurückgegeben, wenn Sie ihn nicht angeben.

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
    CloudFrontOriginAccessIdentityConfig    Id
S3CanonicalUserId
    ------------------------------------    --
-----------------
    Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
4b6e...
```

- API-Details finden Sie GetCloudFrontOriginAccessIdentity in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFCloudFrontOriginAccessIdentityConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFCloudFrontOriginAccessIdentityConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Konfigurationsinformationen zu einer einzelnen CloudFront Amazon-Zugriffsidentität zurückgegeben, die durch den Parameter -Id angegeben wird. Fehler treten auf, wenn kein -Id-Parameter angegeben ist.

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

Ausgabe:

```
    CallerReference                                          Comment
    ---------------                                          -------
    mycallerreference: 2/1/2011 1:16:32 PM                   Caller reference:
2/1/2011 1:16:32 PM
```

- API-Details finden Sie GetCloudFrontOriginAccessIdentityConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFCloudFrontOriginAccessIdentityList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFCloudFrontOriginAccessIdentityList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Liste der CloudFront Amazon-Zugriffsidentitäten zurück. Da der MaxItem Parameter - den Wert 2 angibt, enthalten die Ergebnisse zwei Identitäten.

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

Ausgabe:

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- API-Details finden Sie ListCloudFrontOriginAccessIdentities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFDistribution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistribution`

Tools für PowerShell V5

Beispiel 1: Ruft die Informationen für eine bestimmte Distribution ab.

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- API-Details finden Sie GetDistribution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFDistributionConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistributionConfig`

Tools für PowerShell V5

Beispiel 1: Ruft die Konfiguration für eine bestimmte Distribution ab.

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- API-Details finden Sie GetDistributionConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFDistributionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFDistributionList`

Tools für PowerShell V5

Beispiel 1: Gibt Distributionen zurück.

```
Get-CFDistributionList
```

- API-Details finden Sie ListDistributions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CFDistribution`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFDistribution`

Tools für PowerShell V5

Beispiel 1: Erstellt eine CloudFront Basisverteilung, die mit Protokollierung und Caching konfiguriert ist.

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "amzn-s3-demo-bucket.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
```

```
    -DistributionConfig_Comment "Test distribution" `
    -Origins_Item $origin `
    -Origins_Quantity 1 `
    -Logging_Enabled $true `
    -Logging_IncludeCookie $true `
    -Logging_Bucket amzn-s3-demo-logging-bucket.s3.amazonaws.com `
    -Logging_Prefix "help/" `
    -DistributionConfig_CallerReference Client1 `
    -DistributionConfig_DefaultRootObject index.html `
    -DefaultCacheBehavior_TargetOriginId $origin.Id `
    -ForwardedValues_QueryString $true `
    -Cookies_Forward all `
    -WhitelistedNames_Quantity 0 `
    -TrustedSigners_Enabled $false `
    -TrustedSigners_Quantity 0 `
    -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
    -DefaultCacheBehavior_MinTTL 1000 `
    -DistributionConfig_PriceClass "PriceClass_All" `
    -CacheBehaviors_Quantity 0 `
    -Aliases_Quantity 0
```

- API-Details finden Sie CreateDistribution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CFInvalidation`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFInvalidation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Invalidierung für eine Distribution mit der ID EXAMPLENSTXAXE erstellt. Das CallerReference ist eine vom Benutzer gewählte eindeutige ID; in diesem Fall wird ein Zeitstempel verwendet, der den 15. Mai 2019 um 9:00 Uhr darstellt. Die Variable $Paths speichert drei Pfade zu Bild- und Mediendateien, die der Benutzer nicht als Teil des Caches der Distribution haben möchte. Der Wert des Paths_Quantity Parameters - ist die Gesamtzahl der im Paths_Item Parameter - angegebenen Pfade.

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -Paths_Quantity
 3
```

Ausgabe:

```
Invalidation                        Location

------------                        --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- API-Details finden Sie CreateInvalidation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CFSignedCookie`

Das folgende Codebeispiel zeigt die Verwendung. `New-CFSignedCookie`

Tools für PowerShell V5

Beispiel 1: Erstellt mithilfe einer vordefinierten Richtlinie ein signiertes Cookie für die angegebene Ressource. Das Cookie ist ein Jahr lang gültig.

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/image1.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddYears(1)
}
New-CFSignedCookie @params
```

Ausgabe:

```
Expires
-------
[CloudFront-Expires, 1472227284]
```

Beispiel 1: Erstellt mithilfe einer vordefinierten Richtlinie ein signiertes Cookie für die angegebenen Ressourcen. Das Cookie ist innerhalb von 24 Stunden gültig und läuft eine Woche danach ab.

```
$start = (Get-Date).AddHours(24)
```

```
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
}

New-CFSignedCookie @params
```

Ausgabe:

```
Policy
------
[CloudFront-Policy, eyJTd...wIjo...
```

Beispiel 3: Erstellt mithilfe einer vordefinierten Richtlinie ein signiertes Cookie für die angegebenen Ressourcen. Das Cookie ist innerhalb von 24 Stunden gültig und läuft eine Woche danach ab. Der Zugriff auf die Ressourcen ist auf den angegebenen IP-Bereich beschränkt.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="http://xyz.cloudfront.net/content/*.jpeg"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=$start.AddDays(7)
    "ActiveFrom"=$start
 "IpRange"="192.0.2.0/24"
}

New-CFSignedCookie @params
```

Ausgabe:

```
Policy
                                                                  ------

                                       [CloudFront-Policy, eyJTd...wIjo...
```

- API-Details finden Sie New-CFSignedCookie in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CFSignedUrl

Das folgende Codebeispiel zeigt die Verwendung. `New-CFSignedUrl`

Tools für PowerShell V5

Beispiel 1: Erstellt mithilfe einer vordefinierten Richtlinie eine signierte URL für die angegebene Ressource. Die URL ist eine Stunde lang gültig. Ein System.Uri Objekt, das die signierte URL enthält, wird an die Pipeline ausgegeben.

```
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddHours(1)
}
New-CFSignedUrl @params
```

Beispiel 2: Erstellt mithilfe einer vordefinierten Richtlinie eine signierte URL für die angegebene Ressource. Die URL ist nach Ablauf von 24 Stunden gültig und läuft eine Woche später ab.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
    "ActiveFrom"=$start
}
New-CFSignedUrl @params
```

Beispiel 3: Erstellt mithilfe einer vordefinierten Richtlinie eine signierte URL für die angegebene Ressource. Die URL ist nach Ablauf von 24 Stunden gültig und läuft eine Woche später ab. Der Zugriff auf die Ressource ist auf den angegebenen IP-Bereich beschränkt.

```
$start = (Get-Date).AddHours(24)
$params = @{
 "ResourceUri"="https://cdn.example.com/index.html"
 "KeyPairId"="AKIAIOSFODNN7EXAMPLE"
 "PrivateKeyFile"="C:\pk-AKIAIOSFODNN7EXAMPLE.pem"
 "ExpiresOn"=(Get-Date).AddDays(7)
```

```
    "ActiveFrom"=$start
    "IpRange"="192.0.2.0/24"
}
New-CFSignedUrl @params
```

- API-Details finden Sie New-CFSignedUrl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CloudTrail Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CloudTrail

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Find-CTEvent`

Das folgende Codebeispiel zeigt die Verwendung`Find-CTEvent`.

Tools für PowerShell V5

Beispiel 1: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind. Das Cmdlet führt standardmäßig automatisch mehrere Aufrufe aus, um alle Ereignisse bereitzustellen, und beendet die Ausführung, wenn der Service angibt, dass keine weiteren Daten verfügbar sind.

```
Find-CTEvent
```

Beispiel 2: Gibt alle Ereignisse zurück, die in den letzten sieben Tagen aufgetreten sind, und gibt dabei eine Region an, die nicht dem aktuellen Shell-Standard entspricht.

```
Find-CTEvent -Region eu-central-1
```

Beispiel 3: Gibt alle Ereignisse zurück, die mit dem RunInstances API-Aufruf verknüpft sind.

```
Find-CTEvent -LookupAttribute @{ AttributeKey="EventName";
 AttributeValue="RunInstances" }
```

Beispiel 4: Gibt die ersten 5 verfügbaren Ereignisse zurück.

```
Find-CTEvent -MaxResult 5
```

- API-Details finden Sie LookupEvents in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CTTrail`

Tools für PowerShell V5

Beispiel 1: Gibt die Einstellungen aller Trails zurück, die der aktuellen Region Ihres Kontos zugeordnet sind.

```
Get-CTTrail
```

Beispiel 2: Gibt die Einstellungen für die angegebenen Trails zurück.

```
Get-CTTrail -TrailNameList trail1,trail2
```

Beispiel 3: Gibt die Einstellungen für die angegebenen Trails zurück, die in einer anderen Region als der aktuellen Shell-Standardregion (in diesem Fall der Region Frankfurt (eu-central-1)) erstellt wurden).

```
Get-CTTrail -TrailNameList trailABC,trailDEF -Region eu-central-1
```

- API-Details finden Sie DescribeTrails in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CTTrailStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CTTrailStatus`

Tools für PowerShell V5

Beispiel 1: Gibt Statusinformationen für den Trail mit dem Namen „my“ ExampleTrail zurück. Die zurückgegebenen Daten umfassen Informationen zu Zustellungsfehlern, Amazon-SNS- und Amazon-S3-Fehlern sowie Start- und Endzeiten für die Protokollierung des Trails. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Get-CTTrailStatus -Name myExampleTrail
```

Beispiel 2: Gibt Statusinformationen für einen Trail zurück, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall in der Region Frankfurt (eu-central).

```
Get-CTTrailStatus -Name myExampleTrail -Region eu-central-1
```

- API-Details finden Sie GetTrailStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `New-CTTrail`

Tools für PowerShell V5

Beispiel 1: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet.

```
New-CTTrail -Name "awscloudtrail-example" -S3BucketName "amzn-s3-demo-bucket"
```

Beispiel 2: Erzeugt einen Trail, der den Bucket „amzn-s3-demo-bucket“ für die Speicherung von Protokolldateien verwendet. Die S3-Objekte, die die Protokolle darstellen, verfügen über das gemeinsame Schlüsselpräfix „mylogs“. Wenn neue Protokolle an den Bucket geliefert werden, wird eine Benachrichtigung an das SNS-Thema „mlog-deliverytopic” gesendet. In diesem Beispiel wird Splatting verwendet, um die Parameterwerte an das Cmdlet zu übergeben.

```
$params = @{
    Name="awscloudtrail-example"
    S3BucketName="amzn-s3-demo-bucket"
    S3KeyPrefix="mylogs"
    SnsTopicName="mlog-deliverytopic"
```

```
}
New-CTTrail @params
```

- API-Details finden Sie CreateTrail in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CTTrail`

Tools für PowerShell V5

Beispiel 1: Löscht den angegebenen Trail. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigung zu unterdrücken, fügen Sie den Schalterparameter -Force hinzu.

```
Remove-CTTrail -Name "awscloudtrail-example"
```

- API-Details finden Sie DeleteTrail in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-CTLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Start-CTLogging`

Tools für PowerShell V5

Beispiel 1: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „my“ExampleTrail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Start-CTLogging -Name myExampleTrail
```

Beispiel 2: Startet die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Start-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API-Details finden Sie StartLogging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-CTLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CTLogging`

Tools für PowerShell V5

Beispiel 1: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für den Trail mit dem Namen „my“ExampleTrail. In diesem Beispiel wird davon ausgegangen, dass der Trail in derselben Region erstellt wurde wie die aktuelle Shell-Standardregion.

```
Stop-CTLogging -Name myExampleTrail
```

Beispiel 2: Unterbricht die Aufzeichnung von AWS API-Aufrufen und die Bereitstellung von Protokolldateien für einen Trail, der in einer anderen Region als der aktuellen Shell-Standardregion erstellt wurde (in diesem Fall der Region Frankfurt (eu-central-1)).

```
Stop-CTLogging -Name myExampleTrail -Region eu-central-1
```

- API-Details finden Sie StopLogging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CTTrail`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CTTrail`

Tools für PowerShell V5

Beispiel 1: Aktualisiert den angegebenen Trail, sodass globale Serviceereignisse (z. B. von IAM) aufgezeichnet werden, und ändert das gemeinsame Schlüsselpräfix der Protokolldateien künftig in „globallogs“.

```
Update-CTTrail -Name "awscloudtrail-example" -IncludeGlobalServiceEvents $true -
S3KeyPrefix "globallogs"
```

Beispiel 2: Aktualisiert den angegebenen Trail, sodass Benachrichtigungen über neue Protokollbereitstellungen an das angegebene SNS-Thema gesendet werden.

```
Update-CTTrail -Name "awscloudtrail-example" -SnsTopicName "mlog-deliverytopic2"
```

Beispiel 3: Aktualisiert den angegebenen Trail, sodass Protokolle in einen anderen Bucket zugestellt werden.

```
Update-CTTrail -Name "awscloudtrail-example" -S3BucketName "otherlogs"
```

- API-Details finden Sie UpdateTrail in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CloudWatch Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CloudWatch

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-CWDashboard`

Das folgende Codebeispiel zeigt die Verwendung`Get-CWDashboard`.

Tools für PowerShell V5

Beispiel 1: Gibt den ARN und den Inhalt des angegebenen Dashboards zurück.

```
Get-CWDashboard -DashboardName Dashboard1
```

Ausgabe:

```
DashboardArn                                          DashboardBody
------------                                          -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- API-Details finden Sie GetDashboard in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CWDashboardList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CWDashboardList`

Tools für PowerShell V5

Beispiel 1: Gibt die Sammlung von Dashboards für Ihr Konto zurück.

```
Get-CWDashboardList
```

Ausgabe:

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

Beispiel 2: Gibt die Sammlung von Dashboards für Ihr Konto zurück, deren Namen mit dem Präfix „dev“ beginnen.

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- API-Details finden Sie ListDashboards in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CWDashboard`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CWDashboard`

Tools für PowerShell V5

Beispiel 1: Löscht das angegebene Dashboard und bittet um Bestätigung, bevor der Vorgang fortgesetzt wird. Um die Bestätigung zu umgehen, fügen Sie dem Befehl den Schalter -Force hinzu.

```
Remove-CWDashboard -DashboardName Dashboard1
```

- API-Details finden Sie DeleteDashboards in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-CWDashboard`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CWDashboard`

Tools für PowerShell V5

Beispiel 1: Erstellt oder aktualisiert das Dashboard mit dem Namen „Dashboard1“, sodass es zwei Metrik-Widgets nebeneinander enthält.

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
                      "CPUUtilization",
                      "InstanceId",
                      "i-012345"
                   ]
                ],
                "period":300,
                "stat":"Average",
                "region":"us-east-1",
                "title":"EC2 Instance CPU"
             }
        },
        {
             "type":"metric",
             "x":12,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/S3",
                      "BucketSizeBytes",
```

```
                        "BucketName",
                        "amzn-s3-demo-bucket"
                    ]
                ],
                "period":86400,
                "stat":"Maximum",
                "region":"us-east-1",
                "title":"amzn-s3-demo-bucket bytes"
            }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

Beispiel 2: Erstellt oder aktualisiert das Dashboard und leitet den Inhalt, der das Dashboard beschreibt, an das Cmdlet weiter.

```
$dashBody = @"
{
...
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- API-Details finden Sie PutDashboard in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-CWMetricData`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CWMetricData`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues MetricDatum Objekt und schreibt es in Amazon Web Services CloudWatch Metrics.

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
```

```
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- API-Details finden Sie PutMetricData in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CodeCommit Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CodeCommit

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-CCBranch`

Das folgende Codebeispiel zeigt die Verwendung`Get-CCBranch`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Branch für das spezifizierte Repository abgerufen.

```
Get-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch
```

Ausgabe:

```
BranchName                              CommitId
----------                              --------
MyNewBranch                             7763222d...561fc9c9
```

- API-Details finden Sie GetBranch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CCBranchList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCBranchList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Branch-Namen für das angegebene Repository abgerufen.

```
Get-CCBranchList -RepositoryName MyDemoRepo
```

Ausgabe:

```
master
MyNewBranch
```

- API-Details finden Sie ListBranches in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CCRepository`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepository`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen für das angegebene Repository abgerufen.

```
Get-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CreationDate          : 9/8/2015 3:21:33 PM
DefaultBranch         :
LastModifiedDate      : 9/8/2015 3:21:33 PM
```

```
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE
RepositoryName        : MyDemoRepo
```

- API-Details finden Sie GetRepository in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CCRepositoryBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepositoryBatch`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel bestätigt, welche der angegebenen Repositorys gefunden wurden und welche nicht.

```
Get-CCRepositoryBatch -RepositoryName MyDemoRepo, MyNewRepo, AMissingRepo
```

Ausgabe:

```
Repositories                          RepositoriesNotFound
------------                          --------------------
{MyDemoRepo, MyNewRepo}              {AMissingRepo}
```

- API-Details finden Sie BatchGetRepositories in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CCRepositoryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CCRepositoryList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Repositorys in aufsteigender Reihenfolge nach Repository-Namen aufgelistet.

```
Get-CCRepositoryList -Order Ascending -SortBy RepositoryName
```

Ausgabe:

```
RepositoryId                          RepositoryName
```

```
------------                              --------------
c7d0d2b0-ce40-4303-b4c3-38529EXAMPLE    MyDemoRepo
05f30c66-e3e3-4f91-a0cd-1c84aEXAMPLE    MyNewRepo
```

- API-Details finden Sie ListRepositories in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CCBranch

Das folgende Codebeispiel zeigt die Verwendung. `New-CCBranch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer Branch mit dem angegebenen Namen für das angegebene Repository und der angegebenen Commit-ID erstellt.

```
New-CCBranch -RepositoryName MyDemoRepo -BranchName MyNewBranch -CommitId
 7763222d...561fc9c9
```

- API-Details finden Sie CreateBranch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CCRepository

Das folgende Codebeispiel zeigt die Verwendung. `New-CCRepository`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Repository mit dem angegebenen Namen und der angegebenen Beschreibung erstellt.

```
New-CCRepository -RepositoryName MyDemoRepo -RepositoryDescription "This is a
 repository for demonstration purposes."
```

Ausgabe:

```
AccountId             : 80398EXAMPLE
Arn                   : arn:aws:codecommit:us-east-1:80398EXAMPLE:MyDemoRepo
CloneUrlHttp          : https://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
CloneUrlSsh           : ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/
MyDemoRepo
```

```
CreationDate          : 9/18/2015 4:13:25 PM
DefaultBranch         :
LastModifiedDate      : 9/18/2015 4:13:25 PM
RepositoryDescription : This is a repository for demonstration purposes.
RepositoryId          : 43ef2443-3372-4b12-9e78-65c27EXAMPLE
RepositoryName        : MyDemoRepo
```

- API-Details finden Sie CreateRepository in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CCRepository`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CCRepository`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Repository zwangsweise gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Repository ohne Prompt zu löschen.

```
Remove-CCRepository -RepositoryName MyDemoRepo
```

Ausgabe:

```
43ef2443-3372-4b12-9e78-65c27EXAMPLE
```

- API-Details finden Sie DeleteRepository in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CCDefaultBranch`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCDefaultBranch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Standard-Branch für das angegebene Repository in den spezifischen Branch geändert.

```
Update-CCDefaultBranch -RepositoryName MyDemoRepo -DefaultBranchName MyNewBranch
```

- API-Details finden Sie UpdateDefaultBranch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CCRepositoryDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCRepositoryDescription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Repository geändert.

```
Update-CCRepositoryDescription -RepositoryName MyDemoRepo -RepositoryDescription
 "This is an updated description."
```

- API-Details finden Sie UpdateRepositoryDescription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CCRepositoryName`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CCRepositoryName`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Name des angegebenen Repositorys geändert.

```
Update-CCRepositoryName -NewName MyDemoRepo2 -OldName MyDemoRepo
```

- API-Details finden Sie UpdateRepositoryName in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CodeDeploy Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CodeDeploy

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-CDOnPremiseInstanceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-CDOnPremiseInstanceTag`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Tag einer On-Premises-Instance mit dem angegebenen Schlüssel und Wert für die angegebene On-Premises-Instance hinzugefügt.

```
Add-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" = "Name";
 "Value" = "CodeDeployDemo-OnPrem"}
```

- API-Details finden Sie AddTagsToOnPremisesInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplication`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendung abgerufen.

```
Get-CDApplication -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName             CreateTime
         LinkedToGitHub
-------------                           ---------------             ----------
         --------------
```

```
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
```

- API-Details finden Sie GetApplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDApplicationBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Anwendungen abgerufen.

```
Get-CDApplicationBatch -ApplicationName CodeDeployDemoApplication,
 CodePipelineDemoApplication
```

Ausgabe:

```
ApplicationId                           ApplicationName              CreateTime
         LinkedToGitHub
-------------                           ---------------              ----------
         --------------
e07fb938-091e-4f2f-8963-4d3e8EXAMPLE    CodeDeployDemoApplication    7/20/2015
 9:49:48 PM    False
1ecfd602-62f1-4038-8f0d-06688EXAMPLE    CodePipelineDemoApplication  8/13/2015
 5:53:26 PM    False
```

- API-Details finden Sie BatchGetApplications in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDApplicationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Anwendungen abgerufen.

```
Get-CDApplicationList
```

Ausgabe:

```
CodeDeployDemoApplication
CodePipelineDemoApplication
```

- API-Details finden Sie ListApplications in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDApplicationRevision`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationRevision`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Anwendungsrevision abgerufen.

```
$revision = Get-CDApplicationRevision -ApplicationName CodeDeployDemoApplication
 -S3Location_Bucket amzn-s3-demo-bucket -Revision_RevisionType S3 -
S3Location_Key 5xd27EX.zip -S3Location_BundleType zip -S3Location_ETag
 4565c1ac97187f190c1a90265EXAMPLE
Write-Output ("Description = " + $revision.RevisionInfo.Description + ",
 RegisterTime = " + $revision.RevisionInfo.RegisterTime)
```

Ausgabe:

```
Description = Application revision registered by Deployment ID: d-CX9CHN3EX,
 RegisterTime = 07/20/2015 23:46:42
```

- API-Details finden Sie GetApplicationRevision in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDApplicationRevisionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDApplicationRevisionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den verfügbaren Revisionen für die angegebene Anwendung abgerufen.

```
ForEach ($revision in (Get-CDApplicationRevisionList -ApplicationName
 CodeDeployDemoApplication -Deployed Ignore)) {
>>   If ($revision.RevisionType -Eq "S3") {
>>     Write-Output ("Type = S3, Bucket = " + $revision.S3Location.Bucket
 + ", BundleType = " + $revision.S3Location.BundleType + ", ETag = " +
 $revision.S3Location.ETag + ", Key = " + $revision.S3Location.Key)
>>   }
>>   If ($revision.RevisionType -Eq "GitHub") {
>>     Write-Output ("Type = GitHub, CommitId = " +
 $revision.GitHubLocation.CommitId + ", Repository = " +
 $revision.GitHubLocation.Repository)
>>   }
>> }
>>
```

Ausgabe:

```
Type = S3, Bucket = amzn-s3-demo-bucket, BundleType = zip, ETag =
 4565c1ac97187f190c1a90265EXAMPLE, Key = 5xd27EX.zip
Type = GitHub, CommitId = f48933c3...76405362, Repository = MyGitHubUser/
CodeDeployDemoRepo
```

- API-Details finden Sie ListApplicationRevisions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeployment`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeployment`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellung abgerufen.

```
Get-CDDeployment -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
ApplicationName                : CodeDeployDemoApplication
CompleteTime                   : 7/23/2015 11:26:04 PM
CreateTime                     : 7/23/2015 11:24:43 PM
```

```
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

Beispiel 2: In diesem Beispiel werden Informationen zum Status der Instances abgerufen, die an der angegebenen Bereitstellung beteiligt sind.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).DeploymentOverview
```

Ausgabe:

```
Failed     : 0
InProgress : 0
Pending    : 0
Skipped    : 0
Succeeded  : 3
```

Beispiel 3: Dieses Beispiel ruft Informationen zur Anwendungsrevision für die angegebene Bereitstellung ab.

```
(Get-CDDeployment -DeploymentId d-QZMRGSTEX).Revision.S3Location
```

Ausgabe:

```
Bucket     : amzn-s3-demo-bucket
BundleType : zip
ETag       : cfbb81b304ee5e27efc21adaed3EXAMPLE
Key        : clzfqEX
Version    :
```

- API-Details finden Sie GetDeployment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Get-CDDeploymentBatch

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen Bereitstellungen abgerufen.

```
Get-CDDeploymentBatch -DeploymentId d-QZMRGSTEX, d-RR0T5KTEX
```

Ausgabe:

```
ApplicationName               : CodeDeployDemoApplication
CompleteTime                  : 7/23/2015 11:26:04 PM
CreateTime                    : 7/23/2015 11:24:43 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodeDeployDemoFleet
DeploymentId                  : d-QZMRGSTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded

ApplicationName               : CodePipelineDemoApplication
CompleteTime                  : 7/23/2015 6:07:30 PM
CreateTime                    : 7/23/2015 6:06:29 PM
Creator                       : user
DeploymentConfigName          : CodeDeployDefault.OneAtATime
DeploymentGroupName           : CodePipelineDemoFleet
DeploymentId                  : d-RR0T5KTEX
DeploymentOverview            : Amazon.CodeDeploy.Model.DeploymentOverview
Description                   :
ErrorInformation              :
IgnoreApplicationStopFailures : False
Revision                      : Amazon.CodeDeploy.Model.RevisionLocation
StartTime                     : 1/1/0001 12:00:00 AM
Status                        : Succeeded
```

- API-Details finden Sie BatchGetDeployments in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden zusammenfassende Informationen zur angegebenen Bereitstellungskonfiguration abgerufen.

```
Get-CDDeploymentConfig -DeploymentConfigName ThreeQuartersHealthy
```

Ausgabe:

```
CreateTime              DeploymentConfigId                      DeploymentConfigName
    MinimumHealthyHosts
----------              ------------------                      --------------------
    -------------------
10/3/2014 4:32:30 PM    518a3950-d034-46a1-9d2c-3c949EXAMPLE    ThreeQuartersHealthy
    Amazon.CodeDeploy.Model.MinimumHealthyHosts
```

Beispiel 2: In diesem Beispiel werden Informationen zur Definition der angegebenen Bereitstellungskonfiguration abgerufen.

```
Write-Output ((Get-CDDeploymentConfig -DeploymentConfigName
 ThreeQuartersHealthy).MinimumHealthyHosts)
```

Ausgabe:

```
Type            Value
----            -----
FLEET_PERCENT   75
```

- API-Details finden Sie GetDeploymentConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentConfigList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Bereitstellungskonfigurationen abgerufen.

```
Get-CDDeploymentConfigList
```

Ausgabe:

```
ThreeQuartersHealthy
CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime
```

- API-Details finden Sie ListDeploymentConfigs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zur angegebenen Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentGroup -ApplicationName CodeDeployDemoApplication -
DeploymentGroupName CodeDeployDemoFleet
```

Ausgabe:

```
ApplicationName              : CodeDeployDemoApplication
AutoScalingGroups            : {}
DeploymentConfigName         : CodeDeployDefault.OneAtATime
DeploymentGroupId            : 7d7c098a-b444-4b27-96ef-22791EXAMPLE
DeploymentGroupName          : CodeDeployDemoFleet
Ec2TagFilters                : {Name}
```

```
OnPremisesInstanceTagFilters : {}
ServiceRoleArn               : arn:aws:iam::80398EXAMPLE:role/
CodeDeploySampleStack-4ph6EX-CodeDeployTrustRole-O9MWP7XTL8EX
TargetRevision               : Amazon.CodeDeploy.Model.RevisionLocation
```

- API-Details finden Sie GetDeploymentGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentGroupList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungsgruppen für die angegebene Anwendung abgerufen.

```
Get-CDDeploymentGroupList -ApplicationName CodeDeployDemoApplication
```

Ausgabe:

```
ApplicationName              DeploymentGroups
 NextToken
---------------              ----------------
 ---------
CodeDeployDemoApplication    {CodeDeployDemoFleet, CodeDeployProductionFleet}
```

- API-Details finden Sie ListDeploymentGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über die angegebene Instance für die spezifizierte Bereitstellung abgerufen.

```
Get-CDDeploymentInstance -DeploymentId d-QZMRGSTEX -InstanceId i-254e22EX
```

Ausgabe:

```
DeploymentId    : d-QZMRGSTEX
InstanceId      : arn:aws:ec2:us-east-1:80398EXAMPLE:instance/i-254e22EX
LastUpdatedAt   : 7/23/2015 11:25:24 PM
LifecycleEvents : {ApplicationStop, DownloadBundle, BeforeInstall, Install...}
Status          : Succeeded
```

- API-Details finden Sie GetDeploymentInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentInstanceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentInstanceList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Instance-IDs für die angegebene Bereitstellung abgerufen.

```
Get-CDDeploymentInstanceList -DeploymentId d-QZMRGSTEX
```

Ausgabe:

```
i-254e22EX
i-274e22EX
i-3b4e22EX
```

- API-Details finden Sie ListDeploymentInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDDeploymentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDDeploymentList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Bereitstellungs-IDs für die angegebene Anwendung und Bereitstellungsgruppe abgerufen.

```
Get-CDDeploymentList -ApplicationName CodeDeployDemoApplication -DeploymentGroupName
 CodeDeployDemoFleet
```

Ausgabe:

```
d-QZMRGSTEX
d-RR0T5KTEX
```

- API-Details finden Sie ListDeployments in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- API-Details finden Sie GetOnPremisesInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDOnPremiseInstanceBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstanceBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu den angegebenen On-Premises-Instances abgerufen.

```
Get-CDOnPremiseInstanceBatch -InstanceName AssetTag12010298EX, AssetTag12010298EX-2
```

Ausgabe:

```
DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployFRWUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX-2_XmeSz18rEX
InstanceName   : AssetTag12010298EX-2
RegisterTime   : 4/3/2015 6:38:52 PM
Tags           : {Name}

DeregisterTime : 1/1/0001 12:00:00 AM
IamUserArn     : arn:aws:iam::80398EXAMPLE:user/CodeDeployDemoUser
InstanceArn    : arn:aws:codedeploy:us-east-1:80398EXAMPLE:instance/
AssetTag12010298EX_rDH556dxEX
InstanceName   : AssetTag12010298EX
RegisterTime   : 4/3/2015 6:36:24 PM
Tags           : {Name}
```

- API-Details finden Sie BatchGetOnPremisesInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CDOnPremiseInstanceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CDOnPremiseInstanceList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren On-Premises-Instance-Namen abgerufen.

```
Get-CDOnPremiseInstanceList
```

Ausgabe:

```
AssetTag12010298EX
AssetTag12010298EX-2
```

- API-Details finden Sie ListOnPremisesInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CDApplication

Das folgende Codebeispiel zeigt die Verwendung. `New-CDApplication`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Anwendung mit dem angegebenen Namen erstellt.

```
New-CDApplication -ApplicationName MyNewApplication
```

Ausgabe:

```
f19e4b61-2231-4328-b0fd-e57f5EXAMPLE
```

- API-Details finden Sie CreateApplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CDDeployment

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeployment`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellung für die angegebene Anwendung und Bereitstellungsgruppe mit der angegebenen Bereitstellungskonfiguration und Anwendungsversion erstellt.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True -
S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3
```

Ausgabe:

```
d-ZHROG7UEX
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen wird.

```
New-CDDeployment -ApplicationName MyNewApplication -S3Location_Bucket amzn-s3-demo-
bucket -S3Location_BundleType zip -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -IgnoreApplicationStopFailures $True
 -S3Location_Key aws-codedeploy_linux-master.zip -RevisionType S3 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Ausgabe:

```
d-ZHROG7UEX
```

- API-Details finden Sie CreateDeployment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeploymentConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Bereitstellungskonfiguration mit dem angegebenen Namen und Verhalten erstellt.

```
New-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts -
MinimumHealthyHosts_Type HOST_COUNT -MinimumHealthyHosts_Value 2
```

Ausgabe:

```
0f3e8187-44ef-42da-aeed-b6823EXAMPLE
```

- API-Details finden Sie CreateDeploymentConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CDDeploymentGroup

Das folgende Codebeispiel zeigt die Verwendung. `New-CDDeploymentGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Bereitstellungsgruppe mit dem angegebenen Namen, der Auto-Scaling-Gruppe, der Bereitstellungskonfiguration, dem Tag und der Servicerolle für die angegebene Anwendung erstellt.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen wird.

```
New-CDDeploymentGroup -ApplicationName MyNewApplication -AutoScalingGroup
 CodeDeployDemo-ASG -DeploymentConfigName CodeDeployDefault.OneAtATime
 -DeploymentGroupName MyNewDeploymentGroup -Ec2TagFilter @{Key="Name";
 Type="KEY_AND_VALUE"; Value="CodeDeployDemo"} -ServiceRoleArn
 arn:aws:iam::80398EXAMPLE:role/CodeDeployDemo -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

Ausgabe:

```
16bbf199-95fd-40fc-a909-0bbcfEXAMPLE
```

- API-Details finden Sie CreateDeploymentGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Register-CDApplicationRevision

Das folgende Codebeispiel zeigt die Verwendung. `Register-CDApplicationRevision`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Anwendungsrevision mit dem angegebenen Amazon-S3-Standort für die angegebene Anwendung registriert.

```
Register-CDApplicationRevision -ApplicationName MyNewApplication -S3Location_Bucket
 amzn-s3-demo-bucket -S3Location_BundleType zip -S3Location_Key aws-
codedeploy_linux-master.zip -Revision_RevisionType S3
```

- API-Details finden Sie RegisterApplicationRevision in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung`Register-CDOnPremiseInstance`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine On-Premises-Instance mit dem angegebenen Namen und IAM-Benutzer registriert.

```
Register-CDOnPremiseInstance -IamUserArn arn:aws:iam::80398EXAMPLE:user/
CodeDeployDemoUser -InstanceName AssetTag12010298EX
```

- API-Details finden Sie RegisterOnPremisesInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDApplication`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Anwendung mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Anwendung ohne Prompt zu löschen.

```
Remove-CDApplication -ApplicationName MyNewApplication
```

- API-Details finden Sie DeleteApplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CDDeploymentConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDDeploymentConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Bereitstellungskonfiguration mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungskonfiguration ohne Prompt zu löschen.

```
Remove-CDDeploymentConfig -DeploymentConfigName AtLeastTwoHealthyHosts
```

- API-Details finden Sie DeleteDeploymentConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDDeploymentGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Bereitstellungsgruppe mit dem angegebenen Namen für die spezifizierte Anwendung gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Bereitstellungsgruppe ohne Prompt zu löschen.

```
Remove-CDDeploymentGroup -ApplicationName MyNewApplication -DeploymentGroupName
 MyNewDeploymentGroup
```

- API-Details finden Sie DeleteDeploymentGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CDOnPremiseInstanceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CDOnPremiseInstanceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Tag für die On-Premises-Instance mit dem angegebenen Namen gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um das Tag ohne Prompt zu löschen.

```
Remove-CDOnPremiseInstanceTag -InstanceName AssetTag12010298EX -Tag @{"Key" =
 "Name"; "Value" = "CodeDeployDemo-OnPrem"}
```

- API-Details finden Sie RemoveTagsFromOnPremisesInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-CDDeployment`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-CDDeployment`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird versucht, die Bereitstellung mit der angegebenen Bereitstellungs-ID zu beenden.

```
Stop-CDDeployment -DeploymentId d-LJQNREYEX
```

Ausgabe:

```
Status     StatusMessage
------     -------------
Pending    Stopping Pending. Stopping to schedule commands in the deployment
 instances
```

- API-Details finden Sie StopDeployment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-CDOnPremiseInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-CDOnPremiseInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung der On-Premises-Instance mit dem angegebenen Namen aufgehoben.

```
Unregister-CDOnPremiseInstance -InstanceName AssetTag12010298EX
```

- API-Details finden Sie DeregisterOnPremisesInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CDApplication`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CDApplication`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Anwendung geändert.

```
Update-CDApplication -ApplicationName MyNewApplication -NewApplicationName
 MyNewApplication-2
```

- API-Details finden Sie UpdateApplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CDDeploymentGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CDDeploymentGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Name der angegebenen Bereitstellungsgruppe für die angegebene Anwendung geändert.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2
```

Beispiel 2: Dieses Beispiel zeigt, wie Gruppen von EC2-Instance-Tags angegeben werden, anhand derer eine Instance identifiziert werden muss, damit sie in die Ersatzumgebung für eine blue/green Bereitstellung aufgenommen wird.

```
Update-CDDeploymentGroup -ApplicationName MyNewApplication -
CurrentDeploymentGroupName MyNewDeploymentGroup -NewDeploymentGroupName
 MyNewDeploymentGroup-2 -Ec2TagSetList
 @(@{Key="key1";Type="KEY_ONLY"},@{Key="Key2";Type="KEY_AND_VALUE";Value="Value2"}),@(@{Key=
```

- API-Details finden Sie UpdateDeploymentGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

CodePipeline Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. CodePipeline

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Confirm-CPJob`

Das folgende Codebeispiel zeigt die Verwendung`Confirm-CPJob`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status eines bestimmten Auftrags abgerufen.

```
Confirm-CPJob -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE -Nonce 3
```

Ausgabe:

```
Value
```

```
-----
InProgress
```

- API-Details finden Sie AcknowledgeJob in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-CPStageTransition`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-CPStageTransition`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline deaktiviert.

```
Disable-CPStageTransition -PipelineName CodePipelineDemo -Reason "Disabling
 temporarily." -StageName Beta -TransitionType Inbound
```

- API-Details finden Sie DisableStageTransition in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-CPStageTransition`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-CPStageTransition`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der eingehende Übergang für die angegebene Phase in der angegebenen Pipeline aktiviert.

```
Enable-CPStageTransition -PipelineName CodePipelineDemo -StageName Beta -
TransitionType Inbound
```

- API-Details finden Sie EnableStageTransition in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPActionType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPActionType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über alle verfügbaren Aktionen für den angegebenen Besitzer abgerufen.

```
ForEach ($actionType in (Get-CPActionType -ActionOwnerFilter AWS)) {
  Write-Output ("For Category = " + $actionType.Id.Category + ", Owner = " +
 $actionType.Id.Owner + ", Provider = " + $actionType.Id.Provider + ", Version = " +
 $actionType.Id.Version + ":")
  Write-Output ("  ActionConfigurationProperties:")
  ForEach ($acp in $actionType.ActionConfigurationProperties) {
    Write-Output ("    For " + $acp.Name + ":")
    Write-Output ("      Description = " + $acp.Description)
    Write-Output ("      Key = " + $acp.Key)
    Write-Output ("      Queryable = " + $acp.Queryable)
    Write-Output ("      Required = " + $acp.Required)
    Write-Output ("      Secret = " + $acp.Secret)
  }
  Write-Output ("  InputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.InputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.InputArtifactDetails.MinimumCount)
  Write-Output ("  OutputArtifactDetails:")
  Write-Output ("    MaximumCount = " +
 $actionType.OutputArtifactDetails.MaximumCount)
  Write-Output ("    MinimumCount = " +
 $actionType.OutputArtifactDetails.MinimumCount)
  Write-Output ("  Settings:")
  Write-Output ("    EntityUrlTemplate = " + $actionType.Settings.EntityUrlTemplate)
  Write-Output ("    ExecutionUrlTemplate = " +
 $actionType.Settings.ExecutionUrlTemplate)
}
```

Ausgabe:

```
For Category = Deploy, Owner = AWS, Provider = ElasticBeanstalk, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS Elastic Beanstalk Application name
      Key = True
      Queryable = False
      Required = True
```

```
      Secret = False
    For EnvironmentName:
      Description = The AWS Elastic Beanstalk Environment name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/elasticbeanstalk/r/
application/{Config:ApplicationName}
For Category = Deploy, Owner = AWS, Provider = CodeDeploy, Version = 1:
  ActionConfigurationProperties:
    For ApplicationName:
      Description = The AWS CodeDeploy Application name
      Key = True
      Queryable = False
      Required = True
      Secret = False
    For DeploymentGroupName:
      Description = The AWS CodeDeploy Deployment Group name
      Key = True
      Queryable = False
      Required = True
      Secret = False
  InputArtifactDetails:
    MaximumCount = 1
    MinimumCount = 1
  OutputArtifactDetails:
    MaximumCount = 0
    MinimumCount = 0
  Settings:
    EntityUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
applications/{Config:ApplicationName}/deployment-groups/{Config:DeploymentGroupName}
    ExecutionUrlTemplate = https://console.aws.amazon.com/codedeploy/home?#/
deployments/{ExternalExecutionId}
```

- API-Details finden Sie ListActionTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPActionableJobList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPActionableJobList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über alle Aufträge abgerufen, die für die angegebene Aktionskategorie, den Besitzer, den Anbieter, die Version und die Abfrageparameter ausgeführt werden können.

```
Get-CPActionableJobList -ActionTypeId_Category Build -ActionTypeId_Owner Custom
 -ActionTypeId_Provider MyCustomProviderName -ActionTypeId_Version 1 -QueryParam
 @{"ProjectName" = "MyProjectName"}
```

Ausgabe:

```
AccountId       Data                                Id
         Nonce
---------       ----                                --
         -----
80398EXAMPLE    Amazon.CodePipeline.Model.JobData   0de392f5-712d-4f41-ace3-
f57a0EXAMPLE    3
```

- API-Details finden Sie PollForJobs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPJobDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPJobDetail`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zum angegebenen Auftrag abgerufen.

```
Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Ausgabe:

```
AccountId       Data                                Id
---------       ----                                --
80398EXAMPLE    Amazon.CodePipeline.Model.JobData
 f570dc12-5ef3-44bc-945a-6e133EXAMPLE
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zum angegebenen Auftrag abgerufen.

```
$jobDetails = Get-CPJobDetail -JobId f570dc12-5ef3-44bc-945a-6e133EXAMPLE
Write-Output ("For Job " + $jobDetails.Id + ":")
Write-Output ("  AccountId = " + $jobDetails.AccountId)
$jobData = $jobDetails.Data
Write-Output ("  Configuration:")
ForEach ($key in $jobData.ActionConfiguration.Keys) {
  $value = $jobData.ActionConfiguration.$key
  Write-Output ("    " + $key + " = " + $value)
}
Write-Output ("  ActionTypeId:")
Write-Output ("    Category = " + $jobData.ActionTypeId.Category)
Write-Output ("    Owner = " + $jobData.ActionTypeId.Owner)
Write-Output ("    Provider = " + $jobData.ActionTypeId.Provider)
Write-Output ("    Version = " + $jobData.ActionTypeId.Version)
Write-Output ("  ArtifactCredentials:")
Write-Output ("    AccessKeyId = " + $jobData.ArtifactCredentials.AccessKeyId)
Write-Output ("    SecretAccessKey = " +
 $jobData.ArtifactCredentials.SecretAccessKey)
Write-Output ("    SessionToken = " + $jobData.ArtifactCredentials.SessionToken)
Write-Output ("  InputArtifacts:")
ForEach ($ia in $jobData.InputArtifacts) {
  Write-Output ("    " + $ia.Name)
}
Write-Output ("  OutputArtifacts:")
ForEach ($oa in $jobData.OutputArtifacts) {
  Write-Output ("    " + $oa.Name)
}
Write-Output ("  PipelineContext:")
$context = $jobData.PipelineContext
Write-Output ("    Name = " + $context.Action.Name)
Write-Output ("    PipelineName = " + $context.PipelineName)
Write-Output ("    Stage = " + $context.Stage.Name)
```

Ausgabe:

```
For Job f570dc12-5ef3-44bc-945a-6e133EXAMPLE:
  AccountId = 80398EXAMPLE
  Configuration:
  ActionTypeId:
    Category = Build
    Owner = Custom
    Provider = MyCustomProviderName
    Version = 1
  ArtifactCredentials:
    AccessKeyId = ASIAIEI3...IXI6YREX
    SecretAccessKey = cqAFDhEi...RdQyfa2u
    SessionToken = AQoDYXdz...5u+lsAU=
  InputArtifacts:
    MyApp
  OutputArtifacts:
    MyAppBuild
  PipelineContext:
    Name = Build
    PipelineName = CodePipelineDemo
    Stage = Build
```

- API-Details finden Sie GetJobDetails in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipeline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zur angegebenen Pipeline abgerufen.

```
Get-CPPipeline -Name CodePipelineDemo -Version 1
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Build, Beta, TestStage}
Version       : 1
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen zur angegebenen Pipeline abgerufen.

```
$pipeline = Get-CPPipeline -Name CodePipelineDemo
Write-Output ("Name = " + $pipeline.Name)
Write-Output ("RoleArn = " + $pipeline.RoleArn)
Write-Output ("Version = " + $pipeline.Version)
Write-Output ("ArtifactStore:")
Write-Output ("  Location = " + $pipeline.ArtifactStore.Location)
Write-Output ("  Type = " + $pipeline.ArtifactStore.Type.Value)
Write-Output ("Stages:")
ForEach ($stage in $pipeline.Stages) {
  Write-Output ("  Name = " + $stage.Name)
  Write-Output ("    Actions:")
  ForEach ($action in $stage.Actions) {
    Write-Output ("      Name = " + $action.Name)
 Write-Output ("        Category = " + $action.ActionTypeId.Category)
 Write-Output ("        Owner = " + $action.ActionTypeId.Owner)
 Write-Output ("        Provider = " + $action.ActionTypeId.Provider)
 Write-Output ("        Version = " + $action.ActionTypeId.Version)
 Write-Output ("        Configuration:")
 ForEach ($key in $action.Configuration.Keys) {
   $value = $action.Configuration.$key
   Write-Output ("          " + $key + " = " + $value)
 }
 Write-Output ("        InputArtifacts:")
 ForEach ($ia in $action.InputArtifacts) {
   Write-Output ("          " + $ia.Name)
 }
 ForEach ($oa in $action.OutputArtifacts) {
   Write-Output ("          " + $oa.Name)
 }
 Write-Output ("        RunOrder = " + $action.RunOrder)
  }
}
```

Ausgabe:

```
Name = CodePipelineDemo
RoleArn = arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Version = 3
ArtifactStore:
  Location = amzn-s3-demo-bucket
```

```
  Type = S3
Stages:
  Name = Source
    Actions:
      Name = Source
        Category = Source
        Owner = ThirdParty
        Provider = GitHub
        Version = 1
        Configuration:
          Branch = master
          OAuthToken = ****
          Owner = my-user-name
          Repo = MyRepoName
        InputArtifacts:
          MyApp
        RunOrder = 1
  Name = Build
    Actions:
      Name = Build
        Category = Build
        Owner = Custom
        Provider = MyCustomProviderName
        Version = 1
        Configuration:
          ProjectName = MyProjectName
        InputArtifacts:
          MyApp
          MyAppBuild
        RunOrder = 1
  Name = Beta
    Actions:
      Name = CodePipelineDemoFleet
        Category = Deploy
        Owner = AWS
        Provider = CodeDeploy
        Version = 1
        Configuration:
          ApplicationName = CodePipelineDemoApplication
          DeploymentGroupName = CodePipelineDemoFleet
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
  Name = TestStage
```

```
    Actions:
      Name = MyJenkinsTestAction
        Category = Test
        Owner = Custom
        Provider = MyCustomTestProvider
        Version = 1
        Configuration:
          ProjectName = MyJenkinsProjectName
        InputArtifacts:
          MyAppBuild
        RunOrder = 1
```

- API-Details finden Sie GetPipeline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPPipelineList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipelineList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der verfügbaren Pipelines abgerufen.

```
Get-CPPipelineList
```

Ausgabe:

```
Created                  Name                Updated                  Version
-------                  ----                -------                  -------
8/13/2015 10:17:54 PM    CodePipelineDemo    8/13/2015 10:17:54 PM    3
7/8/2015 2:41:53 AM      MyFirstPipeline     7/22/2015 9:06:37 PM     7
```

- API-Details finden Sie ListPipelines in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CPPipelineState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CPPipelineState`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden allgemeine Informationen zu den Phasen der angegebenen Pipeline abgerufen.

```
Get-CPPipelineState -Name CodePipelineDemo
```

Ausgabe:

```
Created         : 8/13/2015 10:17:54 PM
PipelineName    : CodePipelineDemo
PipelineVersion : 1
StageStates     : {Source, Build, Beta, TestStage}
Updated         : 8/13/2015 10:17:54 PM
```

Beispiel 2: In diesem Beispiel werden detaillierte Informationen über den Status der angegebenen Pipeline abgerufen.

```
ForEach ($stageState in (Get-CPPipelineState -Name $arg).StageStates) {
  Write-Output ("For " + $stageState.StageName + ":")
  Write-Output ("  InboundTransitionState:")
  Write-Output ("    DisabledReason = " +
 $stageState.InboundTransitionState.DisabledReason)
  Write-Output ("    Enabled = " + $stageState.InboundTransitionState.Enabled)
  Write-Output ("    LastChangedAt = " +
 $stageState.InboundTransitionState.LastChangedAt)
  Write-Output ("    LastChangedBy = " +
 $stageState.InboundTransitionState.LastChangedBy)
  Write-Output ("  ActionStates:")
  ForEach ($actionState in $stageState.ActionStates) {
    Write-Output ("    For " + $actionState.ActionName + ":")
 Write-Output ("      CurrentRevision:")
    Write-Output ("        Created = " + $actionState.CurrentRevision.Created)
 Write-Output ("        RevisionChangeId = " +
 $actionState.CurrentRevision.RevisionChangeId)
 Write-Output ("        RevisionId = " + $actionState.CurrentRevision.RevisionId)
 Write-Output ("      EntityUrl = " + $actionState.EntityUrl)
 Write-Output ("      LatestExecution:")
    Write-Output ("        ErrorDetails:")
    Write-Output ("          Code = " +
 $actionState.LatestExecution.ErrorDetails.Code)
 Write-Output ("          Message = " +
 $actionState.LatestExecution.ErrorDetails.Message)
 Write-Output ("        ExternalExecutionId = " +
 $actionState.LatestExecution.ExternalExecutionId)
 Write-Output ("        ExternalExecutionUrl = " +
 $actionState.LatestExecution.ExternalExecutionUrl)
```

```
 Write-Output ("        LastStatusChange = " +
 $actionState.LatestExecution.LastStatusChange)
 Write-Output ("        PercentComplete = " +
 $actionState.LatestExecution.PercentComplete)
 Write-Output ("        Status = " + $actionState.LatestExecution.Status)
 Write-Output ("        Summary = " + $actionState.LatestExecution.Summary)
 Write-Output ("      RevisionUrl = " + $actionState.RevisionUrl)
  }
}
```

Ausgabe:

```
For Source:
  InboundTransitionState:
    DisabledReason =
    Enabled =
    LastChangedAt =
    LastChangedBy =
  ActionStates:
    For Source:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://github.com/my-user-name/MyRepoName/tree/master
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/20/2015 23:28:45
        PercentComplete = 0
        Status = Succeeded
        Summary =
      RevisionUrl =
For Build:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
```

```
    For Build:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code = TimeoutError
          Message = The action failed because a job worker exceeded its time limit.
 If this is a custom action, make sure that the job worker is configured correctly.
        ExternalExecutionId =
        ExternalExecutionUrl =
        LastStatusChange = 07/21/2015 00:29:29
        PercentComplete = 0
        Status = Failed
        Summary =
      RevisionUrl =
For Beta:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For CodePipelineDemoFleet:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = https://console.aws.amazon.com/codedeploy/home?#/applications/
CodePipelineDemoApplication/deployment-groups/CodePipelineDemoFleet
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = d-D5LTCZXEX
        ExternalExecutionUrl = https://console.aws.amazon.com/codedeploy/home?#/
deployments/d-D5LTCZXEX
        LastStatusChange = 07/08/2015 22:07:42
        PercentComplete = 0
        Status = Succeeded
        Summary = Deployment Succeeded
      RevisionUrl =
```

```
For TestStage:
  InboundTransitionState:
    DisabledReason =
    Enabled = True
    LastChangedAt = 01/01/0001 00:00:00
    LastChangedBy =
  ActionStates:
    For MyJenkinsTestAction25:
      CurrentRevision:
        Created =
        RevisionChangeId =
        RevisionId =
      EntityUrl = http://54.174.131.1EX/job/MyJenkinsDemo
      LatestExecution:
        ErrorDetails:
          Code =
          Message =
        ExternalExecutionId = 5
        ExternalExecutionUrl = http://54.174.131.1EX/job/MyJenkinsDemo/5
        LastStatusChange = 07/08/2015 22:09:03
        PercentComplete = 0
        Status = Succeeded
        Summary = Finished
      RevisionUrl =
```

- API-Details finden Sie GetPipelineState in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-CPCustomActionType

Das folgende Codebeispiel zeigt die Verwendung. `New-CPCustomActionType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue benutzerdefinierte Aktion mit den angegebenen Eigenschaften erstellt.

```
New-CPCustomActionType -Category Build -ConfigurationProperty @{"Description"
 = "The name of the build project must be provided when this action is added
 to the pipeline."; "Key" = $True; "Name" = "ProjectName"; "Queryable"
 = $False; "Required" = $True; "Secret" = $False; "Type" = "String"} -
Settings_EntityUrlTemplate "https://my-build-instance/job/{Config:ProjectName}/"
```

```
 -Settings_ExecutionUrlTemplate "https://my-build-instance/job/mybuildjob/
lastSuccessfulBuild{ExternalExecutionId}/" -InputArtifactDetails_MaximumCount
 1 -OutputArtifactDetails_MaximumCount 1 -InputArtifactDetails_MinimumCount 0 -
OutputArtifactDetails_MinimumCount 0 -Provider "MyBuildProviderName" -Version 1
```

Ausgabe:

```
ActionConfigurationProperties : {ProjectName}
Id                            : Amazon.CodePipeline.Model.ActionTypeId
InputArtifactDetails          : Amazon.CodePipeline.Model.ArtifactDetails
OutputArtifactDetails         : Amazon.CodePipeline.Model.ArtifactDetails
Settings                      : Amazon.CodePipeline.Model.ActionTypeSettings
```

- API-Details finden Sie CreateCustomActionType in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `New-CPPipeline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Pipeline mit den angegebenen Einstellungen erstellt.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"
```

```
$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "Source"
$deployStage.Name = "Beta"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)

$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

New-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {Source, Beta}
Version       : 1
```

- API-Details finden Sie CreatePipeline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CPCustomActionType`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CPCustomActionType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene benutzerdefinierte Aktion gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die benutzerdefinierte Aktion ohne Prompt zu löschen.

```
Remove-CPCustomActionType -Category Build -Provider MyBuildProviderName -Version 1
```

- API-Details finden Sie DeleteCustomActionType in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CPPipeline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Pipeline gelöscht. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Parameter -Force hinzu, um die Pipeline ohne Prompt zu löschen.

```
Remove-CPPipeline -Name CodePipelineDemo
```

- API-Details finden Sie DeletePipeline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-CPPipelineExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Start-CPPipelineExecution`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird mit der Ausführung der angegebenen Pipeline begonnen.

```
Start-CPPipelineExecution -Name CodePipelineDemo
```

- API-Details finden Sie StartPipelineExecution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CPPipeline`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CPPipeline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene vorhandene Pipeline mit den angegebenen Einstellungen aktualisiert.

```
$pipeline = New-Object Amazon.CodePipeline.Model.PipelineDeclaration

$sourceStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration
$deployStageAction = New-Object Amazon.CodePipeline.Model.ActionDeclaration

$sourceStageActionOutputArtifact = New-Object
 Amazon.CodePipeline.Model.OutputArtifact
$sourceStageActionOutputArtifact.Name = "MyApp"

$sourceStageAction.ActionTypeId = @{"Category" = "Source"; "Owner" = "AWS";
 "Provider" = "S3"; "Version" = 1}
$sourceStageAction.Configuration.Add("S3Bucket", "amzn-s3-demo-bucket")
$sourceStageAction.Configuration.Add("S3ObjectKey", "my-object-key-name.zip")
$sourceStageAction.OutputArtifacts.Add($sourceStageActionOutputArtifact)
$sourceStageAction.Name = "Source"

$deployStageActionInputArtifact = New-Object Amazon.CodePipeline.Model.InputArtifact
$deployStageActionInputArtifact.Name = "MyApp"

$deployStageAction.ActionTypeId = @{"Category" = "Deploy"; "Owner" = "AWS";
 "Provider" = "CodeDeploy"; "Version" = 1}
$deployStageAction.Configuration.Add("ApplicationName",
 "CodePipelineDemoApplication")
$deployStageAction.Configuration.Add("DeploymentGroupName", "CodePipelineDemoFleet")
$deployStageAction.InputArtifacts.Add($deployStageActionInputArtifact)
$deployStageAction.Name = "CodePipelineDemoFleet"

$sourceStage = New-Object Amazon.CodePipeline.Model.StageDeclaration
$deployStage = New-Object Amazon.CodePipeline.Model.StageDeclaration

$sourceStage.Name = "MyInputFiles"
$deployStage.Name = "MyTestDeployment"

$sourceStage.Actions.Add($sourceStageAction)
$deployStage.Actions.Add($deployStageAction)
```

```
$pipeline.ArtifactStore = @{"Location" = "amzn-s3-demo-bucket"; "Type" = "S3"}
$pipeline.Name = "CodePipelineDemo"
$pipeline.RoleArn = "arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole"
$pipeline.Stages.Add($sourceStage)
$pipeline.Stages.Add($deployStage)
$pipeline.Version = 1

Update-CPPipeline -Pipeline $pipeline
```

Ausgabe:

```
ArtifactStore : Amazon.CodePipeline.Model.ArtifactStore
Name          : CodePipelineDemo
RoleArn       : arn:aws:iam::80398EXAMPLE:role/CodePipelineServiceRole
Stages        : {InputFiles, TestDeployment}
Version       : 2
```

- API-Details finden Sie UpdatePipeline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Amazon Cognito Identity mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon Cognito Identity verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CGIIdentityPool`

Tools für PowerShell V5

Beispiel 1: Ruft Informationen über einen bestimmten Identitätspool anhand seiner ID ab.

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : CommonTests1
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 142
HttpStatusCode                 : OK
```

- API-Details finden Sie DescribeIdentityPool in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CGIIdentityPoolList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CGIIdentityPoolList`

Tools für PowerShell V5

Beispiel 1: Ruft eine Liste vorhandener Identitätspools ab.

```
Get-CGIIdentityPoolList
```

Ausgabe:

```
IdentityPoolId                                                  IdentityPoolName
--------------                                                  ----------------
```

```
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- API-Details finden Sie ListIdentityPools in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CGIIdentityPoolRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CGIIdentityPoolRole`

Tools für PowerShell V5

Beispiel 1: Ruft die Informationen über Rollen für einen bestimmten Identitätspool ab.

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

Ausgabe:

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- API-Details finden Sie GetIdentityPoolRoles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `New-CGIIdentityPool`

Tools für PowerShell V5

Beispiel 1: Erstellt einen neuen Identitätspool, der nicht authentifizierte Identitäten zulässt.

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- API-Details finden Sie CreateIdentityPool in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CGIIdentityPool`

Tools für PowerShell V5

Beispiel 1: Löscht einen bestimmten Identitätspool.

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- API-Details finden Sie DeleteIdentityPool in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-CGIIdentityPoolRole`

Das folgende Codebeispiel zeigt die Verwendung. `Set-CGIIdentityPoolRole`

Tools für PowerShell V5

Beispiel 1: Konfiguriert den spezifischen Identitätspool so, dass er über eine nicht authentifizierte IAM-Rolle verfügt.

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- API-Details finden Sie SetIdentityPoolRoles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-CGIIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung. `Update-CGIIdentityPool`

Tools für PowerShell V5

Beispiel 1: Aktualisiert einige Eigenschaften des Identitätspools, in diesem Fall den Namen des Identitätspools.

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

Ausgabe:

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
HttpStatusCode                 : OK
```

- API-Details finden Sie UpdateIdentityPool in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS Config Beispiele mit Tools for PowerShell V5

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS Config

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-CFGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-CFGResourceTag`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ordnet das angegebene Tag dem Ressourcen-ARN zu, in diesem Fall config- rule/config -rule-16iyn0.

```
Add-CFGResourceTag -ResourceArn arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-16iyn0 -Tag @{Key="Release";Value="Beta"}
```

- API-Details finden Sie TagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateComplianceByConfigRuleList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateComplianceByConfigRuleList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ruft die Details aus der ConfigurationAggregator 'kaju'-Filterung für die angegebene Konfigurationsregel und die 'Konformität' expands/returns der Regel ab.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName kaju
 -Filters_ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK | Select-Object -
ExpandProperty Compliance
```

Ausgabe:

```
ComplianceContributorCount                            ComplianceType
--------------------------                            --------------
Amazon.ConfigService.Model.ComplianceContributorCount NON_COMPLIANT
```

Beispiel 2: Dieses Beispiel ruft Details aus der angegebenen Datei ab ConfigurationAggregator, filtert sie für das angegebene Konto für alle im Aggregator abgedeckten Regionen und gibt die Einhaltung aller Regeln weiter zurück.

```
Get-CFGAggregateComplianceByConfigRuleList -ConfigurationAggregatorName
 kaju -Filters_AccountId 123456789012 | Select-Object ConfigRuleName,
 @{N="Compliance";E={$_.Compliance.ComplianceType}}
```

Ausgabe:

```
ConfigRuleName                      Compliance
--------------                      ----------
ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK NON_COMPLIANT
ec2-instance-no-public-ip           NON_COMPLIANT
desired-instance-type               NON_COMPLIANT
```

- API-Details finden Sie DescribeAggregateComplianceByConfigRules in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateComplianceDetailsByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateComplianceDetailsByConfigRule`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die Bewertungsergebnisse zurück und wählt die Ausgabe mit Ressourcen-ID und Ressourcentyp für die AWS Konfigurationsregel 'desired-instance-type' aus, die sich für das angegebene Konto, den Aggregator, die Region und die Konfigurationsregel im Status 'COMPLIANT' befinden

```
Get-CFGAggregateComplianceDetailsByConfigRule -AccountId 123456789012 -
AwsRegion eu-west-1 -ComplianceType COMPLIANT -ConfigRuleName desired-
instance-type -ConfigurationAggregatorName raju | Select-Object -
```

```
ExpandProperty EvaluationResultIdentifier | Select-Object -ExpandProperty
 EvaluationResultQualifier
```

Ausgabe:

```
ConfigRuleName        ResourceId          ResourceType
--------------        ----------          ------------
desired-instance-type i-0f1bf2f34c5678d12 AWS::EC2::Instance
desired-instance-type i-0fd12dd3456789123 AWS::EC2::Instance
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). GetAggregateComplianceDetailsByConfigRule AWS -Tools für PowerShell

`Get-CFGAggregateConfigRuleComplianceSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateConfigRuleComplianceSummary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Anzahl der nicht konformen Regeln für den bestimmten Aggregator zurückgegeben.

```
(Get-CFGAggregateConfigRuleComplianceSummary -ConfigurationAggregatorName
 raju).AggregateComplianceCounts.ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       5
```

- API-Details finden Sie GetAggregateConfigRuleComplianceSummary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateDiscoveredResourceCount`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateDiscoveredResourceCount`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Ressourcenanzahl für den bestimmten Aggregator zurückgegeben, gefiltert nach der Region us-east-1.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1
```

Ausgabe:

```
GroupByKey GroupedResourceCounts NextToken TotalDiscoveredResources
---------- -------------------- --------- ------------------------
           {}                               455
```

Beispiel 2: In diesem Beispiel wird die Ressourcenanzahl gruppiert nach RESOURCE_TYPE für die gefilterte Region für den bestimmten Aggregator zurückgegeben.

```
Get-CFGAggregateDiscoveredResourceCount -ConfigurationAggregatorName Master -
Filters_Region us-east-1 -GroupByKey RESOURCE_TYPE |
   Select-Object -ExpandProperty GroupedResourceCounts
```

Ausgabe:

```
GroupName                        ResourceCount
---------                        -------------
AWS::CloudFormation::Stack       12
AWS::CloudFront::Distribution    1
AWS::CloudTrail::Trail           1
AWS::DynamoDB::Table             1
AWS::EC2::EIP                    2
AWS::EC2::FlowLog                2
AWS::EC2::InternetGateway        4
AWS::EC2::NatGateway             2
AWS::EC2::NetworkAcl             4
AWS::EC2::NetworkInterface       12
AWS::EC2::RouteTable             13
AWS::EC2::SecurityGroup          18
AWS::EC2::Subnet                 16
AWS::EC2::VPC                    4
AWS::EC2::VPCEndpoint            2
AWS::EC2::VPCPeeringConnection   1
```

```
AWS::IAM::Group                    2
AWS::IAM::Policy                   51
AWS::IAM::Role                     78
AWS::IAM::User                     7
AWS::Lambda::Function              3
AWS::RDS::DBSecurityGroup          1
AWS::S3::Bucket                    3
AWS::SSM::AssociationCompliance    107
AWS::SSM::ManagedInstanceInventory 108
```

- API-Details finden Sie GetAggregateDiscoveredResourceCounts in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateDiscoveredResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateDiscoveredResourceList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Ressourcen-IDs für den angegebenen Ressourcentyp zurückgegeben, aggregiert im Aggregator „Irland“. Die Liste der Ressourcentypen finden Sie unter? https://docs.aws.amazon.com/sdkfornet/v4/apidocs/index.html page= ConfigService/TConfigServiceResourceType.html Amazon_ConfigService_ResourceType &tocid=.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName Ireland -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSAutoScalingAutoScalingGroup)
```

Ausgabe:

```
ResourceId      : arn:aws:autoscaling:eu-
west-1:123456789012:autoScalingGroup:12e3b4fc-1234-1234-
a123-1d2ba3c45678:autoScalingGroupName/asg-1
ResourceName    : asg-1
ResourceType    : AWS::AutoScaling::AutoScalingGroup
SourceAccountId : 123456789012
SourceRegion    : eu-west-1
```

Beispiel 2: In diesem Beispiel wird der Ressourcentyp **`AwsEC2SecurityGroup`** namens „default“ für den bestimmten Aggregator zurückgegeben, gefiltert nach der Region us-east-1.

```
Get-CFGAggregateDiscoveredResourceList -ConfigurationAggregatorName raju -
ResourceType ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
Filters_Region us-east-1 -Filters_ResourceName default
```

Ausgabe:

```
ResourceId      : sg-01234bd5dbfa67c89
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-0123a4ebbf56789be
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1

ResourceId      : sg-4fc1d234
ResourceName    : default
ResourceType    : AWS::EC2::SecurityGroup
SourceAccountId : 123456789102
SourceRegion    : us-east-1
```

- API-Details finden Sie ListAggregateDiscoveredResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateResourceConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateResourceConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Konfigurationselement für die angegebene Ressource zusammengefasst zurückgegeben und die Konfiguration erweitert.

```
(Get-CFGAggregateResourceConfig -ResourceIdentifier_SourceRegion
 us-east-1 -ResourceIdentifier_SourceAccountId 123456789012 -
ResourceIdentifier_ResourceId sg-4fc1d234 -ResourceIdentifier_ResourceType
 ([Amazon.ConfigService.ResourceType]::AWSEC2SecurityGroup) -
ConfigurationAggregatorName raju).Configuration | ConvertFrom-Json
```

Ausgabe:

```
{"description":"default VPC security group","groupName":"default","ipPermissions":
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[{"groupId":"sg-4fc1d234","userId":"123456789012"}],"ipv4Ranges":
[],"ipRanges":[]},{"fromPort":3389,"ipProtocol":"tcp","ipv6Ranges":
[],"prefixListIds":[],"toPort":3389,"userIdGroupPairs":[],"ipv4Ranges":
[{"cidrIp":"54.240.197.224/29","description":"office subnet"},
{"cidrIp":"72.21.198.65/32","description":"home pc"}],"ipRanges":
["54.240.197.224/29","72.21.198.65/32"]}],"ownerId":"123456789012","groupId":"sg-4fc1d234","
[{"ipProtocol":"-1","ipv6Ranges":[],"prefixListIds":[],"userIdGroupPairs":
[],"ipv4Ranges":[{"cidrIp":"0.0.0.0/0"}],"ipRanges":["0.0.0.0/0"]}],"tags":
[],"vpcId":"vpc-2d1c2e34"}
```

- API-Details finden Sie GetAggregateResourceconfig-service in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregateResourceConfigBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregateResourceConfigBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das aktuelle Konfigurationselement für die Ressource (identifiziert) abgerufen, die im angegebenen Aggregator vorhanden ist.

```
$resIdentifier=[Amazon.ConfigService.Model.AggregateResourceIdentifier]@{
  ResourceId= "i-012e3cb4df567e8aa"
  ResourceName = "arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa"
  ResourceType = [Amazon.ConfigService.ResourceType]::AWSEC2Instance
  SourceAccountId = "123456789012"
  SourceRegion = "eu-west-1"
 }

 Get-CFGAggregateResourceConfigBatch -ResourceIdentifier $resIdentifier -
ConfigurationAggregatorName raju
```

Ausgabe:

```
BaseConfigurationItems UnprocessedResourceIdentifiers
---------------------- -----------------------------
```

```
{}                      {arn:aws:ec2:eu-west-1:123456789012:instance/
i-012e3cb4df567e8aa}
```

- API-Details finden Sie BatchGetAggregateResourceconfig-service in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGAggregationAuthorizationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGAggregationAuthorizationList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Autorisierungen abgerufen, die Aggregatoren erteilt wurden.

```
Get-CFGAggregationAuthorizationList
```

Ausgabe:

```
AggregationAuthorizationArn
   AuthorizedAccountId AuthorizedAwsRegion CreationTime
---------------------------
   ------------------- ------------------- ------------
arn:aws:config-service:eu-west-1:123456789012:aggregation-
authorization/123456789012/eu-west-1 123456789012        eu-west-1
 8/26/2019 12:55:27 AM
```

- API-Details finden Sie DescribeAggregationAuthorizations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceByConfigRule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Konformitätsdetails für die Regel „ebs-optimized-instance“ abgerufen, für die derzeit keine Bewertungsergebnisse vorliegen. Daher wird „INSUFFICIENT_DATA“ zurückgegeben

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ebs-optimized-instance).Compliance
```

Ausgabe:

```
ComplianceContributorCount ComplianceType
------------------------- --------------
                           INSUFFICIENT_DATA
```

Beispiel 2: In diesem Beispiel wird die Anzahl der nicht konformen Ressourcen für die Regel „ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK“ zurückgegeben.

```
(Get-CFGComplianceByConfigRule -ConfigRuleName ALB_HTTP_TO_HTTPS_REDIRECTION_CHECK -
ComplianceType NON_COMPLIANT).Compliance.ComplianceContributorCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       2
```

- API-Details finden Sie DescribeComplianceByConfigRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceByResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceByResource`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Ressourcentyp **`AWS::SSM::ManagedInstanceInventory`** auf den Konformitätstyp „COMPLIANT“ überprüft.

```
Get-CFGComplianceByResource -ComplianceType COMPLIANT -ResourceType
 AWS::SSM::ManagedInstanceInventory
```

Ausgabe:

```
Compliance                            ResourceId          ResourceType
----------                            ----------          ------------
Amazon.ConfigService.Model.Compliance i-0123bcf4b567890e3
 AWS::SSM::ManagedInstanceInventory
```

```
Amazon.ConfigService.Model.Compliance i-0a1234f6f5d6b78f7
 AWS::SSM::ManagedInstanceInventory
```

- API-Details finden Sie DescribeComplianceByResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceDetailsByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceDetailsByConfigRule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Auswertungsergebnisse für die Regel access-keys-rotated abgerufen und die Ausgabe nach Compliance-Typ gruppiert zurückgegeben

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-keys-rotated | Group-
Object ComplianceType
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    2 COMPLIANT                 {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult}
    5 NON_COMPLIANT             {Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationResult,
 Amazon.ConfigService.Model.EvaluationRes...
```

Beispiel 2: In diesem Beispiel werden Konformitätsdetails für die Regel access-keys-rotated für COMPLIANT-Ressourcen abgefragt.

```
Get-CFGComplianceDetailsByConfigRule -ConfigRuleName access-
keys-rotated -ComplianceType COMPLIANT | ForEach-Object
 {$_.EvaluationResultIdentifier.EvaluationResultQualifier}
```

Ausgabe:

```
ConfigRuleName      ResourceId            ResourceType
--------------      ----------            ------------
```

```
access-keys-rotated BCAB1CDJ2LITAPVEW3JAH AWS::IAM::User
access-keys-rotated BCAB1CDJ2LITL3EHREM4Q AWS::IAM::User
```

- API-Details finden Sie GetComplianceDetailsByConfigRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceDetailsByResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceDetailsByResource`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel für die Auswertungsergebnisse für die angegebene Ressource.

```
Get-CFGComplianceDetailsByResource -ResourceId ABCD5STJ4EFGHIVEW6JAH -ResourceType
 'AWS::IAM::User'
```

Ausgabe:

```
Annotation                 :
ComplianceType             : COMPLIANT
ConfigRuleInvokedTime      : 8/25/2019 11:34:56 PM
EvaluationResultIdentifier : Amazon.ConfigService.Model.EvaluationResultIdentifier
ResultRecordedTime         : 8/25/2019 11:34:56 PM
ResultToken                :
```

- API-Details finden Sie GetComplianceDetailsByResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceSummaryByConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceSummaryByConfigRule`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die Anzahl der Config-Regeln zurück, die nicht konform sind.

```
Get-CFGComplianceSummaryByConfigRule -Select
 ComplianceSummary.NonCompliantResourceCount
```

Ausgabe:

```
CapExceeded CappedCount
----------- -----------
False       9
```

- API-Details finden Sie GetComplianceSummaryByConfigRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGComplianceSummaryByResourceType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGComplianceSummaryByResourceType`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die Anzahl der Ressourcen zurück, die konform oder nicht konform sind, und konvertiert die Ausgabe in JSON.

```
Get-CFGComplianceSummaryByResourceType -Select
 ComplianceSummariesByResourceType.ComplianceSummary | ConvertTo-Json
{
  "ComplianceSummaryTimestamp": "2019-12-14T06:14:49.778Z",
  "CompliantResourceCount": {
    "CapExceeded": false,
    "CappedCount": 2
  },
  "NonCompliantResourceCount": {
    "CapExceeded": true,
    "CappedCount": 100
  }
}
```

- API-Details finden Sie GetComplianceSummaryByResourceType in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigRule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Konfigurationsregeln für das Konto mit ausgewählten Eigenschaften aufgeführt.

```
Get-CFGConfigRule | Select-Object ConfigRuleName, ConfigRuleId, ConfigRuleArn,
 ConfigRuleState
```

Ausgabe:

```
ConfigRuleName                                     ConfigRuleId       ConfigRuleArn
                                                       ConfigRuleState
--------------                                     ------------       -------------
                                                       ---------------
ALB_REDIRECTION_CHECK                              config-rule-12iyn3 arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-12iyn3 ACTIVE
access-keys-rotated                                config-rule-aospfr arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-aospfr ACTIVE
autoscaling-group-elb-healthcheck-required         config-rule-cn1f2x arn:aws:config-
service:eu-west-1:123456789012:config-rule/config-rule-cn1f2x ACTIVE
```

- API-Details finden Sie DescribeConfigRules in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigRuleEvaluationStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigRuleEvaluationStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Statusinformationen für die angegebenen Konfigurationsregeln zurückgegeben.

```
Get-CFGConfigRuleEvaluationStatus -ConfigRuleName root-account-mfa-enabled, vpc-
flow-logs-enabled
```

Ausgabe:

```
ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-kvq1wk
ConfigRuleId                 : config-rule-kvq1wk
```

```
ConfigRuleName               : root-account-mfa-enabled
FirstActivatedTime           : 8/27/2019 8:05:17 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 8:12:03 AM
LastSuccessfulInvocationTime : 12/13/2019 8:12:03 AM

ConfigRuleArn                : arn:aws:config:eu-west-1:123456789012:config-rule/
config-rule-z1s23b
ConfigRuleId                 : config-rule-z1s23b
ConfigRuleName               : vpc-flow-logs-enabled
FirstActivatedTime           : 8/14/2019 6:23:44 AM
FirstEvaluationStarted       : True
LastErrorCode                :
LastErrorMessage             :
LastFailedEvaluationTime     : 1/1/0001 12:00:00 AM
LastFailedInvocationTime     : 1/1/0001 12:00:00 AM
LastSuccessfulEvaluationTime : 12/13/2019 7:12:01 AM
LastSuccessfulInvocationTime : 12/13/2019 7:12:01 AM
```

- API-Details finden Sie DescribeConfigRuleEvaluationStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigurationAggregatorList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationAggregatorList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt alle Aggregatoren für den zurück. region/account

```
Get-CFGConfigurationAggregatorList
```

Ausgabe:

```
AccountAggregationSources     :
 {Amazon.ConfigService.Model.AccountAggregationSource}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-xabca1me
```

```
ConfigurationAggregatorName   : IrelandMaster
CreationTime                  : 8/25/2019 11:42:39 PM
LastUpdatedTime               : 8/25/2019 11:42:39 PM
OrganizationAggregationSource :

AccountAggregationSources     : {}
ConfigurationAggregatorArn    : arn:aws:config-service:eu-
west-1:123456789012:config-aggregator/config-aggregator-qubqabcd
ConfigurationAggregatorName   : raju
CreationTime                  : 8/11/2019 8:39:25 AM
LastUpdatedTime               : 8/11/2019 8:39:25 AM
OrganizationAggregationSource :
 Amazon.ConfigService.Model.OrganizationAggregationSource
```

- API-Details finden Sie DescribeConfigurationAggregators in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigurationAggregatorSourcesStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationAggregatorSourcesStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden angeforderte Felder für die Quellen im angegebenen Aggregator angezeigt.

```
Get-CFGConfigurationAggregatorSourcesStatus -ConfigurationAggregatorName raju |
 select SourceType, LastUpdateStatus, LastUpdateTime, SourceId
```

Ausgabe:

```
SourceType   LastUpdateStatus LastUpdateTime        SourceId
----------   ---------------- --------------        --------
ORGANIZATION SUCCEEDED        12/31/2019 7:45:06 AM Organization
ACCOUNT      SUCCEEDED        12/31/2019 7:09:38 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:12:53 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:18:10 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:17 AM 933301234567
ACCOUNT      SUCCEEDED        12/31/2019 7:25:49 AM 612641234567
ACCOUNT      SUCCEEDED        12/31/2019 7:26:11 AM 612641234567
```

- API-Details finden Sie DescribeConfigurationAggregatorSourcesStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigurationRecorder`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationRecorder`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details von Konfigurations-Recorders zurückgegeben.

```
Get-CFGConfigurationRecorder | Format-List
```

Ausgabe:

```
Name           : default
RecordingGroup : Amazon.ConfigService.Model.RecordingGroup
RoleARN        : arn:aws:iam::123456789012:role/aws-service-role/
config.amazonaws.com/AWSServiceRoleForConfig
```

- API-Details finden Sie DescribeConfigurationRecorders in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConfigurationRecorderStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConfigurationRecorderStatus`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt den Status der Konfigurations-Recorder zurück.

```
Get-CFGConfigurationRecorderStatus
```

Ausgabe:

```
LastErrorCode        :
LastErrorMessage     :
LastStartTime        : 10/11/2019 10:13:51 AM
LastStatus           : Success
```

```
LastStatusChangeTime : 12/31/2019 6:14:12 AM
LastStopTime         : 10/11/2019 10:13:46 AM
Name                 : default
Recording            : True
```

- API-Details finden Sie DescribeConfigurationRecorderStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGConformancePack`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Conformance Packs aufgeführt.

```
Get-CFGConformancePack
```

Ausgabe:

```
ConformancePackArn             : arn:aws:config:eu-west-1:123456789012:conformance-
pack/dono/conformance-pack-p0acq8bpz
ConformancePackId              : conformance-pack-p0acabcde
ConformancePackInputParameters : {}
ConformancePackName            : dono
CreatedBy                      :
DeliveryS3Bucket               : kt-ps-examples
DeliveryS3KeyPrefix            :
LastUpdateRequestedTime        : 12/31/2019 8:45:31 AM
```

- API-Details finden Sie DescribeConformancePacks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGDeliveryChannel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGDeliveryChannel`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Lieferkanal für die Region abgerufen und Details angezeigt.

```
Get-CFGDeliveryChannel -Region eu-west-1 | Select-Object Name, S3BucketName,
 S3KeyPrefix,
 @{N="DeliveryFrequency";E={$_.ConfigSnapshotDeliveryProperties.DeliveryFrequency}}
```

Ausgabe:

```
Name    S3BucketName               S3KeyPrefix DeliveryFrequency
----    ------------               ----------- -----------------
default config-bucket-NA my          TwentyFour_Hours
```

- API-Details finden Sie DescribeDeliveryChannels in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-CFGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-CFGResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die zugehörigen Tags für die angegebene Ressource aufgelistet

```
Get-CFGResourceTag -ResourceArn $rules[0].ConfigRuleArn
```

Ausgabe:

```
Key     Value
---     -----
Version 1.3
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-CFGConformancePack`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Konformitätspaket zusammen mit allen Regeln, Behebungsmaßnahmen und Bewertungsergebnissen für das Paket entfernt.

```
Remove-CFGConformancePack -ConformancePackName dono
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-CFGConformancePack (DeleteConformancePack)" on
 target "dono".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteConformancePack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-CFGConformancePack`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CFGConformancePack`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Konformitätspaket erstellt, wobei die Vorlage aus der angegebenen Yaml-Datei abgerufen wird.

```
Write-CFGConformancePack -ConformancePackName dono -DeliveryS3Bucket amzn-s3-demo-
bucket -TemplateBody (Get-Content C:\windows\temp\template.yaml -Raw)
```

- API-Details finden Sie PutConformancePack in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-CFGDeliveryChannel`

Das folgende Codebeispiel zeigt die Verwendung. `Write-CFGDeliveryChannel`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die deliveryFrequency-Eigenschaft eines vorhandenen Lieferkanals geändert.

```
Write-CFGDeliveryChannel -ConfigSnapshotDeliveryProperties_DeliveryFrequency
 TwentyFour_Hours -DeliveryChannelName default -DeliveryChannel_S3BucketName amzn-
s3-demo-bucket -DeliveryChannel_S3KeyPrefix my
```

- API-Details finden Sie PutDeliveryChannel in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Gerätefarmen, die Tools für V5 verwenden PowerShell

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Device Farm Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`New-DFUpload`

Das folgende Codebeispiel zeigt die Verwendung`New-DFUpload`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein AWS Device Farm-Upload für eine Android-App erstellt. Sie können den Projekt-ARN aus der Ausgabe von New-DFProject oder abrufen Get-DFProjectList. Verwenden Sie die signierte URL in der New-DFUpload Ausgabe, um eine Datei auf Device Farm hochzuladen.

```
New-DFUpload -ContentType "application/octet-stream" -ProjectArn
 "arn:aws:devicefarm:us-west-2:123456789012:project:EXAMPLEa-7ec1-4741-9c1f-
d3e04EXAMPLE" -Name "app.apk" -Type ANDROID_APP
```

- API-Details finden Sie CreateUpload in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Directory Service Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. Directory Service

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-DSIpRoute`

Das folgende Codebeispiel zeigt die Verwendung`Add-DSIpRoute`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt das Ressourcen-Tag, das dem angegebenen zugewiesen ist Directory-id

```
Add-DSIpRoute -DirectoryId d-123456ijkl -IpRoute @{CidrIp ="203.0.113.5/32"} -
UpdateSecurityGroupForDirectoryController $true
```

- API-Details finden Sie AddIpRoutes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-DSResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl fügt das Resource Tag dem angegebenen Tag hinzu Directory-id

```
Add-DSResourceTag -ResourceId d-123456ijkl -Tag @{Key="myTag"; Value="mytgValue"}
```

- API-Details finden Sie AddTagsToResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Approve-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Approve-DSTrust`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der AWS VerifyTrust Verzeichnisdienst-API-Vorgang für die angegebene Trustid aufgerufen.

```
Approve-DSTrust -TrustId t-9067157123
```

- API-Details finden Sie VerifyTrust in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Confirm-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Confirm-DSSharedDirectory`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel akzeptiert eine Anfrage zur gemeinsamen Nutzung eines Verzeichnisses, die vom Verzeichnisbesitzer AWS-Konto gesendet wurde.

```
Confirm-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Ausgabe:

```
CreatedDateTime     : 12/30/2019 4:20:27 AM
```

```
LastUpdatedDateTime : 12/30/2019 4:21:40 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod         :
ShareNotes          : This is test sharing
ShareStatus         : Sharing
```

- API-Details finden Sie AcceptSharedDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Connect-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Connect-DSDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein AD Connector für die Verbindung zu einem lokalen Verzeichnis erstellt.

```
Connect-DSDirectory -Name contoso.com -ConnectSettings_CustomerUserName
 Administrator -Password $Password -ConnectSettings_CustomerDnsIp 172.31.36.96
 -ShortName CONTOSO -Size Small -ConnectSettings_VpcId vpc-123459da -
ConnectSettings_SubnetId subnet-1234ccaa, subnet-5678ffbb
```

- API-Details finden Sie ConnectDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Deny-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Deny-DSSharedDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Verzeichnisfreigabeanforderung, die vom Konto des Verzeichniseigentümers gesendet wurde, abgelehnt.

```
Deny-DSSharedDirectory -SharedDirectoryId d-9067012345
```

Ausgabe:

```
d-9067012345
```

- API-Details finden Sie RejectSharedDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-DSDirectoryShare`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSDirectoryShare`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Verzeichnisfreigabe für das Verzeichniseigentümer- und -verbraucherkonto gestoppt.

```
Disable-DSDirectoryShare -DirectoryId d-123456ijkl -UnshareTarget_Id 123456784321 -
UnshareTarget_Type ACCOUNT
```

Ausgabe:

```
d-9067012345
```

- API-Details finden Sie UnshareDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-DSLDAPS`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSLDAPS`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden sichere LDAP-Aufrufe für das angegebene Verzeichnis deaktiviert.

```
Disable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Weitere API-Informationen finden Sie unter DisableLDAPS in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Disable-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung`Disable-DSRadius`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der RADIUS-Server deaktiviert, der für einen AD Connector oder ein Microsoft AD-Verzeichnis konfiguriert ist.

```
Disable-DSRadius -DirectoryId d-123456ijkl
```

- API-Details finden Sie DisableRadius in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-DSSso`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-DSSso`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird Single Sign-On für ein Verzeichnis deaktiviert.

```
Disable-DSSso -DirectoryId d-123456ijkl
```

- API-Details finden Sie DisableSso in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-DSDirectoryShare`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSDirectoryShare`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel teilt ein bestimmtes Verzeichnis in Ihrem AWS Konto mithilfe der Handshake-Methode mit einem anderen AWS Konto.

```
Enable-DSDirectoryShare -DirectoryId d-123456ijkl -ShareTarget_Id 123456784321 -
ShareMethod HANDSHAKE -ShareTarget_Type ACCOUNT
```

Ausgabe:

```
d-9067012345
```

- API-Details finden Sie ShareDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-DSLDAPS`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSLDAPS`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Schalter für das jeweilige Verzeichnis aktiviert, um immer sichere LDAP-Aufrufe zu verwenden.

```
Enable-DSLDAPS -DirectoryId d-123456ijkl -Type Client
```

- Weitere API-Informationen finden Sie unter EnableLDAPS in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Enable-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung`Enable-DSRadius`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Multi-Faktor-Authentifizierung (MFA) mit der bereitgestellten RADIUS-Serverkonfiguration für einen AD Connector oder ein Microsoft AD-Verzeichnis aktiviert.

```
Enable-DSRadius -DirectoryId d-123456ijkl
-RadiusSettings_AuthenticationProtocol PAP
-RadiusSettings_DisplayLabel Radius
-RadiusSettings_RadiusPort 1812
-RadiusSettings_RadiusRetry 4
-RadiusSettings_RadiusServer 10.4.185.113
-RadiusSettings_RadiusTimeout 50
-RadiusSettings_SharedSecret wJalrXUtnFEMI
```

- API-Details finden Sie EnableRadius in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-DSSso`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-DSSso`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird Single Sign-On für ein Verzeichnis aktiviert.

```
Enable-DSSso -DirectoryId d-123456ijkl
```

- API-Details finden Sie EnableSso in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über das Zertifikat angezeigt, das für eine gesicherte LDAP-Verbindung registriert ist.

```
Get-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

Ausgabe:

```
CertificateId      : c-906731e34f
CommonName         : contoso-EC2AMAZ-CTGG2NM-CA
ExpiryDateTime     : 4/15/2025 6:34:15 PM
RegisteredDateTime : 4/15/2020 6:38:56 PM
State              : Registered
StateReason        : Certificate registered successfully.
```

- API-Details finden Sie DescribeCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSCertificateList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Zertifikate aufgelistet, die für eine gesicherte LDAP-Verbindung für ein angegebenes Verzeichnis registriert sind.

```
Get-DSCertificateList -DirectoryId d-123456ijkl
```

Ausgabe:

```
CertificateId CommonName                 ExpiryDateTime       State
------------- ----------                 --------------       -----
c-906731e34f  contoso-EC2AMAZ-CTGG2NM-CA 4/15/2025 6:34:15 PM Registered
```

- API-Details finden Sie ListCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSConditionalForwarder`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden alle konfigurierten Conditional Forwarders abgerufen. Directory-id

```
Get-DSConditionalForwarder -DirectoryId d-123456ijkl
```

Ausgabe:

```
DnsIpAddrs      RemoteDomainName ReplicationScope
----------      ---------------- ----------------
{172.31.77.239} contoso.com      Domain
```

- API-Details finden Sie DescribeConditionalForwarders in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDirectory`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft Informationen über die Verzeichnisse ab, die zu diesem Konto gehören.

```
Get-DSDirectory | Select-Object DirectoryId, Name, DnsIpAddrs, Type
```

Ausgabe:

```
DirectoryId  Name           DnsIpAddrs                     Type
-----------  ----           ----------                     ----
d-123456abcd abcd.example.com {172.31.74.189, 172.31.13.145} SimpleAD
d-123456efgh wifi.example.com {172.31.16.108, 172.31.10.56}  ADConnector
d-123456ijkl lan2.example.com {172.31.10.56, 172.31.16.108}  MicrosoftAD
```

- API-Details finden Sie DescribeDirectories in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSDirectoryLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDirectoryLimit`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Informationen über das Verzeichnislimit für die Region us-east-1 angezeigt.

```
Get-DSDirectoryLimit -Region us-east-1
```

Ausgabe:

```
CloudOnlyDirectoriesCurrentCount : 1
CloudOnlyDirectoriesLimit        : 10
CloudOnlyDirectoriesLimitReached : False
CloudOnlyMicrosoftADCurrentCount : 1
CloudOnlyMicrosoftADLimit        : 20
CloudOnlyMicrosoftADLimitReached : False
ConnectedDirectoriesCurrentCount : 1
ConnectedDirectoriesLimit        : 10
```

- API-Details finden Sie GetDirectoryLimits in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSDomainControllerList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSDomainControllerList`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird die detaillierte Liste der Domaincontroller abgerufen, die für die angegebene directory-id gestartet wurden

```
Get-DSDomainControllerList -DirectoryId d-123456ijkl
```

Ausgabe:

```
AvailabilityZone          : us-east-1b
DirectoryId               : d-123456ijkl
DnsIpAddr                 : 172.31.16.108
DomainControllerId        : dc-1234567aa6
LaunchTime                : 4/4/2019 4:53:43 AM
Status                    : Active
StatusLastUpdatedDateTime : 4/24/2019 1:37:54 PM
StatusReason              :
SubnetId                  : subnet-1234kkaa
VpcId                     : vpc-123459d

AvailabilityZone          : us-east-1d
DirectoryId               : d-123456ijkl
DnsIpAddr                 : 172.31.10.56
DomainControllerId        : dc-1234567aa7
LaunchTime                : 4/4/2019 4:53:43 AM
Status                    : Active
StatusLastUpdatedDateTime : 4/4/2019 5:14:31 AM
StatusReason              :
SubnetId                  : subnet-5678ffbb
VpcId                     : vpc-123459d
```

- API-Details finden Sie DescribeDomainControllers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSEventTopic`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl zeigt Informationen zum konfigurierten SNS-Thema als Benachrichtigung an, wenn sich der Verzeichnisstatus ändert.

```
Get-DSEventTopic -DirectoryId d-123456ijkl
```

Ausgabe:

```
CreatedDateTime : 12/13/2019 11:15:32 AM
DirectoryId     : d-123456ijkl
Status          : Registered
TopicArn        : arn:aws:sns:us-east-1:123456781234:snstopicname
TopicName       : snstopicname
```

- API-Details finden Sie DescribeEventTopics in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSIpRouteList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSIpRouteList`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden die im Verzeichnis-IP-Routing konfigurierten öffentlichen IP-Adressblöcke abgerufen

```
Get-DSIpRouteList -DirectoryId d-123456ijkl
```

Ausgabe:

```
AddedDateTime       : 12/13/2019 12:27:22 PM
CidrIp              : 203.0.113.5/32
Description         : Public IP of On-Prem DNS Server
DirectoryId         : d-123456ijkl
IpRouteStatusMsg    : Added
IpRouteStatusReason :
```

- API-Details finden Sie ListIpRoutes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSLDAPSSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSLDAPSSetting`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status der LDAP-Sicherheit für das angegebene Verzeichnis beschrieben.

```
Get-DSLDAPSSetting -DirectoryId d-123456ijkl
```

Ausgabe:

```
LastUpdatedDateTime  LDAPSStatus LDAPSStatusReason
-------------------  ----------- -----------------
4/15/2020 6:51:03 PM Enabled     LDAPS is enabled successfully.
```

- Weitere API-Informationen finden Sie unter DescribeLDAPSSettings in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Get-DSLogSubscriptionList`

Das folgende Codebeispiel zeigt die Verwendung`Get-DSLogSubscriptionList`.

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden die Protokollabonnementinformationen der angegebenen directory-id abgerufen

```
Get-DSLogSubscriptionList -DirectoryId d-123456ijkl
```

Ausgabe:

```
DirectoryId  LogGroupName
 SubscriptionCreatedDateTime
-----------  ------------
 ---------------------------
d-123456ijkl /aws/directoryservice/d-123456ijkl-lan2.example.com 12/14/2019 9:05:23
 AM
```

- API-Details finden Sie ListLogSubscriptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSResourceTag`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden alle Tags des angegebenen Verzeichnisses abgerufen.

```
Get-DSResourceTag -ResourceId d-123456ijkl
```

Ausgabe:

```
Key   Value
---   -----
myTag myTagValue
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSchemaExtension`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Schemaerweiterungen, die auf ein Microsoft AD-Verzeichnis angewendet werden, aufgelistet.

```
Get-DSSchemaExtension -DirectoryId d-123456ijkl
```

Ausgabe:

```
Description                 : ManagedADSchemaExtension
DirectoryId                 : d-123456ijkl
EndDateTime                 : 4/12/2020 10:30:49 AM
SchemaExtensionId           : e-9067306643
```

```
SchemaExtensionStatus       : Completed
SchemaExtensionStatusReason : Schema updates are complete.
StartDateTime               : 4/12/2020 10:28:42 AM
```

- API-Details finden Sie ListSchemaExtensions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSSharedDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSharedDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die geteilten Verzeichnisse Ihres Kontos abgerufen AWS

```
Get-DSSharedDirectory -OwnerDirectoryId d-123456ijkl -SharedDirectoryId d-9067012345
```

Ausgabe:

```
CreatedDateTime     : 12/30/2019 4:34:37 AM
LastUpdatedDateTime : 12/30/2019 4:35:22 AM
OwnerAccountId      : 123456781234
OwnerDirectoryId    : d-123456ijkl
SharedAccountId     : 123456784321
SharedDirectoryId   : d-9067012345
ShareMethod         : HANDSHAKE
ShareNotes          : This is a test Sharing
ShareStatus         : Shared
```

- API-Details finden Sie DescribeSharedDirectories in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSnapshot`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft Informationen über die angegebenen Verzeichnis-Snapshots ab, die zu diesem Konto gehören.

```
Get-DSSnapshot -DirectoryId d-123456ijkl
```

Ausgabe:

```
DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bd1234
StartTime   : 12/13/2019 6:33:01 PM
Status      : Completed
Type        : Auto

DirectoryId : d-123456ijkl
Name        :
SnapshotId  : s-9064bb4321
StartTime   : 12/9/2019 9:48:11 PM
Status      : Completed
Type        : Auto
```

- API-Details finden Sie DescribeSnapshots in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSSnapshotLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSSnapshotLimit`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden die manuellen Snapshot-Limits für ein angegebenes Verzeichnis abgerufen.

```
Get-DSSnapshotLimit -DirectoryId d-123456ijkl
```

Ausgabe:

```
ManualSnapshotsCurrentCount ManualSnapshotsLimit ManualSnapshotsLimitReached
--------------------------- -------------------- ---------------------------
0                           5                    False
```

- API-Details finden Sie GetSnapshotLimits in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DSTrust`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft die Informationen zu den Vertrauensbeziehungen ab, die für die angegebene directory-id erstellt wurden.

```
Get-DSTrust -DirectoryId d-123456abcd
```

Ausgabe:

```
CreatedDateTime          : 7/5/2019 4:55:42 AM
DirectoryId              : d-123456abcd
LastUpdatedDateTime      : 7/5/2019 4:56:04 AM
RemoteDomainName         : contoso.com
SelectiveAuth            : Disabled
StateLastUpdatedDateTime : 7/5/2019 4:56:04 AM
TrustDirection           : One-Way: Incoming
TrustId                  : t-9067157123
TrustState               : Created
TrustStateReason         :
TrustType                : Forest
```

- API-Details finden Sie DescribeTrusts in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Alias für ein Verzeichnis erstellt und der Alias der angegebenen directory-id zugewiesen.

```
New-DSAlias -DirectoryId d-123456ijkl -Alias MyOrgName
```

Ausgabe:

```
Alias     DirectoryId
-----     -----------
myorgname d-123456ijkl
```

- API-Details finden Sie CreateAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-DSComputer

Das folgende Codebeispiel zeigt die Verwendung. `New-DSComputer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Active-Directory-Computerobjekt erstellt.

```
New-DSComputer -DirectoryId d-123456ijkl -ComputerName ADMemberServer -Password
 $Password
```

Ausgabe:

```
ComputerAttributes                  ComputerId
 ComputerName
------------------                  ----------
 ------------
{WindowsSamName, DistinguishedName} S-1-5-21-1191241402-978882507-2717148213-1662
 ADMemberServer
```

- API-Details finden Sie CreateComputer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-DSConditionalForwarder

Das folgende Codebeispiel zeigt die Verwendung. `New-DSConditionalForwarder`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Conditional Forwarder in specified erstellt. AWS Directory-id

```
New-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.10.56 -RemoteDomainName contoso.com
```

- API-Details finden Sie CreateConditionalForwarder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Simple-AD-Verzeichnis erstellt.

```
New-DSDirectory -Name corp.example.com -Password $Password -Size Small -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API-Details finden Sie CreateDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSLogSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSLogSubscription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Abonnement erstellt, um die Sicherheitsprotokolle des Directory Service-Domänencontrollers in Echtzeit an die angegebene CloudWatch Amazon-Protokollgruppe in Ihrem AWS-Konto weiterzuleiten.

```
New-DSLogSubscription -DirectoryId d-123456ijkl -LogGroupName /aws/directoryservice/
d-123456ijkl-lan2.example.com
```

- API-Details finden Sie CreateLogSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSMicrosoftAD`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSMicrosoftAD`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Microsoft AD-Verzeichnis in AWS Cloud erstellt.

```
New-DSMicrosoftAD -Name corp.example.com -Password $Password -edition Standard -
VpcSettings_VpcId vpc-123459d -VpcSettings_SubnetIds subnet-1234kkaa,subnet-5678ffbb
```

- API-Details finden Sie unter CreateMicrosoft AD in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSSnapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Verzeichnis-Snapshot erstellt

```
New-DSSnapshot -DirectoryId d-123456ijkl
```

- API-Details finden Sie CreateSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `New-DSTrust`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Two-Way gesamtheitliche Vertrauensstellung zwischen Ihrem AWS verwalteten Microsoft AD-Verzeichnis und dem vorhandenen lokalen Microsoft Active Directory hergestellt.

```
New-DSTrust -DirectoryId d-123456ijkl -RemoteDomainName contoso.com -TrustDirection
 Two-Way -TrustType Forest -TrustPassword $Password -ConditionalForwarderIpAddr
 172.31.36.96
```

Ausgabe:

```
t-9067157123
```

- API-Details finden Sie CreateTrust in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Register-DSCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Zertifikat für eine gesicherte LDAP-Verbindung registriert.

```
$Certificate = Get-Content contoso.cer -Raw
Register-DSCertificate -DirectoryId d-123456ijkl -CertificateData $Certificate
```

Ausgabe:

```
c-906731e350
```

- API-Details finden Sie RegisterCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Register-DSEventTopic`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Verzeichnis als Publisher einem SNS-Thema zugeordnet.

```
Register-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API-Details finden Sie RegisterEventTopic in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSConditionalForwarder`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Conditional Forwarder entfernt, der für Ihr AWS Verzeichnis eingerichtet wurde.

```
Remove-DSConditionalForwarder -DirectoryId d-123456ijkl -RemoteDomainName
 contoso.com
```

- API-Details finden Sie DeleteConditionalForwarder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSDirectory`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel löscht ein AWS Verzeichnisdienstverzeichnis (Simple AD/Microsoft AD/AD Connector)

```
Remove-DSDirectory -DirectoryId d-123456ijkl
```

- API-Details finden Sie DeleteDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSIpRoute`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSIpRoute`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt die angegebene IP aus den konfigurierten IP-Routen von Directory-id.

```
Remove-DSIpRoute -DirectoryId d-123456ijkl -CidrIp 203.0.113.5/32
```

- API-Details finden Sie RemoveIpRoutes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSLogSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSLogSubscription`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt das Protokollabonnement der angegebenen Directory-id

```
Remove-DSLogSubscription -DirectoryId d-123456ijkl
```

- API-Details finden Sie DeleteLogSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt das Ressourcen-Tag, das dem angegebenen zugewiesen ist Directory-id

```
Remove-DSResourceTag -ResourceId d-123456ijkl -TagKey myTag
```

- API-Details finden Sie RemoveTagsFromResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSSnapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der manuell erstellte Snapshot entfernt.

```
Remove-DSSnapshot -SnapshotId s-9068b488kc
```

- API-Details finden Sie DeleteSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DSTrust`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die bestehende Vertrauensstellung zwischen Ihrem AWS Managed AD Directory und einer externen Domain entfernt.

```
Get-DSTrust -DirectoryId d-123456ijkl -Select Trusts.TrustId | Remove-DSTrust
```

Ausgabe:

```
t-9067157123
```

- API-Details finden Sie DeleteTrust in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-DSUserPassword`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-DSUserPassword`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Passwort des Active Directory-Benutzers ADUser in AWS Managed Microsoft AD oder Simple AD Directory zurückgesetzt

```
Reset-DSUserPassword -UserName ADuser -DirectoryId d-123456ijkl -NewPassword
 $Password
```

- API-Details finden Sie ResetUserPassword in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Restore-DSFromSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Restore-DSFromSnapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Verzeichnis mithilfe eines vorhandenen Verzeichnis-Snapshots wiederhergestellt.

```
Restore-DSFromSnapshot -SnapshotId s-9068b488kc
```

- API-Details finden Sie RestoreFromSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-DSDomainControllerCount`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DSDomainControllerCount`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Anzahl der Domaincontroller für die angegebene directory-id auf 3 gesetzt.

```
Set-DSDomainControllerCount -DirectoryId d-123456ijkl -DesiredNumber 3
```

- API-Details finden Sie UpdateNumberOfDomainControllers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Start-DSSchemaExtension`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Schemaerweiterung auf ein Microsoft AD-Verzeichnis angewendet.

```
$ldif = Get-Content D:\Users\Username\Downloads\ExtendedSchema.ldf -Raw
Start-DSSchemaExtension -DirectoryId d-123456ijkl -
CreateSnapshotBeforeSchemaExtension $true -Description ManagedADSchemaExtension -
LdifContent $ldif
```

Ausgabe:

```
e-9067306643
```

- API-Details finden Sie StartSchemaExtension in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-DSSchemaExtension`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-DSSchemaExtension`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine aktive Schemaerweiterung zu einem Microsoft AD-Verzeichnis abgebrochen.

```
Stop-DSSchemaExtension -DirectoryId d-123456ijkl -SchemaExtensionId e-9067306643
```

- API-Details finden Sie CancelSchemaExtension in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-DSCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-DSCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Zertifikat gelöscht, das für eine gesicherte LDAP-Verbindung registriert wurde, aus dem System gelöscht.

```
Unregister-DSCertificate -DirectoryId d-123456ijkl -CertificateId c-906731e34f
```

- API-Details finden Sie DeregisterCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-DSEventTopic`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-DSEventTopic`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Verzeichnis als Herausgeber für das angegebene SNS-Thema entfernt.

```
Unregister-DSEventTopic -DirectoryId d-123456ijkl -TopicName snstopicname
```

- API-Details finden Sie DeregisterEventTopic in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-DSConditionalForwarder`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSConditionalForwarder`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktualisiert eine bedingte Weiterleitung, die für Ihr AWS Verzeichnis eingerichtet wurde.

```
Update-DSConditionalForwarder -DirectoryId d-123456ijkl -DnsIpAddr
 172.31.36.96,172.31.16.108 -RemoteDomainName contoso.com
```

- API-Details finden Sie UpdateConditionalForwarder in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-DSRadius`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSRadius`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Informationen des RADIUS-Servers für einen AD Connector oder ein Microsoft AD-Verzeichnis aktualisiert.

```
Update-DSRadius -DirectoryId d-123456ijkl -RadiusSettings_RadiusRetry 3
```

- API-Details finden Sie UpdateRadius in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-DSTrust`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DSTrust`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der SelectiveAuth Parameter der angegebenen Trust-ID von Disabled auf Enabled aktualisiert.

```
Update-DSTrust -TrustId t-9067157123 -SelectiveAuth Enabled
```

Ausgabe:

```
RequestId                            TrustId
---------                            -------
138864a7-c9a8-4ad1-a828-eae479e85b45 t-9067157123
```

- API-Details finden Sie UpdateTrust in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS DMS Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS DMS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`New-DMSReplicationTask`

Das folgende Codebeispiel zeigt die Verwendung`New-DMSReplicationTask`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Replikationsaufgabe für den AWS Database Migration Service erstellt, die CdcStartTime anstelle von CdcStartPosition verwendet. Der MigrationType ist auf „full-load-and-cdc“ gesetzt, was bedeutet, dass die Zieltabelle leer sein muss. Die neue Aufgabe wird mit einem Tag versehen, dessen Schlüssel „Stufe“ lautet und dessen Schlüsselwert „Test“ ist. Weitere Informationen zu den von diesem Cmdlet verwendeten Werten finden Sie unter Creating a Task (https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.Creating.html) im Database Migration Service-Benutzerhandbuch. AWS

```
New-DMSReplicationTask -ReplicationInstanceArn "arn:aws:dms:us-
east-1:123456789012:rep:EXAMPLE66XFJUWATDJGBEXAMPLE"`
```

```
  -CdcStartTime "2019-08-08T12:12:12"`
  -CdcStopPosition "server_time:2019-08-09T12:12:12"`
  -MigrationType "full-load-and-cdc"`
  -ReplicationTaskIdentifier "task1"`
  -ReplicationTaskSetting ""`
  -SourceEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEW5UANC7Y3P4EEXAMPLE"`
  -TableMapping "file:////home/testuser/table-mappings.json"`
  -Tag @{"Key"="Stage";"Value"="Test"}`
  -TargetEndpointArn "arn:aws:dms:us-
east-1:123456789012:endpoint:EXAMPLEJZASXWHTWCLNEXAMPLE"
```

- API-Details finden Sie CreateReplicationTask in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

DynamoDB-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit DynamoDB verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-DDBIndexSchema`

Das folgende Codebeispiel zeigt die Verwendung. `Add-DDBIndexSchema`

Tools für PowerShell V5

Beispiel 1: Erstellt ein leeres TableSchema Objekt und fügt ihm eine neue lokale sekundäre Indexdefinition hinzu, bevor das TableSchema Objekt in die Pipeline geschrieben wird.

```
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema = New-DDBTableSchema
```

Ausgabe:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{LastPostDateTime}                           {}
    {LastPostIndex}
```

Beispiel 2: Fügt dem bereitgestellten TableSchema Objekt eine neue lokale sekundäre Indexdefinition hinzu, bevor das TableSchema Objekt zurück in die Pipeline geschrieben wird. Das TableSchema Objekt kann auch mit dem Parameter -Schema bereitgestellt werden.

```
New-DDBTableSchema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
```

Ausgabe:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{LastPostDateTime}                           {}
    {LastPostIndex}
```

- API-Details finden Sie Add-DDBIndexSchema in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-DDBKeySchema`

Das folgende Codebeispiel zeigt die Verwendung. `Add-DDBKeySchema`

Tools für PowerShell V5

Beispiel 1: Erstellt ein leeres TableSchema Objekt und fügt ihm mithilfe der angegebenen Schlüsseldaten Schlüssel- und Attributdefinitionseinträge hinzu, bevor das TableSchema Objekt

in die Pipeline geschrieben wird. Der Schlüsseltyp ist standardmäßig als 'HASH' deklariert; verwenden Sie den KeyType Parameter - mit dem Wert 'RANGE', um einen Bereichsschlüssel zu deklarieren.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Ausgabe:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{ForumName}                                  {ForumName}
    {}
```

Beispiel 2: Fügt dem bereitgestellten TableSchema Objekt neue Schlüssel- und Attributdefinitionseinträge hinzu, bevor das TableSchema Objekt in die Pipeline geschrieben wird. Der Schlüsseltyp ist standardmäßig als 'HASH' deklariert; verwenden Sie den KeyType Parameter - mit dem Wert 'RANGE', um einen Bereichsschlüssel zu deklarieren. Das TableSchema Objekt kann auch mithilfe des -Schema-Parameters bereitgestellt werden.

```
New-DDBTableSchema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
```

Ausgabe:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
---------------                              ---------
    -------------------------
{ForumName}                                  {ForumName}
    {}
```

- API-Details finden Sie Add-DDBKeySchema in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`ConvertFrom-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `ConvertFrom-DDBItem`

Tools für PowerShell V5

Beispiel 1: ConvertFrom-DDBItem wird verwendet, um das Ergebnis einer Hashtabelle Get-DDBItem von DynamoDB in eine Hashtabelle gängiger Typen wie string und double AttributeValues zu konvertieren.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- API-Details finden Sie ConvertFrom-DDBItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`ConvertTo-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `ConvertTo-DDBItem`

Tools für PowerShell V5

Beispiel 1: Ein Beispiel für die Konvertierung einer Hash-Tabelle in ein Wörterbuch mit DynamoDB-Attributwerten.

```
@{
    SongTitle = 'Somewhere Down The Road'
    Artist    = 'No One You Know'
} | ConvertTo-DDBItem
```

```
Key       Value
---       -----
SongTitle Amazon.DynamoDBv2.Model.AttributeValue
Artist    Amazon.DynamoDBv2.Model.AttributeValue
```

Beispiel 2: Ein Beispiel für die Konvertierung einer Hash-Tabelle in ein Wörterbuch mit DynamoDB-Attributwerten.

```
@{
    MyMap        = @{
        MyString = 'my string'
    }
    MyStringSet  = [System.Collections.Generic.HashSet[String]]@('my', 'string')
    MyNumericSet = [System.Collections.Generic.HashSet[Int]]@(1, 2, 3)
    MyBinarySet  = [System.Collections.Generic.HashSet[System.IO.MemoryStream]]@(
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('my'))),
        ([IO.MemoryStream]::new([Text.Encoding]::UTF8.GetBytes('string')))
    )
    MyList1      = @('my', 'string')
    MyList2      = [System.Collections.Generic.List[Int]]@(1, 2)
    MyList3      = [System.Collections.ArrayList]@('one', 2, $true)
} | ConvertTo-DDBItem
```

Ausgabe:

```
Key          Value
---          -----
MyStringSet  Amazon.DynamoDBv2.Model.AttributeValue
MyList1      Amazon.DynamoDBv2.Model.AttributeValue
MyNumericSet Amazon.DynamoDBv2.Model.AttributeValue
MyList2      Amazon.DynamoDBv2.Model.AttributeValue
MyBinarySet  Amazon.DynamoDBv2.Model.AttributeValue
MyMap        Amazon.DynamoDBv2.Model.AttributeValue
MyList3      Amazon.DynamoDBv2.Model.AttributeValue
```

- API-Details finden Sie ConvertTo-DDBItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DDBBatchItem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBBatchItem`

Tools für PowerShell V5

Beispiel 1: Ruft das Objekt mit dem Titel SongTitle „Somewhere Down The Road“ aus den DynamoDB-Tabellen „Music“ und „Songs“ ab.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$keysAndAttributes = New-Object Amazon.DynamoDBv2.Model.KeysAndAttributes
$list = New-Object
 'System.Collections.Generic.List[System.Collections.Generic.Dictionary[String,
 Amazon.DynamoDBv2.Model.AttributeValue]]'
$list.Add($key)
$keysAndAttributes.Keys = $list

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
    'Songs' = [Amazon.DynamoDBv2.Model.KeysAndAttributes]$keysAndAttributes
}

$batchItems = Get-DDBBatchItem -RequestItem $requestItem
$batchItems.GetEnumerator() | ForEach-Object {$PSItem.Value} | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
Artist                         No One You Know
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
CriticRating                   10
Genre                          Country
Price                          1.94
```

- API-Details finden Sie BatchGetItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBItem`

Tools für PowerShell V5

Beispiel 1: Gibt das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist zurück.

```
$key = @{
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
} | ConvertTo-DDBItem

Get-DDBItem -TableName 'Music' -Key $key | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
SongTitle                      Somewhere Down The Road
Price                          1.94
Artist                         No One You Know
CriticRating                   9
AlbumTitle                     Somewhat Famous
```

- API-Details finden Sie GetItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBTable`

Tools für PowerShell V5

Beispiel 1: Gibt Details der angegebenen Tabelle zurück.

```
Get-DDBTable -TableName "myTable"
```

- API-Details finden Sie DescribeTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-DDBTableList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-DDBTableList`

Tools für PowerShell V5

Beispiel 1: Gibt Details aller Tabellen zurück und iteriert automatisch, bis der Service anzeigt, dass keine weiteren Tabellen vorhanden sind.

```
Get-DDBTableList
```

- API-Details finden Sie ListTables in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Invoke-DDBQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Invoke-DDBQuery`

Tools für PowerShell V5

Beispiel 1: Ruft eine Abfrage auf, die DynamoDB-Elemente mit dem angegebenen und Artist zurückgibt. SongTitle

```
$invokeDDBQuery = @{
    TableName = 'Music'
    KeyConditionExpression = ' SongTitle = :SongTitle and Artist = :Artist'
    ExpressionAttributeValues = @{
        ':SongTitle' = 'Somewhere Down The Road'
        ':Artist' = 'No One You Know'
    } | ConvertTo-DDBItem
}
Invoke-DDBQuery @invokeDDBQuery | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
```

```
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Invoke-DDBScan`

Das folgende Codebeispiel zeigt die Verwendung`Invoke-DDBScan`.

Tools für PowerShell V5

Beispiel 1: Gibt alle Elemente in der Tabelle „Music“ zurück.

```
Invoke-DDBScan -TableName 'Music' | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                            Value
----                            -----
Genre                           Country
Artist                          No One You Know
Price                           1.94
CriticRating                    9
SongTitle                       Somewhere Down The Road
AlbumTitle                      Somewhat Famous
Genre                           Country
Artist                          No One You Know
Price                           1.98
CriticRating                    8.4
SongTitle                       My Dog Spot
AlbumTitle                      Hey Now
```

Beispiel 2: Gibt Elemente in der Musik-Tabelle zurück, deren Wert CriticRating größer oder gleich neun ist.

```
$scanFilter = @{
        CriticRating = [Amazon.DynamoDBv2.Model.Condition]@{
            AttributeValueList = @(@{N = '9'})
            ComparisonOperator = 'GE'
```

```
        }
    }
    Invoke-DDBScan -TableName 'Music' -ScanFilter $scanFilter | ConvertFrom-DDBItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Country
Artist                         No One You Know
Price                          1.94
CriticRating                   9
SongTitle                      Somewhere Down The Road
AlbumTitle                     Somewhat Famous
```

- Weitere API-Informationen finden Sie unter Scan in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`New-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung`New-DDBTable`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, die einen Primärschlüssel hat, der aus ForumName '(Schlüsseltyp-Hash) und' Subject '(Schlüsseltypbereich) besteht. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyType RANGE -KeyDataType "S"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
```

```
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {}
```

Beispiel 2: In diesem Beispiel wird eine Tabelle mit dem Namen Thread erstellt, die einen Primärschlüssel hat, der aus ForumName '(Schlüsseltyp-Hash) und' Subject '(Schlüsseltypbereich) besteht. Ein lokaler sekundärer Index wird ebenfalls definiert. Der Schlüssel des lokalen sekundären Indexes wird automatisch anhand des primären Hashschlüssels in der Tabelle (ForumName) festgelegt. Das zur Erstellung der Tabelle verwendete Schema kann wie gezeigt an jedes Cmdlet übergeben oder mit dem Parameter „-Schema“ angegeben werden.

```
$schema = New-DDBTableSchema
$schema | Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S"
$schema | Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S"
$schema | Add-DDBIndexSchema -IndexName "LastPostIndex" -RangeKeyName
 "LastPostDateTime" -RangeKeyDataType "S" -ProjectionType "keys_only"
$schema | New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

Beispiel 3: Dieses Beispiel zeigt, wie eine einzelne Pipeline verwendet wird, um eine Tabelle mit dem Namen Thread zu erstellen, die einen Primärschlüssel enthält, der aus ForumName '' (Schlüsseltyp-Hash) und 'Subject' (Schlüsseltypbereich) besteht, sowie einen lokalen sekundären Index. Add-DDBKeySchema Und Add-DDBIndexSchema erstellen Sie ein neues TableSchema Objekt für Sie, falls keines von der Pipeline oder dem -Schema-Parameter bereitgestellt wird.

```
New-DDBTableSchema |
```

```
  Add-DDBKeySchema -KeyName "ForumName" -KeyDataType "S" |
  Add-DDBKeySchema -KeyName "Subject" -KeyDataType "S" |
  Add-DDBIndexSchema -IndexName "LastPostIndex" `
                     -RangeKeyName "LastPostDateTime" `
                     -RangeKeyDataType "S" `
                     -ProjectionType "keys_only" |
  New-DDBTable -TableName "Thread" -ReadCapacity 10 -WriteCapacity 5
```

Ausgabe:

```
AttributeDefinitions   : {ForumName, LastPostDateTime, Subject}
TableName              : Thread
KeySchema              : {ForumName, Subject}
TableStatus            : CREATING
CreationDateTime       : 10/28/2013 4:39:49 PM
ProvisionedThroughput  : Amazon.DynamoDBv2.Model.ProvisionedThroughputDescription
TableSizeBytes         : 0
ItemCount              : 0
LocalSecondaryIndexes  : {LastPostIndex}
```

- API-Details finden Sie CreateTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-DDBTableSchema`

Das folgende Codebeispiel zeigt die Verwendung. `New-DDBTableSchema`

Tools für PowerShell V5

Beispiel 1: Erstellt ein leeres TableSchema Objekt, das bereit ist, Schlüssel- und Indexdefinitionen für die Erstellung einer neuen Amazon DynamoDB-Tabelle zu akzeptieren. Das zurückgegebene Objekt kann über die Pipeline an die Add-DDBKeySchema New-DDBTable Cmdlets Add-DDBIndexSchema und übergeben werden oder mithilfe des -Schema-Parameters für jedes Cmdlet an sie übergeben werden.

```
New-DDBTableSchema
```

Ausgabe:

```
AttributeSchema                              KeySchema
    LocalSecondaryIndexSchema
```

```
---------------                                 ---------
    -------------------------
{}                                              {}
    {}
```

- API-Details finden Sie New-DDBTableSchema in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DDBItem`

Tools für PowerShell V5

Beispiel 1: Entfernt das DynamoDB-Element, das dem angegebenen Schlüssel entspricht.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem
Remove-DDBItem -TableName 'Music' -Key $key -Confirm:$false
```

- API-Details finden Sie DeleteItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-DDBTable`

Tools für PowerShell V5

Beispiel 1: Löscht die angegebene Tabelle. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable"
```

Beispiel 2: Löscht die angegebene Tabelle. Sie werden nicht zum Bestätigen aufgefordert, bevor der Vorgang fortgesetzt wird.

```
Remove-DDBTable -TableName "myTable" -Force
```

- API-Details finden Sie DeleteTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-DDBBatchItem`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DDBBatchItem`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes durch ein neues Element in den DynamoDB-Tabellen „Music“ und „Songs“.

```
$item = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
    AlbumTitle = 'Somewhat Famous'
    Price = 1.94
    Genre = 'Country'
    CriticRating = 10.0
} | ConvertTo-DDBItem

$writeRequest = New-Object Amazon.DynamoDBv2.Model.WriteRequest
$writeRequest.PutRequest = [Amazon.DynamoDBv2.Model.PutRequest]$item

$requestItem = @{
    'Music' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
    'Songs' = [Amazon.DynamoDBv2.Model.WriteRequest]($writeRequest)
}

Set-DDBBatchItem -RequestItem $requestItem
```

- API-Details finden Sie BatchWriteItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Set-DDBItem`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues Element oder ersetzt ein vorhandenes Element durch ein neues.

```
$item = @{
```

```
  SongTitle = 'Somewhere Down The Road'
  Artist = 'No One You Know'
        AlbumTitle = 'Somewhat Famous'
        Price = 1.94
        Genre = 'Country'
        CriticRating = 9.0
} | ConvertTo-DDBItem
Set-DDBItem -TableName 'Music' -Item $item
```

- API-Details finden Sie PutItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-DDBItem`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DDBItem`

Tools für PowerShell V5

Beispiel 1: Setzt das Genre-Attribut für das DynamoDB-Element mit dem Partitionsschlüssel SongTitle und dem Sortierschlüssel Artist auf „Rap“.

```
$key = @{
    SongTitle = 'Somewhere Down The Road'
    Artist = 'No One You Know'
} | ConvertTo-DDBItem

$updateDdbItem = @{
    TableName = 'Music'
    Key = $key
    UpdateExpression = 'set Genre = :val1'
    ExpressionAttributeValue = (@{
        ':val1' = ([Amazon.DynamoDBv2.Model.AttributeValue]'Rap')
    })
}
Update-DDBItem @updateDdbItem
```

Ausgabe:

```
Name                           Value
----                           -----
Genre                          Rap
```

- API-Details finden Sie UpdateItem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-DDBTable`

Das folgende Codebeispiel zeigt die Verwendung. `Update-DDBTable`

Tools für PowerShell V5

Beispiel 1: Aktualisiert den bereitgestellten Durchsatz für die angegebene Tabelle.

```
Update-DDBTable -TableName "myTable" -ReadCapacity 10 -WriteCapacity 5
```

- API-Details finden Sie UpdateTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon EC2-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon EC2 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2CapacityReservation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Kapazitätsreservierung mit den angegebenen Attributen erstellt

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Ausgabe:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- API-Details finden Sie CreateCapacityReservation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2InternetGateway`

Tools für PowerShell V5

Beispiel 1: Im folgenden Beispiel wird der spezifischen VPC das angegebene Internet-Gateway angehängt.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel werden eine VPC und ein Internet-Gateway erstellt und anschließend das Internet-Gateway mit der VPC verbunden.

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- API-Details finden Sie AttachInternetGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2NetworkInterface`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle an die spezifizierte Instance angehängt.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

Ausgabe:

```
eni-attach-1a2b3c4d
```

- API-Details finden Sie AttachNetworkInterface in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2Volume`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Volume an die spezifizierte Instance angehängt und mit dem angegebenen Gerätenamen verfügbar gemacht.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- API-Details finden Sie AttachVolume in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Add-EC2VpnGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway an die spezifizierte VPC angehängt.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Ausgabe:

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- API-Details finden Sie AttachVpnGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Approve-EC2VpcPeeringConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Approve-EC2VpcPeeringConnection`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel genehmigt die angeforderte Datei pcx-1dfad234b56ff78be VpcPeeringConnectionId

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Ausgabe:

```
AccepterVpcInfo        : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime         : 1/1/0001 12:00:00 AM
RequesterVpcInfo       : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status                 : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                   : {}
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). AcceptVpcPeeringConnection AWS -Tools für PowerShell

`Confirm-EC2ProductInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Confirm-EC2ProductInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der spezifischen Instance zugeordnet ist.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- API-Details finden Sie ConfirmProductInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Copy-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung`Copy-EC2Image`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene AMI in der Region „EU (Irland)“ in die Region „USA West (Oregon)“ kopiert. Wenn -Region nicht angegeben ist, wird die aktuelle Standardregion als Zielregion verwendet.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-west-2
 -Name "Copy of ami-12345678"
```

Ausgabe:

```
ami-87654321
```

- API-Details finden Sie CopyImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Copy-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-EC2Snapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot von der Region EU (Irland) in die Region USA West (Oregon) kopiert.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region us-
west-2
```

Beispiel 2: Wenn Sie eine Standardregion festlegen und den Parameter Region weglassen, ist die Standardzielregion die Standardregion.

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- API-Details finden Sie CopySnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Deny-EC2VpcPeeringConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Deny-EC2VpcPeeringConnection`

Tools für PowerShell V5

Beispiel 1: Das obige Beispiel lehnt die Anfrage für die Anforderungs-ID VpcPeering pcx-01a2b3ce45fe67eb8 ab

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). RejectVpcPeeringConnection AWS -Tools für PowerShell

`Disable-EC2VgwRoutePropagation`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VgwRoutePropagation`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel deaktiviert das VGW, damit es Routen nicht automatisch an die angegebene Routing-Tabelle weitergibt.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API-Details finden Sie DisableVgwRoutePropagation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VpcClassicLink`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel deaktiviert EC2VpcClassicLink für den vpc-01e23c4a5d6db78e9. Es wird entweder „True“ oder „False“ zurückgegeben

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- API-Details finden Sie DisableVpcClassicLink in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel deaktiviert die ClassicLink DNS-Unterstützung für den vpc-0b12d3456a7e8910d

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- API-Details finden DisableVpcClassicLinkDnsSupport Sie AWS -Tools für PowerShell in der Cmdlet-Referenz (V5).

`Dismount-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2InternetGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- API-Details finden Sie DetachInternetGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2NetworkInterface`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Verbindung zwischen einer Netzwerkschnittstelle und einer Instance entfernt.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API-Details finden Sie DetachNetworkInterface in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2Volume`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
```

```
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

Beispiel 2: Sie können auch die Instance-ID und den Gerätenamen angeben, um sicherzustellen, dass Sie das richtige Volume trennen.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- API-Details finden Sie DetachVolume in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Dismount-EC2VpnGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway von der angegebenen VPC getrennt.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- API-Details finden Sie DetachVpnGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2CapacityReservation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Wert CapacityReservationId cr-0c1f2345db6f7cdba geändert, indem die Anzahl der Instanzen auf 1 geändert wird

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Ausgabe:

```
True
```

- API-Details ModifyCapacityReservation finden Sie in AWS -Tools für PowerShell der Cmdlet-Referenz (V5).

`Edit-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2Host`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die AutoPlacement Einstellungen für den dedizierten Host h-01e23f4cd567890f3 auf Off geändert

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Ausgabe:

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- API-Details ModifyHosts finden Sie in AWS -Tools für PowerShell der Cmdlet-Referenz (V5).

`Edit-EC2IdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2IdFormat`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp aktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Beispiel 2: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp deaktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API-Details finden Sie ModifyIdFormat in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2ImageAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI aktualisiert.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Beispiel 2: Dieses Beispiel macht das AMI öffentlich (zum Beispiel, damit jeder es verwenden AWS-Konto kann).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

Beispiel 3: In diesem Beispiel wird das AMI privat gemacht (beispielsweise, damit nur Sie als Besitzer es verwenden können).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

Beispiel 4: Dieses Beispiel erteilt dem angegebenen Objekt die Startberechtigung AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

Beispiel 5: In diesem Beispiel wird die Startberechtigung aus dem angegebenen Objekt entfernt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- API-Details finden Sie ModifyImageAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2InstanceAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Instance-Typ der angegebenen Instance geändert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Beispiel 2: In diesem Beispiel wird Enhanced Networking für die angegebene Instance aktiviert, indem „simple“ als Wert des Netzwerkunterstützungsparameters Single Root I/O Virtualization (SR-IOV) angegeben wird, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance geändert. Die Instance muss sich in einer VPC befinden. Sie müssen die ID jeder Sicherheitsgruppe angeben, nicht den Namen.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

Beispiel 4: Dieses Beispiel aktiviert die I/O EBS-Optimierung für die angegebene Instance. Diese Feature ist nicht für alle Instance-Typen verfügbar. Bei der Nutzung einer EBS-optimized Instance fallen zusätzliche Nutzungsgebühren an.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Beispiel 5: Dieses Beispiel ermöglicht die source/destination Überprüfung der angegebenen Instanz. Damit eine NAT-Instance NAT ausführen kann, muss der Wert „False“ sein.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Beispiel 6: In diesem Beispiel wird die Beendigung für die angegebene Instance deaktiviert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Beispiel 7: In diesem Beispiel wird die angegebene Instance so geändert, dass sie beendet wird, wenn der Shutdown von der Instance aus initiiert wird.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceInitiatedShutdownBehavior
 terminate
```

- API-Details finden Sie ModifyInstanceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2InstanceCreditSpecification`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2InstanceCreditSpecification`

Tools für PowerShell V5

Beispiel 1: Dadurch werden unbegrenzte T2-Gutschriften für die Instance i-01234567890abcdef aktiviert.

```
$Credit = New-Object -TypeName Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API-Details finden Sie ModifyInstanceCreditSpecification in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2NetworkInterfaceAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle so geändert, dass die spezifizierte Anlage beim Beenden gelöscht wird.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Beispiel 2: In diesem Beispiel wird die Beschreibung der angegebenen Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description "my
 description"
```

Beispiel 3: In diesem Beispiel wird die Sicherheitsgruppe für die angegebene Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

Beispiel 4: In diesem Beispiel wird die Suche nach source/destination der angegebenen Netzwerkschnittstelle deaktiviert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
 $false
```

- API-Details finden Sie ModifyNetworkInterfaceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2ReservedInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2ReservedInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Availability Zone, die Anzahl der Instances und die Plattform für die angegebenen Reserved Instances geändert.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE", "0CC556F3-7AB8-4C00-
B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- API-Details finden Sie ModifyReservedInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SnapshotAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot veröffentlicht, indem sein CreateVolumePermission Attribut gesetzt wird.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- API-Details finden Sie ModifySnapshotAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SpotFleetRequest`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Zielkapazität der angegebenen Spot-Flottenanforderung aktualisiert.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Ausgabe:

```
True
```

- API-Details finden Sie ModifySpotFleetRequest in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2SubnetAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2SubnetAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz aktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Beispiel 2: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz deaktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API-Details finden Sie ModifySubnetAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2VolumeAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2VolumeAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ändert das angegebene Attribut des angegebenen Volumes. I/O Der Betrieb des Datenträgers wird automatisch wieder aufgenommen, nachdem er aufgrund möglicherweise inkonsistenter Daten unterbrochen wurde.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API-Details finden Sie ModifyVolumeAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-EC2VpcAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-EC2VpcAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wirdt die Unterstützung von DNS-Hostnamen für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Beispiel 2: In diesem Beispiel wird die Unterstützung für DNS-Hostnamen für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Beispiel 3: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC aktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Beispiel 4: In diesem Beispiel wird die Unterstützung für die DNS-Auflösung für die angegebene VPC deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API-Details finden Sie ModifyVpcAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-EC2VgwRoutePropagation`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VgwRoutePropagation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die automatische Weitergabe von Routen durch die angegebene VGW an die angegebene Routing-Tabelle aktiviert.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API-Details finden Sie EnableVgwRoutePropagation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-EC2VolumeIO`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VolumeIO`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden I/O Operationen für das angegebene Volume aktiviert, wenn die I/O Operationen deaktiviert waren.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API-Details finden Sie EnableVolumeIo in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VpcClassicLink`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktiviert VPC vpc-0123456b789b0d12f für ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Ausgabe:

```
True
```

- API-Details finden EnableVpcClassicLink Sie in AWS -Tools für PowerShell der Cmdlet-Referenz (V5).

`Enable-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ermöglicht es vpc-0b12d3456a7e8910d, die DNS-Hostnamenauflösung zu unterstützen für ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API-Details finden EnableVpcClassicLinkDnsSupport Sie AWS -Tools für PowerShell in der Cmdlet-Referenz (V5).

`Get-EC2AccountAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2AccountAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt, ob Sie Instances in EC2-Classic und EC2-VPC in der Region oder nur in EC2-VPC starten können.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
EC2
VPC
```

Beispiel 2: In diesem Beispiel Ihre Standard-VPC beschrieben oder es zeigt „none“, wenn Sie in der Region keine Standard-VPC haben.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
vpc-12345678
```

Beispiel 3: Dieses Beispiel beschreibt die maximale Anzahl von On-Demand Instances, die Sie ausführen können.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Ausgabe:

```
AttributeValue
--------------
20
```

- API-Details finden Sie DescribeAccountAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Address`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Ausgabe:

```
AllocationId            : eipalloc-12345678
AssociationId           : eipassoc-12345678
Domain                  : vpc
InstanceId              : i-87654321
NetworkInterfaceId      : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress        : 10.0.2.172
PublicIp                : 198.51.100.2
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Elastic-IP-Adressen für Instances in einer VPC. Diese Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Ausgabe:

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
```

```
PublicIp                : 203.0.113.17
```

Beispiel 4: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in EC2-Classic. Diese Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Beispiel 5: In diesem Beispiel werden alle Ihre Elastic IP-Adressen beschrieben.

```
Get-EC2Address
```

Beispiel 6: Dieses Beispiel gibt die öffentliche und private IP für die im Filter angegebene Instance-ID zurück

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Ausgabe:

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

Beispiel 7: In diesem Beispiel werden alle Elastic IPs mit ihrer Zuweisungs-ID, Zuordnungs-ID und Instance-IDs abgerufen

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

Ausgabe:

```
InstanceId          AssociationId              AllocationId               PublicIp
----------          -------------              ------------               --------
                                               eipalloc-012e3b456789e1fad
 17.212.120.178
i-0c123dfd3415bac67 eipassoc-0e123456bb7890bdb eipalloc-01cd23ebf45f7890c
 17.212.124.77
                                               eipalloc-012345678eeabcfad
 17.212.225.7
```

```
i-0123d405c67e89a0c eipassoc-0c123b456783966ba eipalloc-0123cdd456a8f7892
 37.216.52.173
i-0f1bf2f34c5678d09 eipassoc-0e12934568a952d96 eipalloc-0e1c23e4d5e6789e4
 37.218.222.278
i-012e3cb4df567e8aa eipassoc-0d1b2fa4d67d03810 eipalloc-0123f456f78a01b58
 37.210.82.27
i-0123bcf4b567890e1 eipassoc-01d2345f678903fb1 eipalloc-0e1db23cfef5c45c7
 37.215.222.270
```

Beispiel 8: Dieses Beispiel ruft eine Liste der EC2-IP-Adressen ab, die mit dem Tag-Schlüssel „Category” mit dem Wert „Prod” übereinstimmen'

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Ausgabe:

```
AllocationId            : eipalloc-0123f456f81a01b58
AssociationId           : eipassoc-0d1b23a456d103810
CustomerOwnedIp         :
CustomerOwnedIpv4Pool   :
Domain                  : vpc
InstanceId              : i-012e3cb4df567e1aa
NetworkBorderGroup      : eu-west-1
NetworkInterfaceId      : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress        : 192.168.1.84
PublicIp                : 34.250.81.29
PublicIpv4Pool          : amazon
Tags                    : {Category, Name}
```

- API-Details finden Sie DescribeAddresses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2AvailabilityZone`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2AvailabilityZone`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Availability Zones für die aktuelle Region beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2AvailabilityZone
```

Ausgabe:

```
Messages    RegionName    State        ZoneName
--------    ----------    -----        --------
{}          us-west-2     available    us-west-2a
{}          us-west-2     available    us-west-2b
{}          us-west-2     available    us-west-2c
```

Beispiel 2: In diesem Beispiel werden alle Availability Zones beschrieben, die sich in einem beeinträchtigten Zustand befinden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object zum Erstellen des Filters verwenden.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- API-Details finden Sie DescribeAvailabilityZones in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2BundleTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2BundleTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Bündelungsaufgabe beschrieben.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Beispiel 2: In diesem Beispiel werden die Bündelungsaufgaben beschrieben, deren Status entweder „Abgeschlossen“ oder „Fehlgeschlagen“ lautet.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API-Details finden Sie DescribeBundleTasks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2CapacityReservation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden eine oder mehrere Ihrer Kapazitätsreservierungen für die Region beschrieben

```
Get-EC2CapacityReservation -Region eu-west-1
```

Ausgabe:

```
AvailabilityZone       : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId  : cr-0c1f2345db6f7cdba
CreateDate             : 3/28/2019 9:29:41 AM
EbsOptimized           : True
EndDate                : 1/1/0001 12:00:00 AM
EndDateType            : unlimited
EphemeralStorage       : False
InstanceMatchCriteria  : open
InstancePlatform       : Windows
InstanceType           : m4.xlarge
State                  : active
Tags                   : {}
Tenancy                : default
TotalInstanceCount     : 2
```

- API-Details finden Sie DescribeCapacityReservations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ConsoleOutput`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ConsoleOutput`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen. Die Konsolenausgabe ist verschlüsselt.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Ausgabe:

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

Beispiel 2: In diesem Beispiel wird die verschlüsselte Konsolenausgabe in einer Variablen gespeichert und anschließend entschlüsselt.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API-Details finden Sie GetConsoleOutput in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2CustomerGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2CustomerGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway beschrieben.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
```

```
State              : available
Tags               : {}
Type               : ipsec.1
```

Beispiel 2: In diesem Beispiel wird jedes Kunden-Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Kunden-Gateways beschrieben.

```
Get-EC2CustomerGateway
```

- API-Details finden Sie DescribeCustomerGateways in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2DhcpOption`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Ihre DHCP-Optionssätze aufgelistet.

```
Get-EC2DhcpOption
```

Ausgabe:

```
DhcpConfigurations                   DhcpOptionsId    Tag
------------------                   -------------    ---
{domain-name, domain-name-servers}   dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}   dopt-2a3b4c5d    {}
{domain-name-servers}                dopt-3a4b5c6d    {}
```

Beispiel 2: In diesem Beispiel werden Konfigurationsdetails für den angegebenen DHCP-Optionssatz abgerufen.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Ausgabe:

```
Key                    Values
---                    ------
domain-name            {abc.local}
domain-name-servers    {10.0.0.101, 10.0.0.102}
```

- API-Details finden Sie DescribeDhcpOptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2FlowLog`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden ein oder mehrere Flow-Protokolle mit dem Protokollzieltyp „s3“ beschrieben.

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

Ausgabe:

```
CreationTime             : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage  :
DeliverLogsPermissionArn :
DeliverLogsStatus        : SUCCESS
FlowLogId                : fl-01b2e3d45f67f8901
FlowLogStatus            : ACTIVE
LogDestination           : arn:aws:s3:::amzn-s3-demo-bucket-dd-tata
LogDestinationType       : s3
LogGroupName             :
ResourceId               : eni-01d2dda3456b7e890
TrafficType              : ALL
```

- API-Details finden Sie DescribeFlowLogs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Host`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die EC2-Hostdetails zurückgegeben

```
Get-EC2Host
```

Ausgabe:

```
AllocationTime    : 3/23/2019 4:55:22 PM
AutoPlacement     : off
AvailabilityZone  : eu-west-1b
AvailableCapacity : Amazon.EC2.Model.AvailableCapacity
ClientToken       :
HostId            : h-01e23f4cd567890f1
HostProperties    : Amazon.EC2.Model.HostProperties
HostReservationId :
Instances         : {}
ReleaseTime       : 1/1/0001 12:00:00 AM
State             : available
Tags              : {}
```

Beispiel 2: Dieses Beispiel fragt nach dem Host AvailableInstanceCapacity h-01e23f4cd567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Ausgabe:

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). DescribeHosts AWS -Tools für PowerShell

`Get-EC2HostReservationOffering`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2HostReservationOffering`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die Dedicated Host-Reservierungen, die für den angegebenen Filter 'instance-family' käuflich erworben werden können, wobei PaymentOption " NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

Ausgabe:

```
CurrencyCode   :
Duration       : 94608000
HourlyPrice    : 1.307
InstanceFamily : m4
OfferingId     : hro-0c1f234567890d9ab
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000

CurrencyCode   :
Duration       : 31536000
HourlyPrice    : 1.830
InstanceFamily : m4
OfferingId     : hro-04ad12aaaf34b5a67
PaymentOption  : NoUpfront
UpfrontPrice   : 0.000
```

- API-Details finden Sie DescribeHostReservationOfferings in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2HostReservationPurchasePreview`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2HostReservationPurchasePreview`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt eine Vorschau einer Reservierung mit Konfigurationen, die denen Ihres Dedicated Host h-01e23f4cd567890f1 entsprechen

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- API-Details finden Sie GetHostReservationPurchasePreview in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2IdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2IdFormat`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das ID-Format für den angegebenen Ressourcentyp beschrieben.

```
Get-EC2IdFormat -Resource instance
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
instance       False
```

Beispiel 2: In diesem Beispiel werden die ID-Formate für alle Ressourcentypen beschrieben, die längere IDs unterstützen.

```
Get-EC2IdFormat
```

Ausgabe:

```
Resource       UseLongIds
--------       ----------
reservation    False
instance       False
```

- API-Details finden Sie DescribeIdFormat in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2IdentityIdFormat`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2IdentityIdFormat`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt das ID-Format für die Ressource „image“ für die angegebene Rolle zurück

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -Resource
 image
```

Ausgabe:

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- API-Details finden Sie DescribeIdentityIdFormat in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene AMI beschrieben.

```
Get-EC2Image -ImageId ami-12345678
```

Ausgabe:

```
Architecture        : x86_64
BlockDeviceMappings : {/dev/xvda}
CreationDate        : 2014-10-20T00:56:28.000Z
Description         : My image
Hypervisor          : xen
ImageId             : ami-12345678
ImageLocation       : 123456789012/my-image
ImageOwnerAlias     :
```

```
ImageType           : machine
KernelId            :
Name                : my-image
OwnerId             : 123456789012
Platform            :
ProductCodes        : {}
Public              : False
RamdiskId           :
RootDeviceName      : /dev/xvda
RootDeviceType      : ebs
SriovNetSupport     : simple
State               : available
StateReason         :
Tags                : {Name}
VirtualizationType  : hvm
```

Beispiel 2: In diesem Beispiel die AMIs beschrieben, die Sie besitzen.

```
Get-EC2Image -owner self
```

Beispiel 3: In diesem Beispiel werden die öffentlichen AMIs beschrieben, auf denen Microsoft Windows Server ausgeführt wird.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Beispiel 4: In diesem Beispiel werden alle öffentlichen AMIs in der Region „us-west-2“ beschrieben.

```
Get-EC2Image -Region us-west-2
```

- API-Details finden Sie DescribeImages in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImageAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 2: In diesem Beispiel werden die Startberechtigungen für das angegebene AMI abgerufen.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

Beispiel 3: In diesem Beispiel wird überprüft, ob Enhanced Networking aktiviert ist.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
```

```
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- API-Details finden Sie DescribeImageAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ImportImageTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImportImageTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Ausgabe:

```
Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Images beschrieben.

```
Get-EC2ImportImageTask
```

Ausgabe:

```
Architecture    :
Description     : Windows Image 1
```

```
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {}
Status          : deleted
StatusMessage   : User initiated task cancelation

Architecture    : x86_64
Description     : Windows Image 2
Hypervisor      :
ImageId         : ami-1a2b3c4d
ImportTaskId    : import-ami-hgfedcba
LicenseType     : AWS
Platform        : Windows
Progress        :
SnapshotDetails : {/dev/sda1}
Status          : completed
StatusMessage   :
```

- API-Details finden Sie DescribeImportImageTasks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ImportSnapshotTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ImportSnapshotTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Aufgabe zum Importieren eines Snapshots beschrieben.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------
```

```
Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

Beispiel 2: In diesem Beispiel werden alle Ihre Aufgaben zum Importieren von Snapshops beschrieben.

```
Get-EC2ImportSnapshotTask
```

Ausgabe:

```
Description             ImportTaskId               SnapshotTaskDetail

-----------------       --------------------       ------------------

Disk Image Import 1     import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2     import-snap-hgfedcba
 Amazon.EC2.Model.SnapshotTaskDetail
```

- API-Details finden Sie DescribeImportSnapshotTasks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Instance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance beschrieben.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Ausgabe:

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
EbsOptimized          : False
Hypervisor            : xen
IamInstanceProfile    : Amazon.EC2.Model.IamInstanceProfile
```

```
ImageId               : ami-12345678
InstanceId            : i-12345678
InstanceLifecycle     :
InstanceType          : t2.micro
KernelId              :
KeyName               : my-key-pair
LaunchTime            : 12/4/2015 4:44:40 PM
Monitoring            : Amazon.EC2.Model.Monitoring
NetworkInterfaces     : {ip-10-0-2-172.us-west-2.compute.internal}
Placement             : Amazon.EC2.Model.Placement
Platform              : Windows
PrivateDnsName        : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress      : 10.0.2.172
ProductCodes          : {}
PublicDnsName         :
PublicIpAddress       :
RamdiskId             :
RootDeviceName        : /dev/sda1
RootDeviceType        : ebs
SecurityGroups        : {default}
SourceDestCheck       : True
SpotInstanceRequestId :
SriovNetSupport       :
State                 : Amazon.EC2.Model.InstanceState
StateReason           :
StateTransitionReason :
SubnetId              : subnet-12345678
Tags                  : {Name}
VirtualizationType    : hvm
VpcId                 : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Instances in der aktuellen Region beschrieben, gruppiert nach Reservierungen. Um die Instance-Details anzuzeigen, erweitern Sie die Sammlung der Instances innerhalb jedes Reservierungsobjekts.

```
Get-EC2Instance
```

Ausgabe:

```
GroupNames    : {}
Groups        : {}
Instances     : {}
```

```
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c

GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Beispiel 3: Dieses Beispiel veranschaulicht die Verwendung eines Filters zur Abfrage von EC2-Instances in einem bestimmten Subnetz einer VPC.

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 4: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten zur Abfrage von EC2-Instances, die sowohl ausgeführt werden als auch angehalten sind

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress PublicIpAddress
 SecurityGroups SubnetId        VpcId
----------          ------------ -------- ---------------- ---------------
 -------------- --------        -----
i-05a9...f6c5f46e18 t3.medium             10.0.1.7                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11                       ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6                        ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

Beispiel 5: Dieses Beispiel veranschaulicht die Verwendung eines Filters mit mehreren Werten zur Abfrage von EC2-Instances, die sowohl ausgeführt als auch gestoppt werden, und die Verwendung des Select-Object Cmdlets zur Auswahl bestimmter Werte für die Ausgabe.

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_.Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

Ausgabe:

```
InstanceId          InstanceType Platform PrivateIpAddress Name         State
----------          ------------ -------- ---------------- ----         -----
```

```
i-05a9...f6c5f46e18 t3.medium             10.0.1.7         ec2-name-01  running
i-02cf...945c4fdd07 t3.medium    Windows  10.0.1.8         ec2-name-02  stopped
i-0ac0...c037f9f3a1 t3.xlarge    Windows  10.0.1.10        ec2-name-03  running
i-066b...57b7b08888 t3.medium    Windows  10.0.1.11        ec2-name-04  stopped
i-0fee...82e83ccd72 t3.medium    Windows  10.0.1.5         ec2-name-05  running
i-0a68...274cc5043b t3.medium    Windows  10.0.1.6         ec2-name-06  stopped
```

Beispiel 6: In diesem Beispiel werden Berechtigungen für das Abrufen von EC2-Instances mithilfe des DryRun Parameters validiert, ohne sie tatsächlich abzurufen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Get-EC2Tag -DryRun $true
```

Ausgabe:

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- API-Details finden Sie DescribeInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt den Instance-Typ der angegebenen Instance.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Ausgabe:

```
InstanceType                     : t2.micro
```

Beispiel 2: In diesem Beispiel wird beschrieben, ob Enhanced Networking für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
SriovNetSupport                    : simple
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance beschrieben.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Ausgabe:

```
GroupId
-------
sg-12345678
sg-45678901
```

Beispiel 4: In diesem Beispiel wird beschrieben, ob die EBS-Optimierung für die angegebene Instance aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ausgabe:

```
EbsOptimized                       : False
```

Beispiel 5: Dieses Beispiel beschreibt das ApiTermination 'Disable'-Attribut der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ausgabe:

```
DisableApiTermination              : False
```

Beispiel 6: Dieses Beispiel beschreibt das InitiatedShutdownBehavior 'instanz'-Attribut der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

Ausgabe:

```
InstanceInitiatedShutdownBehavior : stop
```

- API-Details finden Sie DescribeInstanceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2InstanceMetadata`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceMetadata`

Tools für PowerShell V5

Beispiel 1: Listet die verfügbaren Kategorien von Instance-Metadaten auf, die abgefragt werden können.

```
Get-EC2InstanceMetadata -ListCategory
```

Ausgabe:

```
AmiId
LaunchIndex
ManifestPath
AncestorAmiId
BlockDeviceMapping
InstanceId
InstanceType
LocalHostname
LocalIpv4
KernelId
AvailabilityZone
ProductCode
PublicHostname
PublicIpv4
PublicKey
RamdiskId
Region
```

```
ReservationId
SecurityGroup
UserData
InstanceMonitoring
IdentityDocument
IdentitySignature
IdentityPkcs7
```

Beispiel 2: Gibt die ID des Amazon Machine Image (AMI) zurück, das zum Starten der Instance verwendet wurde.

```
Get-EC2InstanceMetadata -Category AmiId
```

Ausgabe:

```
ami-b2e756ca
```

Beispiel 3: In diesem Beispiel wird das JSON-formatted Ausweisdokument für die Instanz abgefragt.

```
Get-EC2InstanceMetadata -Category IdentityDocument
{
  "availabilityZone" : "us-west-2a",
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : null,
  "version" : "2017-09-30",
  "instanceId" : "i-01ed50f7e2607f09e",
  "billingProducts" : [ "bp-6ba54002" ],
  "instanceType" : "t2.small",
  "pendingTime" : "2018-03-07T16:26:04Z",
  "imageId" : "ami-b2e756ca",
  "privateIp" : "10.0.0.171",
  "accountId" : "111122223333",
  "architecture" : "x86_64",
  "kernelId" : null,
  "ramdiskId" : null,
  "region" : "us-west-2"
}
```

Beispiel 4: In diesem Beispiel werden mithilfe einer Pfadabfrage die MAC-Adressen der Netzwerkschnittstellen für die Instance abgerufen.

```
Get-EC2InstanceMetadata -Path "/network/interfaces/macs"
```

Ausgabe:

```
02:80:7f:ef:4c:e0/
```

Beispiel 5: Wenn der Instance eine IAM-Rolle zugeordnet ist, werden Informationen darüber zurückgegeben, wann das Instance-Profil zuletzt aktualisiert wurde, einschließlich des LastUpdated Datums der Instance InstanceProfileArn, und InstanceProfileId.

```
Get-EC2InstanceMetadata -Path "/iam/info"
```

Ausgabe:

```
{
  "Code" : "Success",
  "LastUpdated" : "2018-03-08T03:38:40Z",
  "InstanceProfileArn" : "arn:aws:iam::111122223333:instance-profile/
MyLaunchRole_Profile",
  "InstanceProfileId" : "AIPAI4...WVK2RW"
}
```

- API-Details finden Sie Get-EC2InstanceMetadata in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2InstanceStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InstanceStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status der angegebenen Instance beschrieben.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Ausgabe:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
```

```
InstanceState    : Amazon.EC2.Model.InstanceState
Status           : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Ausgabe:

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

Ausgabe:

```
Details           Status
-------           ------
{reachability}    ok
```

- API-Details finden Sie DescribeInstanceStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2InternetGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway beschrieben.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

Beispiel 2: In diesem Beispiel werden alle Ihre Internet-Gateways beschrieben.

```
Get-EC2InternetGateway
```

Ausgabe:

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- API-Details finden Sie DescribeInternetGateways in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2KeyPair`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar beschrieben.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
KeyFingerprint                                          KeyName
--------------                                          -------
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Beispiel 2: In diesem Beispiel werden alle Ihre Schlüsselpaare beschrieben.

```
Get-EC2KeyPair
```

- API-Details finden Sie DescribeKeyPairs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkAcl`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL beschrieben.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden die Regeln für die angegebene Netzwerk-ACL beschrieben.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Ausgabe:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
```

```
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Beispiel 3: In diesem Beispiel werden alle Ihre Netzwerk-ACLs beschrieben.

```
Get-EC2NetworkAcl
```

- API-Details finden Sie DescribeNetworkAcls in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkInterface`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Association        :
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone   : us-west-2c
Description        :
Groups             : {my-security-group}
MacAddress         : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : in-use
SubnetId           : subnet-1a2b3c4d
```

```
TagSet             : {}
VpcId              : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Netzwerkschnittstellen beschrieben.

```
Get-EC2NetworkInterface
```

- API-Details finden Sie DescribeNetworkInterfaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2NetworkInterfaceAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

Ausgabe:

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Beispiel 2: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

Ausgabe:

```
Description        : My description
```

Beispiel 3: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

Ausgabe:

```
Groups             : {my-security-group}
```

Beispiel 4: In diesem Beispiel wird die angegebene Netzwerkschnittstelle beschrieben.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

Ausgabe:

```
SourceDestCheck    : True
```

- API-Details finden Sie DescribeNetworkInterfaceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2PasswordData`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PasswordData`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Passwort entschlüsselt, das Amazon EC2 dem Administratorkonto für die angegebene Windows-Instance zugeordnet hat. Da eine PEM-Datei angegeben wurde, wird automatisch die Einstellung des Schalters -Decrypt übernommen.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 2: (PowerShell nur Windows) Prüft die Instanz, um den Namen des Schlüsselpaars zu ermitteln, das zum Starten der Instanz verwendet wurde, und versucht dann, die entsprechenden Schlüsselpaardaten im Konfigurationsspeicher des Toolkits für Visual Studio zu finden. AWS Wenn die Schlüsselpaardaten gefunden werden, wird das Passwort entschlüsselt.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 3: Gibt die verschlüsselten Passwortdaten für die Instance zurück.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Ausgabe:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API-Details finden Sie GetPasswordData in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PlacementGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe beschrieben.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- API-Details finden Sie DescribePlacementGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2PrefixList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2PrefixList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ruft das AWS-Services in einem Präfixlistenformat verfügbare Format für die Region ab

```
Get-EC2PrefixList
```

Ausgabe:

```
Cidrs                                          PrefixListId PrefixListName
-----                                          ------------ --------------
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006  com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19}               pl-6da54004  com.amazonaws.eu-
west-1.s3
```

- API-Details finden Sie DescribePrefixLists in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Region`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Region`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Regionen beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2Region
```

Ausgabe:

```
Endpoint                          RegionName
--------                          ----------
ec2.eu-west-1.amazonaws.com       eu-west-1
ec2.ap-southeast-1.amazonaws.com  ap-southeast-1
ec2.ap-southeast-2.amazonaws.com  ap-southeast-2
ec2.eu-central-1.amazonaws.com    eu-central-1
ec2.ap-northeast-1.amazonaws.com  ap-northeast-1
ec2.us-east-1.amazonaws.com       us-east-1
ec2.sa-east-1.amazonaws.com       sa-east-1
```

```
ec2.us-west-1.amazonaws.com         us-west-1
ec2.us-west-2.amazonaws.com         us-west-2
```

- API-Details finden Sie DescribeRegions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2RouteTable`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Ihre Routing-Tabellen beschrieben.

```
Get-EC2RouteTable
```

Ausgabe:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId  :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId      :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId  :
```

Beispiel 2: In diesem Beispiel werden Details für die angegebene Routing-Tabelle zurückgegeben.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Beispiel 3: In diesem Beispiel werden die Routing-Tabellen für die angegebene VPC beschrieben.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Ausgabe:

```
Associations    : {rtbassoc-12345678}
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- API-Details finden Sie DescribeRouteTables in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ScheduledInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ScheduledInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance beschrieben.

```
Get-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
```

```
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

Beispiel 2: In diesem Beispiel werden alle Ihre geplanten Instances beschrieben.

```
Get-EC2ScheduledInstance
```

- API-Details finden Sie DescribeScheduledInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2ScheduledInstanceAvailability`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2ScheduledInstanceAvailability`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Zeitplan beschrieben, der jede Woche am Sonntag beginnt und am angegebenen Datum startet.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
 Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
AvailableInstanceCount      : 20
FirstSlotStartTime          : 1/31/2016 8:00:00 AM
HourlyPrice                 : 0.095
InstanceType                : c4.large
MaxTermDurationInDays       : 366
MinTermDurationInDays       : 366
NetworkPlatform             : EC2-VPC
Platform                    : Linux/UNIX
PurchaseToken               : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours         : 23
TotalScheduledInstanceHours : 1219
```

```
...
```

Beispiel 2: Um die Ergebnisse einzugrenzen, können Sie Filter für Kriterien wie Betriebssystem, Netzwerk und Instance-Typ hinzufügen.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API-Details finden Sie DescribeScheduledInstanceAvailability in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SecurityGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für eine VPC beschrieben. Wenn Sie mit Sicherheitsgruppen arbeiten, die zu einer VPC gehören, müssen Sie die Sicherheitsgruppen-ID (- GroupId Parameter) verwenden, nicht den Namen (- GroupName Parameter), um auf die Gruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Description         : default VPC security group
GroupId             : sg-12345678
GroupName           : default
IpPermissions       : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId             : 123456789012
Tags                : {}
VpcId               : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für EC2-Classic. Wenn Sie mit Sicherheitsgruppen für arbeiten, können EC2-Classic Sie entweder den Gruppennamen (- GroupName Parameter) oder die Gruppen-ID (- GroupId Parameter) verwenden, um auf die Sicherheitsgruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Ausgabe:

```
Description         : my security group
GroupId             : sg-45678901
GroupName           : my-security-group
IpPermissions       : {Amazon.EC2.Model.IpPermission, Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId             : 123456789012
Tags                : {}
VpcId               :
```

Beispiel 3: In diesem Beispiel werden alle Sicherheitsgruppen für die Datei vpc-0fc1ff23456b789eb abgerufen

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Snapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot beschrieben.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from
 vol-12345678
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
```

```
OwnerId             : 123456789012
Progress            : 100%
SnapshotId          : snap-12345678
StartTime           : 10/23/2014 6:01:28 AM
State               : completed
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 8
```

Beispiel 2: In diesem Beispiel werden die Snapshots beschrieben, die ein „Name”-Tag enthalten.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Beispiel 3: Dieses Beispiel beschreibt die Schnappschüsse, die ein 'Name'-Tag mit dem Wert " TestValue haben.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

Beispiel 4: In diesem Beispiel werden alle Ihre Snapshots beschrieben.

```
Get-EC2Snapshot -Owner self
```

- API-Details finden Sie DescribeSnapshots in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SnapshotAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Ausgabe:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

Beispiel 2: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots beschrieben.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

Ausgabe:

```
Group    UserId
-----    ------
all
```

- API-Details finden Sie DescribeSnapshotAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotDatafeedSubscription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Daten-Feed für Ihre Spot-Instance beschrieben.

```
Get-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- API-Details finden Sie DescribeSpotDatafeedSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotFleetInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Instances beschrieben, die der angegebenen Spot-Flottenanforderung zugeordnet sind.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- API-Details finden Sie DescribeSpotFleetInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetRequest`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
 | format-list
```

Ausgabe:

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Flottenanforderungen beschrieben.

```
Get-EC2SpotFleetRequest
```

- API-Details finden Sie DescribeSpotFleetRequests in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotFleetRequestHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotFleetRequestHistory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Verlauf der angegebenen Spot-Flottenanforderung beschrieben.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Ausgabe:

```
HistoryRecords     : {Amazon.EC2.Model.HistoryRecord,
 Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Ausgabe:

```
EventInformation                    EventType             Timestamp
----------------                    ---------             ---------
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    fleetRequestChange    12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation    launched              12/26/2015 8:25:34 AM
```

```
Amazon.EC2.Model.EventInformation    launched             12/26/2015 8:25:05 AM
```

- API-Details finden Sie DescribeSpotFleetRequestHistory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotInstanceRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotInstanceRequest`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung beschrieben.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 4/8/2015 2:51:33 PM
Fault                    :
InstanceId               : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.020000
State                    : active
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {Name}
Type                     : one-time
```

Beispiel 2: In diesem Beispiel werden alle Ihre Spot-Instance-Anforderungen beschrieben.

```
Get-EC2SpotInstanceRequest
```

- API-Details finden Sie DescribeSpotInstanceRequests in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2SpotPriceHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2SpotPriceHistory`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ruft die letzten 10 Einträge im Spot-Preisverlauf für den angegebenen Instance-Typ und die angegebene Availability Zone ab. Beachten Sie, dass der für den AvailabilityZone Parameter - angegebene Wert für den Region-Wert gültig sein muss, der entweder an den Parameter -Region des Cmdlets übergeben wird (im Beispiel nicht gezeigt) oder in der Shell als Standard festgelegt ist. Bei diesem Beispielbefehl wird davon ausgegangen, dass die Standardregion „us-west-2“ in der Umgebung festgelegt wurde.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Ausgabe:

```
AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:38:29 AM

AvailabilityZone   : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 6:57:13 AM
...
```

- API-Details finden Sie DescribeSpotPriceHistory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Subnet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz beschrieben.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : available
SubnetId                : subnet-1a2b3c4d
Tags                    : {}
VpcId                   : vpc-12345678
```

Beispiel 2: In diesem Beispiel werden alle Ihre Subnetze beschrieben.

```
Get-EC2Subnet
```

- API-Details finden Sie DescribeSubnets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tags für den Ressourcentyp „image“ abgerufen

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
Name        ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

Beispiel 2: In diesem Beispiel werden alle Tags für alle Ressourcen abgerufen und nach Ressourcentyp gruppiert

```
Get-EC2Tag | Group-Object resourcetype
```

Ausgabe:

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   53 instance                  {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    5 security-group            {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
   30 volume                    {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription...}
    1 internet-gateway          {Amazon.EC2.Model.TagDescription}
    3 network-interface         {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
    4 elastic-ip                {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    1 dhcp-options              {Amazon.EC2.Model.TagDescription}
    2 image                     {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription}
    3 vpc                       {Amazon.EC2.Model.TagDescription,
 Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
```

Beispiel 3: In diesem Beispiel werden alle Ressourcen mit dem Tag „auto-delete“ und dem Wert „no“ für die angegebene Region angezeigt

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

Ausgabe:

```
Key         ResourceId            ResourceType Value
---         ----------            ------------ -----
auto-delete i-0f1bce234d5dd678b   instance     no
auto-delete vol-01d234aa5678901a2 volume       no
auto-delete vol-01234bfb5def6f7b8 volume       no
auto-delete vol-01ccb23f4c5e67890 volume       no
```

Beispiel 4: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' ohne Wert abgerufen und weitere Filter in der nächsten Pipe, um nur die Ressourcentypen 'Instanz' zu analysieren, und erstellt schließlich das 'ThisInstance' Tag für jede Instanzressource, wobei der Wert die Instanz-ID selbst ist

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"} |
 Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -ResourceId
 $_.ResourceId -Tag @{Key="ThisInstance";Value=$_.ResourceId}}
```

Beispiel 5: In diesem Beispiel werden Tags für alle Instance-Ressourcen sowie „Name“-Schlüssel abgerufen und in Tabellenform angezeigt

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Ausgabe:

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

Beispiel 6: In diesem Beispiel werden die Berechtigungen zum Abrufen von EC2-Tags mithilfe des DryRun Parameters überprüft, ohne sie tatsächlich abzurufen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Get-EC2Tag -DryRun $true
```

Ausgabe:

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Volume`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene EBS-Volume beschrieben.

```
Get-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size             : 30
SnapshotId       : snap-12345678
State            : in-use
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : standard
```

Beispiel 2: Dieses Beispiel beschreibt Ihre EBS-Volumes, die den Status „Verfügbar” aufweisen.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
```

```
Iops             : 60
KmsKeyId         :
Size             : 20
SnapshotId       : snap-12345678
State            : available
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
...
```

Beispiel 3: In diesem Beispiel werden alle Ihre EBS-Volumes beschrieben.

```
Get-EC2Volume
```

- API-Details finden Sie DescribeVolumes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VolumeAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VolumeAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Volumes beschrieben.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Ausgabe:

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- API-Details finden Sie DescribeVolumeAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VolumeStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VolumeStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Volumes beschrieben.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Ausgabe:

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Ausgabe:

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Ausgabe:

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- API-Details finden Sie DescribeVolumeStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2Vpc`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPC beschrieben.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

Beispiel 2: In diesem Beispiel wird die Standard-VPC beschrieben (es kann nur eine pro Region geben). Wenn Ihr Konto diese EC2-Classic Region unterstützt, gibt es keine Standard-VPC.

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

Ausgabe:

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
VpcId           : vpc-45678901
```

Beispiel 3: Dieses Beispiel beschreibt die VPCs, die dem angegebenen Filter entsprechen (d. h., sie haben eine CIDR, die dem Wert '10.0.0' entspricht. 0/16'und befinden sich im Status' verfügbar ').

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Beispiel 2: In diesem Beispiel werden alle Ihre VPCs beschrieben.

```
Get-EC2Vpc
```

- API-Details finden Sie DescribeVpcs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpcAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt das DnsSupport 'Enable'-Attribut.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Ausgabe:

```
EnableDnsSupport
----------------
True
```

Beispiel 2: Dieses Beispiel beschreibt das DnsHostnames 'Enable'-Attribut.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Ausgabe:

```
EnableDnsHostnames
------------------
True
```

- API-Details finden Sie DescribeVpcAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpcClassicLink`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcClassicLink`

Tools für PowerShell V5

Beispiel 1: Das obige Beispiel gibt alle VPCs mit ihrem ClassicLinkEnabled Bundesstaat für die Region zurück

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Ausgabe:

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- API-Details finden Sie DescribeVpcClassicLink in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpcClassicLinkDnsSupport`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcClassicLinkDnsSupport`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt den ClassicLink DNS-Unterstützungsstatus von VPCs für die Region eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Ausgabe:

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- API-Details finden Sie DescribeVpcClassicLinkDnsSupport in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcEndpoint`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden ein oder mehrere Ihrer VPC-Endpunkte für die Region eu-west-1 beschrieben. Anschließend leitet es die Ausgabe an den nächsten Befehl weiter, der die VpcEndpointId Eigenschaft auswählt und die Array-VPC-ID als String-Array zurückgibt

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty VpcEndpointId
```

Ausgabe:

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

Beispiel 2: Dieses Beispiel beschreibt alle VPC-Endpunkte für die Region eu-west-1 und wählt VpcEndpointId, ServiceName und PrivateDnsEnabled Eigenschaften aus VpcId, um sie in einem tabellarischen Format darzustellen

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Ausgabe:

```
VpcEndpointId          VpcId                 ServiceName
 PrivateDnsEnabled
-------------          -----                 -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
          True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
          True
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
          True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
          True
```

Beispiel 3: Dieses Beispiel exportiert das Richtliniendokument für den VPC-Endpunkt vpce-01a2ab3f4f5cc6f7d in eine JSON-Datei

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d | Select-
Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- API-Details finden Sie DescribeVpcEndpoints in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpcEndpointService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpcEndpointService`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der EC2 VPC-Endpunktservice mit dem angegebenen Filter beschrieben, in diesem Fall com.amazonaws.eu-west-1.ecs. Darüber hinaus erweitert es auch die ServiceDetails Eigenschaft und zeigt die Details an

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

Ausgabe:

```
AcceptanceRequired         : False
AvailabilityZones          : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames       : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                      : amazon
PrivateDnsName             : ecs.eu-west-1.amazonaws.com
ServiceName                : com.amazonaws.eu-west-1.ecs
ServiceType                : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Beispiel 2: In diesem Beispiel werden alle EC2-VPC-Endpunktdienste abgerufen und das ServiceNames passende „ssm“ zurückgegeben

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

Ausgabe:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- API-Details finden Sie DescribeVpcEndpointServices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpnConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EC2VpnConnection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung beschrieben.

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      : Amazon.EC2.Model.VpnConnectionOptions
Routes                       : {Amazon.EC2.Model.VpnStaticRoute}
State                        : available
Tags                         : {}
Type                         : ipsec.1
VgwTelemetry                 : {Amazon.EC2.Model.VgwTelemetry,
 Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel werden alle VPN-Verbindungen beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

Beispiel 4: In diesem Beispiel werden alle Ihre VPN-Verbindungen beschrieben.

```
Get-EC2VpnConnection
```

- API-Details finden Sie DescribeVpnConnections in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung`Get-EC2VpnGateway`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird jedes Virtual Private Gateway beschrieben, dessen Status entweder „Ausstehend“ oder „Verfügbar“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

Beispiel 3: In diesem Beispiel werden alle Ihre Virtual Private Gateways beschrieben.

```
Get-EC2VpnGateway
```

- API-Details finden Sie DescribeVpnGateways in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Grant-EC2SecurityGroupEgress`

Das folgende Codebeispiel zeigt die Verwendung. `Grant-EC2SecurityGroupEgress`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel definiert eine Ausgangsregel für die angegebene Sicherheitsgruppe für. EC2-VPC Die Regel gewährt Zugriff auf den angegebenen IP-Adressbereich über den TCP-Port 80. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der angegebenen Quellsicherheitsgruppe Zugriff über den TCP-Port 80 gewährt 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API-Details finden Sie AuthorizeSecurityGroupEgress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Grant-EC2SecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung. `Grant-EC2SecurityGroupIngress`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel definiert Eingangsregeln für eine Sicherheitsgruppe für. EC2-VPC Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Beachten Sie, dass Sie Sicherheitsgruppen identifizieren müssen, um die Sicherheitsgruppen-ID und nicht den Sicherheitsgruppennamen zu EC2-VPC verwenden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 3: Dieses Beispiel definiert Eingangsregeln für eine Sicherheitsgruppe für EC2-Classic. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
 IpRanges="203.0.113.25/32" }
```

```
Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
 $ip2 )
```

Beispiel 4: In PowerShell Version 2 müssen Sie verwenden, New-Object um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission @( $ip1,
 $ip2 )
```

Beispiel 5: Dieses Beispiel gewährt TCP-Port 8081 Zugriff von der angegebenen Quellsicherheitsgruppe (sg-1a2b3c4d) auf die angegebene Sicherheitsgruppe (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Beispiel 6: In diesem Beispiel wird der CIDR 5.5.5 hinzugefügt. 5/32 zu den Eingangsregeln der Sicherheitsgruppe sg-1234abcd für TCP-Port-22-Verkehr mit einer Beschreibung.

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
```

```
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Import-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein virtuelles Maschinen-Image mit einem einzigen Datenträger aus dem angegebenen Amazon-S3-Bucket mit einem Idempotenz-Token in Amazon EC2 importiert. Das Beispiel setzt voraus, dass eine VM-Import-Servicerolle mit dem Standardnamen „vmimport“ vorhanden ist, deren Richtlinie Amazon EC2 den Zugriff auf den angegebenen Bucket erlaubt. Dies wird im Abschnitt über die Voraussetzungen für den VM-Import erläutert. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters **`-RoleName`** an.

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

Ausgabe:

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
```

```
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- API-Details finden Sie ImportImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Import-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2KeyPair`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein öffentlicher Schlüssel nach EC2 importiert. Die erste Zeile speichert den Inhalt der öffentlichen Schlüsseldatei (*.pub) in der Variablen **`$publickey`**. Als Nächstes konvertiert das Beispiel das UTF8-Format der öffentlichen Schlüsseldatei in eine Base64-encoded Zeichenfolge und speichert die konvertierte Zeichenfolge in der Variablen. **`$pkbase64`** In der letzten Zeile wird der konvertierte öffentliche Schlüssel in EC2 importiert. Das Cmdlet gibt den Fingerabdruck und den Namen des Schlüssels als Ergebnisse zurück.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
 [System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Ausgabe:

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- API-Details finden Sie ImportKeyPair in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Import-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Import-EC2Snapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein VM-Datenträger-Image im Format „VMDK“ in einen Amazon EBS-Snapshot importiert. Das Beispiel erfordert eine VM-Import-Servicerolle mit dem Standardnamen 'vmimport' und einer Richtlinie, die Amazon EC2 den Zugriff auf den angegebenen Bucket ermöglicht, wie im **`VM Import Prequisites`** Thema unter http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html. So verwenden Sie eine benutzerdefinierte Rolle, geben Sie den Rollennamen mithilfe des Parameters an. **`-RoleName`**

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

Ausgabe:

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- API-Details finden Sie ImportSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Move-EC2AddressToVpc`

Das folgende Codebeispiel zeigt die Verwendung. `Move-EC2AddressToVpc`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine EC2-Instance mit der öffentlichen IP-Adresse 12.345.67.89 auf die EC2-VPC Plattform in der Region USA Ost (Nord-Virginia) verschoben.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Beispiel 2: Dieses Beispiel leitet die Ergebnisse eines Get-EC2Instance Befehls über die Pipeline an das Move-EC2AddressToVpc Cmdlet weiter. Der Get-EC2Instance Befehl ruft eine Instanz ab, die durch die Instanz-ID angegeben ist, und gibt dann die öffentliche IP-Adresseigenschaft der Instanz zurück.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- API-Details finden Sie MoveAddressToVpc in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Address`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird einer Instance in einer VPC eine Elastic-IP-Adresse zugewiesen.

```
New-EC2Address -Domain Vpc
```

Ausgabe:

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

Beispiel 2: In diesem Beispiel wird eine Elastic IP-Adresse zugewiesen, die zusammen mit einer Instance verwendet werden soll. EC2-Classic

```
New-EC2Address
```

Ausgabe:

```
AllocationId         Domain      PublicIp
```

```
------------         ------      --------
                     standard    203.0.113.17
```

- API-Details finden Sie AllocateAddress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2CustomerGateway

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2CustomerGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway erstellt.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Ausgabe:

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- API-Details finden Sie CreateCustomerGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2DhcpOption

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2DhcpOption`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Satz erstellt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
```

```
New-EC2DhcpOption -DhcpConfiguration $options
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jede DHCP-Option verwenden New-Object , um sie zu erstellen.

```
$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @("10.0.0.101","10.0.0.102")

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)
```

Ausgabe:

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- API-Details finden Sie CreateDhcpOptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2FlowLog`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein EC2-Flow-Protokoll für das Subnetz „subnet-1d234567“ im CloudWatch-Protokoll mit dem Namen „subnet1-log“ für den gesamten „REJECT“-Datenverkehr erstellt, wobei die Berechtigungen der Rolle „Admin“ verwendet werden

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Ausgabe:

```
ClientToken                                  FlowLogIds             Unsuccessful
-----------                                  ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- API-Details finden Sie CreateFlowLogs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Host

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Host`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird Ihrem Konto ein Dedicated Host für den angegebenen Instance-Typ und die angegebene Availability Zone zugewiesen

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType m4.xlarge -
Quantity 1
```

Ausgabe:

```
h-01e23f4cd567890f3
```

- API-Details finden Sie AllocateHosts in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2HostReservation

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2HostReservation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Reservierung mit der Nummer hro-0c1f23456789d0ab erworben, deren Konfigurationen mit denen Ihres Dedicated Hosts h-01e23f4cd567890f1 übereinstimmen

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

Ausgabe:

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- API-Details finden Sie PurchaseHostReservation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Image

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 versucht, die Instance vor dem Erstellen des Images ordnungsgemäß herunterzufahren, und startet die Instance nach Abschluss des Vorgangs neu.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

Beispiel 2: In diesem Beispiel wird aus der angegebenen Instance ein AMI mit dem spezifizierten Namen und der Beschreibung erstellt. Amazon EC2 erstellt das Image, ohne die Instance herunterzufahren und neu zu starten. Daher kann die Integrität des Dateisystems auf dem erstellten Image nicht garantiert werden.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

Beispiel 3: In diesem Beispiel wird ein AMI mit drei Volumes erstellt. Das erste Volume basiert auf einem Amazon-EBS-Snapshot. Bei dem zweiten Volume handelt es sich um ein leeres 100-

GiB-Amazon-EBS-Volume. Das dritte Volume ist ein Instance-Speicher-Volume. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
 "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
$ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
sdc";VirtualName="ephemeral0"})
```

- API-Details finden Sie CreateImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Instance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in EC2-Classic oder einer Standard-VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Beispiel 2: In diesem Beispiel wird eine einzelne Instance des angegebenen AMI in einer VPC gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

Beispiel 3: Um ein EBS-Volume oder ein Instance-Speicher-Volume hinzuzufügen, definieren Sie eine Blockgerät-Zuweisung und fügen Sie diese dem Befehl hinzu. In diesem Beispiel wird als Instance-Speicher-Volume hinzugefügt.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"
```

```
New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Beispiel 4: Um eines der aktuellen Windows-AMIs anzugeben, rufen Sie dessen AMI-ID mit ab Get-SSMLatestEC2Image. In diesem Beispiel wird eine Instance aus dem aktuellen Basis-AMI für Windows Server 2016 gestartet 2016.

```
$ami = (Get-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
 'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

Beispiel 5: Startet eine Instance in der angegebenen Dedicated-Host-Umgebung.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

Beispiel 6: Diese Anforderung startet zwei Instances und weist diesen Instances ein Tag mit dem Schlüssel „webserver” und dem Wert „production” zu. Die Anfrage wendet außerdem ein Tag mit dem Schlüssel „cost-center“ und dem Wert „cc123“ auf die erstellten Volumes an (in diesem Fall das Stammvolume für jede Instance).

```
$tag1 = @{ Key="webserver"; Value="production" }
$tag2 = @{ Key="cost-center"; Value="cc123" }

$tagspec1 = new-object Amazon.EC2.Model.TagSpecification
$tagspec1.ResourceType = "instance"
$tagspec1.Tags.Add($tag1)

$tagspec2 = new-object Amazon.EC2.Model.TagSpecification
$tagspec2.ResourceType = "volume"
$tagspec2.Tags.Add($tag2)

New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,
$tagspec2
```

Beispiel 7: In diesem Beispiel werden die Berechtigungen zum Starten einer EC2-Instance mithilfe des DryRun Parameters überprüft, ohne die Instance tatsächlich zu erstellen. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-key-
pair' -Region 'us-west-2' -DryRun $true
```

- API-Details finden Sie RunInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2InstanceExportTask

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2InstanceExportTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine angehaltene Instance, **`i-0800b00a00EXAMPLE`**, als virtuelle Festplatte (VHD) in den S3-Bucket **`testbucket-export-instances-2019`** exportiert. Die Zielumgebung ist**`Microsoft`**, und der Parameter Region wird hinzugefügt, da sich die Instanz in der **`us-east-1`** Region befindet, während die AWS Standardregion des Benutzers nicht us-east-1 ist. Um den Status der Exportaufgabe abzurufen, kopieren Sie den **`ExportTaskId`**-Wert aus den Ergebnissen dieses Befehls und führen Sie anschließend **`Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.`** aus.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket" -
TargetEnvironment Microsoft -Region us-east-1
```

Ausgabe:

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- API-Details finden Sie CreateInstanceExportTask in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2InternetGateway

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2InternetGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Internet-Gateway erstellt.

```
New-EC2InternetGateway
```

Ausgabe:

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- API-Details finden Sie CreateInternetGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2KeyPair`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Schlüsselpaar erstellt und der private PEM-encoded RSA-Schlüssel in einer Datei mit dem angegebenen Namen erfasst. Wenn Sie verwenden PowerShell, muss die Kodierung auf ASCII gesetzt sein, um einen gültigen Schlüssel zu generieren. Weitere Informationen finden Sie unter Erstellen, Anzeigen und Löschen von Amazon EC2-Schlüsselpaaren (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) im AWS Command Line Interface-Benutzerhandbuch.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- API-Details finden Sie CreateKeyPair in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkAcl`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Netzwerk-ACL für die angegebene VPC erstellt.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Ausgabe:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry, Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {}
VpcId        : vpc-12345678
```

- API-Details finden Sie CreateNetworkAcl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkAclEntry`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Eintrag für die angegebene Netzwerk-ACL erstellt. Die Regel erlaubt eingehenden Verkehr von überall (0.0.0. 0/0) auf dem UDP-Port 53 (DNS) in ein beliebiges zugehöriges Subnetz.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -RuleAction
 allow
```

- API-Details finden Sie CreateNetworkAclEntry in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2NetworkInterface`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle erstellt.

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Ausgabe:

```
Association        :
Attachment         :
AvailabilityZone   : us-west-2c
Description        : my network interface
Groups             : {my-security-group}
MacAddress         : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId            : 123456789012
PrivateDnsName     : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress   : 10.0.0.17
PrivateIpAddresses : {}
RequesterId        :
RequesterManaged   : False
SourceDestCheck    : True
Status             : pending
SubnetId           : subnet-1a2b3c4d
TagSet             : {}
VpcId              : vpc-12345678
```

- API-Details finden Sie CreateNetworkInterface in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2PlacementGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Platzierungsgruppe mit dem angegebenen Namen erstellt.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- API-Details finden Sie CreatePlacementGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Route

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Route`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle erstellt. Die Route entspricht dem gesamten Datenverkehr und leitet ihn an das angegebene Internet-Gateway weiter.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -GatewayId
 igw-1a2b3c4d
```

Ausgabe:

```
True
```

- API-Details finden Sie CreateRoute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2RouteTable

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2RouteTable`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Routing-Tabelle für die angegebene VPC erstellt.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Ausgabe:

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- API-Details finden Sie CreateRouteTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2ScheduledInstance`

Das folgende Codebeispiel zeigt die Verwendung`New-EC2ScheduledInstance`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene geplante Instance gestartet.

```
New-EC2ScheduledInstance -ScheduledInstanceId sci-1234-1234-1234-1234-123456789012 -
InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678`
-LaunchSpecification_SecurityGroupId sg-12345678
```

- API-Details finden Sie RunScheduledInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2ScheduledInstancePurchase`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2ScheduledInstancePurchase`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine geplante Instance gekauft.

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

Ausgabe:

```
AvailabilityZone            : us-west-2b
CreateDate                  : 1/25/2016 1:43:38 PM
HourlyPrice                 : 0.095
InstanceCount               : 1
InstanceType                : c4.large
NetworkPlatform             : EC2-VPC
NextSlotStartTime           : 1/31/2016 1:00:00 AM
Platform                    : Linux/UNIX
```

```
PreviousSlotEndTime         :
Recurrence                  : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId         : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours         : 32
TermEndDate                 : 1/31/2017 1:00:00 AM
TermStartDate               : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696
```

- API-Details finden Sie PurchaseScheduledInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2SecurityGroup

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2SecurityGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Sicherheitsgruppe für die angegebene VPC erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -
VpcId vpc-12345678
```

Ausgabe:

```
sg-12345678
```

Beispiel 2: In diesem Beispiel wird eine Sicherheitsgruppe für EC2-Classic erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Ausgabe:

```
sg-45678901
```

- API-Details finden Sie CreateSecurityGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Snapshot

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Snapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Snapshot des angegebenen Volumes erstellt.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Ausgabe:

```
DataEncryptionKeyId :
Description         : This is a test
Encrypted           : False
KmsKeyId            :
OwnerAlias          :
OwnerId             : 123456789012
Progress            :
SnapshotId          : snap-12345678
StartTime           : 12/22/2015 1:28:42 AM
State               : pending
StateMessage        :
Tags                : {}
VolumeId            : vol-12345678
VolumeSize          : 20
```

- API-Details finden Sie CreateSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2SpotDatafeedSubscription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Daten-Feed für eine Spot Instance erstellt.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Ausgabe:

```
Bucket  : amzn-s3-demo-bucket
Fault   :
```

```
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- API-Details finden Sie CreateSpotDatafeedSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Subnet

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Subnet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Subnetz mit dem angegebenen CIDR erstellt.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Ausgabe:

```
AvailabilityZone        : us-west-2c
AvailableIpAddressCount : 251
CidrBlock               : 10.0.0.0/24
DefaultForAz            : False
MapPublicIpOnLaunch     : False
State                   : pending
SubnetId                : subnet-1a2b3c4d
Tag                     : {}
VpcId                   : vpc-12345678
```

- API-Details finden Sie CreateSubnet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2Tag

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebenen Ressource ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist TagValue 'my'. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags aktualisiert oder der angegebenen Ressource hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

Beispiel 3: In PowerShell Version 2 müssen Sie verwenden, New-Object um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API-Details finden Sie CreateTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Volume`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Volume erstellt.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size             : 50
SnapshotId       :
```

```
State            : creating
Tags             : {}
VolumeId         : vol-12345678
VolumeType       : gp2
```

Beispiel 2: Diese Beispielanforderung erstellt ein Volume und versieht es mit einem Tag mit dem Schlüssel „stack“ und dem Wert „production.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- API-Details finden Sie CreateVolume in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2Vpc`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine VPC mit dem angegebenen CIDR erstellt. Amazon VPC erstellt für die VPC außerdem Folgendes: einen standardmäßigen DHCP-Optionssatz, eine Haupt-Routing-Tabelle und eine Standard-Netzwerk-ACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Ausgabe:

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- API-Details finden Sie CreateVpc in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2VpcEndpoint

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpcEndpoint`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer VPC-Endpunkt für den Service com.amazonaws.eu-west-1.s3 in der VPC vpc-0fc1ff23f45b678eb erstellt

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

Ausgabe:

```
ClientToken VpcEndpoint
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- API-Details finden Sie CreateVpcEndpoint in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EC2VpnConnection

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnConnection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine VPN-Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt. Die Ausgabe enthält die Konfigurationsinformationen, die Ihr Netzwerkadministrator benötigt, im XML-Format.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
```

```
Tags                          : {}
Type                          :
VgwTelemetry                  : {}
VpnConnectionId               : vpn-12345678
VpnGatewayId                  : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird die VPN-Verbindung hergestellt und die Konfiguration in einer Datei mit dem angegebenen Namen erfasst.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Beispiel 3: In diesem Beispiel wird eine VPN-Verbindung mit statischem Routing zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway erstellt.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API-Details finden Sie CreateVpnConnection in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2VpnConnectionRoute`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnConnectionRoute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene statische Route für die angegebene VPN-Verbindung erstellt.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- API-Details finden Sie CreateVpnConnectionRoute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `New-EC2VpnGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway erstellt.

```
New-EC2VpnGateway -Type ipsec.1
```

Ausgabe:

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- API-Details finden Sie CreateVpnGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2Address`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse der spezifizierten Instance in einer VPC zugeordnet.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Ausgabe:

```
eipassoc-12345678
```

Beispiel 2: Dieses Beispiel verknüpft die angegebene Elastic IP-Adresse mit der angegebenen Instance in EC2-Classic.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- API-Details finden Sie AssociateAddress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2DhcpOption`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebenen VPC der genannte DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel wird der angegebenen VPC der standardmäßige DHCP-Optionssatz zugeordnet.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- API-Details finden Sie AssociateDhcpOptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein AMI mithilfe der angegebenen Manifestdatei in Amazon S3 registriert.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- API-Details finden Sie RegisterImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-EC2PrivateIpAddress`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2PrivateIpAddress`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die genannten sekundären privaten IP-Adressen der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

Beispiel 2: In diesem Beispiel werden zwei sekundäre private IP-Adressen erstellt und der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- API-Details finden Sie AssignPrivateIpAddresses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Register-EC2RouteTable`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem festgelegten Subnetz zugeordnet.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
rtbassoc-12345678
```

- API-Details finden Sie AssociateRouteTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Address`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse für Instances in einer VPC freigegeben.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in EC2-Classic freigegeben.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API-Details finden Sie ReleaseAddress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2CapacityReservation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2CapacityReservation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Kapazitätsreservierung cr-0c1f2345db6f7cdba storniert

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation (CancelCapacityReservation)"
 on target "cr-0c1f2345db6f7cdba".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
True
```

- API-Details finden Sie CancelCapacityReservation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2CustomerGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2CustomerGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on Target
 "cgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteCustomerGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2DhcpOption`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2DhcpOption`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene DHCP-Optionssatz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
 "dopt-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteDhcpOptions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2FlowLog`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2FlowLog`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel entfernt das angegebene FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target
 "fl-01a2b3456a789c01".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). DeleteFlowLogs AWS -Tools für PowerShell

`Remove-EC2Host`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Host`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Host-ID h-0badafd1dcb2f3456 freigegeben

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
 "h-0badafd1dcb2f3456".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Successful            Unsuccessful
----------            ------------
{h-0badafd1dcb2f3456} {}
```

- API-Details finden Sie ReleaseHosts in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Instance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance beendet (die Instance kann sich im laufenden oder im angehaltenen Status befinden). Das Cmdlet fordert vor dem Fortfahren eine Bestätigung an, verwenden Sie den Schalter -Force, um den Prompt zu unterdrücken.

```
Remove-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API-Details finden Sie TerminateInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2InternetGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2InternetGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on Target
 "igw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteInternetGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2KeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2KeyPair`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Schlüsselpaar gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteKeyPair in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2NetworkAcl`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkAcl`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteNetworkAcl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkAclEntry`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus der spezifizierten Netzwerk-ACL entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on Target
 "acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteNetworkAclEntry in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2NetworkInterface`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2NetworkInterface`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on Target
 "eni-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteNetworkInterface in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2PlacementGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2PlacementGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
 "my-placement-group".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeletePlacementGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Route`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Route`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Route aus der spezifizierten Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteRoute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2RouteTable`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
 "rtb-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteRouteTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2SecurityGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für gelöscht. EC2-VPC Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
 "sg-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2-Classic gelöscht.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API-Details finden Sie DeleteSecurityGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Snapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Snapshot`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
 "snap-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteSnapshot in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2SpotDatafeedSubscription`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2SpotDatafeedSubscription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Daten-Feed für die Spot Instance gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
 (DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteSpotDatafeedSubscription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Subnet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target "subnet-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteSubnet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Beispiel 2: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Beispiel 3: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Beispiel 4: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API-Details finden Sie DeleteTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Volume`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Volume`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target "vol-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteVolume in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2Vpc`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2Vpc`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPC gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteVpc in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2VpnConnection`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnConnection`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPN-Verbindung gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
 "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteVpnConnection in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2VpnConnectionRoute`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnConnectionRoute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN-Verbindung entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on
 Target "vpn-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteVpnConnectionRoute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EC2VpnGateway`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EC2VpnGateway`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Virtual Private Gateway gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target
 "vgw-1a2b3c4d".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteVpnGateway in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Request-EC2SpotFleet`

Das folgende Codebeispiel zeigt die Verwendung. `Request-EC2SpotFleet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Spot-Flottenanforderung in der Availability Zone mit dem niedrigsten Preis für den angegebenen Instance-Typ erstellt. Wenn Ihr Konto EC2-VPC

nur Unterstützung bietet, startet die Spot-Flotte die Instances in der Availability Zone mit dem niedrigsten Preis, die über ein Standardsubnetz verfügt. Wenn Ihr Konto dies unterstützt EC2-Classic, startet die Spot-Flotte die Instances EC2-Classic in der Availability Zone mit dem niedrigsten Preis. Beachten Sie, dass der von Ihnen gezahlte Preis den angegebenen Spot-Preis für die Anforderung nicht überschreitet.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-fleet-
role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- API-Details finden Sie RequestSpotFleet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Request-EC2SpotInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Request-EC2SpotInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine einmalige Spot Instance im angegebenen Subnetz angefordert. Beachten Sie, dass die Sicherheitsgruppe für die VPC erstellt werden muss, die das angegebene Subnetz enthält, und dass sie über die Netzwerkschnittstelle anhand der ID angegeben werden muss. Wenn Sie eine Netzwerkschnittstelle angeben, müssen Sie die Subnetz-ID unter Verwendung der Netzwerkschnittstelle angeben.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
```

```
-LaunchSpecification_NetworkInterface $n
```

Ausgabe:

```
ActualBlockHourlyPrice   :
AvailabilityZoneGroup    :
BlockDurationMinutes     : 0
CreateTime               : 12/26/2015 7:44:10 AM
Fault                    :
InstanceId               :
LaunchedAvailabilityZone :
LaunchGroup              :
LaunchSpecification      : Amazon.EC2.Model.LaunchSpecification
ProductDescription       : Linux/UNIX
SpotInstanceRequestId    : sir-12345678
SpotPrice                : 0.050000
State                    : open
Status                   : Amazon.EC2.Model.SpotInstanceStatus
Tags                     : {}
Type                     : one-time
```

- API-Details finden Sie RequestSpotInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-EC2ImageAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2ImageAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Attribut „LaunchPermission“ auf seinen Standardwert zurückgesetzt. Standardmäßig sind alle AMIs privat.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API-Details finden Sie ResetImageAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-EC2InstanceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2InstanceAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel setzt das Attribut 'sriov für die NetSupport angegebene Instanz zurück.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Beispiel 2: In diesem Beispiel wird das Attribut „ebsOptimized“ für die angegebene Instance zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Beispiel 3: Dieses Beispiel setzt das DestCheck 'source'-Attribut für die angegebene Instanz zurück.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Beispiel 4: Dieses Beispiel setzt das ApiTermination 'Disable'-Attribut für die angegebene Instanz zurück.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Beispiel 5: Dieses Beispiel setzt das InitiatedShutdownBehavior 'instanz'-Attribut für die angegebene Instanz zurück.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- API-Details finden Sie ResetInstanceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-EC2NetworkInterfaceAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2NetworkInterfaceAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel setzt die source/destination Prüfung für die angegebene Netzwerkschnittstelle zurück.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -SourceDestCheck
```

- API-Details finden Sie ResetNetworkInterfaceAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-EC2SnapshotAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-EC2SnapshotAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des spezifizierten Snapshots zurückgesetzt.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- API-Details finden Sie ResetSnapshotAttribute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Restart-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Restart-EC2Instance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance neu gestartet.

```
Restart-EC2Instance -InstanceId i-12345678
```

- API-Details finden Sie RebootInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Revoke-EC2SecurityGroupEgress`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-EC2SecurityGroupEgress`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Regel für die angegebene Sicherheitsgruppe für EC2-VPC entfernt. Dadurch wird der Zugriff auf den angegebenen IP-Adressbereich am TCP-Port 80 aufgehoben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf die angegebene Quellsicherheitsgruppe auf TCP-Port 80 widerrufen.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
 @( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API-Details finden Sie RevokeSecurityGroupEgress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Revoke-EC2SecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung. `Revoke-EC2SecurityGroupIngress`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel widerruft den Zugriff auf den TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für. EC2-VPC Beachten Sie, dass

Sie Sicherheitsgruppen identifizieren müssen, um die Sicherheitsgruppen-ID und nicht den Sicherheitsgruppennamen zu EC2-VPC verwenden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: Dieses Beispiel widerruft den Zugriff auf den TCP-Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2-Classic. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22"; IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Beispiel 4: In PowerShell Version 2 müssen Sie verwenden, New-Object um das IpPermission Objekt zu erstellen.

```
$ip = New-Object Amazon.EC2.Model.IpPermission
$ip.IpProtocol = "tcp"
$ip.FromPort = 22
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- API-Details finden Sie RevokeSecurityGroupIngress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Send-EC2InstanceStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Send-EC2InstanceStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird Feedback zum Status der angegebenen Instance gegeben.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- API-Details finden Sie ReportInstanceStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-EC2NetworkAclAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2NetworkAclAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerk-ACL dem Subnetz für die angegebene Netzwerk-ACL-Zuordnung zugeordnet.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

Ausgabe:

```
aclassoc-87654321
```

- API-Details finden Sie ReplaceNetworkAclAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-EC2NetworkAclEntry`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2NetworkAclEntry`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Eintrag für die spezifizierte Netzwerk-ACL ersetzt. Die neue Regel erlaubt eingehenden Verkehr von der angegebenen Adresse zu jedem zugehörigen Subnetz.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -
RuleAction allow
```

- API-Details finden Sie ReplaceNetworkAclEntry in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-EC2Route`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2Route`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Route für die spezifizierte Routing-Tabelle ersetzt. Die neue Route leitet den angegebenen Datenverkehr an das spezifizierte Virtual Private Gateway.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -GatewayId
 vgw-1a2b3c4d
```

- API-Details finden Sie ReplaceRoute in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-EC2RouteTableAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Set-EC2RouteTableAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routing-Tabelle zugeordnet.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

Ausgabe:

```
rtbassoc-87654321
```

- API-Details finden Sie ReplaceRouteTableAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung`Start-EC2Instance`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance gestartet.

```
Start-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

Beispiel 2: In diesem Beispiel werden die angegebenen Instances gestartet.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Beispiel 3: In diesem Beispiel wird die Gruppe der Instances gestartet, die derzeit angehalten sind. Die Instanzobjekte, die von zurückgegeben werden, Get-EC2Instance werden über die Pipeline an weitergeleitet. Start-EC2Instance Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name"; Values="stopped"}).Instances
 | Start-EC2Instance
```

Beispiel 4: In PowerShell Version 2 müssen Sie verwenden, New-Object um den Filter für den Parameter Filter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

Beispiel 5: In diesem Beispiel werden die Berechtigungen für das Starten einer EC2-Instance mithilfe des DryRun Parameters überprüft, ohne die Instance tatsächlich zu starten. Hinweis: Im Erfolgsfall wird eine Ausnahme ausgelöst, was das erwartete Verhalten ist.

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- API-Details finden Sie StartInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-EC2InstanceMonitoring`

Das folgende Codebeispiel zeigt die Verwendung. `Start-EC2InstanceMonitoring`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance aktiviert.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API-Details finden Sie MonitorInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-EC2ImportTask`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2ImportTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Importaufgabe (entweder Snapshot- oder Bildimport) abgebrochen. Gegebenenfalls kann über den **`-CancelReason`**-Parameter ein Grund angegeben werden.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- API-Details finden Sie CancelImportTask in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-EC2Instance`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2Instance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Instance angehalten.

```
Stop-EC2Instance -InstanceId i-12345678
```

Ausgabe:

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- API-Details finden Sie StopInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-EC2InstanceMonitoring`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2InstanceMonitoring`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instance deaktiviert.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- API-Details finden Sie UnmonitorInstances in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-EC2SpotFleetRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2SpotFleetRequest`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen und die zugehörigen Spot Instances beendet.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Beispiel 2: In diesem Beispiel wird die angegebene Spot-Flottenanforderung abgebrochen, ohne die zugehörigen Spot Instances zu beenden.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- API-Details finden Sie CancelSpotFleetRequests in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-EC2SpotInstanceRequest`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-EC2SpotInstanceRequest`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anforderung abgebrochen.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- API-Details finden Sie CancelSpotInstanceRequests in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-EC2Address`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2Address`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Elastic-IP-Adresse von der angegebenen Instance in einer VPC getrennt.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse von der angegebenen Instance in getrennt. EC2-Classic

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API-Details finden Sie DisassociateAddress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-EC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung des angegebenen AMI aufgehoben.

```
Unregister-EC2Image -ImageId ami-12345678
```

- API-Details finden Sie DeregisterImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-EC2PrivateIpAddress`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2PrivateIpAddress`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- API-Details finden Sie UnassignPrivateIpAddresses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-EC2RouteTable`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-EC2RouteTable`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Zuordnung zwischen einer Routing-Tabelle und einem Subnetz entfernt.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API-Details finden Sie DisassociateRouteTable in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-EC2SecurityGroupRuleIngressDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Update-EC2SecurityGroupRuleIngressDescription`

Tools für PowerShell V5

Beispiel 1: Aktualisiert die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
```

```
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithUpdatedDescription
```

Beispiel 2: Entfernt die Beschreibung einer vorhandenen eingehenden Sicherheitsgruppenregel (indem der Parameter in der Anforderung weggelassen wird).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
 "sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId $existingInboundRule.GroupId
 -SecurityGroupRuleDescription $ruleWithoutDescription
```

- API-Details finden Sie UpdateSecurityGroupRuleDescriptionsIngress in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon ECR-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon ECR verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-ECRLoginCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECRLoginCommand`

Tools für PowerShell V5

Beispiel 1: Gibt ein PSObject zurück, das Anmeldeinformationen enthält, mit denen Sie sich bei jeder Amazon-ECR-Registrierung authentifizieren können, auf die Ihr IAM-Prinzipal Zugriff hat. Die für den Aufruf zum Abrufen des Autorisierungstokens erforderlichen Anmeldeinformationen und der Regionsendpunkt werden aus den Shell-Standardeinstellungen (die mit den Cmdlets **`Set-AWSCredential/Set-DefaultAWSRegion`** oder **`Initialize-AWSDefaultConfiguration`** eingerichtet wurden) abgerufen. Sie können die Command-Eigenschaft mit verwenden Invoke-Expression , um sich bei der angegebenen Registrierung anzumelden, oder die zurückgegebenen Anmeldeinformationen in anderen Tools verwenden, für die eine Anmeldung erforderlich ist.

```
Get-ECRLoginCommand
```

Ausgabe:

```
Username      : AWS
Password      : eyJwYXlsb2Fk...kRBVEFfS0VZIn0=
ProxyEndpoint : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
Endpoint      : https://123456789012.dkr.ecr.us-west-2.amazonaws.com
ExpiresAt     : 9/26/2017 6:08:23 AM
Command       : docker login --username AWS --password
 eyJwYXlsb2Fk...kRBVEFfS0VZIn0= https://123456789012.dkr.ecr.us-west-2.amazonaws.com
```

Beispiel 2: Ruft ein PSObject mit Anmeldeinformationen ab, das Sie als Eingabe für einen Docker-Anmeldebefehl verwenden. Sie können jede beliebige Registrierungs-URI von Amazon ECR für die Authentifizierung angeben, solange Ihr IAM-Prinzipal Zugriff auf diese Registrierung hat.

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin
 012345678910.dkr.ecr.us-east-1.amazonaws.com
```

- API-Details finden Sie Get-ECRLoginCommand in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon ECS-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon ECS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-ECSClusterDetail`

Das folgende Codebeispiel zeigt die Verwendung`Get-ECSClusterDetail`.

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet beschreibt einen oder mehrere Ihrer ECS-Cluster.

```
Get-ECSClusterDetail -Cluster "LAB-ECS-CL" -Include SETTINGS | Select-Object *
```

Ausgabe:

```
LoggedAt         : 12/27/2019 9:27:41 PM
Clusters         : {LAB-ECS-CL}
Failures         : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 396
HttpStatusCode   : OK
```

- API-Details finden Sie DescribeClusters in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ECSClusterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSClusterList`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet gibt eine Liste vorhandener ECS-Cluster zurück.

```
Get-ECSClusterList
```

Ausgabe:

```
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS-CL
arn:aws:ecs:us-west-2:012345678912:cluster/LAB-ECS
```

- API-Details finden Sie ListClusters in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ECSClusterService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSClusterService`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Services aufgeführt, die in Ihrem Standard-Cluster ausgeführt werden.

```
Get-ECSClusterService
```

Beispiel 2: In diesem Beispiel werden alle Services aufgeführt, die im angegebenen Cluster ausgeführt werden.

```
Get-ECSClusterService -Cluster myCluster
```

- API-Details finden Sie ListServices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ECSService`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service aus Ihrem Standard-Cluster abrufen können.

```
Get-ECSService -Service my-hhtp-service
```

Beispiel 2: Dieses Beispiel zeigt, wie Sie Details zu einem bestimmten Service abrufen können, der in dem genannten Cluster ausgeführt wird.

```
Get-ECSService -Cluster myCluster -Service my-hhtp-service
```

- API-Details finden Sie DescribeServices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ECSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `New-ECSCluster`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet erstellt einen Amazon-ECS-Cluster.

```
New-ECSCluster -ClusterName "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="enabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
```

```
Status                              : ACTIVE
Tags                                : {}
```

- API-Details finden Sie CreateCluster in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `New-ECSService`

Tools für PowerShell V5

Beispiel 1: Dieser Beispielbefehl erstellt in Ihrem Standard-Cluster einen Service namens „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10
```

Beispiel 2: Dieser Beispielbefehl erstellt einen Service hinter einem Load Balancer in Ihrem Standardcluster mit dem Namen „ecs-simple-service“. Der Service verwendet die Aufgabendefinition „ecs-demo“ und verwaltet zehn Instanziierungen dieser Aufgabe.

```
$lb = @{
    LoadBalancerName = "EC2Contai-EcsElast-S06278JGSJCM"
    ContainerName = "simple-demo"
    ContainerPort = 80
}
New-ECSService -ServiceName ecs-simple-service -TaskDefinition ecs-demo -
DesiredCount 10 -LoadBalancer $lb
```

- API-Details finden Sie CreateService in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ECSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ECSCluster`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht den angegebenen ECS-Cluster. Sie müssen die Registrierung aller Container-Instances aus diesem Cluster aufheben, bevor Sie ihn löschen können.

```
Remove-ECSCluster -Cluster "LAB-ECS"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ECSCluster (DeleteCluster)" on target "LAB-ECS".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteCluster in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ECSService`

Tools für PowerShell V5

Beispiel 1: Löscht den Service mit dem Namen „my-http-service“ im Standard-Cluster. Der Service muss eine gewünschte Anzahl und eine laufende Anzahl von 0 aufweisen, bevor Sie ihn löschen können. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Um die Bestätigungsaufforderung zu umgehen, fügen Sie den Schalter -Force hinzu.

```
Remove-ECSService -Service my-http-service
```

Beispiel 2: Löscht den Service mit dem Namen „my-http-service“ im genannten Cluster.

```
Remove-ECSService -Cluster myCluster -Service my-http-service
```

- API-Details finden Sie DeleteService in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-ECSClusterSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ECSClusterSetting`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet ändert die Einstellungen, die für einen ECS-Cluster verwendet werden.

```
Update-ECSClusterSetting -Cluster "LAB-ECS-CL" -Setting @{Name="containerInsights";
 Value="disabled"}
```

Ausgabe:

```
ActiveServicesCount               : 0
Attachments                       : {}
AttachmentsStatus                 :
CapacityProviders                 : {}
ClusterArn                        : arn:aws:ecs:us-west-2:012345678912:cluster/LAB-
ECS-CL
ClusterName                       : LAB-ECS-CL
DefaultCapacityProviderStrategy   : {}
PendingTasksCount                 : 0
RegisteredContainerInstancesCount : 0
RunningTasksCount                 : 0
Settings                          : {containerInsights}
Statistics                        : {}
Status                            : ACTIVE
Tags                              : {}
```

- API-Details finden Sie UpdateClusterSettings in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-ECSService`

Das folgende Codebeispiel zeigt die Verwendung. `Update-ECSService`

Tools für PowerShell V5

Beispiel 1: Dieser Beispielbefehl aktualisiert den Service „my-http-service“, sodass er die Aufgabendefinition „amazon-ecs-sample“ verwendet.

```
Update-ECSService -Service my-http-service -TaskDefinition amazon-ecs-sample
```

Beispiel 2: Dieser Beispielbefehl aktualisiert die gewünschte Anzahl des Service „my-http-service“ auf 10.

```
Update-ECSService -Service my-http-service -DesiredCount 10
```

- API-Details finden Sie UpdateService in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon EFS-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon EFS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Edit-EFSMountTargetSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`Edit-EFSMountTargetSecurityGroup`.

Tools für PowerShell V5

Beispiel 1: Aktualisiert die Sicherheitsgruppen, die für das angegebene Mounting-Ziel aktiv sind Es können bis zu 5 im Format „sg-xxxxxxxx“ angegeben werden.

```
Edit-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d -SecurityGroup sg-
group1,sg-group3
```

- API-Details finden Sie ModifyMountTargetSecurityGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSFileSystem`

Tools für PowerShell V5

Beispiel 1: Gibt die Sammlung aller Dateisysteme zurück, die sich im Besitz des Kontos des Anrufers in der Region befinden.

```
Get-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : available
Name                 :
NumberOfMountTargets : 0
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize

CreationTime         : 5/26/2015 4:06:23 PM
CreationToken        : 2b4daa14-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-4d3c2b1a
...
```

Beispiel 2: Gibt die Details des angegebenen Dateisystems zurück.

```
Get-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

Beispiel 3: Gibt die Details eines Dateisystems unter Verwendung des bei der Erstellung des Dateisystems angegebenen Idempotenz-Erstellungstokens zurück.

```
Get-EFSFileSystem -CreationToken 1a2bff54-85e0-4747-bd95-7bc172c4f555
```

- API-Details finden Sie DescribeFileSystems in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSMountTarget`

Tools für PowerShell V5

Beispiel 1: Gibt die Sammlung von Mount-Zielen zurück, die dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSMountTarget -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : available
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

- API-Details finden Sie DescribeMountTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EFSMountTargetSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSMountTargetSecurityGroup`

Tools für PowerShell V5

Beispiel 1: Gibt die IDs der Sicherheitsgruppen zurück, die derzeit der Netzwerkschnittstelle zugewiesen sind, welche dem Mount-Ziel zugeordnet ist.

```
Get-EFSMountTargetSecurityGroup -MountTargetId fsmt-1a2b3c4d
```

Ausgabe:

```
sg-1a2b3c4d
```

- API-Details finden Sie DescribeMountTargetSecurityGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EFSTag`

Tools für PowerShell V5

Beispiel 1: Gibt die Sammlung von Tags zurück, die derzeit dem angegebenen Dateisystem zugeordnet sind.

```
Get-EFSTag -FileSystemId fs-1a2b3c4d
```

Ausgabe:

```
Key         Value
---         -----
Name        My File System
tagkey1     tagvalue1
tagkey2     tagvalue2
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSFileSystem`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues, leeres Dateisystem. Das Token, das zur Gewährleistung einer idempotenten Erstellung verwendet wird, wird automatisch generiert und kann über das **`CreationToken`**-Element des zurückgegebenen Objekts abgerufen werden.

```
New-EFSFileSystem
```

Ausgabe:

```
CreationTime         : 5/26/2015 4:02:38 PM
CreationToken        : 1a2bff54-85e0-4747-bd95-7bc172c4f555
FileSystemId         : fs-1a2b3c4d
LifeCycleState       : creating
Name                 :
NumberOfMountTargets : 0
```

```
OwnerId              : 123456789012
SizeInBytes          : Amazon.ElasticFileSystem.Model.FileSystemSize
```

Beispiel 2: Erstellt ein neues, leeres Dateisystem unter Verwendung eines benutzerdefinierten Tokens, um eine idempotente Erstellung sicherzustellen.

```
New-EFSFileSystem -CreationToken "MyUniqueToken"
```

- API-Details finden Sie CreateFileSystem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSMountTarget`

Tools für PowerShell V5

Beispiel 1: Erstellt ein neues Mounting-Ziel für ein Dateisystem. Das angegebene Subnetz wird verwendet, um die Virtual Private Cloud (VPC) zu bestimmen, in der das Mount-Ziel erstellt wird, und die IP-Adresse, die automatisch zugewiesen wird (aus dem Adressbereich des Subnetzes). Die zugewiesene IP-Adresse kann verwendet werden, um dieses Dateisystem auf einer Amazon-EC2-Instance zu mounten. Da keine Sicherheitsgruppen angegeben wurden, ist die für das Ziel erstellte Netzwerkschnittstelle der Standardsicherheitsgruppe für die VPC des Subnetzes zugeordnet.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
FileSystemId       : fs-1a2b3c4d
IpAddress          : 10.0.0.131
LifeCycleState     : creating
MountTargetId      : fsmt-1a2b3c4d
NetworkInterfaceId : eni-1a2b3c4d
OwnerId            : 123456789012
SubnetId           : subnet-1a2b3c4d
```

Beispiel 2: Erzeugt ein neues Mount-Ziel für das angegebene Dateisystem mit automatisch zugewiesener IP-Adresse. Die für das Mount-Ziel erstellte Netzwerkschnittstelle ist den

angegebenen Sicherheitsgruppen zugeordnet (es können bis zu 5 im Format „sg-xxxxxxxx“ angegeben werden).

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -
SecurityGroup sg-group1,sg-group2,sg-group3
```

Beispiel 3: Erstellt ein neues Mount-Ziel für das angegebene Dateisystem mit der angegebenen IP-Adresse.

```
New-EFSMountTarget -FileSystemId fs-1a2b3c4d -SubnetId subnet-1a2b3c4d -IpAddress
 10.0.0.131
```

- API-Details finden Sie CreateMountTarget in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `New-EFSTag`

Tools für PowerShell V5

Beispiel 1: Wendet die Sammlung von Tags auf das angegebene Dateisystem an. Wenn ein Tag mit dem angegebenen Schlüssel bereits im Dateisystem vorhanden ist, wird der Wert des Tags aktualisiert.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag
 @{Key="tagkey1";Value="tagvalue1"},@{Key="tagkey2";Value="tagvalue2"}
```

Beispiel 2: Legt das Name-Tag für das angegebene Dateisystem fest. Dieser Wert wird zusammen mit anderen Dateisystemdetails zurückgegeben, wenn das Get-EFSFileSystem Cmdlet verwendet wird.

```
New-EFSTag -FileSystemId fs-1a2b3c4d -Tag @{Key="Name";Value="My File System"}
```

- API-Details finden Sie CreateTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EFSFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSFileSystem`

Tools für PowerShell V5

Beispiel 1: Löscht das angegebene Dateisystem, das nicht mehr verwendet wird (wenn das Dateisystem Mount-Ziele hat, müssen diese zuerst entfernt werden). Sie werden um eine Bestätigung gebeten, bevor das Cmdlet fortfährt. Um die Bestätigung zu unterdrücken, verwenden Sie den **`-Force`**-Schalter.

```
Remove-EFSFileSystem -FileSystemId fs-1a2b3c4d
```

- API-Details finden Sie DeleteFileSystem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EFSMountTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSMountTarget`

Tools für PowerShell V5

Beispiel 1: Löscht das angegebene Mount-Ziel. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird. Verwenden Sie den **`-Force`**-Schalter, um das Prompt zu unterdrücken. Beachten Sie, dass dieser Vorgang alle Mounts des Dateisystems über das Ziel zwangsweise beendet. Sie sollten daher in Erwägung ziehen, den Mount des Dateisystems aufzuheben, bevor Sie diesen Befehl ausführen, sofern dies möglich ist.

```
Remove-EFSMountTarget -MountTargetId fsmt-1a2b3c4d
```

- API-Details finden Sie DeleteMountTarget in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EFSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EFSTag`

Tools für PowerShell V5

Beispiel 1: Löscht die Sammlung eines oder mehrerer Tags aus einem Dateisystem. Sie werden um eine Bestätigung gebeten, bevor das Cmdlet fortfährt. Um die Bestätigung zu unterdrücken, verwenden Sie den **`-Force`**-Schalter.

```
Remove-EFSTag -FileSystemId fs-1a2b3c4d -TagKey "tagkey1","tagkey2"
```

- API-Details finden Sie DeleteTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon EKS-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Amazon EKS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-EKSResourceTag`.

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet ordnet die angegebenen Tags einer Ressource mit dem angegebenen resourceArn zu.

```
Add-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD" -
Tag @{Name = "EKSPRODCLUSTER"}
```

- API-Details finden Sie TagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSCluster`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Amazon-EKS-Cluster zurück.

```
Get-EKSCluster -Name "PROD"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/PROD
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 6:46:17 AM
Endpoint             : https://669608765450FBBE54D1D78A3D71B72C.gr8.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : PROD
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : ACTIVE
Tags                 : {}
Version              : 1.14
```

- API-Details finden Sie DescribeCluster in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSClusterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSClusterList`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Cluster in Ihrem AWS-Konto in der angegebenen Region auf.

```
Get-EKSClusterList
```

Ausgabe:

```
PROD
```

- API-Details finden Sie ListClusters in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSFargateProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSFargateProfile`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen zu einem Fargate-Profil zurück. AWS

```
Get-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : ACTIVE
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- API-Details finden Sie DescribeFargateProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSFargateProfileList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSFargateProfileList`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die AWS Fargate-Profile auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSFargateProfileList -ClusterName "TEST"
```

Ausgabe:

```
EKSFargate
EKSFargateProfile
```

- API-Details finden Sie ListFargateProfiles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSNodegroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSNodegroup`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über eine Amazon-EKS-Knotengruppe zurück.

```
Get-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
```

```
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- API-Details finden Sie DescribeNodegroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSNodegroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSNodegroupList`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die Amazon EKS-Knotengruppen auf, die dem angegebenen Cluster in Ihrem AWS-Konto in der angegebenen Region zugeordnet sind.

```
Get-EKSNodegroupList -ClusterName PROD
```

Ausgabe:

```
ProdEKSNodeGroup
```

- API-Details finden Sie ListNodegroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die Tags für eine Amazon-EKS-Ressource auf.

```
Get-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
```

Ausgabe:

```
Key  Value
---  -----
```

```
Name EKSPRODCLUSTER
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSUpdate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSUpdate`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet gibt beschreibende Informationen über ein Update für Ihren Amazon-EKS-Cluster oder die zugewiesene verwaltete Knotengruppe zurück.

```
Get-EKSUpdate -Name "PROD" -UpdateId "ee708232-7d2e-4ed7-9270-d0b5176f0726"
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : Successful
Type      : LoggingUpdate
```

- API-Details finden Sie DescribeUpdate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-EKSUpdateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-EKSUpdateList`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet listet die Updates auf, die einem Amazon EKS-Cluster oder einer verwalteten Knotengruppe in Ihrem AWS-Konto, in der angegebenen Region zugeordnet sind.

```
Get-EKSUpdateList -Name "PROD"
```

Ausgabe:

```
ee708232-7d2e-4ed7-9270-d0b5176f0726
```

- API-Details finden Sie ListUpdates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EKSCluster

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSCluster`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer Cluster namens „prod“ erstellt.

```
New-EKSCluster -Name prod -ResourcesVpcConfig
 @{SubnetIds=@("subnet-0a1b2c3d","subnet-3a2b1c0d");SecurityGroupIds="sg-6979fe18"}
 -RoleArn "arn:aws:iam::012345678901:role/eks-service-role"
```

Ausgabe:

```
Arn                  : arn:aws:eks:us-west-2:012345678901:cluster/prod
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/10/2018 9:25:31 PM
Endpoint             :
Name                 : prod
PlatformVersion      : eks.3
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678901:role/eks-service-role
Status               : CREATING
Version              : 1.10
```

- API-Details finden Sie CreateCluster in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-EKSFargateProfile

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSFargateProfile`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet erstellt ein AWS Fargate-Profil für Ihren Amazon EKS-Cluster. Sie müssen mindestens ein Fargate-Profil in einem Cluster haben, um Pods auf der Fargate-Infrastruktur planen zu können.

```
New-EKSFargateProfile -FargateProfileName EKSFargateProfile -ClusterName TEST -
Subnet "subnet-02f6ff500ff2067a0", "subnet-0cd976f08d5fbfaae" -PodExecutionRoleArn
 arn:aws:iam::012345678912:role/AmazonEKSFargatePodExecutionRole -Selector
 @{Namespace="default"}
```

Ausgabe:

```
ClusterName         : TEST
CreatedAt           : 12/26/2019 12:38:21 PM
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargateProfile/20b7a11b-8292-41c1-bc56-ffa5e60f6224
FargateProfileName  : EKSFargateProfile
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : CREATING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- API-Details finden Sie CreateFargateProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-EKSNodeGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-EKSNodeGroup`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet erstellt eine verwaltete Worker-Kknotengruppe für einen Amazon EKS-Cluster. Sie können nur eine Knotengruppe für Ihren Cluster erstellen, die der aktuellen Kubernetes-Version für den Cluster entspricht. Alle Knotengruppen werden mit der neuesten AMI-Release-Version für die jeweilige Kubernetes-Nebenversion des Clusters erstellt.

```
New-EKSNodeGroup -NodeGroupName "ProdEKSNodeGroup" -AmiType "AL2_x86_64"
 -DiskSize 40 -ClusterName "PROD" -ScalingConfig_DesiredSize 2 -
ScalingConfig_MinSize 2 -ScalingConfig_MaxSize 5 -InstanceType t3.large
 -NodeRole "arn:aws:iam::012345678912:role/NodeInstanceRole" -Subnet
 "subnet-0d1a9fff35efa7691","subnet-0a3f4928edbc224d4"
```

Ausgabe:

```
AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 10:16:45 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      :
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : CREATING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- API-Details finden Sie CreateNodegroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EKSCluster`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSCluster`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht die Steuerebene des Amazon-EKS-Clusters.

```
Remove-EKSCluster -Name "DEV-KUBE-CL"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSCluster (DeleteCluster)" on target "DEV-KUBE-
CL".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

```
Arn                  : arn:aws:eks:us-west-2:012345678912:cluster/DEV-KUBE-CL
CertificateAuthority : Amazon.EKS.Model.Certificate
ClientRequestToken   :
CreatedAt            : 12/25/2019 9:33:25 AM
Endpoint             : https://02E6D31E3E4F8C15D7BE7F58D527776A.yl4.us-
west-2.eks.amazonaws.com
Identity             : Amazon.EKS.Model.Identity
Logging              : Amazon.EKS.Model.Logging
Name                 : DEV-KUBE-CL
PlatformVersion      : eks.7
ResourcesVpcConfig   : Amazon.EKS.Model.VpcConfigResponse
RoleArn              : arn:aws:iam::012345678912:role/eks-iam-role
Status               : DELETING
Tags                 : {}
Version              : 1.14
```

- API-Details finden Sie DeleteCluster in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EKSFargateProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSFargateProfile`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht ein Fargate-Profil. AWS Wenn Sie ein Fargate-Profil löschen, werden alle Pods, die auf Fargate ausgeführt werden und mit diesem Profil erstellt wurden, ebenfalls gelöscht.

```
Remove-EKSFargateProfile -FargateProfileName "EKSFargate" -ClusterName "TEST"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSFargateProfile (DeleteFargateProfile)" on target
 "EKSFargate".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

ClusterName         : TEST
CreatedAt           : 12/26/2019 12:34:47 PM
```

```
FargateProfileArn   : arn:aws:eks:us-east-2:012345678912:fargateprofile/TEST/
EKSFargate/42b7a119-e16b-a279-ce97-bdf303adec92
FargateProfileName  : EKSFargate
PodExecutionRoleArn : arn:aws:iam::012345678912:role/
AmazonEKSFargatePodExecutionRole
Selectors           : {Amazon.EKS.Model.FargateProfileSelector}
Status              : DELETING
Subnets             : {subnet-0cd976f08d5fbfaae, subnet-02f6ff500ff2067a0}
Tags                : {}
```

- API-Details finden Sie DeleteFargateProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EKSNodegroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSNodegroup`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht eine Amazon-EKS-Knotengruppe für einen Cluster.

```
Remove-EKSNodegroup -NodegroupName "ProdEKSNodeGroup" -ClusterName "PROD"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSNodegroup (DeleteNodegroup)" on target
 "ProdEKSNodeGroup".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

AmiType        : AL2_x86_64
ClusterName    : PROD
CreatedAt      : 12/25/2019 10:16:45 AM
DiskSize       : 40
Health         : Amazon.EKS.Model.NodegroupHealth
InstanceTypes  : {t3.large}
Labels         : {}
ModifiedAt     : 12/25/2019 11:01:16 AM
NodegroupArn   : arn:aws:eks:us-west-2:012345678912:nodegroup/PROD/
ProdEKSNodeGroup/7eb79e47-82b6-04d9-e984-95110db6fa85
```

```
NodegroupName  : ProdEKSNodeGroup
NodeRole       : arn:aws:iam::012345678912:role/NodeInstanceRole
ReleaseVersion : 1.14.7-20190927
RemoteAccess   :
Resources      : Amazon.EKS.Model.NodegroupResources
ScalingConfig  : Amazon.EKS.Model.NodegroupScalingConfig
Status         : DELETING
Subnets        : {subnet-0d1a9fff35efa7691, subnet-0a3f4928edbc224d4}
Tags           : {}
Version        : 1.14
```

- API-Details finden Sie DeleteNodegroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-EKSResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-EKSResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet löscht angegebene Tags aus einer EKS-Ressource.

```
Remove-EKSResourceTag -ResourceArn "arn:aws:eks:us-west-2:012345678912:cluster/PROD"
 -TagKey "Name"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EKSResourceTag (UntagResource)" on target
 "arn:aws:eks:us-west-2:012345678912:cluster/PROD".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie UntagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-EKSClusterConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Update-EKSClusterConfig`

Tools für PowerShell V5

Beispiel 1: Aktualisiert eine Konfiguration von Amazon EKS Cluster. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterConfig -Name "PROD" -Logging_ClusterLogging
 @{Types="api","audit","authenticator","controllerManager","scheduler",Enabled="True"}
```

Ausgabe:

```
CreatedAt : 12/25/2019 5:03:07 PM
Errors    : {}
Id        : ee708232-7d2e-4ed7-9270-d0b5176f0726
Params    : {Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : LoggingUpdate
```

- API-Details finden Sie UpdateClusterConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-EKSClusterVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Update-EKSClusterVersion`

Tools für PowerShell V5

Beispiel 1: Dieses Cmdlet aktualisiert einen Amazon-EKS-Cluster auf die angegebene Kubernetes-Version. Ihr Cluster funktioniert während des Updates weiterhin.

```
Update-EKSClusterVersion -Name "PROD-KUBE-CL" -Version 1.14
```

Ausgabe:

```
CreatedAt : 12/26/2019 9:50:37 AM
Errors    : {}
Id        : ef186eff-3b3a-4c25-bcfc-3dcdf9e898a8
Params    : {Amazon.EKS.Model.UpdateParam, Amazon.EKS.Model.UpdateParam}
Status    : InProgress
Type      : VersionUpdate
```

- API-Details finden Sie UpdateClusterVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Elastic Load Balancing — Beispiele für Version 1 unter Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Elastic Load Balancing — Version 1 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-ELBLoadBalancerToSubnet`

Das folgende Codebeispiel zeigt die Verwendung`Add-ELBLoadBalancerToSubnet`.

Tools für PowerShell V5

Beispiel 1: Hier wird das angegebene Subnetz zu den Subnetzen hinzugefügt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe enthält die vollständige Liste der Subnetze.

```
Add-ELBLoadBalancerToSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-12345678
subnet-87654321
```

- API-Details finden Sie AttachLoadBalancerToSubnets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-ELBResourceTag`

Tools für PowerShell V5

Beispiel 1: Hier werden die angegebenen Tags zum spezifizierten Load Balancer hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag
 @{ Key="project";Value="lima" },@{ Key="department";Value="digital-media" }
```

Beispiel 2: In PowerShell Version 2 müssen Sie verwenden, New-Object um ein Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.Tag
$tag.Key = "project"
$tag.Value = "lima"
Add-ELBResourceTag -LoadBalancerName my-load-balancer -Tag $tag
```

- API-Details finden Sie AddTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-ELBAvailabilityZoneForLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-ELBAvailabilityZoneForLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone aus dem spezifizierten Load Balancer entfernt. Die Ausgabe umfasst die verbleibenden Availability Zones.

```
Disable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2b
```

- API-Details finden Sie DisableAvailabilityZonesForLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Dismount-ELBLoadBalancerFromSubnet`

Das folgende Codebeispiel zeigt die Verwendung`Dismount-ELBLoadBalancerFromSubnet`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz aus den Subnetzen entfernt, die für den spezifizierten Load Balancer konfiguriert sind. Die Ausgabe umfasst die verbleibenden Subnetze.

```
Dismount-ELBLoadBalancerFromSubnet -LoadBalancerName my-load-balancer -Subnet
 subnet-12345678
```

Ausgabe:

```
subnet-87654321
```

- API-Details finden Sie DetachLoadBalancerFromSubnets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELBLoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELBLoadBalancerAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das zonenübergreifende Load Balancing für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
CrossZoneLoadBalancing_Enabled $true
```

Beispiel 2: In diesem Beispiel wird das Connection Draining für den angegebenen Load Balancer deaktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer -
ConnectionDraining_Enabled $false
```

Beispiel 3: In diesem Beispiel wird die Zugriffsprotokollierung für den angegebenen Load Balancer aktiviert.

```
Edit-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer `
>> -AccessLog_Enabled $true `
>> -AccessLog_S3BucketName amzn-s3-demo-logging-bucket `
>> -AccessLog_S3BucketPrefix my-app/prod `
>> -AccessLog_EmitInterval 60
```

- API-Details finden Sie ModifyLoadBalancerAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-ELBAvailabilityZoneForLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-ELBAvailabilityZoneForLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Availability Zone zum angegebenen Load Balancer hinzugefügt. Die Ausgabe enthält die vollständige Liste der Availability Zones.

```
Enable-ELBAvailabilityZoneForLoadBalancer -LoadBalancerName my-load-balancer -
AvailabilityZone us-west-2a
```

Ausgabe:

```
us-west-2a
us-west-2b
```

- API-Details finden Sie EnableAvailabilityZonesForLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBInstanceHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBInstanceHealth`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status der Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Description                   InstanceId                    ReasonCode
      State
-----------                   ----------                    ----------
      -----
N/A                           i-87654321                    N/A
      InService
Instance has failed at lea... i-12345678                    Instance
      OutOfService
```

Beispiel 2: In diesem Beispiel wird der Status der angegebenen Instance beschrieben, die beim angegebenen Load Balancer registriert ist.

```
Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance i-12345678
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des Status der angegebenen Instance angezeigt.

```
(Get-ELBInstanceHealth -LoadBalancerName my-load-balancer -Instance
 i-12345678).Description
```

Ausgabe:

```
Instance has failed at least the UnhealthyThreshold number of health checks
 consecutively.
```

- API-Details finden Sie DescribeInstanceHealth in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Namen Ihrer Load Balancer aufgelistet.

```
Get-ELBLoadBalancer | format-table -property LoadBalancerName
```

Ausgabe:

```
LoadBalancerName
----------------
my-load-balancer
my-other-load-balancer
my-internal-load-balancer
```

Beispiel 2: In diesem Beispiel wird der angegebene Load Balancer beschrieben.

```
Get-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AvailabilityZones         : {us-west-2a, us-west-2b}
BackendServerDescriptions :
 {Amazon.ElasticLoadBalancing.Model.BackendServerDescription}
CanonicalHostedZoneName   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
CanonicalHostedZoneNameID : Z3DZXE0EXAMPLE
CreatedTime               : 4/11/2015 12:12:45 PM
DNSName                   : my-load-balancer-1234567890.us-west-2.elb.amazonaws.com
HealthCheck               : Amazon.ElasticLoadBalancing.Model.HealthCheck
Instances                 : {i-207d9717, i-afefb49b}
ListenerDescriptions      : {Amazon.ElasticLoadBalancing.Model.ListenerDescription}
LoadBalancerName          : my-load-balancer
Policies                  : Amazon.ElasticLoadBalancing.Model.Policies
Scheme                    : internet-facing
SecurityGroups            : {sg-a61988c3}
SourceSecurityGroup       : Amazon.ElasticLoadBalancing.Model.SourceSecurityGroup
Subnets                   : {subnet-15aaab61}
VPCId                     : vpc-a01106c2
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Load Balancer in der aktuellen AWS Region.

```
Get-ELBLoadBalancer
```

Beispiel 4: In diesem Beispiel werden alle Ihre Load Balancer in allen verfügbaren AWS-Regionen beschrieben.

```
Get-AWSRegion | % { Get-ELBLoadBalancer -Region $_ }
```

- API-Details finden Sie DescribeLoadBalancers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBLoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Attribute für den angegebenen Load Balancer beschrieben.

```
Get-ELBLoadBalancerAttribute -LoadBalancerName my-load-balancer
```

Ausgabe:

```
AccessLog              : Amazon.ElasticLoadBalancing.Model.AccessLog
AdditionalAttributes   : {}
ConnectionDraining     : Amazon.ElasticLoadBalancing.Model.ConnectionDraining
ConnectionSettings     : Amazon.ElasticLoadBalancing.Model.ConnectionSettings
CrossZoneLoadBalancing : Amazon.ElasticLoadBalancing.Model.CrossZoneLoadBalancing
```

- API-Details finden Sie DescribeLoadBalancerAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Richtlinien beschrieben, die dem angegebenen Load Balancer zugeordnet sind.

```
Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{ProxyProtocol}                         my-ProxyProtocol-policy
 ProxyProtocolPolicyType
{CookieName}                            my-app-cookie-policy
 AppCookieStickinessPolicyType
```

Beispiel 2: In diesem Beispiel werden die Attribute der angegebenen Richtlinie beschrieben.

```
(Get-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy).PolicyAttributeDescriptions
```

Ausgabe:

```
AttributeName    AttributeValue
-------------    --------------
ProxyProtocol    true
```

Beispiel 3: In diesem Beispiel werden die vordefinierten Richtlinien beschrieben, einschließlich der Beispielrichtlinien. Die Namen der Beispielrichtlinien beginnen mit dem Präfix „ELBSample-“.

```
Get-ELBLoadBalancerPolicy
```

Ausgabe:

```
PolicyAttributeDescriptions             PolicyName
 PolicyTypeName
---------------------------             ----------
 --------------
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-05
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-03
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2015-02
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-10
 SSLNegotiationPolicyType
```

```
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2014-01
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSecurityPolicy-2011-08
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-ELBDefaultCipherPolicy
 SSLNegotiationPolicyType
{Protocol-SSLv2, Protocol-TLSv1, Pro... ELBSample-OpenSSLDefaultCipherPolicy
 SSLNegotiationPolicyType
```

- API-Details finden Sie DescribeLoadBalancerPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBLoadBalancerPolicyType`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBLoadBalancerPolicyType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Richtlinientypen abgerufen, die von Elastic Load Balancing unterstützt werden.

```
Get-ELBLoadBalancerPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Stickiness policy with session lifet... {CookieExpirationPeriod}
 LBCookieStickinessPolicyType
Policy that controls authentication ... {PublicKeyPolicyName}
 BackendServerAuthenticationPolicyType
Listener policy that defines the cip... {Protocol-SSLv2, Protocol-TLSv1, Pro...
 SSLNegotiationPolicyType
Policy containing a list of public k... {PublicKey}
 PublicKeyPolicyType
Stickiness policy with session lifet... {CookieName}
 AppCookieStickinessPolicyType
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 2: In diesem Beispiel wird der angegebene Richtlinientyp beschrieben.

```
Get-ELBLoadBalancerPolicyType -PolicyTypeName ProxyProtocolPolicyType
```

Ausgabe:

```
Description                             PolicyAttributeTypeDescriptions
 PolicyTypeName
-----------                             -------------------------------
 --------------
Policy that controls whether to incl... {ProxyProtocol}
 ProxyProtocolPolicyType
```

Beispiel 3: In diesem Beispiel wird die vollständige Beschreibung des angegebenen Richtlinientyps angezeigt.

```
(Get-ELBLoadBalancerPolicyType -PolicyTypeName).Description
```

Ausgabe:

```
Policy that controls whether to include the IP address and port of the originating
 request for TCP messages.
This policy operates on TCP/SSL listeners only
```

- API-Details finden Sie DescribeLoadBalancerPolicyTypes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELBResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tags für die angegebenen Load Balancers aufgelistet.

```
Get-ELBResourceTag -LoadBalancerName @("my-load-balancer","my-internal-load-
balancer")
```

Ausgabe:

```
LoadBalancerName             Tags
----------------             ----
my-load-balancer             {project, department}
my-internal-load-balancer    {project, department}
```

Beispiel 2: In diesem Beispiel werden die Tags für den angegebenen Load Balancers beschrieben.

```
(Get-ELBResourceTag -LoadBalancerName my-load-balancer).Tags
```

Ausgabe:

```
Key           Value
---           -----
project       lima
department    digital-media
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Join-ELBSecurityGroupToLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Join-ELBSecurityGroupToLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die aktuelle Sicherheitsgruppe für den angegebenen Load Balancer durch die spezifizierte Sicherheitsgruppe ersetzt.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup sg-87654321
```

Ausgabe:

```
sg-87654321
```

Beispiel 2: Um die aktuelle Sicherheitsgruppe beizubehalten und eine zusätzliche Sicherheitsgruppe anzugeben, geben Sie sowohl die vorhandene als auch die neue Sicherheitsgruppe an.

```
Join-ELBSecurityGroupToLoadBalancer -LoadBalancerName my-load-balancer -
SecurityGroup @("sg-12345678", "sg-87654321")
```

Ausgabe:

```
sg-12345678
sg-87654321
```

- API-Details finden Sie ApplySecurityGroupsToLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELBAppCookieStickinessPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBAppCookieStickinessPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Stickiness-Richtlinie erstellt, die sich an die Lebensdauer des angegebenen, von der Anwendung generierten Cookies für Sticky Sessions hält.

```
New-ELBAppCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
app-cookie-policy -CookieName my-app-cookie
```

- API-Details finden Sie CreateAppCookieStickinessPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELBLBCookieStickinessPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLBCookieStickinessPolicy`

Tools für PowerShell V5

Beispiel 1: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions durch die festgelegte Ablaufzeit (in Sekunden) bestimmt wird).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy -CookieExpirationPeriod 60
```

Beispiel 2: Hier wird eine Stickiness-Richtlinie erstellt, bei der die Dauer der Sticky Sessions von der Lebensdauer des Browsers (User-Agent) abhängt).

```
New-ELBLBCookieStickinessPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

- API-Details finden Sie CreateLbCookieStickinessPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener in einer VPC erstellt.

```
$httpListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpListener.Protocol = "http"
$httpListener.LoadBalancerPort = 80
$httpListener.InstanceProtocol = "http"
$httpListener.InstancePort = 80
New-ELBLoadBalancer -LoadBalancerName my-vpc-load-balancer -SecurityGroup sg-
a61988c3 -Subnet subnet-15aaab61 -Listener $httpListener

my-vpc-load-balancer-1234567890.us-west-2.elb.amazonaws.com
```

Beispiel 2: In diesem Beispiel wird ein Load Balancer mit einem HTTP-Listener erstellt. EC2-Classic

```
New-ELBLoadBalancer -LoadBalancerName my-classic-load-balancer -AvailabilityZone us-
west-2a` -Listener $httpListener
```

Ausgabe:

```
my-classic-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

Beispiel 3: In diesem Beispiel wird ein Load Balancer mit einem HTTPS-Listener erstellt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
```

```
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "http"
$httpsListener.InstancePort = 80
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancer -LoadBalancerName my-load-balancer -AvailabilityZone us-west-2a
 -Listener $httpsListener

my-load-balancer-123456789.us-west-2.elb.amazonaws.com
```

- API-Details finden Sie CreateLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELBLoadBalancerListener`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancerListener`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer ein HTTPS-Listener hinzugefügt.

```
$httpsListener = New-Object Amazon.ElasticLoadBalancing.Model.Listener
$httpsListener.Protocol = "https"
$httpsListener.LoadBalancerPort = 443
$httpsListener.InstanceProtocol = "https"
$httpsListener.InstancePort = 443
$httpsListener.SSLCertificateId="arn:aws:iam::123456789012:server-certificate/my-
server-cert"
New-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -Listener
 $httpsListener
```

- API-Details finden Sie CreateLoadBalancerListeners in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELBLoadBalancerPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Proxyprotokollrichtlinie für einen angegebenen Load Balancer erstellt.

```
$attribute = New-Object Amazon.ElasticLoadBalancing.Model.PolicyAttribute -Property
 @{
         AttributeName="ProxyProtocol"
         AttributeValue="True"
    }
New-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
ProxyProtocol-policy -PolicyTypeName ProxyProtocolPolicyType -PolicyAttribute
 $attribute
```

- API-Details finden Sie CreateLoadBalancerPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-ELBInstanceWithLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Register-ELBInstanceWithLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance beim spezifizierten Load Balancer registriert.

```
Register-ELBInstanceWithLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
InstanceId
----------
i-12345678
i-87654321
```

- API-Details finden Sie RegisterInstancesWithLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELBInstanceFromLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBInstanceFromLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene EC2-Instance aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBInstanceFromLoadBalancer -LoadBalancerName my-load-balancer -Instance
 i-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBInstanceFromLoadBalancer
 (DeregisterInstancesFromLoadBalancer)" on Target
"Amazon.ElasticLoadBalancing.Model.Instance".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):

InstanceId
----------
i-87654321
```

- API-Details finden Sie DeregisterInstancesFromLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELBLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancer -LoadBalancerName my-load-balancer
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancer (DeleteLoadBalancer)" on Target "my-
load-balancer".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELBLoadBalancerListener`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancerListener`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Listener auf Port 80 für den angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerListener -LoadBalancerName my-load-balancer -LoadBalancerPort
 80
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerListener (DeleteLoadBalancerListeners)"
 on Target "80".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteLoadBalancerListeners in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELBLoadBalancerPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBLoadBalancerPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Richtlinie aus dem angegebenen Load Balancer gelöscht. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an.

```
Remove-ELBLoadBalancerPolicy -LoadBalancerName my-load-balancer -PolicyName my-
duration-cookie-policy
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ELBLoadBalancerPolicy (DeleteLoadBalancerPolicy)" on
 Target "my-duration-cookie-policy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

- API-Details finden Sie DeleteLoadBalancerPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELBResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELBResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Tag aus dem spezifizierten Load Balancer entfernt. Sie werden um eine Bestätigung gebeten, bevor der Vorgang fortgesetzt wird, es sei denn, Sie geben zusätzlich den Force-Parameter an. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-ELBResourceTag -LoadBalancerName my-load-balancer -Tag @{ Key="project" }
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELBResourceTag (RemoveTags)" on target
 "Amazon.ElasticLoadBalancing.Model.TagKeyOnly".
```

```
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

Beispiel 2: Mit Powershell-Version 2 New-Object müssen Sie das Tag für den Tag-Parameter erstellen.

```
$tag = New-Object Amazon.ElasticLoadBalancing.Model.TagKeyOnly
$tag.Key = "project"
Remove-ELBResourceTag -Tag $tag -Force
```

- API-Details finden Sie RemoveTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELBHealthCheck`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBHealthCheck`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Einstellungen für die Zustandsprüfung für den angegebenen Load Balancer konfiguriert.

```
Set-ELBHealthCheck -LoadBalancerName my-load-balancer `
>> -HealthCheck_HealthyThreshold 2 `
>> -HealthCheck_UnhealthyThreshold 2 `
>> -HealthCheck_Target "HTTP:80/ping" `
>> -HealthCheck_Interval 30 `
>> -HealthCheck_Timeout 3
```

Ausgabe:

```
HealthyThreshold   : 2
Interval           : 30
Target             : HTTP:80/ping
Timeout            : 3
UnhealthyThreshold : 2
```

- API-Details finden Sie ConfigureHealthCheck in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELBLoadBalancerListenerSSLCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerListenerSSLCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Zertifikat ersetzt, das die SSL-Verbindungen für den angegebenen Listener beendet.

```
Set-ELBLoadBalancerListenerSSLCertificate -LoadBalancerName my-load-balancer `
>> -LoadBalancerPort 443 `
>> -SSLCertificateId "arn:aws:iam::123456789012:server-certificate/new-server-cert"
```

- API-Details finden Sie SetLoadBalancerListenerSslCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELBLoadBalancerPolicyForBackendServer`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerPolicyForBackendServer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Port durch die spezifizierte Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80 -PolicyName my-ProxyProtocol-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Port zugeordnet sind.

```
Set-ELBLoadBalancerPolicyForBackendServer -LoadBalancerName my-load-balancer -
InstancePort 80
```

- API-Details finden Sie SetLoadBalancerPoliciesForBackendServer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELBLoadBalancerPolicyOfListener`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELBLoadBalancerPolicyOfListener`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Richtlinien für den angegebenen Listener durch die angegebene Richtlinie ersetzt.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443 -PolicyName my-SSLNegotiation-policy
```

Beispiel 2: In diesem Beispiel werden alle Richtlinien entfernt, die dem angegebenen Listener zugeordnet sind.

```
Set-ELBLoadBalancerPolicyOfListener -LoadBalancerName my-load-balancer -
LoadBalancerPort 443
```

- API-Details finden Sie SetLoadBalancerPoliciesOfListener in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Elastic Load Balancing — Beispiele für Version 2 unter Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Elastic Load Balancing — Version 2 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`Add-ELB2ListenerCertificate`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird dem angegebenen Listener ein zusätzliches Zertifikat hinzugefügt.

```
Add-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618' -
Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'}
```

Ausgabe:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97
 False
```

- API-Details finden Sie AddListenerCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-ELB2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Tag zur angegebenen Ressource **`AWS.Tools.ElasticLoadBalancingV2`** hinzugefügt.

```
Add-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Tag @{Key =
 'productVersion'; Value = '1.0.0'}
```

- API-Details finden Sie AddTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2Listener`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ändert die Standardaktion des angegebenen Listeners in eine feststehende Antwort.

```
$newDefaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
    "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}

Edit-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/d19f2f14974db685' -Port
 8080 -DefaultAction $newDefaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/d19f2f14974db685
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8080
Protocol        : HTTP
SslPolicy       :
```

- API-Details finden Sie ModifyListener in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELB2LoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2LoadBalancerAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Attribute des angegebenen Load Balancers geändert.

```
Edit-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Attribute @{Key =
 'deletion_protection.enabled'; Value = 'true'}
```

Ausgabe:

```
Key                                            Value
---                                            -----
deletion_protection.enabled                    true
access_logs.s3.enabled                         false
access_logs.s3.bucket
access_logs.s3.prefix
idle_timeout.timeout_seconds                   60
routing.http2.enabled                          true
routing.http.drop_invalid_header_fields.enabled false
```

- API-Details finden Sie ModifyLoadBalancerAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2Rule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die angegebenen Listener-Regelkonfigurationen geändert.

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
    "PathPatternConfig" = @{
    "Values" = "/login1","/login2","/login3"
  }
  "Field" = "path-pattern"
}

Edit-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/testALB/3e2f03b558e19676/1c84f02aec143e80/
f4f51dfaa033a8cc' -Condition $newRuleCondition
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- API-Details finden Sie ModifyRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-ELB2TargetGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Eigenschaften der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -HealthCheckIntervalSecond
 60 -HealthCheckPath '/index.html' -HealthCheckPort 8080
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 60
HealthCheckPath            : /index.html
HealthCheckPort            : 8080
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- API-Details finden Sie ModifyTargetGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-ELB2TargetGroupAttribute`

Das folgende Codebeispiel zeigt die Verwendung`Edit-ELB2TargetGroupAttribute`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Attribut deregistration_delay der angegebenen Zielgruppe geändert.

```
Edit-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Attribute @{Key =
 'deregistration_delay.timeout_seconds'; Value = 600}
```

Ausgabe:

```
Key                                 Value
---                                 -----
stickiness.enabled                  false
deregistration_delay.timeout_seconds  600
stickiness.type                     lb_cookie
stickiness.lb_cookie.duration_seconds 86400
slow_start.duration_seconds         0
load_balancing.algorithm.type       round_robin
```

- API-Details finden Sie ModifyTargetGroupAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2AccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2AccountLimit`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl listet die ELB2-Kontolimits für eine bestimmte Region auf.

```
Get-ELB2AccountLimit
```

Ausgabe:

```
Max  Name
---  ----
3000 target-groups
1000 targets-per-application-load-balancer
50   listeners-per-application-load-balancer
100  rules-per-application-load-balancer
50   network-load-balancers
3000 targets-per-network-load-balancer
500  targets-per-availability-zone-per-network-load-balancer
50   listeners-per-network-load-balancer
5    condition-values-per-alb-rule
5    condition-wildcards-per-alb-rule
100  target-groups-per-application-load-balancer
5    target-groups-per-action-on-application-load-balancer
1    target-groups-per-action-on-network-load-balancer
50   application-load-balancers
```

- API-Details finden Sie DescribeAccountLimits in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Listener`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die Listener der angegebenen. ALB/NLB

```
Get-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/1dac07c21187d41e
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 80
Protocol        : HTTP
```

```
SslPolicy       :

Certificates    : {Amazon.ElasticLoadBalancingV2.Model.Certificate}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/test-alb/3651b4394dd9a24f
Port            : 443
Protocol        : HTTPS
SslPolicy       : ELBSecurityPolicy-2016-08
```

- API-Details finden Sie DescribeListeners in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2ListenerCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Zertifikat für den angegebenen Listener beschrieben.

```
Get-ELB2ListenerCertificate -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
CertificateArn
 IsDefault
--------------
 ---------
arn:aws:acm:us-east-1:123456789012:certificate/5fc7c092-68bf-4862-969c-22fd48b6e17c
 True
```

- API-Details finden Sie DescribeListenerCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2LoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Load Balancer für die angegebene Region angezeigt.

```
Get-ELB2LoadBalancer
```

Ausgabe:

```
AvailabilityZones     : {us-east-1c}
CanonicalHostedZoneId : Z26RNL4JYFTOTI
CreatedTime           : 6/22/18 11:21:50 AM
DNSName               : test-elb1234567890-238d34ad8d94bc2e.elb.us-
east-1.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb1234567890/238d34ad8d94bc2e
LoadBalancerName      : test-elb1234567890
Scheme                : internet-facing
SecurityGroups        : {}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : network
VpcId                 : vpc-2cf00000
```

- API-Details finden Sie DescribeLoadBalancers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2LoadBalancerAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2LoadBalancerAttribute`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl beschreibt die Attribute eines bestimmten Load Balancers.

```
Get-ELB2LoadBalancerAttribute -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/net/test-elb/238d34ad8d94bc2e'
```

Ausgabe:

```
Key                                Value
```

```
---                                 -----
access_logs.s3.enabled              false
load_balancing.cross_zone.enabled true
access_logs.s3.prefix
deletion_protection.enabled         false
access_logs.s3.bucket
```

- API-Details finden Sie DescribeLoadBalancerAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Rule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Listener-Regeln für den angegebenen Listener-ARN beschrieben.

```
Get-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 1
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/2286fff5055e0f79

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 2
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/14e7b036567623ba

Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {}
IsDefault  : True
```

```
Priority   : default
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b/853948cf3aa9b2bf
```

- API-Details finden Sie DescribeRules in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2SSLPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2SSLPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet alle verfügbaren Listener-Richtlinien für auf. ElasticLoadBalancingV2

```
Get-ELB2SSLPolicy
```

Ausgabe:

```
Ciphers
                              Name                                  SslProtocols
-------
                              ----                                  ------------
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2016-08            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-2017-01     {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-1-2017-01     {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-2-Ext-2018-06 {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-2018-06         {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-2015-05            {TLSv1,
 TLSv1.1, TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-TLS-1-0-2015-04     {TLSv1,
 TLSv1.1, TLSv1.2}
```

```
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-Res-2019-08  {TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-1-2019-08      {TLSv1.1,
 TLSv1.2}
{ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-RSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES128-
SHA256, ECDHE-RSA-AES128-SHA256} ELBSecurityPolicy-FS-1-2-2019-08      {TLSv1.2}
```

- API-Details finden Sie DescribeSslPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tags für die angegebene Ressource aufgelistet.

```
Get-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
ResourceArn
           Tags
-----------
           ----
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f {stage, internalName, version}
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe beschrieben.

```
Get-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /
HealthCheckPort            : traffic-port
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 5
LoadBalancerArns           : {arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970
TargetGroupName            : test-tg
TargetType                 : instance
UnhealthyThresholdCount    : 2
VpcId                      : vpc-2cfd7000
```

- API-Details finden Sie DescribeTargetGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2TargetGroupAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetGroupAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Attribute der angegebenen Zielgruppe beschrieben.

```
Get-ELB2TargetGroupAttribute -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
Key                                     Value
```

```
---                                     -----
stickiness.enabled                      false
deregistration_delay.timeout_seconds    300
stickiness.type                         lb_cookie
stickiness.lb_cookie.duration_seconds   86400
slow_start.duration_seconds             0
load_balancing.algorithm.type           round_robin
```

- API-Details finden Sie DescribeTargetGroupAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ELB2TargetHealth`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ELB2TargetHealth`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Zustand der in der angegebenen Zielgruppe vorhandenen Ziele zurückgegeben.

```
Get-ELB2TargetHealth -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970'
```

Ausgabe:

```
HealthCheckPort Target                                                  TargetHealth
--------------- ------                                                  ------------
80              Amazon.ElasticLoadBalancingV2.Model.TargetDescription
 Amazon.ElasticLoadBalancingV2.Model.TargetHealth
```

- API-Details finden Sie DescribeTargetHealth in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2Listener`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer ALB-Listener mit der Standardaktion „Weiterleiten“ erstellt, um Datenverkehr an die angegebene Zielgruppe zu senden.

```
$defaultAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  ForwardConfig = @{
    TargetGroups = @(
      @{ TargetGroupArn = "arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testAlbTG/3d61c2f20aa5bccb" }
    )
    TargetGroupStickinessConfig = @{
      DurationSeconds = 900
      Enabled = $true
    }
  }
  Type = "Forward"
}

New-ELB2Listener -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/testALB/3e2f03b558e19676' -Port 8001 -Protocol
 "HTTP" -DefaultAction $defaultAction
```

Ausgabe:

```
Certificates    : {}
DefaultActions  : {Amazon.ElasticLoadBalancingV2.Model.Action}
ListenerArn     : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/
testALB/3e2f03b558e19676/1c84f02aec143e80
LoadBalancerArn : arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/
app/testALB/3e2f03b558e19676
Port            : 8001
Protocol        : HTTP
SslPolicy       :
```

- API-Details finden Sie CreateListener in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2LoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer, mit dem Internet verbundener Application Load Balancer mit zwei Subnetzen erstellt.

```
New-ELB2LoadBalancer -Type application -Scheme internet-facing -IpAddressType
 ipv4 -Name 'New-Test-ALB' -SecurityGroup 'sg-07c3414abb8811cbd' -subnet 'subnet-
c37a67a6','subnet-fc02eea0'
```

Ausgabe:

```
AvailabilityZones     : {us-east-1b, us-east-1a}
CanonicalHostedZoneId : Z35SXDOTRQ7X7K
CreatedTime           : 12/28/19 2:58:03 PM
DNSName               : New-Test-ALB-1391502222.us-east-1.elb.amazonaws.com
IpAddressType         : ipv4
LoadBalancerArn       : arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/New-Test-ALB/dab2e4d90eb51493
LoadBalancerName      : New-Test-ALB
Scheme                : internet-facing
SecurityGroups        : {sg-07c3414abb8811cbd}
State                 : Amazon.ElasticLoadBalancingV2.Model.LoadBalancerState
Type                  : application
VpcId                 : vpc-2cfd7000
```

- API-Details finden Sie CreateLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2Rule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Listener-Regel mit einer Aktion mit fester Antwort erstellt, die auf dem Kunden-Header-Wert für den angegebenen Listener basiert.

```
$newRuleAction = [Amazon.ElasticLoadBalancingV2.Model.Action]@{
  "FixedResponseConfig" = @{
    "ContentType" = "text/plain"
    "MessageBody" = "Hello World"
    "StatusCode" = "200"
  }
  "Type" = [Amazon.ElasticLoadBalancingV2.ActionTypeEnum]::FixedResponse
}
```

```
$newRuleCondition = [Amazon.ElasticLoadBalancingV2.Model.RuleCondition]@{
  "httpHeaderConfig" = @{
    "HttpHeaderName" = "customHeader"
    "Values" = "header2","header1"
  }
  "Field" = "http-header"
}

New-ELB2Rule -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/testALB/3e2f03b558e19676/1c84f02aec143e80' -Action
 $newRuleAction -Condition $newRuleCondition -Priority 10
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 10
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
testALB/3e2f03b558e19676/1c84f02aec143e80/f4f51dfaa033a8cc
```

- API-Details finden Sie CreateRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-ELB2TargetGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Zielgruppe mit den angegebenen Parametern erstellt.

```
New-ELB2TargetGroup -HealthCheckEnabled 1 -HealthCheckIntervalSeconds 30 -
HealthCheckPath '/index.html' -HealthCheckPort 80 -HealthCheckTimeoutSecond 5 -
HealthyThresholdCount 2 -UnhealthyThresholdCount 5 -Port 80 -Protocol 'HTTP' -
TargetType instance -VpcId 'vpc-2cfd7000' -Name 'NewTargetGroup'
```

Ausgabe:

```
HealthCheckEnabled         : True
HealthCheckIntervalSeconds : 30
HealthCheckPath            : /index.html
```

```
HealthCheckPort            : 80
HealthCheckProtocol        : HTTP
HealthCheckTimeoutSeconds  : 5
HealthyThresholdCount      : 2
LoadBalancerArns           : {}
Matcher                    : Amazon.ElasticLoadBalancingV2.Model.Matcher
Port                       : 80
Protocol                   : HTTP
TargetGroupArn             : arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/NewTargetGroup/534e484681d801bf
TargetGroupName            : NewTargetGroup
TargetType                 : instance
UnhealthyThresholdCount    : 5
VpcId                      : vpc-2cfd7000
```

- API-Details finden Sie CreateTargetGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-ELB2Target`

Das folgende Codebeispiel zeigt die Verwendung. `Register-ELB2Target`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Instance „i-0672a4c4cdeae3111” bei der angegebenen Zielgruppe registriert.

```
Register-ELB2Target -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/test-tg/a4e04b3688be1970' -Target @{Port = 80; Id =
 'i-0672a4c4cdeae3111'}
```

- API-Details finden Sie RegisterTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2Listener`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Listener`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Listener gelöscht.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/66e10e3aaf5b6d9b'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/66e10e3aaf5b6d9b".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

Beispiel 2: In diesem Beispiel wird der angegebene Listener aus dem Load Balancer entfernt.

```
Remove-ELB2Listener -ListenerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Listener (DeleteListener)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie DeleteListener in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2ListenerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2ListenerCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Zertifikat aus der spezifizierten Zielgruppe entfernt.

```
Remove-ELB2ListenerCertificate -Certificate @{CertificateArn = 'arn:aws:acm:us-
east-1:123456789012:certificate/19478bd5-491d-47d4-b1d7-5217feba1d97'} -ListenerArn
```

```
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2ListenerCertificate
 (RemoveListenerCertificates)" on target "arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener/app/test-alb/3651b4394dd9a24f/3873f123b98f7618".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie RemoveListenerCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2LoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2LoadBalancer`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene Load Balancer gelöscht.

```
Remove-ELB2LoadBalancer -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2LoadBalancer (DeleteLoadBalancer)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie DeleteLoadBalancer in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2Rule`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Rule`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus dem Listener entfernt

```
Remove-ELB2Rule -RuleArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2Rule (DeleteRule)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/3873f123b98f7618/4b25eb10a42e33ab".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie DeleteRule in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2Tag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2Tag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Tag für den angegebenen Schlüssel entfernt.

```
Remove-ELB2Tag -ResourceArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -TagKey
 'productVersion'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-ELB2Tag (RemoveTags)" on target
 "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/app/test-
alb/3651b4394dd9a24f".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie RemoveTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-ELB2TargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-ELB2TargetGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Zielgruppe entfernt.

```
Remove-ELB2TargetGroup -TargetGroupArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:targetgroup/testsssss/4e0b6076bc6483a7'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ELB2TargetGroup (DeleteTargetGroup)" on
 target "arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/
testsssss/4e0b6076bc6483a7".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): y
```

- API-Details finden Sie DeleteTargetGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELB2IpAddressType`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2IpAddressType`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IP-Adresstyp des Load Balancers von 'IPv4' auf '' geändert. DualStack

```
Set-ELB2IpAddressType -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -IpAddressType
 dualstack
```

Ausgabe:

```
Value
-----
dualstack
```

- API-Details finden Sie SetIpAddressType in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELB2RulePriority`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2RulePriority`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Priorität der angegebenen Listener-Regel geändert.

```
Set-ELB2RulePriority -RulePriority -RulePriority @{Priority = 11; RuleArn =
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/test-
alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8'}
```

Ausgabe:

```
Actions    : {Amazon.ElasticLoadBalancingV2.Model.Action}
Conditions : {Amazon.ElasticLoadBalancingV2.Model.RuleCondition}
IsDefault  : False
Priority   : 11
RuleArn    : arn:aws:elasticloadbalancing:us-east-1:123456789012:listener-rule/app/
test-alb/3651b4394dd9a24f/a4eb199fa5046f80/dbf4c6dcef3ec6f8
```

- API-Details finden Sie SetRulePriorities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELB2SecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2SecurityGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird dem angegebenen Load Balancer die Sicherheitsgruppe „sg-07c3414abb8811cbd” hinzugefügt.

```
Set-ELB2SecurityGroup -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -SecurityGroup
 'sg-07c3414abb8811cbd'
```

Ausgabe:

```
sg-07c3414abb8811cbd
```

- API-Details finden Sie SetSecurityGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-ELB2Subnet`

Das folgende Codebeispiel zeigt die Verwendung. `Set-ELB2Subnet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Subnetze des angegebenen Load Balancers geändert.

```
Set-ELB2Subnet -LoadBalancerArn 'arn:aws:elasticloadbalancing:us-
east-1:123456789012:loadbalancer/app/test-alb/3651b4394dd9a24f' -Subnet
 'subnet-7d8a0a51','subnet-c37a67a6'
```

Ausgabe:

```
LoadBalancerAddresses SubnetId        ZoneName
--------------------- --------        --------
{}                    subnet-7d8a0a51 us-east-1c
{}                    subnet-c37a67a6 us-east-1b
```

- API-Details finden Sie SetSubnets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-ELB2Target`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-ELB2Target`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung der Instance „i-0672a4c4cdeae3111” aus der angegebenen Zielgruppe aufgehoben.

```
$targetDescription = New-Object
 Amazon.ElasticLoadBalancingV2.Model.TargetDescription
$targetDescription.Id = 'i-0672a4c4cdeae3111'
Unregister-ELB2Target -Target $targetDescription -TargetGroupArn
 'arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/test-tg/
a4e04b3688be1970'
```

- API-Details finden Sie DeregisterTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon FSx-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon FSx verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-FSXResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-FSXResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden der angegebenen Ressource Tags hinzugefügt.

```
Add-FSXResourceTag -ResourceARN "arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a" -Tag @{Key="Users";Value="Test"}
```

Ausgabe:

```
arn:aws:fsx:eu-west-1:123456789012:file-system/fs-01cd23bc4bdf5678a
```

- API-Details finden Sie TagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-FSXBackup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXBackup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Backups abgerufen, die seit gestern für die angegebene Dateisystem-ID erstellt wurden.

```
Get-FSXBackup -Filter @{Name="file-system-id";Values=$fsx.FileSystemId} | Where-
Object CreationTime -gt (Get-Date).AddDays(-1)
```

Ausgabe:

```
BackupId        : backup-01dac234e56782bcc
CreationTime    : 6/14/2019 3:35:14 AM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f1-
e1234c5af123
Lifecycle       : AVAILABLE
ProgressPercent : 100
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/backup-01dac234e56782bcc
Tags            : {}
Type            : AUTOMATIC
```

- API-Details finden Sie DescribeBackups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXFileSystem`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung der angegebenen filesystemId zurückgegeben.

```
Get-FSXFileSystem -FileSystemId fs-01cd23bc4bdf5678a
```

Ausgabe:

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-8bde-
a9f0-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-07d1dda1322b7e209}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-01cd23bc4bdf5678a
StorageCapacity      : 300
SubnetIds            : {subnet-7d123456}
Tags                 : {FSx-Service}
VpcId                : vpc-41cf2b3f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- API-Details finden Sie DescribeFileSystems in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-FSXResourceTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-FSXResourceTagList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Tags für die bereitgestellte Ressource arn aufgelistet.

```
Get-FSXResourceTagList -ResourceARN $fsx.ResourceARN
```

Ausgabe:

```
Key         Value
---         -----
FSx-Service Windows
Users       Dev
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-FSXBackup

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXBackup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Sicherung des angegebenen Dateisystems erstellt.

```
New-FSXBackup -FileSystemId fs-0b1fac2345623456ba
```

Ausgabe:

```
BackupId        : backup-0b1fac2345623456ba
CreationTime    : 6/14/2019 5:37:17 PM
FailureDetails  :
FileSystem      : Amazon.FSx.Model.FileSystem
KmsKeyId        : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-a1f3-
e1234c5af678
Lifecycle       : CREATING
ProgressPercent : 0
ResourceARN     : arn:aws:fsx:eu-west-1:123456789012:backup/
backup-0b1fac2345623456ba
Tags            : {}
Type            : USER_INITIATED
```

- API-Details finden Sie CreateBackup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-FSXFileSystem

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXFileSystem`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Windows-Dateisystem mit einer Größe von 300 GB erstellt, auf das von dem angegebenen Subnetz aus zugegriffen werden kann und das einen Durchsatz von bis zu 8 Megabyte pro Sekunde unterstützt. Das neue Dateisystem wird automatisch mit dem angegebenen Microsoft Active Directory verbunden.

```
New-FSXFileSystem -FileSystemType WINDOWS -StorageCapacity
 300 -SubnetId subnet-1a2b3c4d5e6f -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId='d-1a2b3c4d'}
```

Ausgabe:

```
CreationTime         : 12/10/2018 6:06:59 PM
DNSName              : fs-abcdef01234567890.example.com
FailureDetails       :
FileSystemId         : fs-abcdef01234567890
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:us-west-2:123456789012:key/a1234567-252c-45e9-
afaa-123456789abc
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 123456789012
ResourceARN          : arn:aws:fsx:us-west-2:123456789012:file-system/fs-
abcdef01234567890
StorageCapacity      : 300
SubnetIds            : {subnet-1a2b3c4d5e6f}
Tags                 : {}
VpcId                : vpc-1a2b3c4d5e6f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- API-Details finden Sie CreateFileSystem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-FSXFileSystemFromBackup

Das folgende Codebeispiel zeigt die Verwendung. `New-FSXFileSystemFromBackup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Amazon FSx-Dateisystem aus einer vorhandenen Sicherung von Amazon FSx für Windows File Server erstellt.

```
New-FSXFileSystemFromBackup -BackupId $backupID -Tag @{Key="tag:Name";Value="from-
manual-backup"} -SubnetId $SubnetID -SecurityGroupId $SG_ID -WindowsConfiguration
 @{ThroughputCapacity=8;ActiveDirectoryId=$DirectoryID}
```

Ausgabe:

```
CreationTime         : 8/8/2019 12:59:58 PM
DNSName              : fs-012ff34e56789120.ktmsad.local
FailureDetails       :
FileSystemId         : fs-012ff34e56789120
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-5b67-1bde-
a2f3-e4567c8a9321
Lifecycle            : CREATING
LustreConfiguration  :
NetworkInterfaceIds  : {}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:123456789012:file-system/
fs-012ff34e56789120
StorageCapacity      : 300
SubnetIds            : {subnet-fa1ae23c}
Tags                 : {tag:Name}
VpcId                : vpc-12cf3b4f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- API-Details finden Sie CreateFileSystemFromBackup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-FSXBackup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXBackup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Sicherungs-ID entfernt.

```
Remove-FSXBackup -BackupId $backupID
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXBackup (DeleteBackup)" on target
 "backup-0bbca1e2345678e12".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

BackupId                Lifecycle
--------                ---------
backup-0bbca1e2345678e12 DELETED
```

- API-Details finden Sie DeleteBackup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXFileSystem`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene FSX-Dateisystem-ID entfernt.

```
Remove-FSXFileSystem -FileSystemId fs-012ff34e567890120
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXFileSystem (DeleteFileSystem)" on target
 "fs-012ff34e567890120".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

FileSystemId         Lifecycle WindowsResponse
------------         --------- ---------------
fs-012ff34e567890120 DELETING  Amazon.FSx.Model.DeleteFileSystemWindowsResponse
```

- API-Details finden Sie DeleteFileSystem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-FSXResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-FSXResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Ressourcen-Tag für die angegebene FSX-Dateisystemressource ARN entfernt.

```
Remove-FSXResourceTag -ResourceARN $FSX.ResourceARN -TagKey Users
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-FSXResourceTag (UntagResource)" on target
 "arn:aws:fsx:eu-west-1:933303704102:file-system/fs-07cd45bc6bdf2674a".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie UntagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-FSXFileSystem`

Das folgende Codebeispiel zeigt die Verwendung. `Update-FSXFileSystem`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das automatische Aufbewahrungsdatum für Backups des FSX-Dateisystems über aktualisiert. UpdateFileSystemWindowsConfiguration

```
$UpdateFSXWinConfig = [Amazon.FSx.Model.UpdateFileSystemWindowsConfiguration]::new()
 $UpdateFSXWinConfig.AutomaticBackupRetentionDays = 35
 Update-FSXFileSystem -FileSystemId $FSX.FileSystemId -WindowsConfiguration
 $UpdateFSXWinConfig
```

Ausgabe:

```
CreationTime         : 1/17/2019 9:55:30 AM
DNSName              : fs-01cd23bc4bdf5678a.ktmsad.local
FailureDetails       :
FileSystemId         : fs-01cd23bc4bdf5678a
FileSystemType       : WINDOWS
KmsKeyId             : arn:aws:kms:eu-west-1:123456789012:key/f1af23c4-1b23-1bde-
a1f2-e1234c5af678
Lifecycle            : AVAILABLE
LustreConfiguration  :
NetworkInterfaceIds  : {eni-01cd23bc4bdf5678a}
OwnerId              : 933303704102
ResourceARN          : arn:aws:fsx:eu-west-1:933303704102:file-system/
fs-07cd45bc6bdf2674a
StorageCapacity      : 300
SubnetIds            : {subnet-1d234567}
Tags                 : {FSx-Service}
VpcId                : vpc-23cf4b5f
WindowsConfiguration : Amazon.FSx.Model.WindowsFileSystemConfiguration
```

- API-Details finden Sie UpdateFileSystem in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon Glacier-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon Glacier verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-GLCJob`

Das folgende Codebeispiel zeigt die Verwendung`Get-GLCJob`.

Tools für PowerShell V5

Beispiel 1: Gibt Details des angegebenen Auftrags zurück. Wenn der Auftrag erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Auftrags (ein Archiv oder eine Inventarliste) in das lokale Dateisystem abzurufen.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Ausgabe:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
StatusCode                   : InProgress
StatusMessage                :
Tier                         : Standard
VaultARN                     : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- API-Details finden Sie DescribeJob in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-GLCVault

Das folgende Codebeispiel zeigt die Verwendung. `New-GLCVault`

Tools für PowerShell V5

Beispiel 1: Erstellt einen neuen Tresor für das Benutzerkonto. Da für den AccountId Parameter - kein Wert angegeben wurde, verwenden die Cmdlets den Standardwert „-“, der das aktuelle Konto angibt.

```
New-GLCVault -VaultName myvault
```

Ausgabe:

```
/01234567812/vaults/myvault
```

- API-Details finden Sie CreateVault in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Read-GLCJobOutput

Das folgende Codebeispiel zeigt die Verwendung. `Read-GLCJobOutput`

Tools für PowerShell V5

Beispiel 1: Lädt den Archivinhalt herunter, dessen Abruf im angegebenen Auftrag geplant war, und speichert den Inhalt in einer Datei auf der Festplatte. Der Download überprüft für Sie die Prüfsumme, sofern eine verfügbar ist. Auf Wunsch kann die gesamte Antwort einschließlich der Prüfsumme durch Angabe von **`-Select '*'`** zurückgegeben werden.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:\temp
\blue.bin"
```

- API-Details finden Sie GetJobOutput in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Start-GLCJob

Das folgende Codebeispiel zeigt die Verwendung. `Start-GLCJob`

Tools für PowerShell V5

Beispiel 1: Startet einen Auftrag zum Abrufen eines Archivs aus dem angegebenen Tresor, dessen Eigentümer der Benutzer ist. Der Status des Jobs kann mit dem Get-GLCJob Cmdlet überprüft werden. Wenn der Auftrag erfolgreich abgeschlossen wurde, kann das Read-GCJobOutput Cmdlet verwendet werden, um den Inhalt des Archivs in das lokale Dateisystem abzurufen.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Ausgabe:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/op1xe...I4HqCHkSJSbthM
```

- API-Details finden Sie InitiateJob in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-GLCArchive`

Das folgende Codebeispiel zeigt die Verwendung. `Write-GLCArchive`

Tools für PowerShell V5

Beispiel 1: Lädt eine einzelne Datei in den angegebenen Tresor hoch und gibt die Archiv-ID und die berechnete Prüfsumme zurück.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Beispiel 2: Lädt den Inhalt einer Ordnerhierarchie in den angegebenen Tresor im Benutzerkonto hoch. Für jede hochgeladene Datei gibt das Cmdlet den Dateinamen, die entsprechende Archiv-ID und die berechnete Prüfsumme des Archivs aus.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Ausgabe:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- API-Details finden Sie UploadArchive in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS Glue Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS Glue

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`New-GLUEJob`

Das folgende Codebeispiel zeigt die Verwendung`New-GLUEJob`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt einen neuen Job in AWS Glue. Der Befehlsnamenwert ist immer**`glueetl`**. AWS Glue unterstützt die Ausführung von Jobskripten, die in Python oder

Scala geschrieben sind. In diesem Beispiel ist das Job-Skript (MyTestGlueJob.py) in Python geschrieben. Python-Parameter werden in der **$DefArgs** Variablen angegeben und dann an den PowerShell Befehl im **DefaultArguments** Parameter übergeben, der eine Hashtabelle akzeptiert. Die Parameter in der **$JobParams** Variablen stammen aus der CreateJob API, was im Thema Jobs (https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-jobs-job.html) der AWS Glue-API-Referenz dokumentiert ist.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://amzn-s3-demo-source-bucket/admin/MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://amzn-s3-demo-bucket/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- API-Details finden Sie CreateJob in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS Health Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS Health

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-HLTHEvent`

Das folgende Codebeispiel zeigt die Verwendung`Get-HLTHEvent`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt Ereignisse aus dem AWS Personal Health Dashboard zurück. Der Benutzer fügt den Parameter -Region hinzu, um Ereignisse anzuzeigen, die für den Service in der Region USA Ost (Nord-Virginia) verfügbar sind. Der Filter_Region Parameter - filtert jedoch nach Ereignissen, die in den Regionen EU (London) und USA West (Oregon) (eu-west-2 und us-west-2) protokolliert werden. Der Filter_StartTime Parameter - filtert nach einer Zeitspanne, zu der Ereignisse beginnen können, während der Filter_EndTime Parameter - nach einer Zeitspanne filtert, zu der Ereignisse enden können. Das Ergebnis ist ein geplantes Wartungsereignis für RDS, das innerhalb des angegebenen Filter_StartTime Bereichs beginnt und innerhalb des geplanten Filter_EndTime Bereichs endet.

```
Get-HLTHEvent -Region us-east-1 -Filter_Region "eu-west-2","us-west-2" -
Filter_StartTime @{from="3/14/2019 6:30:00AM";to="3/15/2019 5:00:00PM"} -
Filter_EndTime @{from="3/21/2019 7:00:00AM";to="3/21/2019 5:00:00PM"}
```

Ausgabe:

```
Arn               : arn:aws:health:us-west-2::event/RDS/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED/
AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED_USW2_20190314_20190321
AvailabilityZone  :
EndTime           : 3/21/2019 2:00:00 PM
EventTypeCategory : scheduledChange
EventTypeCode     : AWS_RDS_HARDWARE_MAINTENANCE_SCHEDULED
LastUpdatedTime   : 2/28/2019 2:26:07 PM
Region            : us-west-2
Service           : RDS
StartTime         : 3/14/2019 2:00:00 PM
StatusCode        : open
```

- API-Details finden Sie DescribeEvents in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

IAM-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit IAM Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-IAMClientIDToOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMClientIDToOpenIDConnectProvider`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird die Client-ID (oder Zielgruppe) **`my-application-ID`** zum vorhandenen OIDC-Anbieter mit dem Namen **`server.example.com`** hinzugefügt.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- API-Details finden Sie AddClientIdToOpenIdConnectProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-IAMRoleTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMRoleTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Rolle im Identity Management Service ein Tag hinzugefügt

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API-Details finden Sie TagRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-IAMRoleToInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMRoleToInstanceProfile`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird die Rolle mit dem Namen **`S3Access`** einem vorhandenen Instance-Profil mit dem Namen **`webserver`** hinzugefügt. Verwenden Sie den **`New-IAMInstanceProfile`**-Befehl, um das Instance-Profil zu erstellen. Nachdem Sie das Instance-Profil erstellt und es mithilfe dieses Befehls einer Rolle zugeordnet haben, können Sie es an eine EC2 Instance anfügen. Verwenden Sie dazu das **`New-EC2Instance`** Cmdlet mit dem Parameter **`InstanceProfile_Arn`** oder **`InstanceProfile-Name`**, um die neue Instance zu starten.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName "webserver"
```

- API-Details finden Sie AddRoleToInstanceProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-IAMUserTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMUserTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird dem Benutzer im Identity Management Service ein Tag hinzugefügt

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API-Details finden Sie TagUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Add-IAMUserToGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Add-IAMUserToGroup`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird der Benutzer mit dem Namen **`Bob`** der Gruppe mit dem Namen **`Admins`** hinzugefügt.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API-Details finden Sie AddUserToGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Disable-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Disable-IAMMFADevice`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird das Hardware-MFA-Gerät deaktiviert, das dem Benutzer **`Bob`** mit der Seriennummer **`123456789012`** zugeordnet ist.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

Beispiel 2: Mit diesem Befehl wird das virtuelle MFA-Gerät deaktiviert, das dem Benutzer **`David`** mit der ARN **`arn:aws:iam::210987654321:mfa/David`** zugeordnet ist. Beachten Sie, dass das virtuelle MFA-Gerät nicht aus dem Konto gelöscht wird. Das virtuelle Gerät ist noch vorhanden und wird in der Ausgabe des Befehls **`Get-IAMVirtualMFADevice`** angezeigt. Bevor Sie ein neues virtuelles MFA-Gerät für denselben Benutzer erstellen können, müssen Sie das alte mit dem Befehl **`Remove-IAMVirtualMFADevice`** löschen.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber "arn:aws:iam::210987654321:mfa/
David"
```

- API-Details finden Sie DeactivateMfaDevice in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-IAMPassword`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-IAMPassword`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl wird das Passwort des Benutzers geändert, der den Befehl ausführt. Dieser Befehl kann nur von IAM-Benutzern aufgerufen werden. Wenn dieser Befehl aufgerufen wird, während Sie mit AWS Kontoanmeldeinformationen (Root) angemeldet sind, gibt der Befehl einen Fehler zurück. **`InvalidUserType`**

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API-Details finden Sie ChangePassword in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Enable-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Enable-IAMMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Befehl wird das Hardware-MFA-Gerät mit der Seriennummer **`987654321098`** aktiviert und das Gerät dem Benutzer **`Bob`** zugeordnet. Er enthält nacheinander die ersten beiden Codes des Geräts.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

Beispiel 2: In diesem Beispiel wird ein virtuelles MFA-Gerät erstellt und aktiviert. Mit dem ersten Befehl wird das virtuelle Gerät erstellt und die Objektdarstellung des Geräts in der Variablen **`$MFADevice`** zurückgegeben. Sie können die Eigenschaften **`.Base32StringSeed`** oder **`QRCodePng`** verwenden, um die Softwar-Aanwendung des Benutzers zu konfigurieren. Mit dem letzten Befehl wird das Gerät dem Benutzer **`David`** zugewiesen und das Gerät anhand seiner Seriennummer identifiziert. Der Befehl synchronisiert das Gerät auch mit, AWS indem die ersten beiden Codes nacheinander vom virtuellen MFA-Gerät aufgenommen werden.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- API-Details finden Sie EnableMfaDevice in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccessKey`

Tools für PowerShell V5

Beispiel 1: Mit diesem Befehl werden die Zugriffsschlüssel für den IAM-Benutzer namens **`Bob`** aufgelistet. Beachten Sie, dass Sie die geheimen Zugriffsschlüssel für IAM-Benutzer nicht auflisten können. Bei Verlust der geheimen Zugriffsschlüssel müssen mit dem **`New-IAMAccessKey`** Cmdlet neue Zugriffsschlüssel erstellt werden.

```
Get-IAMAccessKey -UserName "Bob"
```

Ausgabe:

```
AccessKeyId                CreateDate                  Status             UserName
-----------                ----------                  ------             --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active             Bob
```

```
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM          Inactive            Bob
```

- API-Details finden Sie ListAccessKeys in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccessKeyLastUsed`

Tools für PowerShell V5

Beispiel 1: Gibt den Benutzernamen des Besitzers und die Informationen zur letzten Verwendung für den angegebenen Zugriffsschlüssel zurück.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API-Details finden Sie GetAccessKeyLastUsed in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountAlias`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Kontoalias für das AWS-Konto zurück.

```
Get-IAMAccountAlias
```

Ausgabe:

```
ExampleCo
```

- API-Details finden Sie ListAccountAliases in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccountAuthorizationDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountAuthorizationDetail`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Autorisierungsdetails zu den Identitäten im AWS Konto abgerufen und die Elementliste des zurückgegebenen Objekts angezeigt, einschließlich Benutzer, Gruppen und Rollen. Beispielsweise zeigt die **`UserDetailList`**-Eigenschaft Details zu den Benutzern an. Ähnliche Informationen sind in den **`RoleDetailList`**- und **`GroupDetailList`**-Eigenschaften verfügbar.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

Ausgabe:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

Ausgabe:

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/BackupToS3
CreateDate     : 1/27/2015 10:15:08 AM
```

```
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- API-Details finden Sie GetAccountAuthorizationDetails in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountPasswordPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zur Passwortrichtlinie für das aktuelle Konto zurückgegeben. Wenn keine Passwortrichtlinie für das Konto definiert ist, gibt der Befehl einen **`NoSuchEntity`**-Fehler zurück.

```
Get-IAMAccountPasswordPolicy
```

Ausgabe:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAccountSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAccountSummary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über die aktuelle Nutzung der IAM-Entität und die aktuellen Kontingente der IAM-Entität im AWS-Konto zurückgegeben.

```
Get-IAMAccountSummary
```

Ausgabe:

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
PolicyVersionsInUse                     4
InstanceProfiles                        1
```

```
...
```

- API-Details finden Sie GetAccountSummary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAttachedGroupPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedGroupPolicyList`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der **`Admins`** im Konto genannten IAM-Gruppe zugeordnet sind. AWS Verwenden Sie den Befehl **`Get-IAMGroupPolicyList`**, um die Liste der in der Gruppe eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

Ausgabe:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- API-Details finden Sie ListAttachedGroupPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAttachedRolePolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedRolePolicyList`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien zurück, die der IAM-Rolle mit dem Namen **`SecurityAuditRole`** im AWS -Konto zugeordnet sind. Verwenden Sie den **`Get-IAMRolePolicyList`**-Befehl, um die Liste der in die Rolle eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/SecurityAudit                    SecurityAudit
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMAttachedUserPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMAttachedUserPolicyList`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Namen und ARNs der verwalteten Richtlinien für den **`Bob`** im Konto genannten IAM-Benutzer zurück. AWS Verwenden Sie den **`Get-IAMUserPolicyList`**-Befehl, um die Liste der im IAM-Benutzer eingebetteten Inline-Richtlinien anzuzeigen.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

Ausgabe:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/TesterPolicy                     TesterPolicy
```

- API-Details finden Sie ListAttachedUserPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMContextKeysForCustomPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMContextKeysForCustomPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die im bereitgestellten Richtlinien-JSON vorhanden sind. Um mehrere Richtlinien bereitzustellen, können Sie eine durch Kommas getrennte Werteliste bereitstellen.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- API-Details finden Sie GetContextKeysForCustomPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMContextKeysForPrincipalPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMContextKeysForPrincipalPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Kontextschlüssel abgerufen, die in der bereitgestellten Richtlinien-JSON-Datei und den an die IAM-Entität angehängten Richtlinien (user/role usw.) enthalten sind. Für - PolicyInputList Sie können eine Liste mit mehreren Werten als kommagetrennte Werte angeben.

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- API-Details finden Sie GetContextKeysForPrincipalPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMCredentialReport`

Tools für PowerShell V5

Example 1: In diesem Beispiel wird der zurückgegebene Bericht geöffnet und als Array von Textzeilen an die Pipeline ausgegeben. Die erste Zeile ist die Kopfzeile mit durch Kommas getrennten Spaltennamen. Jede nachfolgende Zeile ist die Detailzeile für einen Benutzer, wobei jedes Feld durch Kommas getrennt ist. Bevor Sie den Bericht anzeigen können, müssen Sie ihn mit dem **`Request-IAMCredentialReport`** Cmdlet generieren. Um den Bericht als einzelne Zeichenfolge abzurufen, verwenden Sie **`-Raw`** anstelle von **`-AsTextArray`**. Der Alias **`-SplitLines`** wird auch für den **`-AsTextArray`** Switch akzeptiert. Die vollständige Liste der Spalten in der Ausgabe finden Sie in der Service-API-Referenz. Beachten Sie, dass Sie, wenn Sie **`-AsTextArray`** oder **`-SplitLines`** nicht verwenden, Sie den Text mithilfe der Klasse .NET **`StreamReader`** aus der **`.Content`**-Eigenschaft extrahieren müssen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

Ausgabe:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,passw
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015-04
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:00+0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/A,N/
A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- API-Details finden Sie GetCredentialReport in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMEntitiesForPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMEntitiesForPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Liste der IAM-Gruppen, Rollen und Benutzer zurück, denen die Richtlinie **`arn:aws:iam::123456789012:policy/TestPolicy`** zugeordnet ist.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

Ausgabe:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- API-Details finden Sie ListEntitiesForPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zur IAM-Gruppe **`Testers`** zurückgegeben, einschließlich einer Sammlung aller IAM-Benutzer, die zur Gruppe gehören.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

Ausgabe:

```
Group                                     IsTruncated           Marker
  Users
-----                                     -----------           ------
  -----
Amazon.IdentityManagement.Model.Group     False
  {Theresa, David}
```

```
$results.Group
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- API-Details finden Sie GetGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMGroupForUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupForUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der IAM-Gruppen zurückgegeben, zu denen der IAM-Benutzer **`David`** angehört.

```
Get-IAMGroupForUser -UserName David
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- API-Details finden Sie ListGroupsForUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Sammlung aller IAM-Gruppen zurück, die in der aktuellen Version definiert sind. AWS-Konto

```
Get-IAMGroupList
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- API-Details finden Sie ListGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zur eingebetteten Inline-Richtlinie mit dem Namen **`PowerUserAccess-Testers`** für die Gruppe **`Testers`** zurückgegeben. Die **`PolicyDocument`**-Eigenschaft ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-Testers
$results
```

Ausgabe:

```
GroupName     PolicyDocument                                                PolicyName
---------     --------------                                                ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- API-Details finden Sie GetGroupPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMGroupPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMGroupPolicyList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste der Inline-Richtlinien zurückgegeben, die in der Gruppe **`Testers`** eingebettet sind. Verwenden Sie den Befehl **`Get-IAMAttachedGroupPolicyList`**, um die verwalteten Richtlinien abzurufen, die der Gruppe zugeordnet sind.

```
Get-IAMGroupPolicyList -GroupName Testers
```

Ausgabe:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- API-Details finden Sie ListGroupPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfile`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt Details des benannten Instanzprofils zurück**`ec2instancerole`**, das im aktuellen AWS Konto definiert ist.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API-Details finden Sie GetInstanceProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMInstanceProfileForRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfileForRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details des Instance-Profils zurückgegeben, das der Rolle **`ec2instancerole`** zugeordnet ist.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

Ausgabe:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
```

```
    Path                : /
    Roles               : {ec2instancerole}
```

- API-Details finden Sie ListInstanceProfilesForRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMInstanceProfileList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMInstanceProfileList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Sammlung der Instanzprofile zurück, die in der aktuellen AWS-Konto Version definiert sind.

```
Get-IAMInstanceProfileList
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API-Details finden Sie ListInstanceProfiles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMLoginProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Erstellungsdatum des Passworts zurückgegeben und ob für den IAM-Benutzer **`David`** ein Passwortzurücksetzung erforderlich ist.

```
Get-IAMLoginProfile -UserName David
```

Ausgabe:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- API-Details finden Sie GetLoginProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zum MFA-Gerät zurückgegeben, das dem IAM-Benutzer **`David`** zugewiesen ist. In diesem Beispiel können Sie erkennen, dass es sich um ein virtuelles Gerät handelt, da die **`SerialNumber`** eine ARN und nicht die tatsächliche Seriennummer eines physischen Geräts ist.

```
Get-IAMMFADevice -UserName David
```

Ausgabe:

```
EnableDate                 SerialNumber                         UserName
----------                 ------------                         --------
4/8/2015 9:41:10 AM        arn:aws:iam::123456789012:mfa/David  David
```

- API-Details finden Sie ListMfaDevices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMOpenIDConnectProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zum OpenID-Connect-Anbieter zurückgegeben, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/accounts.google.com`**

lautet. Die **`ClientIDList`**-Eigenschaft ist eine Sammlung, die alle für diesen Anbieter definierten Client-IDs enthält.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

Ausgabe:

```
ClientIDList         CreateDate                ThumbprintList
        Url
------------         ----------                --------------
        ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- API-Details finden Sie GetOpenIdConnectProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMOpenIDConnectProviderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMOpenIDConnectProviderList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von ARNs aller OpenID Connect-Anbieter zurückgegeben, die im aktuellen AWS-Konto definiert sind.

```
Get-IAMOpenIDConnectProviderList
```

Ausgabe:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- API-Details finden Sie ListOpenIdConnectProviders in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zur verwalteten Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- API-Details finden Sie GetPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Sammlung der ersten drei verwalteten Richtlinien zurück, die im aktuellen AWS Konto verfügbar sind. Da nicht angegeben **`-scope`** ist, sind standardmäßig sowohl AWS verwaltete als auch vom Kunden verwaltete Richtlinien enthalten. **`all`**

```
Get-IAMPolicyList -MaxItem 3
```

Ausgabe:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
```

```
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

Beispiel 2: Dieses Beispiel gibt eine Sammlung der ersten beiden vom Kunden verwalteten Policen zurück, die im AWS Girokonto verfügbar sind. Es wird verwendet **`-Scope local`**, um die Ausgabe nur auf vom Kunden verwaltete Richtlinien zu beschränken.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
```

```
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- API-Details finden Sie ListPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyVersion`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt das Richtliniendokument für die **`v2`**-Version der Richtlinie zurück, deren ARN **`arn:aws:iam::123456789012:policy/MyManagedPolicy`** lautet. Das Richtliniendokument in der **`Document`**-Eigenschaft ist URL-codiert und wird in diesem Beispiel mit der .NET-Methode **`UrlDecode`** decodiert.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

Ausgabe:

```
CreateDate             Document
 IsDefaultVersion     VersionId
```

```
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
        v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- API-Details finden Sie GetPolicyVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMPolicyVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMPolicyVersionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der verfügbaren Versionen der Richtlinie zurückgegeben, deren ARN **`arn:aws:iam::123456789012:policy/MyManagedPolicy`** lautet. Um das Richtliniendokument für eine bestimmte Version abzurufen, verwenden Sie den Befehl **`Get-IAMPolicyVersion`** und geben Sie die **`VersionId`** der gewünschten Version an.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/MyManagedPolicy
```

Ausgabe:

```
CreateDate                  Document                IsDefaultVersion
    VersionId
```

```
----------                    --------                    ----------------
    ---------
2/12/2015 9:39:53 AM                                      True
    v2
2/12/2015 9:39:09 AM                                      False
    v1
```

- API-Details finden Sie ListPolicyVersions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRole`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die Details von **`lamda_exec_role`** zurück. Es enthält das Vertrauensrichtlinien-Dokument, das angibt, wer diese Rolle übernehmen kann. Das Richtliniendokument ist URL-codiert und kann mit der .NET-Methode **`UrlDecode`**decodiert werden. In diesem Beispiel wurden aus der ursprünglichen Richtlinie sämtliche Leerzeichen entfernt, bevor sie in die Richtlinie hochgeladen wurden. Um die Berechtigungsrichtliniendokumente anzuzeigen, die bestimmen, was jemand, der die Rolle übernimmt, tun kann, verwenden Sie **`Get-IAMRolePolicy`** für Inline-Richtlinien und **`Get-IAMPolicyVersion`** für angefügte verwaltete Richtlinien.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

Ausgabe:

```
Arn                      : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A
%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
```

```
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Statement":[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- API-Details finden Sie GetRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMRoleList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRoleList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste aller IAM-Rollen im AWS-Konto abgerufen.

```
Get-IAMRoleList
```

- API-Details finden Sie ListRoles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`Get-IAMRolePolicy`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Berechtigungs-Richtliniendokument für die Richtlinie mit dem Namen **`oneClick_lambda_exec_role_policy`** zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet ist. Das resultierende Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der .NET-Methode **`UrlDecode`**.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

Ausgabe:

```
PolicyDocument                                          PolicyName
           UserName
--------------                                          ----------
           --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

Ausgabe:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- API-Details finden Sie GetRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMRolePolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRolePolicyList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der Namen von Inline-Richtlinien zurückgegeben, die in die IAM-Rolle **`lamda_exec_role`** eingebettet sind. Um die Details einer Inline-Richtlinie anzuzeigen, verwenden Sie den Befehl **`Get-IAMRolePolicy`**.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

Ausgabe:

```
oneClick_lambda_exec_role_policy
```

- API-Details finden Sie ListRolePolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMRoleTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMRoleTagList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das der Rolle zugeordnete Tag abgerufen.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API-Details finden Sie ListRoleTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSAMLProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details über den SAML 2.0-Anbieter abgerufen, dessen ARM arn:aws:iam: :123456789012:saml- ist. provider/SAMLADFS Die Antwort enthält das Metadatendokument, das Sie vom Identitätsanbieter zur Erstellung der AWS SAML-Provider-Entität erhalten haben, sowie die Erstellungs- und Ablaufdaten.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

Ausgabe:

```
CreateDate                 SAMLMetadataDocument
     ValidUntil
----------                 --------------------
     ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-example1...
    12/23/2114 12:16:54 PM
```

- API-Details finden Sie GetSamlProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMSAMLProviderList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSAMLProviderList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der im aktuellen AWS-Konto erstellten SAML 2.0-Anbieter abgerufen. Es gibt den ARN, das Erstellungs- und das Ablaufdatum für jeden SAML-Anbieter zurück.

```
Get-IAMSAMLProviderList
```

Ausgabe:

```
Arn                                                CreateDate
 ValidUntil
---                                                ----------
 ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
 12/23/2114 12:16:54 PM
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Get-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`Get-IAMServerCertificate`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zum Serverzertifikat mit dem Namen **`MyServerCertificate`** abgerufen. Sie finden die Zertifikatsdetails in den Eigenschaften **`CertificateBody`** und **`ServerCertificateMetadata`**.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

Ausgabe:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
```

```
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- API-Details finden Sie GetServerCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMServerCertificateList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServerCertificateList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der Serverzertifikate abgerufen, die in das aktuelle AWS-Konto hochgeladen wurden.

```
Get-IAMServerCertificateList
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- API-Details finden Sie ListServerCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMServiceLastAccessedDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServiceLastAccessedDetail`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zum Service bereitgestellt, auf den die im Anforderungsaufruf zugeordnete IAM-Entität (Benutzer, Gruppe, Rolle oder Richtlinie) zuletzt zugegriffen hat.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

Ausgabe:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- API-Details finden Sie GetServiceLastAccessedDetails in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMServiceLastAccessedDetailWithEntity`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMServiceLastAccessedDetailWithEntity`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel stellt den Zeitstempel des letzten Zugriffs für den Service in der Anfrage durch die entsprechende IAM-Entität bereit.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

Ausgabe:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
```

```
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

Ausgabe:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

Ausgabe:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- API-Details finden Sie GetServiceLastAccessedDetailsWithEntities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMSigningCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details über das Signaturzertifikat abgerufen, das dem Benutzer mit dem Namen **`Bob`** zugeordnet ist.

```
Get-IAMSigningCertificate -UserName Bob
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
                  FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- API-Details finden Sie ListSigningCertificates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zum Benutzer mit dem Namen **`David`** abgerufen.

```
Get-IAMUser -UserName David
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
```

```
UserName         : David
```

Beispiel 2: In diesem Beispiel werden Details zum aktuell angemeldeten IAM-Benutzer abgerufen.

```
Get-IAMUser
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- API-Details finden Sie GetUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMUserList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Sammlung von Benutzern in der aktuellen Version abgerufen. AWS-Konto

```
Get-IAMUserList
```

Ausgabe:

```
      Arn              : arn:aws:iam::123456789012:user/Administrator
      CreateDate       : 10/16/2014 9:03:09 AM
      PasswordLastUsed : 3/4/2015 12:12:33 PM
      Path             : /
      UserId           : 7K3GJEANSKZF2EXAMPLE1
      UserName         : Administrator

      Arn              : arn:aws:iam::123456789012:user/Bob
      CreateDate       : 4/6/2015 12:54:42 PM
      PasswordLastUsed : 1/1/0001 12:00:00 AM
```

```
      Path             : /
      UserId           : L3EWNONDOM3YUEXAMPLE2
      UserName         : bab

      Arn              : arn:aws:iam::123456789012:user/David
      CreateDate       : 12/10/2014 3:39:27 PM
      PasswordLastUsed : 3/19/2015 8:44:04 AM
      Path             : /
      UserId           : Y4FKWQCXTA52QEXAMPLE3
      UserName         : David
```

- API-Details finden Sie ListUsers in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details der Inline-Richtlinie mit dem Namen **`Davids_IAM_Admin_Policy`** abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet ist. Das Richtliniendokument ist URL-codiert.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

Ausgabe:

```
PolicyDocument                                              PolicyName
    UserName
--------------                                              ----------
    --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
    David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- API-Details finden Sie GetUserPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMUserPolicyList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserPolicyList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der Namen der Inline-Richtlinien abgerufen, die in den IAM-Benutzer mit dem Namen **`David`** eingebettet sind.

```
Get-IAMUserPolicyList -UserName David
```

Ausgabe:

```
Davids_IAM_Admin_Policy
```

- API-Details finden Sie ListUserPolicies in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMUserTagList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMUserTagList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das dem Benutzer zugeordnete Tag abgerufen.

```
Get-IAMUserTagList -UserName joe
```

- API-Details finden Sie ListUserTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Get-IAMVirtualMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Sammlung der virtuellen MFA-Geräte abgerufen, die Benutzern im Konto zugewiesen sind. AWS Bei jeder **`User`**-Eigenschaft handelt es sich um ein Objekt mit Angaben zum IAM-Benutzer, dem das Gerät zugewiesen ist.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

Ausgabe:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- API-Details finden Sie ListVirtualMfaDevices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMAccessKey`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Paar aus Zugriffsschlüssel und geheimem Zugriffsschlüssel erstellt und dem Benutzer **`David`** zugewiesen. Stellen Sie sicher, dass Sie die **`AccessKeyId`**- und **`SecretAccessKey`**-Werte in einer Datei speichern, da Sie den

`SecretAccessKey` nur zu diesem Zeitpunkt abrufen können. Sie können es später nicht abrufen. Bei Verlust des geheimen Schlüssels müssen Sie ein neues Zugriffsschlüsselpaar erstellen.

```
New-IAMAccessKey -UserName David
```

Ausgabe:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- API-Details finden Sie CreateAccessKey in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMAccountAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Kontoalias für Ihr AWS Konto in geändert. **`mycompanyaws`** <accountidnumber>Die Adresse der Benutzeranmeldeseite wird in https://mycompanyaws.signin.aws.amazon.com/console. Die ursprüngliche URL geändert, wobei Ihre Konto-ID-Nummer anstelle des Alias verwendet wird (https://.signin.aws.amazon. com/console) funktioniert weiterhin. Alle zuvor definierten Alias-basierten URLs funktionieren jedoch nicht mehr.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- API-Details finden Sie CreateAccountAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue IAM-Gruppe mit dem Namen **`Developers`** erstellt.

```
New-IAMGroup -GroupName Developers
```

Ausgabe:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- API-Details finden Sie CreateGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMInstanceProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues IAM Instance-Profil mit dem Namen **`ProfileForDevEC2Instance`** erstellt. Sie müssen den Befehl **`Add-IAMRoleToInstanceProfile`** separat ausführen, um das Instance-Profil einer vorhandenen IAM-Rolle zuzuordnen, die Berechtigungen für die Instance bereitstellt. Fügen Sie abschließend das Instance-Profil einer EC2-Instance an, wenn Sie diese starten. Verwenden Sie dazu das Cmdlet **`New-EC2Instance`** entweder mit dem Parameter **`InstanceProfile_Arn`** oder **`InstanceProfile_Name`**.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

Ausgabe:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- API-Details finden Sie CreateInstanceProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-IAMLoginProfile

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMLoginProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein (temporäres) Passwort für den IAM-Benutzer Bob erstellt und festgelegt, dass der Benutzer das Passwort bei der nächsten Anmeldung von **Bob** ändern muss.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

Ausgabe:

```
CreateDate                    PasswordResetRequired                UserName
----------                    ---------------------                --------
4/14/2015 12:26:30 PM         True                                 Bob
```

- API-Details finden Sie CreateLoginProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-IAMOpenIDConnectProvider

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMOpenIDConnectProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein IAM-OIDC-Anbieter erstellt, der dem OIDC-kompatiblen Anbieter-Service zugeordnet ist, der unter der URL **https://example.oidcprovider.com** und der Client-ID **my-testapp-1** zu finden ist. Der OIDC-Anbieter stellt den Fingerabdruck bereit. Um den Fingerabdruck zu authentifizieren, folgen Sie den Schritten unter http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList my-
testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

Ausgabe:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API-Details finden Sie CreateOpenIdConnectProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-IAMPolicy

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird im aktuellen AWS Konto eine neue IAM-Richtlinie mit dem Namen **MySamplePolicy** Die Datei **MySamplePolicy.json** stellt den Richtlinieninhalt bereit. Beachten Sie, dass Sie den Switch-Parameter **-Raw** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- API-Details finden Sie CreatePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-IAMPolicyVersion

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMPolicyVersion`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue „v2“-Version der IAM-Richtlinie erstellt, deren ARN **arn:aws:iam::123456789012:policy/MyPolicy** lautet, und als standardmäßige Version festgelegt. Die Datei **NewPolicyVersion.json** enthält den Richtlinieninhalt. Beachten

Sie, dass Sie den Switch-Parameter **-Raw** verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

Ausgabe:

```
CreateDate                          Document                 IsDefaultVersion
        VersionId
----------                          --------                 ----------------
        ---------
4/15/2015 10:54:54 AM                                        True
        v2
```

- API-Details finden Sie CreatePolicyVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-IAMRole

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Rolle mit dem Namen **MyNewRole** erstellt und die in der Datei **NewRoleTrustPolicy.json** gefundene Richtlinie daran angefügt. Beachten Sie, dass Sie den **-Raw**-Switch-Parameter verwenden müssen, um die JSON-Richtliniendatei erfolgreich zu verarbeiten. Das in der Ausgabe angezeigte Richtliniendokument ist URL-codiert. Die Decodierung erfolgt in diesem Beispiel mit der **UrlDecode** .NET-Methode.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

Ausgabe:

```
Arn                      : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0D
%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
```

```
                            %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow%22%2C
%0D%0A%20%20%20%20%20%20
                            %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                            %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                            %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D%0A
%20%20%20%20%7D%0D%0A%20
                            %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- API-Details finden Sie CreateRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMSAMLProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue SAML-Anbieter-Entität in IAM erstellt. Sie ist **`MySAMLProvider`** benannt und wird durch das SAML-Metadatendokument in der Datei **`SAMLMetaData.xml`** beschrieben, die separat von der Website des SAML-Service-Anbieters heruntergeladen wurde.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

Ausgabe:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- Weitere API-Informationen finden Sie unter CreateSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`New-IAMServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`New-IAMServiceLinkedRole`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine serviceverknüpfte Rolle für den Autoscaling-Service erstellt.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support autoscaling"
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein IAM-Benutzer mit dem Namen **`Bob`** erstellt. Wenn Bob sich bei der AWS Konsole anmelden muss, müssen Sie den Befehl separat ausführen, **`New-IAMLoginProfile`** um ein Anmeldeprofil mit einem Passwort zu erstellen. Wenn Bob plattformübergreifende CLI-Befehle ausführen AWS PowerShell oder AWS API-Aufrufe tätigen muss, müssen Sie den **`New-IAMAccessKey`** Befehl separat ausführen, um Zugriffsschlüssel zu erstellen.

```
New-IAMUser -UserName Bob
```

Ausgabe:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- API-Details finden Sie CreateUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `New-IAMVirtualMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **`Base32StringSeed`**, den das virtuelle MFA-Softwareprogramm zum Erstellen eines Kontos benötigt (als Alternative zum QR-Code), extrahiert. Nachdem Sie das Programm mit dem Wert konfiguriert haben, erhalten Sie vom Programm zwei aufeinanderfolgende Authentifizierungs-Codes. Verwenden Sie abschließend den letzten Befehl, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **`Bob`** zu verknüpfen und das Konto mit den beiden Authentifizierungscodes zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

Ausgabe:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to register
 account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

Beispiel 2: In diesem Beispiel wird ein neues virtuelles MFA-Gerät erstellt. In Zeilen 2 und 3 wird der Wert **QRCodePNG** extrahiert und in eine Datei geschrieben. Dieses Bild kann vom virtuellen MFA-Softwareprogramm gescannt werden, um ein Konto zu erstellen (als Alternative zur manuellen Eingabe des Base32StringSeed Werts). Nachdem Sie das Konto in Ihrem virtuellen MFA-Programm erstellt haben, erhalten Sie zwei aufeinanderfolgende Authentifizierungscodes. Geben Sie diese in die letzten Befehle ein, um das virtuelle MFA-Gerät mit dem IAM-Benutzer **Bob** zu verknüpfen und das Konto zu synchronisieren.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path QRCode.png
```

Ausgabe:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- API-Details finden Sie CreateVirtualMfaDevice in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Publish-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-IAMServerCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Serverzertifikat in das IAM-Konto hochgeladen. Die Dateien, die den Text des Zertifikats, den privaten Schlüssel und (optional) die Zertifikatskette enthalten, müssen alle PEM-codiert sein. Beachten Sie, dass die Parameter den tatsächlichen Inhalt der Dateien und nicht die Dateinamen erfordern. Um den Dateiinhalt erfolgreich zu verarbeiten, müssen Sie den **-Raw**-Switch-Parameter verwenden.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

Ausgabe:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- API-Details finden Sie UploadServerCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Publish-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-IAMSigningCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues X.509 Signaturzertifikat hochgeladen und es dem angegebenen IAM-Benutzer zugeordnet. **`Bob`** Die Datei mit dem Zertifikatstext ist PEM-codiert. Der **`CertificateBody`**-Parameter erfordert den tatsächlichen Inhalt der Zertifikatsdatei und nicht den Dateinamen. Sie müssen den **`-Raw`**-Switch-Parameter verwenden, um die Datei erfolgreich zu verarbeiten.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

Ausgabe:

```
CertificateBody : -----BEGIN CERTIFICATE-----
                  MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
                  VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
                  b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
                  BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
                  MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
                  VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
                  b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
                  Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
```

```
                    FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId    : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status           : Active
UploadDate       : 4/20/2015 1:26:01 PM
UserName         : Bob
```

- API-Details finden Sie UploadSigningCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMGroupPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die vom Kunden verwaltete Richtlinie mit dem Namen **`TesterPolicy`** an die IAM-Gruppe **`Testers`** angefügt. Die in der Standardversion dieser Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

Beispiel 2: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen der **`AdministratorAccess`** IAM-Gruppe angehängt. **`Admins`** Die in der neuesten Version dieser Richtlinie definierten Berechtigungen wirken sich unmittelbar auf die Benutzer in dieser Gruppe aus.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- API-Details finden Sie AttachGroupPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMRolePolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen der **`SecurityAudit`** IAM-Rolle angehängt. **`CoSecurityAuditors`** Die Benutzer, die diese Rolle übernehmen, sind unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- API-Details finden Sie AttachRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Register-IAMUserPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die AWS verwaltete Richtlinie mit dem Namen des **`AmazonCognitoPowerUser`** IAM-Benutzers verknüpft. **`Bob`** Der Benutzer ist unmittelbar von den in der neuesten Version dieser Richtlinie definierten Berechtigungen betroffen.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- API-Details finden Sie AttachUserPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccessKey`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das AWS Zugriffsschlüsselpaar mit der Schlüssel-ID **`AKIAIOSFODNN7EXAMPLE`** aus dem benannten Benutzer gelöscht. **`Bob`**

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API-Details finden Sie DeleteAccessKey in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccountAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Konto-Alias von Ihrem AWS-Konto entfernt. Die Benutzeranmeldeseite mit dem Alias unter funktioniert nicht https://mycompanyaws.signin.aws.amazon.com/console mehr. Sie müssen stattdessen die ursprüngliche URL mit Ihrer AWS-Konto ID-Nummer unter https://<accountidnumber>.signin.aws.amazon verwenden. com/console.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API-Details finden Sie DeleteAccountAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMAccountPasswordPolicy`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel löscht die Kennwortrichtlinie für AWS-Konto und setzt alle Werte auf ihre ursprünglichen Standardwerte zurück. Wenn derzeit keine Kennwortrichtlinie vorhanden ist, wird die folgende Fehlermeldung angezeigt: Die Kontorichtlinie mit dem Namen PasswordPolicy kann nicht gefunden werden.

```
Remove-IAMAccountPasswordPolicy
```

- API-Details finden Sie DeleteAccountPasswordPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMClientIDFromOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMClientIDFromOpenIDConnectProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Client-ID **`My-TestApp-3`** aus der Liste der Client-IDs entfernt, die dem IAM-OIDC-Anbieter zugeordnet sind, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- API-Details finden Sie RemoveClientIdFromOpenIdConnectProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe mit dem Namen **`MyTestGroup`** gelöscht. Mit dem ersten Befehl werden alle IAM-Benutzer entfernt, die Mitglieder der Gruppe sind, und mit dem zweiten Befehl wird die IAM-Gruppe gelöscht. Beide Befehle funktionieren ohne weitere Bestätigungsaufforderung.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API-Details finden Sie DeleteGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMGroupPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`TesterPolicy`** aus der IAM-Gruppe **`Testers`** entfernt. Die Benutzer in dieser Gruppe verlieren sofort die in dieser Richtlinie definierten Berechtigungen.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- API-Details finden Sie DeleteGroupPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMInstanceProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das EC2-Instance-Profil mit dem Namen **`MyAppInstanceProfile`** gelöscht. Mit dem ersten Befehl werden alle Rollen vom Instance-Profil getrennt, und mit dem zweiten Befehl wird das Instance-Profil gelöscht.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles | Remove-
IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- API-Details finden Sie DeleteInstanceProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMLoginProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Anmeldeprofil des IAM-Benutzers **`Bob`** gelöscht. Dadurch wird verhindert, dass sich der Benutzer an der Konsole anmeldet. AWS Es hindert den Benutzer nicht daran, AWS CLI- oder API-Aufrufe mit AWS Zugriffsschlüsseln auszuführen PowerShell, die möglicherweise noch an das Benutzerkonto angehängt sind.

```
Remove-IAMLoginProfile -UserName Bob
```

- API-Details finden Sie DeleteLoginProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMOpenIDConnectProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMOpenIDConnectProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM-OIDC-Anbieter gelöscht, der eine Verbindung zum Anbieter **`example.oidcprovider.com`** herstellt. Stellen Sie sicher, dass Sie alle Rollen aktualisieren oder löschen, die im **`Principal`**-Element der Vertrauensrichtlinie der Rolle auf diesen Anbieter verweisen.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API-Details finden Sie DeleteOpenIdConnectProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Richtlinie gelöscht, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet. Bevor Sie die Richtlinie löschen können, müssen Sie zunächst alle Versionen außer der Standardversion löschen, indem Sie **`Remove-IAMPolicyVersion`** ausführen. Sie müssen die Richtlinie auch von allen IAM-Benutzern, -Gruppen oder -Rollen trennen.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen gelöscht werden, die Richtlinie dann von allen zugeordneten IAM-Entitäten getrennt wird und schließlich die Richtlinie selbst gelöscht wird. In der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßige Version gekennzeichnet sind, in einer

Sammlung erfasst und dann wird jede Richtlinie in der Sammlung gelöscht. In der dritten Zeile werden alle Benutzer, Gruppen und Rollen des IAM abgerufen, denen die Richtlinie angefügt ist. Die Zeilen vier bis sechs trennen die Richtlinie von jeder angefügten Entität. In der letzten Zeile wird dieser Befehl zum Entfernen der verwalteten Richtlinie sowie der verbleibenden Standardversion verwendet. Das Beispiel schließt den **`-Force`**-Switch-Parameter in jede Zeile ein, in der er benötigt wird, um Bestätigungsaufforderungen zu unterdrücken.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- API-Details finden Sie DeletePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMPolicyVersion`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die als **`v2`** identifizierte Version aus der Richtlinie gelöscht, deren ARN **`arn:aws:iam::123456789012:policy/MySamplePolicy`** lautet.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy -
VersionID v2
```

Beispiel 2: In diesem Beispiel wird eine Richtlinie gelöscht, indem zuerst alle nicht standardmäßigen Richtlinienversionen und dann die Richtlinie selbst gelöscht werden. In der ersten Zeile wird das Richtlinienobjekt abgerufen. Über die zweite Zeile werden alle Richtlinienversionen, die nicht als standardmäßig gekennzeichnet sind, in einer Sammlung abgerufen und dann wird dieser Befehl zur Löschung jeder einzelnen Richtlinie in der Sammlung verwendet. In der letzten Zeile wird die Richtlinie selbst sowie die verbleibende Standardversion entfernt. Beachten Sie, dass Sie zum erfolgreichen Löschen einer verwalteten Richtlinie die Richtlinie auch von allen Benutzern, Gruppen oder Rollen trennen müssen, indem Sie die Befehle **`Unregister-IAMUserPolicy`**, **`Unregister-IAMGroupPolicy`** und **`Unregister-`**

IAMRolePolicy verwenden. Weitere Informationen finden Sie im Beispiel für das **Remove-IAMPolicy** Cmdlet.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- API-Details finden Sie DeletePolicyVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Remove-IAMRole

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **MyNewRole** aus dem aktuellen IAM-Konto gelöscht. Bevor Sie die Rolle löschen können, müssen Sie zunächst mit dem **Unregister-IAMRolePolicy**-Befehl alle verwalteten Richtlinien entfernen. Inline-Richtlinien werden mit der Rolle gelöscht.

```
Remove-IAMRole -RoleName MyNewRole
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien von der Rolle mit dem Namen **MyNewRole** getrennt und dann die Rolle gelöscht. In der ersten Zeile werden alle verwalteten Richtlinien abgerufen, die der Rolle als Sammlung zugeordnet sind, und dann werden alle Richtlinien in der Sammlung von der Rolle getrennt. In der zweiten Zeile wird die Rolle selbst gelöscht. Inline-Richtlinien werden zusammen mit der Rolle gelöscht.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- API-Details finden Sie DeleteRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Remove-IAMRoleFromInstanceProfile

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRoleFromInstanceProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Rolle mit dem Namen **`MyNewRole`** aus dem EC2-Instance-Profil mit dem Namen **`MyNewRole`** gelöscht. Ein in der IAM-Konsole erstelltes Instance-Profil hat immer denselben Namen wie die Rolle, wie in diesem Beispiel. Wenn Sie sie in der API oder CLI erstellen, können sie unterschiedliche Namen haben.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName MyNewRole
 -Force
```

- API-Details finden Sie RemoveRoleFromInstanceProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMRolePermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRolePermissionsBoundary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie die an eine IAM-Rolle angefügte Berechtigung entfernt wird.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API-Details finden Sie DeleteRolePermissionsBoundary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRolePolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie **`S3AccessPolicy`** gelöscht, die in die IAM-Rolle **`S3BackupRole`** eingebettet ist.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API-Details finden Sie DeleteRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMRoleTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMRoleTag`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel entfernt das Tag aus der Rolle mit dem Namen "MyRoleName" mit dem Tag-Schlüssel „abac“. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API-Details finden Sie UntagRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMSAMLProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM SAML 2.0-Anbieter gelöscht, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider`** lautet.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- Weitere API-Informationen finden Sie unter DeleteSAMLProvider in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Remove-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`Remove-IAMServerCertificate`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Serverzertifikat mit dem Namen **`MyServerCert`** gelöscht.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API-Details finden Sie DeleteServerCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMServiceLinkedRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wurde die serviceverknüpfte Rolle gelöscht. Bitte beachten Sie, dass dieser Befehl fehlschlägt, wenn der Service diese Rolle weiterhin verwendet.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API-Details finden Sie DeleteServiceLinkedRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMSigningCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Signaturzertifikat mit der ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** des IAM-Benutzers mit dem Namen **`Bob`** gelöscht.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API-Details finden Sie DeleteSigningCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **`Bob`** gelöscht.

```
Remove-IAMUser -UserName Bob
```

Beispiel 2: In diesem Beispiel wird der IAM-Benutzer mit dem Namen **`Theresa`** zusammen mit allen Elementen gelöscht, die zuerst gelöscht werden müssen.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn $pol.PolicyArn
 -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

```
# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- API-Details finden Sie DeleteUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMUserFromGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserFromGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** aus der Gruppe **`Testers`** entfernt.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Gruppen gesucht, in denen der IAM-Benutzer **`Theresa`** Mitglied ist, und dann wird **`Theresa`** aus diesen Gruppen entfernt.

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName -
UserName Theresa -Force }
```

Beispiel 3: In diesem Beispiel wird eine alternative Möglichkeit zum Entfernen des IAM-Benutzers **`Bob`** aus der **`Testers`**-Gruppe gezeigt.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -GroupName
 Testers -Force
```

- API-Details finden Sie RemoveUserFromGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMUserPermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserPermissionsBoundary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze entfernt wird, die einem IAM-Benutzer zugeordnet ist.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- API-Details finden Sie DeleteUserPermissionsBoundary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Inline-Richtlinie mit dem Namen **`AccessToEC2Policy`** gelöscht, die in den IAM-Benutzer mit dem Namen **`Bob`** eingebettet ist.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

Beispiel 2: In diesem Beispiel werden alle Inline-Richtlinien gefunden, die in den IAM-Benutzer mit dem Namen **`Theresa`** eingebettet sind, und dann gelöscht.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- API-Details finden Sie DeleteUserPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMUserTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMUserTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Tag vom Benutzer namens „joe“ mit den Tag-Schlüsseln „abac“ und „xyzw“ entfernt. Um mehrere Tags zu entfernen, geben Sie eine durch Kommas getrennte Liste mit Tag-Schlüsseln an.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- API-Details finden Sie UntagUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-IAMVirtualMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-IAMVirtualMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das virtuelle IAM-MFA-Gerät gelöscht, dessen ARN **`arn:aws:iam::123456789012:mfa/bob`** lautet.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

Beispiel 2: In diesem Beispiel wird überprüft, ob der IAM-Benutzerin Theresa ein MFA-Gerät zugewiesen ist. Wenn eines gefunden wird, wird das Gerät für den IAM-Benutzer deaktiviert. Wenn das Gerät virtuell ist, wird es ebenfalls gelöscht.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -SerialNumber
 $mfa.SerialNumber }
}
```

- API-Details finden Sie DeleteVirtualMfaDevice in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Request-IAMCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `Request-IAMCredentialReport`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Erstellung eines neuen Berichts angefordert, was alle vier Stunden erfolgen kann. Wenn der letzte Bericht noch aktuell ist, lautet das Feld „Status“ **`COMPLETE`**. Verwenden Sie **`Get-IAMCredentialReport`**, um den vollständigen Bericht anzuzeigen.

```
Request-IAMCredentialReport
```

Ausgabe:

```
Description                                                    State
```

```
-----------                                                        -----
No report exists. Starting a new report generation task            STARTED
```

- API-Details finden Sie GenerateCredentialReport in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Request-IAMServiceLastAccessedDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Request-IAMServiceLastAccessedDetail`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel entspricht einem API-Cmdlet. GenerateServiceLastAccessedDetails Dies bietet eine Job-ID, die verwendet werden kann in Get-IAMServiceLastAccessedDetail und Get-IAMServiceLastAccessedDetailWithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API-Details finden Sie GenerateServiceLastAccessedDetails in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-IAMDefaultPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMDefaultPolicyVersion`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die **`v2`**-Version der Richtlinie, deren ARN **`arn:aws:iam::123456789012:policy/MyPolicy`** lautet, als standardmäßige aktive Version festgelegt.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
VersionId v2
```

- API-Details finden Sie SetDefaultPolicyVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-IAMRolePermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMRolePermissionsBoundary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für eine IAM-Rolle festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API-Details finden Sie PutRolePermissionsBoundary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-IAMUserPermissionsBoundary`

Das folgende Codebeispiel zeigt die Verwendung. `Set-IAMUserPermissionsBoundary`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie die Berechtigungsgrenze für den Benutzer festgelegt wird. Sie können AWS verwaltete Richtlinien oder benutzerdefinierte Richtlinien als Berechtigungsgrenze festlegen.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API-Details finden Sie PutUserPermissionsBoundary in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Sync-IAMMFADevice`

Das folgende Codebeispiel zeigt die Verwendung. `Sync-IAMMFADevice`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das MFA-Gerät, das dem IAM-Benutzer **`Bob`** zugeordnet ist und dessen ARN **`arn:aws:iam::123456789012:mfa/bob`** lautet, mit einem Authentifizierungs-Programm synchronisiert, das die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

Beispiel 2: In diesem Beispiel wird das IAM-MFA-Gerät, das dem IAM-Benutzer **`Theresa`** zugeordnet ist, mit einem physischen Gerät synchronisiert, das die Seriennummer **`ABCD12345678`** hat und die beiden Authentifizierungs-Codes bereitgestellt hat.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- API-Details finden Sie ResyncMfaDevice in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMGroupPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterAccessPolicy`** lautet, von der Gruppe mit dem Namen **`Testers`** getrennt.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Gruppe mit dem Namen **`Testers`** zugeordnet sind, und von der Gruppe getrennt.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- API-Details finden Sie DetachGroupPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMRolePolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die verwaltete Gruppenrichtlinie, deren ARN **`arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy`** lautet, von der Rolle mit dem Namen **`FedTesterRole`** getrennt.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die der Rolle mit dem Namen **`FedTesterRole`** zugeordnet sind, und von der Rolle getrennt.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy -
Rolename FedTesterRole
```

- API-Details finden Sie DetachRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-IAMUserPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die verwaltete Richtlinie, deren ARN **`arn:aws:iam::123456789012:policy/TesterPolicy`** lautet, vom IAM-Benutzer mit dem Namen **`Bob`** getrennt.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::123456789012:policy/
TesterPolicy
```

Beispiel 2: In diesem Beispiel werden alle verwalteten Richtlinien gesucht, die dem IAM-Benutzer mit dem Namen **`Theresa`** zugeordnet sind, und diese Richtlinien werden vom Benutzer getrennt.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -Username
 Theresa
```

- API-Details finden Sie DetachUserPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAccessKey`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status des Zugriffsschlüssels **`AKIAIOSFODNN7EXAMPLE`** für den IAM-Benutzer mit dem Namen **`Bob`** in **`Inactive`** geändert.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status Inactive
```

- API-Details finden Sie UpdateAccessKey in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAccountPasswordPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Passwortrichtlinie für das Konto mit den angegebenen Einstellungen aktualisiert. Beachten Sie, dass alle Parameter, die nicht im Befehl enthalten sind, nicht unverändert bleiben. Stattdessen werden diese auf die Standardwerte zurückgesetzt.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- API-Details finden Sie UpdateAccountPasswordPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMAssumeRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMAssumeRolePolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die IAM-Rolle mit dem Namen **`ClientRole`** mit einer neuen Vertrauensrichtlinie aktualisiert, deren Inhalt aus der Datei **`ClientRolePolicy.json`** stammt.

Beachten Sie, dass Sie den **-Raw**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- API-Details finden Sie UpdateAssumeRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die IAM-Gruppe **Testers** in **AppTesters** umbenannt.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

Beispiel 2: In diesem Beispiel wird der Pfad der IAM-Gruppe **AppTesters** in **/Org1/Org2/** geändert. Dadurch wird die ARN für die Gruppe in **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters** geändert.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- API-Details finden Sie UpdateGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMLoginProfile`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMLoginProfile`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues temporäres Passwort für den IAM-Benutzer **Bob** festgelegt und der Benutzer muss das Passwort bei der nächsten Anmeldung ändern.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -PasswordResetRequired
 $true
```

- API-Details finden Sie UpdateLoginProfile in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMOpenIDConnectProviderThumbprint`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMOpenIDConnectProviderThumbprint`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Zertifikat-Fingerabdruckliste für den OIDC-Anbieter, dessen ARN **`arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`** lautet, aktualisiert, um einen neuen Fingerabdruck zu verwenden. Der OIDC-Anbieter gibt den neuen Wert frei, wenn sich das dem Anbieter zugeordnete Zertifikat ändert.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- API-Details finden Sie UpdateOpenIdConnectProviderThumbprint in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMRole`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Rollenbeschreibung und der maximale Sitzungsdauerwert (in Sekunden) aktualisiert, für den die Sitzung einer Rolle angefordert werden kann.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- API-Details finden Sie UpdateRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMRoleDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMRoleDescription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung einer IAM-Rolle in Ihrem Konto aktualisiert.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API-Details finden Sie UpdateRoleDescription in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMSAMLProvider`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMSAMLProvider`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der SAML-Provider in IAM, dessen ARN **`arn:aws:iam::123456789012:saml-provider/SAMLADFS`** lautet, mit einem neuen SAML-Metadatendokument aus der Datei **`SAMLMetaData.xml`** aktualisiert. Beachten Sie, dass Sie den **`-Raw`**-Switch-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- API-Details finden Sie UpdateSamlProvider in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMServerCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Zertifikat mit dem Namen **`MyServerCertificate`** in **`MyRenamedServerCertificate`** umbenannt.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

Beispiel 2: Dieses Beispiel verschiebt das benannte Zertifikat **`MyServerCertificate`** in den Pfad/Org1/Org2/. Dadurch wird die ARN für die Ressource zu **`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`** geändert.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- API-Details finden Sie UpdateServerCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMSigningCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMSigningCertificate`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Zertifikat aktualisiert, das dem IAM-Benutzer mit dem Namen **`Bob`** und der Zertifikats-ID **`Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU`** zugeordnet ist, um es als inaktiv zu markieren.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- API-Details finden Sie UpdateSigningCertificate in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-IAMUser`

Das folgende Codebeispiel zeigt die Verwendung. `Update-IAMUser`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der IAM-Benutzer **`Bob`** in **`Robert`** umbenannt.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

Beispiel 2: In diesem Beispiel wird der Pfad des IAM-Benutzers **`Bob`** in **`/Org1/Org2/`** geändert, wodurch der ARN für den Benutzer effektiv zu **`arn:aws:iam::123456789012:user/Org1/Org2/bob`** geändert wird.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- API-Details finden Sie UpdateUser in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-IAMGroupPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMGroupPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`AppTesterPolicy`** erstellt und in die IAM-Gruppe **`AppTesters`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie wird in der Datei **`apptesterpolicy.json`** gespeichert. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- API-Details finden Sie PutGroupPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-IAMRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMRolePolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`FedTesterRolePolicy`** erstellt und in die IAM-Rolle **`FedTesterRole`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`FedTesterPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- API-Details finden Sie PutRolePolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-IAMUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Write-IAMUserPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Inline-Richtlinie mit dem Namen **`EC2AccessPolicy`** erstellt und in den IAM-Benutzer **`Bob`** eingebettet. Wenn bereits eine Inline-Richtlinie mit demselben Namen vorhanden ist, wird diese überschrieben. Der Inhalt der JSON-Richtlinie stammt aus der Datei **`EC2AccessPolicy.json`**. Beachten Sie, dass Sie den **`-Raw`**-Parameter verwenden müssen, um den Inhalt der JSON-Datei erfolgreich zu verarbeiten.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument (Get-
Content -Raw EC2AccessPolicy.json)
```

- API-Details finden Sie PutUserPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Kinesis-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS -Tools für PowerShell V5 mit Kinesis Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-KINRecord`

Das folgende Codebeispiel zeigt die Verwendung. `Get-KINRecord`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird gezeigt, wie Daten aus einer Reihe von einem oder mehreren Datensätzen zurückgegeben und extrahiert werden können. Der angegebene Iterator Get-KINRecord bestimmt die Startposition der zurückzugebenden Datensätze, die in diesem Beispiel in der Variablen $records erfasst werden. Auf jeden einzelnen Datensatz kann dann durch Indizierung der $records-Sammlung zugegriffen werden. Unter der Annahme, dass es sich bei den Daten im Datensatz um UTF-8 codierten Text handelt, zeigt der letzte Befehl, wie Sie die Daten aus dem Objekt extrahieren und als Text MemoryStream an die Konsole zurückgeben können.

```
$records
$records = Get-KINRecord -ShardIterator "AAAAAAAAAAGIc....9VnbiRNaP"
```

Ausgabe:

```
MillisBehindLatest NextShardIterator           Records
------------------ -----------------           -------
0                  AAAAAAAAAAERNIq...uDn11HuUs  {Key1, Key2}
```

```
$records.Records[0]
```

Ausgabe:

```
ApproximateArrivalTimestamp Data                   PartitionKey SequenceNumber
--------------------------- ----                   ------------ --------------
3/7/2016 5:14:33 PM         System.IO.MemoryStream Key1
 4955986459776...931586
```

```
[Text.Encoding]::UTF8.GetString($records.Records[0].Data.ToArray())
```

Ausgabe:

```
test data from string
```

- API-Details finden Sie GetRecords in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-KINShardIterator`

Das folgende Codebeispiel zeigt die Verwendung. `Get-KINShardIterator`

Tools für PowerShell V5

Beispiel 1: Gibt einen Shard-Iterator für den angegebenen Shard und die Startposition zurück. Einzelheiten zu den Shard-Identifikatoren und Sequenznummern können der Ausgabe des Get-KINStream Cmdlets entnommen werden, indem auf die Shards -Auflistung des zurückgegebenen Stream-Objekts verwiesen wird. Der zurückgegebene Iterator kann zusammen mit dem Get-KINRecord Cmdlet verwendet werden, um Datensätze im Shard abzurufen.

```
Get-KINShardIterator -StreamName "mystream" -ShardId "shardId-000000000000" -
ShardIteratorType AT_SEQUENCE_NUMBER -StartingSequenceNumber "495598645..."
```

Ausgabe:

```
AAAAAAAAAAGIc....9VnbiRNaP
```

- API-Details finden Sie GetShardIterator in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `Get-KINStream`

Tools für PowerShell V5

Beispiel 1: Gibt Details des angegebenen Streams zurück.

```
Get-KINStream -StreamName "mystream"
```

Ausgabe:

```
HasMoreShards        : False
RetentionPeriodHours : 24
Shards               : {}
StreamARN            : arn:aws:kinesis:us-west-2:123456789012:stream/mystream
StreamName           : mystream
StreamStatus         : ACTIVE
```

- API-Details finden Sie DescribeStream in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `New-KINStream`

Tools für PowerShell V5

Beispiel 1: Erzeugt einen neuen Stream.

```
New-KINStream -StreamName "mystream" -ShardCount 1
```

- API-Details finden Sie CreateStream in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-KINStream`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-KINStream`

Tools für PowerShell V5

Beispiel 1: Löscht den angegebenen Stream. Sie werden um eine Bestätigung gebeten, bevor der Befehl ausgeführt wird. Verwenden Sie den Schalter -Force, um die Bestätigungsaufforderung zu unterdrücken.

```
Remove-KINStream -StreamName "mystream"
```

- API-Details finden Sie DeleteStream in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-KINRecord`

Das folgende Codebeispiel zeigt die Verwendung`Write-KINRecord`.

Tools für PowerShell V5

Beispiel 1: Schreibt einen Datensatz, der die an den Parameter -Text übergebene Zeichenfolge enthält.

```
Write-KINRecord -Text "test data from string" -StreamName "mystream" -PartitionKey
 "Key1"
```

Beispiel 2: Schreibt einen Datensatz, der die Daten aus der angegebenen Datei enthält. Die Datei wird als Folge von Bytes behandelt. Wenn sie Text enthält, sollte sie daher vor der Verwendung mit diesem Cmdlet mit der erforderlichen Kodierung geschrieben werden.

```
Write-KINRecord -FilePath "C:\TestData.txt" -StreamName "mystream" -PartitionKey
 "Key2"
```

- API-Details finden Sie PutRecord in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Lambda-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Lambda Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-LMResourceTag`

Tools für PowerShell V5

Beispiel 1: Fügt die drei Tags (Washington, Oregon und Kalifornien) und ihre zugehörigen Werte zu der angegebenen Funktion hinzu, die durch ihre ARN identifiziert wird.

```
Add-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -Tag @{ "Washington" = "Olympia"; "Oregon"
 = "Salem"; "California" = "Sacramento" }
```

- API-Details finden Sie TagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMAccountSetting`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMAccountSetting`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel wird angezeigt, um das Kontolimit und die Kontonutzung zu vergleichen

```
Get-LMAccountSetting | Select-Object
 @{Name="TotalCodeSizeLimit";Expression={$_.AccountLimit.TotalCodeSize}},
 @{Name="TotalCodeSizeUsed";Expression={$_.AccountUsage.TotalCodeSize}}
```

Ausgabe:

```
TotalCodeSizeLimit TotalCodeSizeUsed
------------------ -----------------
       80530636800          15078795
```

- API-Details finden Sie GetAccountSettings in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Gewichtungen der Routing-Konfiguration für einen bestimmten Lambda-Funktionsalias abgerufen.

```
Get-LMAlias -FunctionName "MylambdaFunction123" -Name "newlabel1" -Select
 RoutingConfig
```

Ausgabe:

```
AdditionalVersionWeights
------------------------
{[1, 0.6]}
```

- API-Details finden Sie GetAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionConcurrency`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ruft die reservierte Gleichzeitigkeit für die Lambda-Funktion ab

```
Get-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -Select *
```

Ausgabe:

```
ReservedConcurrentExecutions
----------------------------
100
```

- API-Details finden Sie GetFunctionConcurrency in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die versionsspezifische Konfiguration einer Lambda-Funktion zurück.

```
Get-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Qualifier
 "PowershellAlias"
```

Ausgabe:

```
CodeSha256                  : uWOW0R7z+f0VyLuUg7+/D08hkMFsq0SF4seuyUZJ/R8=
CodeSize                    : 1426
DeadLetterConfig            : Amazon.Lambda.Model.DeadLetterConfig
Description                 : Verson 3 to test Aliases
Environment                 : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn                 : arn:aws:lambda:us-
east-1:123456789012:function:MylambdaFunction123
```

```
                                :PowershellAlias
FunctionName                  : MylambdaFunction123
Handler                       : lambda_function.launch_instance
KMSKeyArn                     :
LastModified                  : 2019-12-25T09:52:59.872+0000
LastUpdateStatus              : Successful
LastUpdateStatusReason        :
LastUpdateStatusReasonCode    :
Layers                        : {}
MasterArn                     :
MemorySize                    : 128
RevisionId                    : 5d7de38b-87f2-4260-8f8a-e87280e10c33
Role                          : arn:aws:iam::123456789012:role/service-role/lambda
Runtime                       : python3.8
State                         : Active
StateReason                   :
StateReasonCode               :
Timeout                       : 600
TracingConfig                 : Amazon.Lambda.Model.TracingConfigResponse
Version                       : 4
VpcConfig                     : Amazon.Lambda.Model.VpcConfigDetail
```

- API-Details finden Sie GetFunctionConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMFunctionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMFunctionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Lambda-Funktionen mit sortierter Codegröße angezeigt

```
Get-LMFunctionList | Sort-Object -Property CodeSize | Select-Object FunctionName,
 RunTime, Timeout, CodeSize
```

Ausgabe:

```
FunctionName                                                    Runtime   Timeout
 CodeSize
------------                                                    -------   -------
 --------
```

```
test                                                        python2.7       3
 243
MylambdaFunction123                                         python3.8     600
 659
myfuncpython1                                               python3.8     303
 675
```

- API-Details finden Sie ListFunctions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Get-LMPolicy

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMPolicy`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie der Lambda-Funktion angezeigt

```
Get-LMPolicy -FunctionName test -Select Policy
```

Ausgabe:

```
{"Version":"2012-10-17",       "Id":"default","Statement":
[{"Sid":"xxxx","Effect":"Allow","Principal":
{"Service":"sns.amazonaws.com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:
east-1:123456789102:function:test"}]}
```

- API-Details finden Sie GetPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Get-LMProvisionedConcurrencyConfig

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMProvisionedConcurrencyConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für den angegebenen Alias der Lambda-Funktion abgerufen.

```
C:\>Get-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
Qualifier "NewAlias1"
```

Ausgabe:

```
AllocatedProvisionedConcurrentExecutions : 0
AvailableProvisionedConcurrentExecutions : 0
LastModified                             : 2020-01-15T03:21:26+0000
RequestedProvisionedConcurrentExecutions : 70
Status                                   : IN_PROGRESS
StatusReason                             :
```

- API-Details finden Sie GetProvisionedConcurrencyConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMProvisionedConcurrencyConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMProvisionedConcurrencyConfigList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der bereitgestellten Gleichzeitigkeitskonfigurationen für eine Lambda-Funktion abgerufen.

```
Get-LMProvisionedConcurrencyConfigList -FunctionName "MylambdaFunction123"
```

- API-Details finden Sie ListProvisionedConcurrencyConfigs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMResourceTag`

Tools für PowerShell V5

Beispiel 1: Ruft die Tags und ihre Werte ab, die derzeit für die angegebene Funktion festgelegt sind.

```
Get-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction"
```

Ausgabe:

```
Key        Value
---        -----
California Sacramento
Oregon     Salem
Washington Olympia
```

- API-Details finden Sie ListTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-LMVersionsByFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Get-LMVersionsByFunction`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt die Liste der versionsspezifischen Konfigurationen für jede Version der Lambda-Funktion zurück.

```
Get-LMVersionsByFunction -FunctionName "MylambdaFunction123"
```

Ausgabe:

```
FunctionName        Runtime   MemorySize Timeout CodeSize LastModified
   RoleName
------------        -------   ---------- ------- -------- ------------
   --------
MylambdaFunction123 python3.8        128     600      659
 2020-01-10T03:20:56.390+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:19:02.238+0000 lambda
MylambdaFunction123 python3.8        128       5     1426
 2019-12-25T09:39:36.779+0000 lambda
MylambdaFunction123 python3.8        128     600     1426
 2019-12-25T09:52:59.872+0000 lambda
```

- API-Details finden Sie ListVersionsByFunction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `New-LMAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neuer Lambda-Alias für die angegebene Version und Routing-Konfiguration erstellt, um den Prozentsatz der Aufrufe für dieses System anzugeben.

```
New-LMAlias -FunctionName "MylambdaFunction123" -
RoutingConfig_AdditionalVersionWeight @{Name="1";Value="0.6} -Description "Alias for
 version 4" -FunctionVersion 4 -Name "PowershellAlias"
```

- API-Details finden Sie CreateAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Publish-LMFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-LMFunction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue C#-Funktion (dotnetcore1.0 Runtime) mit dem Namen AWS Lambda erstellt, die die kompilierten Binärdateien für die Funktion aus einer ZIP-Datei MyFunction im lokalen Dateisystem bereitstellt (relative oder absolute Pfade können verwendet werden). C#-Lambda-Funktionen spezifizieren den Handler für die Funktion mit der Bezeichnung AssemblyName:::Namespace.ClassName:. MethodName Sie sollten die Teile Assembly-Name (ohne .dll-Suffix), Namespace, Klassenname und Methodenname der Handler-Spezifikation entsprechend ersetzen. Für die neue Funktion werden die Umgebungsvariablen 'envvar1' und 'envvar2' aus den bereitgestellten Werten eingerichtet.

```
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -ZipFilename .\MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

Ausgabe:

```
CodeSha256       : /NgBMd...gq71I=
CodeSize         : 214784
DeadLetterConfig :
```

```
Description      : My C# Lambda Function
Environment      : Amazon.Lambda.Model.EnvironmentResponse
FunctionArn      : arn:aws:lambda:us-west-2:123456789012:function:ToUpper
FunctionName     : MyFunction
Handler          : AssemblyName::Namespace.ClassName::MethodName
KMSKeyArn        :
LastModified     : 2016-12-29T23:50:14.207+0000
MemorySize       : 128
Role             : arn:aws:iam::123456789012:role/LambdaFullExecRole
Runtime          : dotnetcore1.0
Timeout          : 3
Version          : $LATEST
VpcConfig        :
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, mit dem Unterschied, dass die Funktions-Binärdateien zunächst in einen Amazon S3-Bucket hochgeladen werden (der sich in derselben Region befinden muss wie die vorgesehene Lambda-Funktion) und das resultierende S3-Objekt dann bei der Erstellung der Funktion referenziert wird.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key MyFunctionBinaries.zip -File .
\MyFunctionBinaries.zip
Publish-LMFunction -Description "My C# Lambda Function" `
        -FunctionName MyFunction `
        -BucketName amzn-s3-demo-bucket `
        -Key MyFunctionBinaries.zip `
        -Handler "AssemblyName::Namespace.ClassName::MethodName" `
        -Role "arn:aws:iam::123456789012:role/LambdaFullExecRole" `
        -Runtime dotnetcore1.0 `
        -Environment_Variable @{ "envvar1"="value";"envvar2"="value" }
```

- API-Details finden Sie CreateFunction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Publish-LMVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-LMVersion`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Version für den vorhandenen Snapshot von Lambda-Funktionscode erstellt

```
Publish-LMVersion -FunctionName "MylambdaFunction123" -Description "Publishing
 Existing Snapshot of function code as a  new version through Powershell"
```

- API-Details finden Sie PublishVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMAlias`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die im Befehl erwähnte Lambda-Funktion Alias gelöscht.

```
Remove-LMAlias -FunctionName "MylambdaFunction123" -Name "NewAlias"
```

- API-Details finden Sie DeleteAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMFunction`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMFunction`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine bestimmte Version einer Lambda-Funktion gelöscht

```
Remove-LMFunction -FunctionName "MylambdaFunction123" -Qualifier '3'
```

- API-Details finden Sie DeleteFunction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMFunctionConcurrency`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Funktionsgleichzeitigkeit der Lambda-Funktion aufgehoben.

```
Remove-LMFunctionConcurrency -FunctionName "MylambdaFunction123"
```

- API-Details finden Sie DeleteFunctionConcurrency in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMPermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Funktionsrichtlinie für die angegebene StatementId Lambda-Funktion entfernt.

```
$policy =  Get-LMPolicy -FunctionName "MylambdaFunction123" -Select Policy |
 ConvertFrom-Json| Select-Object -ExpandProperty Statement
Remove-LMPermission -FunctionName "MylambdaFunction123" -StatementId $policy[0].Sid
```

- API-Details finden Sie RemovePermission in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMProvisionedConcurrencyConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMProvisionedConcurrencyConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die bereitgestellte Gleichzeitigkeitskonfiguration für einen bestimmten Alias entfernt.

```
Remove-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -Qualifier
 "NewAlias1"
```

- API-Details finden Sie DeleteProvisionedConcurrencyConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-LMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-LMResourceTag`

Tools für PowerShell V5

Beispiel 1: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen.

```
Remove-LMResourceTag -Resource "arn:aws:lambda:us-
west-2:123456789012:function:MyFunction" -TagKey "Washington","Oregon","California"
```

Beispiel 2: Entfernt die angegebenen Tags aus einer Funktion. Das Cmdlet fordert Sie zur Bestätigung auf, bevor der Vorgang fortgesetzt wird, sofern der Schalter -Force nicht angegeben ist. Sobald der Dienst pro bereitgestelltem Tag aufgerufen wurde.

```
"Washington","Oregon","California" | Remove-LMResourceTag -Resource
 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
```

- API-Details finden Sie UntagResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-LMAlias`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMAlias`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktualisiert die Konfiguration einer vorhandenen Lambda-Funktion-Alias. Der RoutingConfiguration Wert wird aktualisiert, sodass 60% (0,6) des Datenverkehrs auf Version 1 verschoben werden

```
Update-LMAlias -FunctionName "MylambdaFunction123" -Description " Alias for version
 2" -FunctionVersion 2 -Name "newlabel1" -RoutingConfig_AdditionalVersionWeight
 @{Name="1";Value="0.6}
```

- API-Details finden Sie UpdateAlias in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-LMFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMFunctionCode`

Tools für PowerShell V5

Beispiel 1: Aktualisiert die Funktion mit dem Namen 'MyFunction' mit neuem Inhalt, der in der angegebenen Zip-Datei enthalten ist. Für eine C#.NET-Core-Lambda-Funktion sollte die ZIP-Datei die kompilierte Assembly enthalten.

```
Update-LMFunctionCode -FunctionName MyFunction -ZipFilename .\UpdatedCode.zip
```

Beispiel 2: Dieses Beispiel ähnelt dem vorherigen, verwendet aber ein Amazon S3-Objekt, das den aktualisierten Code enthält, um die Funktion zu aktualisieren.

```
Update-LMFunctionCode -FunctionName MyFunction -BucketName amzn-s3-demo-bucket -Key
 UpdatedCode.zip
```

- API-Details finden Sie UpdateFunctionCode in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-LMFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Update-LMFunctionConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel aktualisiert die vorhandene Lambda-Funktionskonfiguration

```
Update-LMFunctionConfiguration -FunctionName "MylambdaFunction123" -Handler
 "lambda_function.launch_instance" -Timeout 600 -Environment_Variable
 @{ "envvar1"="value";"envvar2"="value" } -Role arn:aws:iam::123456789101:role/
service-role/lambda -DeadLetterConfig_TargetArn arn:aws:sns:us-east-1:
 123456789101:MyfirstTopic
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-LMFunctionConcurrency`

Das folgende Codebeispiel zeigt die Verwendung. `Write-LMFunctionConcurrency`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Gleichzeitigkeitseinstellungen für die Funktion als Ganzes angewendet.

```
Write-LMFunctionConcurrency -FunctionName "MylambdaFunction123" -
ReservedConcurrentExecution 100
```

- API-Details finden Sie PutFunctionConcurrency in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-LMProvisionedConcurrencyConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Write-LMProvisionedConcurrencyConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird dem Alias einer Funktion eine bereitgestellte Gleichzeitigkeitskonfiguration hinzugefügt

```
Write-LMProvisionedConcurrencyConfig -FunctionName "MylambdaFunction123" -
ProvisionedConcurrentExecution 20 -Qualifier "NewAlias1"
```

- API-Details finden Sie PutProvisionedConcurrencyConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon ML-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon ML verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-MLBatchPrediction`

Das folgende Codebeispiel zeigt die Verwendung`Get-MLBatchPrediction`.

Tools für PowerShell V5

Beispiel 1: Gibt die detaillierten Metadaten für eine Batch-Vorhersage mit der id-ID zurück.

```
Get-MLBatchPrediction -BatchPredictionId ID
```

- API-Details finden Sie GetBatchPrediction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLBatchPredictionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLBatchPredictionList`

Tools für PowerShell V5

Beispiel 1: Gibt eine Liste aller Datensätze BatchPredictions und ihrer zugehörigen Datensätze zurück, die dem in der Anfrage angegebenen Suchkriterium entsprechen.

```
Get-MLBatchPredictionList
```

Beispiel 2: Gibt eine Liste aller BatchPredictions mit dem Status COMPLETED zurück.

```
Get-MLBatchPredictionList -FilterVariable Status -EQ COMPLETED
```

- API-Details finden Sie DescribeBatchPredictions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLDataSource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLDataSource`

Tools für PowerShell V5

Beispiel 1: Gibt die Metadaten-, Status- und Datendateiinformationen für a DataSource mit der ID-ID zurück

```
Get-MLDataSource -DataSourceId ID
```

- API-Details finden Sie GetDataSource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLDataSourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLDataSourceList`

Tools für PowerShell V5

Beispiel 1: Gibt eine Liste aller Datensätze DataSources und ihrer zugehörigen Datensätze zurück.

```
Get-MLDataSourceList
```

Beispiel 2: Gibt eine Liste aller DataSources mit dem Status COMPLETED zurück.

```
Get-MLDataDourceList -FilterVariable Status -EQ COMPLETED
```

- API-Details finden Sie DescribeDataSources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLEvaluation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLEvaluation`

Tools für PowerShell V5

Beispiel 1: Gibt Metadaten und Status für eine Bewertung mit der id-ID zurück.

```
Get-MLEvaluation -EvaluationId ID
```

- API-Details finden Sie GetEvaluation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLEvaluationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLEvaluationList`

Tools für PowerShell V5

Beispiel 1: Gibt eine Liste aller Evaluierungsressourcen zurück

```
Get-MLEvaluationList
```

Beispiel 2: Gibt eine Liste aller Evaluierungen mit dem Status COMPLETED zurück.

```
Get-MLEvaluationList -FilterVariable Status -EQ COMPLETED
```

- API-Details finden Sie DescribeEvaluations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-MLModel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MLModel`

Tools für PowerShell V5

Beispiel 1: Gibt die Detail-Metadaten, den Status, das Schema und die Datendateiinformationen für ein MLModel mit der id-ID zurück.

```
Get-MLModel -ModelId ID
```

- Weitere API-Informationen finden Sie unter GetMLModel in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Get-MLModelList`

Das folgende Codebeispiel zeigt die Verwendung`Get-MLModelList`.

Tools für PowerShell V5

Beispiel 1: Gibt eine Liste aller Modelle und der zugehörigen Datensätze zurück.

```
Get-MLModelList
```

Beispiel 2: Gibt eine Liste aller Modelle mit dem Status COMPLETED zurück.

```
Get-MLModelList -FilterVariable Status -EQ COMPLETED
```

- Weitere API-Informationen finden Sie unter DescribeMLModels in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Get-MLPrediction`

Das folgende Codebeispiel zeigt die Verwendung`Get-MLPrediction`.

Tools für PowerShell V5

Beispiel 1: Sendet einen Datensatz an die URL des Echtzeit-Prognoseendpunkts für das Modell mit der id-ID.

```
Get-MLPrediction -ModelId ID -PredictEndpoint URL -Record @{"A" = "B"; "C" = "D";}
```

- Weitere API-Informationen finden Sie unter Predict in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`New-MLBatchPrediction`

Das folgende Codebeispiel zeigt die Verwendung`New-MLBatchPrediction`.

Tools für PowerShell V5

Beispiel 1: Erstellen Sie eine neue Batch-Vorhersageanforderung für das Modell mit der id-ID und speichern Sie die Ausgabe am angegebenen S3-Speicherort.

```
New-MLBatchPrediction -ModelId ID -Name NAME -OutputURI s3://...
```

- API-Details finden Sie CreateBatchPrediction in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-MLDataSourceFromS3`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLDataSourceFromS3`

Tools für PowerShell V5

Beispiel 1: Erstellen Sie eine Datenquelle mit Daten für einen S3-Speicherort mit dem Namen NAME und dem Schema SCHEMA.

```
New-MLDataSourceFromS3 -Name NAME -ComputeStatistics $true -DataSpec_DataLocationS3
 "s3://BUCKET/KEY" -DataSchema SCHEMA
```

- API-Details finden Sie CreateDataSourceFromS3 in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-MLEvaluation`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLEvaluation`

Tools für PowerShell V5

Beispiel 1: Erstellen Sie eine Auswertung für eine bestimmte Datenquellen-ID und Modell-ID.

```
New-MLEvaluation -Name NAME -DataSourceId DSID -ModelId MID
```

- API-Details finden Sie CreateEvaluation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-MLModel`

Das folgende Codebeispiel zeigt die Verwendung. `New-MLModel`

Tools für PowerShell V5

Beispiel 1: Erstellen Sie ein neues Modell mit Trainingsdaten.

```
New-MLModel -Name NAME -ModelType BINARY -Parameter @{...} -TrainingDataSourceId ID
```

- Weitere API-Informationen finden Sie unter CreateMLModel in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`New-MLRealtimeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`New-MLRealtimeEndpoint`.

Tools für PowerShell V5

Beispiel 1: Erstellen Sie einen neuen Echtzeit-Vorhersage-Endpunkt für die angegebene Modell-ID.

```
New-MLRealtimeEndpoint -ModelId ID
```

- API-Details finden Sie CreateRealtimeEndpoint in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Macie-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Macie Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-MAC2FindingList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-MAC2FindingList`

Tools für PowerShell V5

Beispiel 1: Gibt eine Liste von FindingIds vier Ergebnissen zurück, die eine Erkennung vertraulicher Daten vom Typ „CREDIT_CARD_NUMBER“ oder „US_SOCIAL_SECURITY_NUMBER“ enthalten

```
$criterionAddProperties = New-Object
 Amazon.Macie2.Model.CriterionAdditionalProperties

$criterionAddProperties.Eq = @(
"CREDIT_CARD_NUMBER"
"US_SOCIAL_SECURITY_NUMBER"
)
```

```
$FindingCriterion = @{
'classificationDetails.result.sensitiveData.detections.type' =
 [Amazon.Macie2.Model.CriterionAdditionalProperties]$criterionAddProperties
}

Get-MAC2FindingList -FindingCriteria_Criterion $FindingCriterion -MaxResult 5
```

- API-Details finden Sie in der Cmdlet-Referenz (V5). ListFindings AWS -Tools für PowerShell

AWS-Preisliste Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS-Preisliste

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-PLSAttributeValue`

Das folgende Codebeispiel zeigt die Verwendung`Get-PLSAttributeValue`.

Tools für PowerShell V5

Beispiel 1: Gibt die Werte für das Attribut „volumeType” für Amazon EC2 in der Region us-east-1 zurück.

```
Get-PLSAttributeValue -ServiceCode AmazonEC2 -AttributeName "volumeType" -region us-
east-1
```

Ausgabe:

```
Value
-----
Cold HDD
General Purpose
Magnetic
Provisioned IOPS
Throughput Optimized HDD
```

- API-Details finden Sie GetAttributeValues in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-PLSProduct`

Das folgende Codebeispiel zeigt die Verwendung. `Get-PLSProduct`

Tools für PowerShell V5

Beispiel 1: Gibt Details zu allen Produkten für Amazon EC2 zurück.

```
Get-PLSProduct -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
{"product":{"productFamily":"Compute Instance","attributes":
{"enhancedNetworkingSupported":"Yes","memory":"30.5
 GiB","dedicatedEbsThroughput":"800 Mbps","vcpu":"4","locationType":"AWS
 Region","storage":"EBS only","instanceFamily":"Memory
 optimized","operatingSystem":"SUSE","physicalProcessor":"Intel Xeon E5-2686 v4
 (Broadwell)","clockSpeed":"2.3 GHz","ecu":"Variable","networkPerformance":"Up
 to 10 Gigabit","servicename":"Amazon Elastic Compute
 Cloud","instanceType":"r4.xlarge","tenancy":"Shared","usagetype":"USW2-
BoxUsage:r4.xlarge","normalizationSizeFactor":"8","processorFeatures":"Intel AVX,
 Intel AVX2, Intel Turbo","servicecode":"AmazonEC2","licenseModel":"No License
 required","currentGeneration":"Yes","preInstalledSw":"NA","location":"US West
 (Oregon)","processorArchitecture":"64-bit","operation":"RunInstances:000g"},...
```

Beispiel 2: Gibt Daten für Amazon EC2 in der Region us-east-1 zurück, gefiltert nach den Datenträgertypen „Allgemeiner Zweck“. SSD-backed

```
Get-PLSProduct -ServiceCode AmazonEC2 -Filter
 @{Type="TERM_MATCH";Field="volumeType";Value="General
```

```
 Purpose"},@{Type="TERM_MATCH";Field="storageMedia";Value="SSD-backed"} -Region us-
east-1
```

Ausgabe:

```
{"product":{"productFamily":"Storage","attributes":{"storageMedia":"SSD-
backed","maxThroughputvolume":"160 MB/sec","volumeType":"General
 Purpose","maxIopsvolume":"10000",...
```

- API-Details finden Sie GetProducts in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-PLSService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-PLSService`

Tools für PowerShell V5

Beispiel 1: Gibt die Metadaten für alle verfügbaren Servicecodes in der Region us-east-1 zurück.

```
Get-PLSService -Region us-east-1
```

Ausgabe:

```
AttributeNames                                                    ServiceCode
--------------                                                    -----------
{productFamily, servicecode, groupDescription, termType...}       AWSBudgets
{productFamily, servicecode, termType, usagetype...}              AWSCloudTrail
{productFamily, servicecode, termType, usagetype...}              AWSCodeCommit
{productFamily, servicecode, termType, usagetype...}              AWSCodeDeploy
{productFamily, servicecode, termType, usagetype...}              AWSCodePipeline
{productFamily, servicecode, termType, usagetype...}              AWSConfig
...
```

Beispiel 2: Gibt die Metadaten für den Amazon-EC2-Service in der Region us-east-1 zurück.

```
Get-PLSService -ServiceCode AmazonEC2 -Region us-east-1
```

Ausgabe:

```
AttributeNames                                                    ServiceCode
```

```
--------------                                                           -----------
{volumeType, maxIopsvolume, instanceCapacity10xlarge, locationType...} AmazonEC2
```

- API-Details finden Sie DescribeServices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Ressourcengruppen, die Tools für V5 verwenden PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Ressourcengruppen verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-RGResourceTag`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Tag-Schlüssel „Instances“ mit dem Wert „workboxes“ zur angegebenen Ressourcengruppe arn hinzugefügt.

```
Add-RGResourceTag -Tag @{Instances="workboxes"} -Arn arn:aws:resource-groups:eu-
west-1:123456789012:group/workboxes
```

Ausgabe:

```
Arn                                                          Tags
---                                                          ----
```

```
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {[Instances,
 workboxes]}
```

- Weitere API-Informationen finden Sie unter Tag in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Find-RGResource`

Das folgende Codebeispiel zeigt die Verwendung`Find-RGResource`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein ResourceQuery For-Inance-Ressourcentyp mit Tag-Filtern erstellt und Ressourcen gefunden.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = ConvertTo-Json -Compress -Depth 4 -InputObject @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
    Key = 'auto'
    Values = @('no')
  })
 }

Find-RGResource -ResourceQuery $query | Select-Object -ExpandProperty
 ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                    ResourceType
-----------                                                    ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123445b6cb7bd67b AWS::EC2::Instance
```

- API-Details finden Sie SearchResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Ressourcengruppe anhand des Gruppennamens abgerufen.

```
Get-RGGroup -GroupName auto-no
```

Ausgabe:

```
Description GroupArn                                                    Name
----------- --------                                                    ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no auto-no
```

- API-Details finden Sie GetGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGGroupList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden bereits erstellte Ressourcengruppen aufgelistet.

```
Get-RGGroupList
```

Ausgabe:

```
GroupArn                                                                GroupName
--------                                                                ---------
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-no            auto-no
arn:aws:resource-groups:eu-west-1:123456789012:group/auto-yes           auto-yes
arn:aws:resource-groups:eu-west-1:123456789012:group/build600           build600
```

- API-Details finden Sie ListGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGGroupQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupQuery`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Ressourcenabfrage für die angegebene Ressourcengruppe abgerufen

```
Get-RGGroupQuery -GroupName auto-no | Select-Object -ExpandProperty ResourceQuery
```

Ausgabe:

```
Query
         Type
-----
         ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"auto","Values":
["no"]}]} TAG_FILTERS_1_0
```

- API-Details finden Sie GetGroupQuery in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGGroupResourceList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGGroupResourceList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Gruppenressourcen auf der Grundlage einer Filterung nach Ressourcentyp aufgelistet

```
Get-RGGroupResourceList -Filter @{Name="resource-type";Values="AWS::EC2::Instance"}
 -GroupName auto-yes | Select-Object -ExpandProperty ResourceIdentifiers
```

Ausgabe:

```
ResourceArn                                                    ResourceType
-----------                                                    ------------
arn:aws:ec2:eu-west-1:123456789012:instance/i-0123bc45b567890e1 AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0a1caf2345f67d8dc AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-012e3cb4df567e8aa AWS::EC2::Instance
arn:aws:ec2:eu-west-1:123456789012:instance/i-0fd12dd3456789012 AWS::EC2::Instance
```

- API-Details finden Sie ListGroupResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Tags für die angegebene Ressourcengruppe arn aufgelistet

```
Get-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes
```

Ausgabe:

```
Key       Value
---       -----
Instances workboxes
```

- API-Details finden Sie GetTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `New-RGGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue tagbasierte AWS Ressourcengruppe mit dem Namen Resource Groups erstellt. TestPowerShellGroup Die Gruppe umfasst Amazon EC2-Instances in der aktuellen Region, die mit dem Tag-Schlüssel „Name“ und dem Tag-Wert „test2“ versehen sind. Der Befehl gibt die Abfrage und den Typ der Gruppe sowie die Ergebnisse der Operation zurück.

```
$ResourceQuery = New-Object -TypeName Amazon.ResourceGroups.Model.ResourceQuery
$ResourceQuery.Type = "TAG_FILTERS_1_0"
$ResourceQuery.Query = '{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":
[{"Key":"Name","Values":["test2"]}]}'
$ResourceQuery
```

```
New-RGGroup -Name TestPowerShellGroup -ResourceQuery $ResourceQuery -Description
 "Test resource group."
```

Ausgabe:

```
Query
            Type
-----
            ----
{"ResourceTypeFilters":["AWS::EC2::Instance"],"TagFilters":[{"Key":"Name","Values":
["test2"]}]} TAG_FILTERS_1_0

LoggedAt         : 11/20/2018 2:40:59 PM
Group            : Amazon.ResourceGroups.Model.Group
ResourceQuery    : Amazon.ResourceGroups.Model.ResourceQuery
Tags             : {}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 338
HttpStatusCode   : OK
```

- API-Details finden Sie CreateGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die benannte Ressourcengruppe entfernt

```
Remove-RGGroup -GroupName non-tag-cfn-elbv2
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGGroup (DeleteGroup)" on target "non-tag-cfn-
elbv2".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

```
Description GroupArn
 Name
----------- --------
 ----
            arn:aws:resource-groups:eu-west-1:123456789012:group/non-tag-cfn-elbv2
 non-tag-cfn-elbv2
```

- API-Details finden Sie DeleteGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-RGResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das erwähnte Tag aus der Ressourcengruppe entfernt

```
Remove-RGResourceTag -Arn arn:aws:resource-groups:eu-west-1:123456789012:group/
workboxes -Key Instances
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-RGResourceTag (Untag)" on target "arn:aws:resource-
groups:eu-west-1:933303704102:group/workboxes".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y

Arn                                                          Keys
---                                                          ----
arn:aws:resource-groups:eu-west-1:123456789012:group/workboxes {Instances}
```

- Weitere API-Informationen finden Sie unter Untag in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Update-RGGroup`

Das folgende Codebeispiel zeigt die Verwendung`Update-RGGroup`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Beschreibung der Gruppe aktualisiert

```
Update-RGGroup -GroupName auto-yes -Description "Instances auto-remove"
```

Ausgabe:

```
Description             GroupArn
  Name
-----------             --------
  ----
Instances to be cleaned arn:aws:resource-groups:eu-west-1:123456789012:group/auto-
yes auto-yes
```

- API-Details finden Sie UpdateGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-RGGroupQuery`

Das folgende Codebeispiel zeigt die Verwendung. `Update-RGGroupQuery`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Abfrageobjekt erstellt und die Abfrage für die Gruppe aktualisiert.

```
$query = [Amazon.ResourceGroups.Model.ResourceQuery]::new()
$query.Type = [Amazon.ResourceGroups.QueryType]::TAG_FILTERS_1_0
$query.Query = @{
  ResourceTypeFilters = @('AWS::EC2::Instance')
  TagFilters = @(@{
  Key='Environment'
  Values='Build600.11'
  })
} | ConvertTo-Json -Compress -Depth 4

Update-RGGroupQuery -GroupName build600 -ResourceQuery $query
```

Ausgabe:

```
GroupName ResourceQuery
```

```
--------- -------------
build600  Amazon.ResourceGroups.Model.ResourceQuery
```

- API-Details finden Sie UpdateGroupQuery in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

API-Beispiele für das Taggen von Ressourcengruppen mithilfe von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS -Tools für PowerShell V5 mit der Resource Groups Tagging API Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-RGTResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Add-RGTResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tag-Schlüssel „stage“ und „version“ mit den Werten „beta“ und „preprod_test“ zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es wird ein einziger Aufruf an den Service gesendet, um die Tags anzuwenden.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Add-RGTResourceTag -ResourceARNList $arn1,$arn2 -Tag @{ "stage"="beta";
 "version"="preprod_test" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags und Werte zu einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle hinzugefügt. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

$arn1,$arn2 | Add-RGTResourceTag -Tag @{ "stage"="beta"; "version"="preprod_test" }
```

- API-Details finden Sie TagResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGTResource`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGTResource`

Tools für PowerShell V5

Beispiel 1: Gibt alle markierten Ressourcen in einer Region und die mit der Ressource verbundenen Tag-Schlüssel zurück. Wenn dem Cmdlet kein Parameter -Region übergeben wird, versucht es, die Region aus den Metadaten der Shell oder der EC2-Instance abzuleiten.

```
Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                           {stage,
 version, othertag}
```

Beispiel 2: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Die Zeichenfolge für jeden Servicenamen und Ressourcentyp entspricht der Zeichenfolge, die im Amazon-Ressourcennamen (ARN) einer Ressource eingebettet ist.

```
Get-RGTResource -ResourceType "s3"
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Beispiel 3: Gibt alle markierten Ressourcen des angegebenen Typs in einer Region zurück. Beachten Sie, dass beim Übergeben der Ressourcentypen an das Cmdlet für jeden angegebenen Ressourcentyp ein Aufruf des Services erfolgt.

```
"dynamodb","s3" | Get-RGTResource
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable           {stage, version}
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Beispiel 4: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" }
```

Ausgabe:

```
ResourceARN                                                     Tags
-----------                                                     ----
arn:aws:s3:::amzn-s3-demo-bucket                                          {stage,
 version, othertag}
```

Beispiel 5: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter und Ressourcentyp entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage" } -ResourceType "dynamodb"
```

Ausgabe:

```
ResourceARN                                                     Tags
```

```
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

Beispiel 6: Gibt alle markierten Ressourcen zurück, die dem angegebenen Filter entsprechen.

```
Get-RGTResource -TagFilter @{ Key="stage"; Values=@("beta","gamma") }
```

Ausgabe:

```
ResourceARN                                                    Tags
-----------                                                    ----
arn:aws:dynamodb:us-west-2:123456789012:table/mytable          {stage, version}
```

- API-Details finden Sie GetResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGTTagKey`

Das folgende Codebeispiel zeigt die Verwendung`Get-RGTTagKey`.

Tools für PowerShell V5

Beispiel 1: Gibt alle Tag-Schlüssel in der angegebenen Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten. Beachten Sie, dass die Tag-Schlüssel nicht in einer bestimmten Reihenfolge zurückgegeben werden.

```
Get-RGTTagKey -region us-west-2
```

Ausgabe:

```
version
stage
```

- API-Details finden Sie GetTagKeys in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-RGTTagValue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-RGTTagValue`

Tools für PowerShell V5

Beispiel 1: Gibt den Wert für das angegebene Tag in einer Region zurück. Wenn der Parameter -Region nicht angegeben ist, versucht das Cmdlet, die Region aus der Standard-Shell-Region oder den Metadaten der EC2-Instance abzuleiten.

```
Get-RGTTagValue -Key "stage" -Region us-west-2
```

Ausgabe:

```
beta
```

- API-Details finden Sie GetTagValues in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-RGTResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-RGTResourceTag`

Tools für PowerShell V5

Beispiel 1: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es wird ein einziger Aufruf an den Dienst gesendet, um die Tags zu entfernen. Bevor die Tags entfernt werden, fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"

Remove-RGTResourceTag -ResourceARNList $arn1,$arn2 -TagKey "stage","version"
```

Beispiel 2: Entfernt die Tag-Schlüssel „stage“ und „version“ sowie die zugehörigen Werte aus einem Amazon-S3-Bucket und einer Amazon-DynamoDB-Tabelle. Es werden zwei Aufrufe an den Service gesendet, einer für jede Ressource-ARN, die in das Cmdlet übergeben wird. Vor jedem Aufruf fordert das Cmdlet zur Bestätigung auf. Um die Bestätigung zu umgehen, fügen Sie den Parameter -Force hinzu.

```
$arn1 = "arn:aws:s3:::amzn-s3-demo-bucket"
$arn2 = "arn:aws:dynamodb:us-west-2:123456789012:table/mytable"
```

```
$arn1,$arn2 | Remove-RGTResourceTag -TagKey "stage","version"
```

- API-Details finden Sie UntagResources in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Route-53-Beispiele mithilfe von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie die AWS -Tools für PowerShell V5 mit Route 53 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Edit-R53ResourceRecordSet`

Das folgende Codebeispiel zeigt die Verwendung`Edit-R53ResourceRecordSet`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein A-Eintrag für www.example.com erstellt und der A-Eintrag für test.example.com von 192.0.2.3 in 192.0.2.1 geändert. Beachten Sie, dass Werte für TXT-type Änderungsdatensätze in doppelten Anführungszeichen stehen müssen. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "TXT"
$change1.ResourceRecordSet.TTL = 600
```

```
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="item 1 item 2 item 3"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "DELETE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "test.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.3"})

$change3 = New-Object Amazon.Route53.Model.Change
$change3.Action = "CREATE"
$change3.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change3.ResourceRecordSet.Name = "test.example.com"
$change3.ResourceRecordSet.Type = "A"
$change3.ResourceRecordSet.TTL = 600
$change3.ResourceRecordSet.ResourceRecords = @()
$change3.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.1"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
 ChangeBatch_Comment="This change batch creates a TXT record for www.example.com.
 and changes the A record for test.example.com. from 192.0.2.3 to 192.0.2.1."
 ChangeBatch_Change=$change1,$change2,$change3
}

Edit-R53ResourceRecordSet @params
```

Beispiel 2: Dieses Beispiel zeigt, wie Alias-Ressourceneinträge erstellt werden. „Z222222222“ ist die ID der von Amazon Route 53 gehosteten Zone, in der Sie den Alias-Ressourceneintragssatz erstellen. „example.com“ ist der Zonenapex, für den Sie einen Alias erstellen möchten, und „www.example.com“ ist eine Subdomain, für die Sie ebenfalls einen Alias erstellen möchten. „Z1111111111111“ ist ein Beispiel für eine gehostete Zonen-ID für den Load Balancer und „example-load-balancer-1111111111.us-east-1.elb.amazonaws.com“ ist ein Beispiel für einen Load-Balancer-Domainnamen, mit dem Amazon Route 53 auf Anfragen für example.com und www.example.com antwortet. Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
```

```
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z222222222"
 ChangeBatch_Comment="This change batch creates two alias resource record sets, one
 for the zone apex, example.com, and one for www.example.com, that both point to
 example-load-balancer-1111111111.us-east-1.elb.amazonaws.com."
 ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 3: In diesem Beispiel werden zwei A-Datensätze für www.example.com erstellt. One-fourth In diesem Fall (1/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden Werten für den ersten Ressourcendatensatz (192.0.2.9 und 192.0.2.10). Three-fourths In allen Fällen (3/ (1+3)) beantwortet Amazon Route 53 Anfragen für www.example.com mit den beiden Werten für den zweiten Ressourcendatensatz (192.0.2.11 und 192.0.2.12). Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
```

```
$change1.ResourceRecordSet.Name = "www.example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Rack 2, Positions 4 and 5"
$change1.ResourceRecordSet.Weight = 1
$change1.ResourceRecordSet.TTL = 600
$change1.ResourceRecordSet.ResourceRecords = @()
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.9"})
$change1.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.10"})

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "www.example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Rack 5, Positions 1 and 2"
$change2.ResourceRecordSet.Weight = 3
$change2.ResourceRecordSet.TTL = 600
$change2.ResourceRecordSet.ResourceRecords = @()
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.11"})
$change2.ResourceRecordSet.ResourceRecords.Add(@{Value="192.0.2.12"})

$params = @{
    HostedZoneId="Z1PA6795UKMFR9"
    ChangeBatch_Comment="This change creates two weighted resource record sets, each
 of which has two values."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 4: Dieses Beispiel zeigt, wie gewichtete Alias-Ressourcendatensätze erstellt werden, vorausgesetzt, example.com ist die Domain, für die Sie gewichtete Alias-Ressourcendatensätze erstellen möchten. SetIdentifier unterscheidet die beiden gewichteten Alias-Ressourcendatensätze voneinander. Dieses Element ist erforderlich, da die Elemente „Name“ und „Type“ für beide Ressourceneintragsätze dieselben Werte haben. Z1111111111111 und Z3333333333333 sind Beispiele für IDs gehosteter Zonen für den ELB-Load-Balancer, die durch den Wert von DNSName angegeben werden. example-load-balancer-2222222222.us-east-1.elb.amazonaws.com und example-load-balancer-4444444444.us-east-1.elb.amazonaws.com sind Beispiele für Domains von Elastic Load Balancing, von denen aus Amazon Route 53 auf Abfragen für example.com antwortet.

Weitere Informationen finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "1"
$change1.ResourceRecordSet.Weight = 3
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-east-1.elb.amazonaws.com."
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "2"
$change2.ResourceRecordSet.Weight = 1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z3333333333333"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-4444444444.us-east-1.elb.amazonaws.com."
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $false

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two weighted alias resource
 record sets. Amazon Route 53 responds to queries for example.com with the first ELB
 domain 3/4ths of the times and the second one 1/4th of the time."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

Beispiel 5: In diesem Beispiel werden zwei Latenz-Alias-Ressourceneinträge erstellt, einer für einen ELB-Load Balancer in der Region US-West (Oregon) (us-west-2) und einer für einen Load Balancer in der Region Asien-Pazifik (Singapur) (ap-southeast-1). Weitere Informationen

finden Sie in der Dokumentation zu Amazon Route 53 Sie können das Get-R53Change Cmdlet verwenden, um abzufragen, wann die Änderungen abgeschlossen sind.

```
$change1 = New-Object Amazon.Route53.Model.Change
$change1.Action = "CREATE"
$change1.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change1.ResourceRecordSet.Name = "example.com"
$change1.ResourceRecordSet.Type = "A"
$change1.ResourceRecordSet.SetIdentifier = "Oregon load balancer 1"
$change1.ResourceRecordSet.Region = us-west-2
$change1.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change1.ResourceRecordSet.AliasTarget.HostedZoneId = "Z1111111111111"
$change1.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-2222222222.us-west-2.elb.amazonaws.com"
$change1.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$change2 = New-Object Amazon.Route53.Model.Change
$change2.Action = "CREATE"
$change2.ResourceRecordSet = New-Object Amazon.Route53.Model.ResourceRecordSet
$change2.ResourceRecordSet.Name = "example.com"
$change2.ResourceRecordSet.Type = "A"
$change2.ResourceRecordSet.SetIdentifier = "Singapore load balancer 1"
$change2.ResourceRecordSet.Region = ap-southeast-1
$change2.ResourceRecordSet.AliasTarget = New-Object Amazon.Route53.Model.AliasTarget
$change2.ResourceRecordSet.AliasTarget.HostedZoneId = "Z2222222222222"
$change2.ResourceRecordSet.AliasTarget.DNSName = "example-load-
balancer-1111111111.ap-southeast-1.elb.amazonaws.com"
$change2.ResourceRecordSet.AliasTarget.EvaluateTargetHealth = $true

$params = @{
    HostedZoneId="Z5555555555"
    ChangeBatch_Comment="This change batch creates two latency resource record
 sets, one for the US West (Oregon) region and one for the Asia Pacific (Singapore)
 region."
    ChangeBatch_Change=$change1,$change2
}

Edit-R53ResourceRecordSet @params
```

- API-Details finden Sie ChangeResourceRecordSets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53AccountLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53AccountLimit`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die maximale Anzahl von gehosteten Zonen zurückgegeben, die mit dem aktuellen Konto erstellt werden können.

```
Get-R53AccountLimit -Type MAX_HOSTED_ZONES_BY_OWNER
```

Ausgabe:

```
15
```

- API-Details finden Sie GetAccountLimit in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53CheckerIpRanges`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53CheckerIpRanges`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die CIDRs für die Route53-Zustandsprüfungen zurückgegeben

```
Get-R53CheckerIpRanges
```

Ausgabe:

```
15.177.2.0/23
15.177.6.0/23
15.177.10.0/23
15.177.14.0/23
15.177.18.0/23
15.177.22.0/23
15.177.26.0/23
15.177.30.0/23
15.177.34.0/23
15.177.38.0/23
```

```
15.177.42.0/23
15.177.46.0/23
15.177.50.0/23
15.177.54.0/23
15.177.58.0/23
15.177.62.0/23
54.183.255.128/26
54.228.16.0/26
54.232.40.64/26
54.241.32.64/26
54.243.31.192/26
54.244.52.192/26
54.245.168.0/26
54.248.220.0/26
54.250.253.192/26
54.251.31.128/26
54.252.79.128/26
54.252.254.192/26
54.255.254.192/26
107.23.255.0/26
176.34.159.192/26
177.71.207.128/26
```

- API-Details finden Sie GetCheckerIpRanges in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZone`

Tools für PowerShell V5

Beispiel 1: Gibt Details der gehosteten Zone mit der ID Z1D633PJN98FT9 zurück.

```
Get-R53HostedZone -Id Z1D633PJN98FT9
```

- API-Details finden Sie GetHostedZone in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53HostedZoneCount`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneCount`

Tools für PowerShell V5

Beispiel 1: Gibt die Gesamtzahl der öffentlichen und privaten gehosteten Zonen für die aktuelle AWS-Konto Zone zurück.

```
Get-R53HostedZoneCount
```

- API-Details finden Sie GetHostedZoneCount in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53HostedZoneLimit`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneLimit`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Limit für die maximale Anzahl von Datensätzen zurückgegeben, die in der angegebenen gehosteten Zone erstellt werden können.

```
Get-R53HostedZoneLimit -HostedZoneId Z3MEQ8T7HAAAAF -Type MAX_RRSETS_BY_ZONE
```

Ausgabe:

```
5
```

- API-Details finden Sie GetHostedZoneLimit in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53HostedZoneList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZoneList`

Tools für PowerShell V5

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen aus.

```
Get-R53HostedZoneList
```

Beispiel 2: Gibt alle gehosteten Zonen aus, die mit dem wiederverwendbaren Delegationssatz mit der ID NZ8X2CISAMPLE verknüpft sind.

```
Get-R53HostedZoneList -DelegationSetId NZ8X2CISAMPLE
```

- API-Details finden Sie ListHostedZones in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53HostedZonesByName`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53HostedZonesByName`

Tools für PowerShell V5

Beispiel 1: Gibt alle Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück.

```
Get-R53HostedZonesByName
```

Beispiel 2: Gibt Ihre öffentlichen und privaten gehosteten Zonen in ASCII-Reihenfolge nach Domainnamen zurück, beginnend mit dem angegebenen DNS-Namen.

```
Get-R53HostedZonesByName -DnsName example2.com
```

- API-Details finden Sie ListHostedZonesByName in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53QueryLoggingConfigList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53QueryLoggingConfigList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Konfigurationen für die DNS-Abfrageprotokollierung zurückgegeben, die mit dem aktuellen AWS-Konto verknüpft sind.

```
Get-R53QueryLoggingConfigList
```

Ausgabe:

```
Id                                   HostedZoneId   CloudWatchLogsLogGroupArn
--                                   ------------   -------------------------
```

```
59b0fa33-4fea-4471-a88c-926476aaa40d Z385PDS6EAAAZR arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example1.com:*
ee528e95-4e03-4fdc-9d28-9e24ddaaa063 Z94SJHBV1AAAAZ arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example2.com:*
e38dddda-ceb6-45c1-8cb7-f0ae56aaaa2b Z3MEQ8T7AAA1BF arn:aws:logs:us-
east-1:111111111112:log-group:/aws/route53/example3.com:*
```

- API-Details finden Sie ListQueryLoggingConfigs in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-R53ReusableDelegationSet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über den angegebenen Delegationssatz abgerufen, einschließlich der vier Nameserver, die dem Delegationssatz zugewiesen sind.

```
Get-R53ReusableDelegationSet -Id N23DS9X4AYEAAA
```

Ausgabe:

```
Id                            CallerReference NameServers
--                            --------------- -----------
/delegationset/N23DS9X4AYEAAA testcaller      {ns-545.awsdns-04.net,
 ns-1264.awsdns-30.org, ns-2004.awsdns-58.co.uk, ns-240.awsdns-30.com}
```

- API-Details finden Sie GetReusableDelegationSet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53HostedZone`

Tools für PowerShell V5

Beispiel 1: Erstellt eine neue gehostete Zone namens „example.com“, die mit einem wiederverwendbaren Delegationssatz verknüpft ist. Beachten Sie, dass Sie einen Wert für den CallerReference Parameter angeben müssen, sodass Anfragen, die bei Bedarf wiederholt werden

müssen, ohne dass das Risiko besteht, dass der Vorgang zweimal ausgeführt wird. Da die gehostete Zone in einer VPC erstellt wird, ist sie automatisch privat und Sie sollten den Parameter - HostedZoneConfig _ PrivateZone nicht festlegen.

```
$params = @{
    Name="example.com"
    CallerReference="myUniqueIdentifier"
    HostedZoneConfig_Comment="This is my first hosted zone"
    DelegationSetId="NZ8X2CISAMPLE"
    VPC_VPCId="vpc-1a2b3c4d"
    VPC_VPCRegion="us-east-1"
}

New-R53HostedZone @params
```

- API-Details finden Sie CreateHostedZone in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-R53QueryLoggingConfig`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53QueryLoggingConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine neue Route53-DNS-Abfrageprotokollierungskonfiguration für die angegebene gehostete Zone erstellt. Amazon Route53 veröffentlicht DNS-Abfrageprotokolle in der angegebenen Cloudwatch-Protokollgruppe.

```
New-R53QueryLoggingConfig -HostedZoneId Z3MEQ8T7HAAAAF -CloudWatchLogsLogGroupArn
 arn:aws:logs:us-east-1:111111111111:log-group:/aws/route53/example.com:*
```

Ausgabe:

```
QueryLoggingConfig                      Location
------------------                      --------
Amazon.Route53.Model.QueryLoggingConfig https://route53.amazonaws.com/2013-04-01/
queryloggingconfig/ee5aaa95-4e03-4fdc-9d28-9e24ddaaaaa3
```

- API-Details finden Sie CreateQueryLoggingConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `New-R53ReusableDelegationSet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein wiederverwendbarer Delegationssatz mit vier Nameservern erstellt, der von mehreren gehosteten Zonen wiederverwendet werden kann.

```
New-R53ReusableDelegationSet -CallerReference testcallerreference
```

Ausgabe:

```
DelegationSet                      Location
-------------                      --------
Amazon.Route53.Model.DelegationSet https://route53.amazonaws.com/2013-04-01/
delegationset/N23DS9XAAAAAXM
```

- API-Details finden Sie CreateReusableDelegationSet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-R53VPCWithHostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Register-R53VPCWithHostedZone`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPC mit der privaten gehosteten Zone verknüpft.

```
Register-R53VPCWithHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa -
VPC_VPCRegion us-east-1
```

Ausgabe:

```
Id                     Status  SubmittedAt         Comment
--                     ------  -----------         -------
/change/C3SCAAA633Z6DX PENDING 01/28/2020 19:32:02
```

- API-Details finden Sie AssociateVPCWithHostedZone in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-R53HostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53HostedZone`

Tools für PowerShell V5

Beispiel 1: Löscht die gehostete Zone mit der angegebenen ID. Sie werden zur Bestätigung aufgefordert, bevor der Befehl ausgeführt wird, es sei denn, Sie fügen den Schalterparameter -Force hinzu.

```
Remove-R53HostedZone -Id Z1PA6795UKMFR9
```

- API-Details finden Sie DeleteHostedZone in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-R53QueryLoggingConfig`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53QueryLoggingConfig`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Konfiguration für die Protokollierung von DNS-Abfragen entfernt.

```
Remove-R53QueryLoggingConfig -Id ee528e95-4e03-4fdc-9d28-9e24daaa20063
```

- API-Details finden Sie DeleteQueryLoggingConfig in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-R53ReusableDelegationSet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-R53ReusableDelegationSet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der angegebene wiederverwendbare Delegationssatz gelöscht.

```
Remove-R53ReusableDelegationSet -Id N23DS9X4AYAAAM
```

- API-Details finden Sie DeleteReusableDelegationSet in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-R53VPCFromHostedZone`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-R53VPCFromHostedZone`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene VPC von der privaten gehosteten Zone getrennt.

```
Unregister-R53VPCFromHostedZone -HostedZoneId Z3MEQ8T7HAAAAF -VPC_VPCId vpc-f1b9aaaa
 -VPC_VPCRegion us-east-1
```

Ausgabe:

```
Id                      Status  SubmittedAt         Comment
--                      ------  -----------         -------
/change/C2XFCAAAA9HKZG PENDING 01/28/2020 10:35:55
```

- API-Details finden Sie DisassociateVPCFromHostedZone in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-R53HostedZoneComment`

Das folgende Codebeispiel zeigt die Verwendung. `Update-R53HostedZoneComment`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl aktualisiert den Kommentar für die angegebene gehostete Zone.

```
Update-R53HostedZoneComment -Id Z385PDS6AAAAAR -Comment "This is my first hosted
 zone"
```

Ausgabe:

```
Id                      : /hostedzone/Z385PDS6AAAAAR
Name                    : example.com.
```

```
CallerReference        : C5B55555-7147-EF04-8341-69131E805C89
Config                 : Amazon.Route53.Model.HostedZoneConfig
ResourceRecordSetCount : 9
LinkedService          :
```

- API-Details finden Sie UpdateHostedZoneComment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon S3-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon S3 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Copy-S3Object`

Das folgende Codebeispiel zeigt die Verwendung`Copy-S3Object`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in denselben Bucket, jedoch mit einem neuen Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
 sample-copy.txt
```

Beispiel 2: Dieser Befehl kopiert das Objekt „sample.txt” aus dem Bucket „test-files” in den Bucket „backup-files” mit dem Schlüssel „sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey
 sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Beispiel 3: Dieser Befehl lädt das Objekt „sample.txt” aus dem Bucket „test-files” in eine lokale Datei mit dem Namen „local-sample.txt” herunter.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:
\downloads
```

Beispiel 5: Lädt alle Objekte, die mit dem angegebenen Schlüsselpräfix übereinstimmen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:
\downloads
```

Beispiel 6: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallelen Downloads in eine lokale Datei herunter, um die Leistung bei großen Dateien zu verbessern.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt -UseMultipartDownload
```

Beispiel 7: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Komponentengröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanfragen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-
large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Beispiel 8: Lädt alle Objekte, die dem Schlüsselpräfix entsprechen, mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Beispiel 9: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um maximalen Durchsatz in Netzwerken mit hoher Bandbreite zu erzielen.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Beispiel 10: Lädt alle Objekte, die dem Schlüsselpräfix entsprechen, mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter, wobei einzelne Dateifehler behoben werden, anstatt den Vorgang abzubrechen.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:
\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API-Details finden Sie CopyObject in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Bucket`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API-Details finden Sie ListBuckets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketACL`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketACL`

Tools für PowerShell V5

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Ausgabe:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API-Details finden Sie GetBucketAcl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketAccelerateConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAccelerateConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Übertragungsbeschleunigungseinstellungen für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
Enabled
```

- API-Details finden Sie GetBucketAccelerateConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketAnalyticsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAnalyticsConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen „testfilter” im angegebenen S3-Bucket zurück.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- API-Details finden Sie GetBucketAnalyticsConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketAnalyticsConfigurationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketAnalyticsConfigurationList`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die ersten 100 Analysekonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie ListBucketAnalyticsConfigurations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketEncryption`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketEncryption in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketInventoryConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketInventoryConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen „testinventory" für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testinventory'
```

- API-Details finden Sie GetBucketInventoryConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketInventoryConfigurationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketInventoryConfigurationList`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie ListBucketInventoryConfigurations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketLocation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketLocation`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket „amzn-s3-demo-bucket" zurück, sofern eine Beschränkung vorhanden ist.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
ap-south-1
```

- API-Details finden Sie GetBucketLocation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketLogging`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketLogging`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Protokollierungsstatus für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- API-Details finden Sie GetBucketLogging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketMetricsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketMetricsConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen „testfilter” für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testfilter'
```

- API-Details finden Sie GetBucketMetricsConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketNotification`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketNotification`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
 TopicConfigurations
```

Ausgabe:

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API-Details finden Sie GetBucketNotification in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketPolicy`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie aus.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketPolicyStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketPolicyStatus`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketPolicyStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketReplication`

Tools für PowerShell V5

Beispiel 1: Gibt die informationen der Replikationskonfiguration zurück, die für den Bucket mit dem Namen „amzn-s3-demo-bucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API-Details finden Sie GetBucketReplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketRequestPayment`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketRequestPayment`

Tools für PowerShell V5

Beispiel 1: Gibt die Zahlungsanforderungskonfiguration für den Bucket mit dem Namen „amzn-s3-demo-bucket“ zurück. Standardmäßig zahlt der Bucket-Eigentümer für die Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API-Details finden Sie GetBucketRequestPayment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketTagging`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die mit dem angegebenen Bucket verknüpft sind.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketTagging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketVersioning`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketVersioning in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3BucketWebsite`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetBucketWebsite in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3CORSConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3CORSConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt ein Objekt zurück, das alle CORS-Konfigurationsregeln enthält, die dem angegebenen S3-Bucket entsprechen.

```
Get-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 Configuration.Rules
```

Ausgabe:

```
AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example1.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {PUT, POST, DELETE}
AllowedOrigins : {http://www.example2.com}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {*}

AllowedMethods : {GET}
AllowedOrigins : {*}
Id             :
ExposeHeaders  : {}
MaxAgeSeconds  : 0
AllowedHeaders : {}
```

- Weitere API-Informationen finden Sie unter GetCORSConfiguration in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Get-S3HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung`Get-S3HeadBucket`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Ausgabe mit dem HTTP-Statuscode 200 OK für einen vorhandenen Bucket zurück, wenn der Benutzer die Berechtigung hat, darauf zuzugreifen. BucketArn Der Parameter wird nur für S3-Verzeichnis-Buckets unterstützt.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Ausgabe:

```
AccessPointAlias   : False
BucketArn          :
BucketLocationName :
BucketLocationType :
BucketRegion       : us-east-2
ResponseMetadata   : Amazon.Runtime.ResponseMetadata
ContentLength      : 0
HttpStatusCode     : OK
```

Beispiel 2: Dieser Befehl gibt einen Fehler mit dem HTTP-Statuscode NotFound für einen nicht vorhandenen Bucket aus.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code
 NotFound. No further error information was returned by the service.
```

Beispiel 3: Dieser Befehl löst einen Fehler mit dem HTTP-Statuscode „Forbidden“ für einen vorhandenen Bucket aus, auf den der Benutzer keine Zugriffsberechtigung hat.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
 Code Forbidden. No further error information was returned by the service.
```

- API-Details finden Sie HeadBucket in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3LifecycleConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Lebenszykluskonfiguration für den Bucket abgerufen.

```
Get-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket
```

Ausgabe:

```
Rules
-----
{Remove-in-150-days, Archive-to-Glacier-in-30-days}
```

- API-Details finden Sie GetLifecycleConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Object`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt „sample.txt“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API-Details finden Sie ListObjects in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3ObjectACL`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectACL`

Tools für PowerShell V5

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
 *).Owner
```

Ausgabe:

```
DisplayName Id
----------- --
testusername      9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API-Details finden Sie GetObjectAcl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3ObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectLockConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value
-----
Enabled
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3ObjectMetadata`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectMetadata`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Metadaten des Objekts mit dem Schlüssel 'ListTrusts.txt' im angegebenen S3-Bucket zurück.

```
Get-S3ObjectMetadata -BucketName 'amzn-s3-demo-bucket' -Key 'ListTrusts.txt'
```

Ausgabe:

```
Headers                                       : Amazon.S3.Model.HeadersCollection
Metadata                                      : Amazon.S3.Model.MetadataCollection
DeleteMarker                                  :
AcceptRanges                                  : bytes
ContentRange                                  :
Expiration                                    :
RestoreExpiration                             :
RestoreInProgress                             : False
LastModified                                  : 01/01/2020 08:02:05
ETag                                          : "d000011112a222e333e3bb4ee5d43d21"
MissingMeta                                   : 0
VersionId                                     : null
Expires                                       : 01/01/0001 00:00:00
WebsiteRedirectLocation                       :
ServerSideEncryptionMethod                    : AES256
ServerSideEncryptionCustomerMethod            :
ServerSideEncryptionKeyManagementServiceKeyId :
ReplicationStatus                             :
PartsCount                                    :
ObjectLockLegalHoldStatus                     :
ObjectLockMode                                :
```

```
ObjectLockRetainUntilDate                       : 01/01/0001 00:00:00
StorageClass                                    :
RequestCharged                                  :
```

- API-Details finden Sie GetObjectMetadata in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3ObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectRetention`

Tools für PowerShell V5

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt aufbewahrt wird.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API-Details finden Sie GetObjectRetention in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3ObjectTagSet`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3ObjectTagSet`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Tags zurückgegeben, die mit dem Objekt im angegebenen S3-Bucket verknüpft sind.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Key  Value
---  -----
test value
```

- API-Details finden Sie GetObjectTagging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3PreSignedURL`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3PreSignedURL`

Tools für PowerShell V5

Beispiel 1: Der Befehl gibt eine vorab signierte URL für einen angegebenen Schlüssel und ein Ablaufdatum zurück.

```
Get-S3PreSignedURL -BucketName 'amzn-s3-demo-bucket' -Key 'testkey' -Expires
 '2023-11-16'
```

Beispiel 2: Der Befehl gibt eine vorab signierte URL für einen Verzeichnis-Bucket mit angegebenem Schlüssel und ein Ablaufdatum zurück.

```
[Amazon.AWSConfigsS3]::UseSignatureVersion4 = $true
      Get-S3PreSignedURL -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -Key
 'testkey' -Expire '2023-11-17'
```

- API-Details finden Sie GetPreSignedURL in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3PublicAccessBlock`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3PublicAccessBlock`

Tools für PowerShell V5

Beispiel 1: Der Befehl gibt die Konfiguration der öffentlichen Zugriffssperre des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie GetPublicAccessBlock in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-S3Version`

Das folgende Codebeispiel zeigt die Verwendung. `Get-S3Version`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Metadaten zu allen Versionen von Objekten im angegebenen S3-Bucket zurück.

```
Get-S3Version -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
IsTruncated         : False
KeyMarker           :
VersionIdMarker     :
NextKeyMarker       :
NextVersionIdMarker :
Versions            : {EC2.txt, EC2MicrosoftWindowsGuide.txt, ListDirectories.json,
 ListTrusts.json}
Name                : amzn-s3-demo-bucket
Prefix              :
MaxKeys             : 1000
CommonPrefixes      : {}
Delimiter           :
```

- API-Details finden Sie ListVersions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `New-S3Bucket`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl erstellt einen neuen privaten Bucket mit dem Namen „sample-bucket“.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl erstellt einen neuen Bucket mit dem Namen „sample-bucket“ mit Lese- und Schreibberechtigungen.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadWrite
```

Beispiel 3: Dieser Befehl erstellt einen neuen Bucket mit dem Namen „sample-bucket“ mit Leseberechtigungen.

```
New-S3Bucket -BucketName amzn-s3-demo-bucket -PublicReadOnly
```

Beispiel 4: Dieser Befehl erstellt einen neuen Directory-Bucket mit dem Namen „amzn-s3-demo-bucket--use1-az5--x-s3" mit. PutBucketConfiguration

```
$bucketConfiguration = @{
      BucketInfo = @{
          DataRedundancy = 'SingleAvailabilityZone'
          Type = 'Directory'
      }
      Location = @{
          Name = 'usw2-az1'
          Type = 'AvailabilityZone'
      }
    }
New-S3Bucket -BucketName amzn-s3-demo-bucket--usw2-az1--x-s3 -BucketConfiguration
 $bucketConfiguration -Region us-west-2
```

- API-Details finden Sie PutBucket in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Read-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Read-S3Object`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft das Element "sample.txt" aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei „local-sample.txt” muss vor dem Aufruf dieses Befehls nicht vorhanden sein.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Beispiel 2: Dieser Befehl ruft das virtuelle Verzeichnis „DIR“ aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einem Ordner mit dem Namen "" am aktuellen Speicherort. Local-DIR Der Ordner "Local-DIR" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf „.json“ enden, aus Buckets, deren Name „config“ enthält, in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Beispiel 4: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallelen Downloads herunter, um die Leistung bei großen Dateien zu verbessern.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
 -UseMultipartDownload
```

Beispiel 5: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Komponentengröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanfragen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-
file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Beispiel 6: Lädt alle Objekte mit dem Schlüsselpräfix „data/“ mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data
 -UseMultipartDownload -DownloadFilesConcurrently
```

Beispiel 7: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um maximalen Durchsatz in Netzwerken mit hoher Bandbreite zu erzielen.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-
file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Beispiel 8: Lädt alle Objekte mit dem Schlüsselpräfix „data/“ mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter und setzt dabei einzelne Dateifehler fort, anstatt den Vorgang abzubrechen.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data
 -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API-Details finden Sie GetObject in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3Bucket`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket „test-files“ und löscht anschließend den Bucket. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass nicht leere Buckets nicht gelöscht werden können.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API-Details finden Sie DeleteBucket in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketAnalyticsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketAnalyticsConfiguration`

Tools für PowerShell V5

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen „testfilter” im angegebenen S3-Bucket.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId
 'testfilter'
```

- API-Details finden Sie DeleteBucketAnalyticsConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketEncryption`

Tools für PowerShell V5

Beispiel 1: Dadurch wird die für den angegebenen S3-Bucket aktivierte Verschlüsselung deaktiviert.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteBucketEncryption in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketInventoryConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketInventoryConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen InventoryName 'test', das dem angegebenen S3-Bucket entspricht.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
 'testInventoryName'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteBucketInventoryConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketMetricsConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketMetricsConfiguration`

Tools für PowerShell V5

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen „testmetrics” im angegebenen S3-Bucket.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
 'testmetrics'
```

- API-Details finden Sie DeleteBucketMetricsConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketPolicy`

Tools für PowerShell V5

Beispiel 1: Der Befehl entfernt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie DeleteBucketPolicy in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketReplication`

Tools für PowerShell V5

Beispiel 1: Löscht die Replikationskonfiguration, die mit dem Bucket namens „amzn-s3-demo-bucket“ verknüpft ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die DeleteReplicationConfiguration Aktion s3: erforderlich ist. Sie werden vor der Ausführung des Vorgangs zur Bestätigung aufgefordert. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API-Details finden Sie DeleteBucketReplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketTagging`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem angegebenen Bucket verknüpft sind.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteBucketTagging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3BucketWebsite`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteBucketWebsite in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3CORSConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3CORSConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt die CORS-Konfiguration für den angegebenen S3-Bucket.

```
Remove-S3CORSConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3CORSConfiguration (DeleteCORSConfiguration)" on
 target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- Weitere API-Informationen finden Sie unter DeleteCORSConfiguration in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

`Remove-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`Remove-S3LifecycleConfiguration`.

Tools für PowerShell V5

Beispiel 1: Der Befehl entfernt alle Lebenszyklusregeln für den angegebenen S3-Bucket.

```
Remove-S3LifecycleConfiguration -BucketName 'amzn-s3-demo-bucket'
```

- API-Details finden Sie DeleteLifecycleConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3MultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3MultipartUpload`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl bricht mehrteilige Uploads ab, die vor mehr als 5 Tagen erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014 erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "Thursday,
 January 02, 2014"
```

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014, 10:45:37, erstellt wurden.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate "2014/01/02
 10:45:37"
```

- API-Details finden Sie AbortMultipartUpload in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3Object`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt das Objekt „sample.txt” aus dem Bucket „test-files”. Sie werden vor der Ausführung des Befehls um eine Bestätigung gebeten. Um den Prompt zu unterdrücken, verwenden Sie den -Force-Schalter.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts „sample.txt“ aus dem Bucket „test-files“, vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert sind.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte „sample1.txt“, „sample2.txt“ und „sample3.txt“ als einzelne Stapelverarbeitung aus dem Bucket „test-files“. Die Antwort des Services listet alle verarbeiteten Schlüssel auf, unabhängig davon, ob die Löschung erfolgreich war oder fehlgeschlagen ist. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Object gibt eine Sammlung von Amazon.S3.Model.S3Object Instanzen zurück, von denen jede ein Key-Element vom Typ string hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
 "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein Schlüsselpräfix "prefix/subprefix" im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden

Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
 Remove-S3Object -Force
```

Beispiel 6: Dieses Beispiel leitet eine Sammlung von Amazon.S3.Model.S3ObjectVersion Instanzen, die Löschmarkierungen darstellen, zum Löschen an das Cmdlet weiter. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
 {$_.IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten stapelweise gelöscht wird (in diesem Fall Markierungen löschen), indem ein Array von Objekten erstellt wird, die mit dem KeyAndVersionCollection Parameter - verwendet werden.

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -Force
```

- API-Details finden Sie DeleteObjects in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3ObjectTagSet`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3ObjectTagSet`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem Objekt mit dem Schlüssel „testfile.txt” im angegebenen S3-Bucket verknüpft sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -Select
 '^Key'
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- API-Details finden Sie DeleteObjectTagging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-S3PublicAccessBlock`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-S3PublicAccessBlock`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlicher Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
 '^BucketName'
```

Ausgabe:

```
amzn-s3-demo-bucket
```

- API-Details finden Sie DeletePublicAccessBlock in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-S3BucketEncryption`

Das folgende Codebeispiel zeigt die Verwendung. `Set-S3BucketEncryption`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl aktiviert die standardmäßige serverseitige AES256-Verschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API-Details finden Sie PutBucketEncryption in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Test-S3Bucket`

Das folgende Codebeispiel zeigt die Verwendung. `Test-S3Bucket`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt „True“ zurück, wenn der Bucket existiert, andernfalls „False“. Der Befehl gibt „True“ zurück, auch wenn der Bucket nicht zum Benutzer gehört.

```
Test-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API-Details finden Sie Test-S3Bucket in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketAccelerateConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketAccelerateConfiguration`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- API-Details finden Sie PutBucketAccelerateConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketNotification`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketNotification`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis konfiguriert ObjectRemovedDelete und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
 $topic
```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen ObjectCreatedAll für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- API-Details finden Sie PutBucketNotification in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketReplication`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketReplication`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Replikationskonfiguration mit einer einzigen Regel festgelegt, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix "" im Bucket 'amzn-s3-demo-bucket' erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglicht. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: Dieses Beispiel legt eine Replikationskonfiguration mit mehreren Regeln fest, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "“ erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglichen. TaxDocs OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }
```

```
$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Beispiel 3: Dieses Beispiel aktualisiert die Replikationskonfiguration im angegebenen Bucket, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "TaxDocs" in den Bucket 'amzn-s3-demo-bucket' steuert.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API-Details finden Sie PutBucketReplication in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketRequestPayment`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketRequestPayment`

Tools für PowerShell V5

Beispiel 1: Aktualisiert die Konfiguration für Zahlungsanforderungen für den Bucket mit dem Namen „amzn-s3-demo-bucket“, sodass der Person, die Downloads aus dem Bucket anfordert, die Kosten für den Download in Rechnung gestellt werden. Standardmäßig zahlt der Bucket-

Eigentümer für Downloads. Um die Zahlungsaufforderung wieder auf die Standardeinstellung zurückzusetzen, verwenden Sie 'BucketOwner' für den Parameter RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- API-Details finden Sie PutBucketRequestPayment in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketTagging`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketTagging`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket namens „**`cloudtrail-test-2018`**“ an: ein Tag mit dem Schlüssel „Stage“ und dem Wert „Test“ sowie ein Tag mit dem Schlüssel „Environment“ und dem Wert „Alpha“. Um zu überprüfen, ob die Tags zum Bucket hinzugefügt wurden, führen Sie **`Get-S3BucketTagging -BucketName bucket_name`** aus. Die Ergebnisse sollten die Tags anzeigen, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **`Write-S3BucketTagging`** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die Cmdlets Resource Groups und Tagging API aus, **`Add-RGTResourceTag`** und **`Remove-RGTResourceTag`**. Verwenden Sie alternativ den Tag Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Dieser Befehl übergibt einen Bucket namens **`cloudtrail-test-2018`** in das **`Write-S3BucketTagging`**-Cmdlet. Es wendet Tags Stage:Production und auf Department:Finance den Bucket an. Beachten Sie, dass **`Write-S3BucketTagging`** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet
 @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- API-Details finden Sie PutBucketTagging in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketVersioning`

Tools für PowerShell V5

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -VersioningConfig_Status
 Enabled
```

- API-Details finden Sie PutBucketVersioning in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3BucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3BucketWebsite`

Tools für PowerShell V5

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indexdokument „index.html“ und dem Fehlerdokument „error.html“.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
 -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API-Details finden Sie PutBucketWebsite in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3LifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3LifecycleConfiguration`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die im $ NewRule angegebene Konfiguration geschrieben/ersetzt. Diese Konfiguration stellt sicher, dass Objekte mit bestimmten Präfix- und Tag-Werten eingeschränkt werden.

```
$NewRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
   Days=  50
  }
  Id =  "Test-From-Write-cmdlet-1"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "non-use"
       "Value" = "yes"
      }
     }
    )
   }
  }
  "Status"= 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 75
  }
 }

 Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
 $NewRule
```

Beispiel 2: Dieses Beispiel legt mehrere Regeln mit Filterung fest. $ ArchiveRule legt fest, dass die Objekte in 30 Tagen auf Glacier archiviert werden sollen und in 120 Tagen DeepArchive. $ ExpireRule läuft für Objekte, bei denen das Präfix 'py' und das tag:key 'archieved' auf 'yes' gesetzt sind, innerhalb von 150 Tagen ab.

```
$ExpireRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  @{
```

```
   Days=  150
  }
  Id =  "Remove-in-150-days"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands=  @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "archived"
       "Value" = "yes"
      }
     }
    )
   }
  }
  Status= 'Enabled'
  NoncurrentVersionExpiration = @{
   NoncurrentDays = 150
  }
 }

 $ArchiveRule = [Amazon.S3.Model.LifecycleRule] @{
  Expiration =  $null
  Id =  "Archive-to-Glacier-in-30-days"
  Filter=  @{
   LifecycleFilterPredicate =  [Amazon.S3.Model.LifecycleAndOperator]@{
    Operands= @(
     [Amazon.S3.Model.LifecyclePrefixPredicate] @{
      "Prefix" =  "py"
     },
     [Amazon.S3.Model.LifecycleTagPredicate] @{
      "Tag"=  @{
       "Key" =  "reviewed"
       "Value" = "yes"
      }
     }
    )
   }
  }
  Status = 'Enabled'
  NoncurrentVersionExpiration = @{
```

```
    NoncurrentDays = 75
   }
   Transitions = @(
    @{
     Days = 30
     "StorageClass"= 'Glacier'
    },
    @{
     Days = 120
     "StorageClass"= [Amazon.S3.S3StorageClass]::DeepArchive
    }
   )
  }

  Write-S3LifecycleConfiguration -BucketName amzn-s3-demo-bucket -Configuration_Rule
  $ExpireRule,$ArchiveRule
```

- API-Details finden Sie PutLifecycleConfiguration in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3Object`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3Object`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl lädt die einzelne Datei „local-sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei „sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“. Wenn der Parameter -Key nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Beispiel 3: Mit diesem Befehl wird die einzelne Datei "local-sample.txt" auf Amazon S3 hochgeladen und ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“ erstellt.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .
\local-sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame Schlüsselpräfix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei der 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle*.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame Schlüsselpräfix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei der 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
 SampleScripts\ -SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel „sample.txt“ enthält.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
 contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -
Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API-Details finden Sie PutObject in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-S3ObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung. `Write-S3ObjectRetention`

Tools für PowerShell V5

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31st Dec 2019 00:00:00“ für das Objekt „testfile.txt“ im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API-Details finden Sie PutObjectRetention in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Security Hub CSPM-Beispiele unter Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS -Tools für PowerShell V5 mit Security Hub CSPM Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-SHUBFinding`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SHUBFinding`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl ruft Security-Hub-Erkenntnisse von Amazon EC2 ab.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.ResourceType = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'AwsEc2'
}
Get-SHUBFinding -Filter $filter
```

Beispiel 2: Mit diesem Befehl werden Security Hub-Ergebnisse von der AWS Konto-ID 123456789012 abgerufen.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.AwsAccountId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = '123456789012'
}
Get-SHUBFinding -Filter $filter
```

Beispiel 3: Dieser Befehl ruft Security-Hub-Erkenntnisse ab, die für den Standard „pci-dss“ generiert wurden.

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.GeneratorId = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'PREFIX'
    Value = 'pci-dss'
}
Get-SHUBFinding -Filter $filter
```

Beispiel 4: Dieser Befehl ruft Security-Hub-Erkenntnisse mit kritischem Schweregrad ab, die den Workflow-Status NOTIFIED haben

```
$filter = New-Object -TypeName Amazon.SecurityHub.Model.AwsSecurityFindingFilters
$filter.SeverityLabel = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter -
Property @{
    Comparison = 'EQUALS'
    Value = 'CRITICAL'
}
$filter.WorkflowStatus = New-Object -TypeName Amazon.SecurityHub.Model.StringFilter
 -Property @{
```

```
    Comparison = 'EQUALS'
    Value = 'NOTIFIED'
}
Get-SHUBFinding -Filter $filter
```

- API-Details finden Sie GetFindings in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon SES-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon SES verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Get-SESIdentity`

Das folgende Codebeispiel zeigt die Verwendung`Get-SESIdentity`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt eine Liste mit allen Identitäten (E-Mail-Adressen und Domänen) für ein bestimmtes AWS Konto zurück, unabhängig vom Bestätigungsstatus.

```
Get-SESIdentity
```

- API-Details finden Sie ListIdentities in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SESSendQuota`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SESSendQuota`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die aktuellen Sende-Limits des Benutzers zurück.

```
Get-SESSendQuota
```

- API-Details finden Sie GetSendQuota in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SESSendStatistic`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SESSendStatistic`

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Sendestatistik des Benutzers zurück. Das Ergebnis ist eine Liste mit Datenpunkten, die die letzten zwei Wochen Sendeaktivität repräsentieren. Jeder Datenpunkt in der Liste enthält Statistiken für ein Intervall von 15 Minuten.

```
Get-SESSendStatistic
```

- API-Details finden Sie GetSendStatistics in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Amazon SES API v2 unter Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon SES API v2 verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Send-SES2Email`

Das folgende Codebeispiel zeigt die Verwendung`Send-SES2Email`.

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt, wie Sie eine Standard-E-Mail-Nachricht versenden.

```
Send-SES2Email -FromEmailAddress "sender@example.com" -Destination_ToAddress
 "recipient@example.com" -Subject_Data "Email Subject" -Text_Data "Email Body"
```

- API-Details finden Sie SendEmail in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon SNS-Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon SNS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Publish-SNSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Publish-SNSMessage`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht mit einer einzelnen Nachricht, die MessageAttribute als Inline deklariert ist.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String';
 StringValue ='AnyCity'}}
```

Beispiel 2: Dieses Beispiel zeigt die Veröffentlichung einer Nachricht, bei der mehrere im Voraus MessageAttributes deklariert wurden.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -Message
 "Hello" -MessageAttribute $messageAttributes
```

- Weitere API-Informationen finden Sie unter Publish in der AWS -Tools für PowerShell -Cmdlet-Referenz (V5).

Amazon SQS-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon SQS verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-SQSPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Add-SQSPermission`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ermöglicht es der angegebenen Person, Nachrichten aus der angegebenen Warteschlange AWS-Konto zu senden.

```
Add-SQSPermission -Action SendMessage -AWSAccountId 80398EXAMPLE -Label
 SendMessagesFromMyQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue
```

- API-Details finden Sie AddPermission in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Clear-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Clear-SQSQueue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Nachrichten aus der angegebenen Warteschlange gelöscht.

```
Clear-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie PurgeQueue in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-SQSMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SQSMessageVisibility`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für die Nachricht mit dem angegebenen Empfangs-Handle in der angegebenen Warteschlange auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden).

```
Edit-SQSMessageVisibility -QueueUrl https://sqs.us-east-1.amazonaws.com/8039EXAMPLE/
MyQueue -ReceiptHandle AQEBgGDh...J/Iqww== -VisibilityTimeout 36000
```

- API-Details finden Sie ChangeMessageVisibility in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-SQSMessageVisibilityBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SQSMessageVisibilityBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Sichtbarkeits-Timeout für zwei Nachrichten mit den angegebenen Empfangs-Handles in der angegebenen Warteschlange geändert. Das Sichtbarkeits-Timeout der ersten Nachricht wird auf 10 Stunden geändert (10 Stunden * 60 Minuten * 60 Sekunden = 36 000 Sekunden). Das Sichtbarkeits-Timeout der zweiten Nachricht wird auf 5 Stunden geändert (5 Stunden * 60 Minuten * 60 Sekunden = 18 000 Sekunden).

```
$changeVisibilityRequest1 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest1.Id = "Request1"
$changeVisibilityRequest1.ReceiptHandle = "AQEBd329...v6gl8Q=="
$changeVisibilityRequest1.VisibilityTimeout = 36000

$changeVisibilityRequest2 = New-Object
 Amazon.SQS.Model.ChangeMessageVisibilityBatchRequestEntry
$changeVisibilityRequest2.Id = "Request2"
$changeVisibilityRequest2.ReceiptHandle = "AQEBgGDh...J/Iqww=="
$changeVisibilityRequest2.VisibilityTimeout = 18000

Edit-SQSMessageVisibilityBatch -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue -Entry $changeVisibilityRequest1,
 $changeVisibilityRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request2, Request1}
```

- API-Details finden Sie ChangeMessageVisibilityBatch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SQSDeadLetterSourceQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSDeadLetterSourceQueue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die URLs aller Warteschlangen aufgelistet, die die angegebene Warteschlange als Warteschlange für unzustellbare Nachrichten verwenden.

```
Get-SQSDeadLetterSourceQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
```

- API-Details finden Sie ListDeadLetterSourceQueues in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Parameter aufgelistet.

```
Get-SQSQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/AnotherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/DeadLetterQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

Beispiel 2: In diesem Beispiel werden alle Warteschlangen auflistet, die mit dem angegebenen Namen beginnen.

```
Get-SQSQueue -QueueNamePrefix My
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyOtherQueue
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyDeadLetterQueue
```

- API-Details finden Sie ListQueues in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SQSQueueAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueueAttribute`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zu den Attributen für die angegebene Warteschlange aufgelistet.

```
Get-SQSQueueAttribute -AttributeName All -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
```

```
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14
                                        495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                        398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                        arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                            : {[QueueArn, arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue], [ApproximateNumberOfMessages, 0],
                                        [ApproximateNumberOfMessagesNotVisible, 0],
 [ApproximateNumberOfMessagesDelayed, 0]...}
```

Beispiel 2: In diesem Beispiel werden nur die angegebenen Attribute für die angegebene Warteschlange separat aufgelistet.

```
Get-SQSQueueAttribute -AttributeName MaximumMessageSize, VisibilityTimeout -QueueUrl
 https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
VisibilityTimeout                     : 30
DelaySeconds                          : 0
MaximumMessageSize                    : 262144
MessageRetentionPeriod                : 345600
ApproximateNumberOfMessages           : 0
ApproximateNumberOfMessagesNotVisible : 0
ApproximateNumberOfMessagesDelayed    : 0
CreatedTimestamp                      : 2/11/2015 5:53:35 PM
LastModifiedTimestamp                 : 12/29/2015 2:23:17 PM
QueueARN                              : arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue
Policy                                : {"Version":"2012-10-17",
 "Id":"arn:aws:sqs:us-east-1:80398EXAMPLE:MyQueue/SQSDefaultPolicy","Statement":
[{"Sid":"Sid14
```

```
                                          495134224EX","Effect":"Allow","Principal":
{"AWS":"*"},"Action":"SQS:SendMessage","Resource":"arn:aws:sqs:us-east-1:80
                                          398EXAMPLE:MyQueue","Condition":
{"ArnEquals":{"aws:SourceArn":"arn:aws:sns:us-east-1:80398EXAMPLE:MyTopic"}}},
{"Sid":

 "SendMessagesFromMyQueue","Effect":"Allow","Principal":
{"AWS":"80398EXAMPLE"},"Action":"SQS:SendMessage","Resource":"
                                          arn:aws:sqs:us-
east-1:80398EXAMPLE:MyQueue"}]}
Attributes                              : {[MaximumMessageSize, 262144],
 [VisibilityTimeout, 30]}
```

- API-Details finden Sie GetQueueAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SQSQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SQSQueueUrl`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die URL der Warteschlange mit dem angegebenen Namen aufgelistet.

```
Get-SQSQueueUrl -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie GetQueueUrl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `New-SQSQueue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Warteschlange mit dem angegebenen Namen erstellt.

```
New-SQSQueue -QueueName MyQueue
```

Ausgabe:

```
https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie CreateQueue in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Receive-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Receive-SQSMessage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen für die nächsten 10 Nachrichten aufgelistet, die in der angegebenen Warteschlange empfangen werden sollen. Die Informationen enthalten Werte für die angegebenen Nachrichtenattribute, sofern sie existieren.

```
Receive-SQSMessage -AttributeName SenderId, SentTimestamp -MessageAttributeName
 StudentName, StudentGrade -MessageCount 10 -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
Attributes             : {[SenderId, AIDAIAZKMSNQ7TEXAMPLE], [SentTimestamp,
 1451495923744]}
Body                   : Information about John Doe's grade.
MD5OfBody              : ea572796e3c231f974fe75d89EXAMPLE
MD5OfMessageAttributes : 48c1ee811f0fe7c4e88fbe0f5EXAMPLE
MessageAttributes      : {[StudentGrade, Amazon.SQS.Model.MessageAttributeValue],
 [StudentName, Amazon.SQS.Model.MessageAttributeValue]}
MessageId              : 53828c4b-631b-469b-8833-c093cEXAMPLE
ReceiptHandle          : AQEBpfGp...20Q5cg==
```

- API-Details finden Sie ReceiveMessage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSMessage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Nachricht mit dem angegebenen Empfangs-Handle aus der angegebenen Warteschlange gelöscht.

```
Remove-SQSMessage -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
 -ReceiptHandle AQEBd329...v6gl8Q==
```

- API-Details finden Sie DeleteMessage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SQSMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSMessageBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Empfangs-Handles aus der angegebenen Warteschlange gelöscht.

```
$deleteMessageRequest1 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest1.Id = "Request1"
$deleteMessageRequest1.ReceiptHandle = "AQEBX2g4...wtJSQg=="

$deleteMessageRequest2 = New-Object Amazon.SQS.Model.DeleteMessageBatchRequestEntry
$deleteMessageRequest2.Id = "Request2"
$deleteMessageRequest2.ReceiptHandle = "AQEBqOVY...KTsLYg=="

Remove-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $deleteMessageRequest1, $deleteMessageRequest2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {Request1, Request2}
```

- API-Details finden Sie DeleteMessageBatch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SQSPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSPermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Berechtigungseinstellungen mit der angegebenen Bezeichnung aus der angegebenen Warteschlange entfernt.

```
Remove-SQSPermission -Label SendMessagesFromMyQueue -QueueUrl https://sqs.us-
east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie RemovePermission in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SQSQueue`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SQSQueue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene Warteschlange gelöscht.

```
Remove-SQSQueue -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie DeleteQueue in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Send-SQSMessage`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SQSMessage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Nachricht mit den angegebenen Attributen und dem angegebenen Nachrichtentext an die angegebene Warteschlange gesendet, wobei die Zustellung der Nachricht um 10 Sekunden verzögert wird.

```
$cityAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
```

```
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Send-SQSMessage -DelayInSeconds 10 -MessageAttributes $messageAttributes -
MessageBody "Information about the largest city in Any Region." -QueueUrl https://
sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

Ausgabe:

```
MD5OfMessageAttributes              MD5OfMessageBody                    MessageId

----------------------              ----------------                    ---------

1d3e51347bc042efbdf6dda31EXAMPLE    51b0a3256d59467f973009b73EXAMPLE    c35fed8f-
c739-4d0c-818b-1820eEXAMPLE
```

- API-Details finden Sie SendMessage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Send-SQSMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SQSMessageBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden zwei Nachrichten mit den angegebenen Attributen und Nachrichtentexten an die angegebene Warteschlange gesendet. Die Zustellung verzögert sich um 15 Sekunden für die erste Nachricht und um 10 Sekunden für die zweite Nachricht.

```
$student1NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1NameAttributeValue.DataType = "String"
$student1NameAttributeValue.StringValue = "John Doe"

$student1GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student1GradeAttributeValue.DataType = "Number"
```

```
$student1GradeAttributeValue.StringValue = "89"

$student2NameAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2NameAttributeValue.DataType = "String"
$student2NameAttributeValue.StringValue = "Jane Doe"

$student2GradeAttributeValue = New-Object Amazon.SQS.Model.MessageAttributeValue
$student2GradeAttributeValue.DataType = "Number"
$student2GradeAttributeValue.StringValue = "93"

$message1 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message1.DelaySeconds = 15
$message1.Id = "FirstMessage"
$message1.MessageAttributes.Add("StudentName", $student1NameAttributeValue)
$message1.MessageAttributes.Add("StudentGrade", $student1GradeAttributeValue)
$message1.MessageBody = "Information about John Doe's grade."

$message2 = New-Object Amazon.SQS.Model.SendMessageBatchRequestEntry
$message2.DelaySeconds = 10
$message2.Id = "SecondMessage"
$message2.MessageAttributes.Add("StudentName", $student2NameAttributeValue)
$message2.MessageAttributes.Add("StudentGrade", $student2GradeAttributeValue)
$message2.MessageBody = "Information about Jane Doe's grade."

Send-SQSMessageBatch -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/
MyQueue -Entry $message1, $message2
```

Ausgabe:

```
Failed    Successful

------    ----------

{}        {FirstMessage, SecondMessage}
```

- API-Details finden Sie SendMessageBatch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Set-SQSQueueAttribute`

Das folgende Codebeispiel zeigt die Verwendung. `Set-SQSQueueAttribute`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt, wie Sie eine Richtlinie festlegen, mit der eine Warteschlange für ein SNS-Thema abonniert wird. Wenn eine Nachricht in einem Thema veröffentlicht wird, wird eine Nachricht an die abonnierte Warteschlange gesendet.

```
# create the queue and topic to be associated
$qurl = New-SQSQueue -QueueName "myQueue"
$topicarn = New-SNSTopic -Name "myTopic"

# get the queue ARN to inject into the policy; it will be returned
# in the output's QueueARN member but we need to put it into a variable
# so text expansion in the policy string takes effect
$qarn = (Get-SQSQueueAttribute -QueueUrl $qurl -AttributeName "QueueArn").QueueARN

# construct the policy and inject arns
$policy = @"
{
  "Version":"2012-10-17",
  "Id": "$qarn/SQSPOLICY",
  "Statement": [
      {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "SQS:SendMessage",
      "Resource": "$qarn",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "$topicarn"
          }
      }
    }
  ]
}
"@

# set the policy
Set-SQSQueueAttribute -QueueUrl $qurl -Attribute @{ Policy=$policy }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Attribute für die angegebene Warteschlange festgelegt.

```
Set-SQSQueueAttribute -Attribute @{"DelaySeconds" = "10"; "MaximumMessageSize" =
 "131072"} -QueueUrl https://sqs.us-east-1.amazonaws.com/80398EXAMPLE/MyQueue
```

- API-Details finden Sie SetQueueAttributes in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS STS Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS STS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Convert-STSAuthorizationMessage`

Das folgende Codebeispiel zeigt die Verwendung`Convert-STSAuthorizationMessage`.

Tools für PowerShell V5

Beispiel 1: Decodiert die zusätzlichen Informationen, die im bereitgestellten codierten Nachrichteninhalt enthalten sind, der als Antwort auf eine Anfrage zurückgegeben wurde. Die Zusatzinformationen werden verschlüsselt, da es sich bei den Details des Autorisierungsstatus um vertrauliche Informationen handeln kann, die der Benutzer, der die Aktion angefordert hat, nicht sehen sollte.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- API-Details finden Sie DecodeAuthorizationMessage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-STSFederationToken`

Das folgende Codebeispiel zeigt die Verwendung. `Get-STSFederationToken`

Tools für PowerShell V5

Beispiel 1: Fordert ein Verbund-Token an, das eine Stunde lang gültig ist, mit „Bob“ als Namen des Verbundbenutzers. Dieser Name kann verwendet werden, um in einer ressourcenbasierten Richtlinie (z. B. einer Amazon-S3-Bucket-Richtlinie) auf den Namen des Verbundbenutzers zu verweisen. Die bereitgestellte IAM-Richtlinie im JSON-Format wird verwendet, um die Berechtigungen, die dem IAM-Benutzer zur Verfügung stehen, einzuschränken. Die angegebene Richtlinie kann nicht mehr Berechtigungen gewähren, als dem anfordernden Benutzer gewährt wurden. Die endgültigen Berechtigungen für den Verbundbenutzer sind der restriktivste Satz, basierend auf der Schnittmenge der übergebenen Richtlinie und der IAM-Benutze-Richtlinie.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- API-Details finden Sie GetFederationToken in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-STSSessionToken`

Das folgende Codebeispiel zeigt die Verwendung. `Get-STSSessionToken`

Tools für PowerShell V5

Beispiel 1: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die für einen festgelegten Zeitraum gültig sind. Die Anmeldeinformationen, die zum Anfordern temporärer Anmeldeinformationen verwendet werden, werden aus den aktuellen Shell-Standardwerten abgeleitet. Verwenden Sie die Parameter -ProfileName oder - AccessKey /-SecretKey , um andere Anmeldeinformationen anzugeben.

```
Get-STSSessionToken
```

Ausgabe:

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
```

```
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Beispiel 2: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Die zum Stellen der Anfrage verwendeten Anmeldeinformationen werden aus dem angegebenen Profil abgerufen.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

Beispiel 3: Gibt eine **`Amazon.RuntimeAWSCredentials`**-Instance mit temporären Anmeldeinformationen zurück, die eine Stunde lang gültig sind. Dabei wird die Identifikationsnummer des MFA-Geräts verwendet, das dem Konto zugeordnet ist, dessen Anmeldeinformationen im Profil „myprofilename“ angegeben sind, und der vom Gerät bereitgestellte Wert.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

Ausgabe:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- API-Details finden Sie GetSessionToken in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Use-STSRole`

Das folgende Codebeispiel zeigt die Verwendung. `Use-STSRole`

Tools für PowerShell V5

Beispiel 1: Gibt eine Reihe temporärer Anmeldeinformationen (Zugriffsschlüssel, geheimer Schlüssel und Sitzungstoken) zurück, die eine Stunde lang verwendet werden können, um auf AWS Ressourcen zuzugreifen, auf die der anfordernde Benutzer normalerweise keinen Zugriff hat. Die zurückgegebenen Anmeldeinformationen verfügen über die Berechtigungen, die von der Zugriffsrichtlinie der übernommenen Rolle und der bereitgestellten Richtlinie zugelassen sind (Sie können die bereitgestellte Richtlinie nicht verwenden, um Berechtigungen zu gewähren, die über die in der Zugriffsrichtlinie der übernommenen Rolle definierten Berechtigungen hinausgehen).

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
Policy "...JSON policy..." -DurationInSeconds 3600
```

Beispiel 2: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine Stunde gültig sind und über dieselben Berechtigungen verfügen, die in der Zugriffsrichtlinie der übernommenen Rolle definiert sind.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600
```

Beispiel 3: Gibt einen Satz temporärer Anmeldeinformationen zurück, die die Seriennummer und das generierte Token von einem MFA bereitstellen, das den Anmeldeinformationen des Benutzers zugeordnet ist, die zur Ausführung des Cmdlet verwendet werden.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

Beispiel 4: Gibt einen Satz temporärer Anmeldeinformationen zurück, die eine in einem Kundenkonto definierte Rolle übernommen haben. Für jede Rolle, die der Drittanbieter übernehmen kann, muss das Kundenkonto eine Rolle mithilfe einer Kennung erstellen, die bei jeder Übernahme der Rolle im ExternalId Parameter - übergeben werden muss.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo" -
DurationInSeconds 3600 -ExternalId "ABC123"
```

- API-Details finden Sie AssumeRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Use-STSWebIdentityRole`

Das folgende Codebeispiel zeigt die Verwendung. `Use-STSWebIdentityRole`

Tools für PowerShell V5

Beispiel 1: Gibt einen temporären Satz Anmeldeinformationen mit einer Gültigkeit von einer Stunde für einen Benutzer zurück, der mit dem Identitätsanbieter „Anmeldung mit Amazon“ authentifiziert wurde. Die Anmeldeinformationen übernehmen die Zugriffsrichtlinie, die der durch die Rollen-ARN identifizierten Rolle zugeordnet ist. Optional können Sie eine JSON-Richtlinie an den Parameter -Richtlinie übergeben, die die Zugriffsberechtigungen weiter verfeinert (Sie können keine Berechtigungen mehr gewähren, als in den mit der Rolle verknüpften Berechtigungen verfügbar sind). Der an - übergebene Wert WebIdentityToken ist die eindeutige Benutzerkennung, die vom Identitätsanbieter zurückgegeben wurde.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- API-Details finden Sie AssumeRoleWithWebIdentity in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Support Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. Support

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-ASACommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung`Add-ASACommunicationToCase`.

Tools für PowerShell V5

Beispiel 1: Fügt den Text einer E-Mail-Kommunikation zum angegebenen Fall hinzu.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CommunicationBody "Some text about the case"
```

Beispiel 2: Fügt den Text einer E-Mail-Nachricht zum angegebenen Fall sowie eine oder mehrere E-Mail-Adressen hinzu, die in der CC-Zeile der E-Mail enthalten sind.

```
Add-ASACommunicationToCase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -
CcEmailAddress @("email1@address.com", "email2@address.com") -CommunicationBody
 "Some text about the case"
```

- API-Details finden Sie AddCommunicationToCase in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASACase`

Tools für PowerShell V5

Beispiel 1: Gibt die Details aller Supportfälle zurück.

```
Get-ASACase
```

Beispiel 2: Gibt die Details aller Supportfälle seit dem angegebenen Datum und der angegebenen Uhrzeit zurück.

```
Get-ASACase -AfterTime "2013-09-10T03:06Z"
```

Beispiel 3: Gibt die Details der ersten 10 Supportfälle zurück, einschließlich derer, die gelöst wurden.

```
Get-ASACase -MaxResult 10 -IncludeResolvedCases $true
```

Beispiel 4: Gibt die Details des einzelnen angegebenen Supportfalls zurück.

```
Get-ASACase -CaseIdList "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 5: Gibt die Details der angegebenen Supportfälle zurück.

```
Get-ASACase -CaseIdList @("case-12345678910-2013-c4c1d2bf33c5cf47",
 "case-18929034710-2011-c4fdeabf33c5cf47")
```

- API-Details finden Sie DescribeCases in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASACommunication`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASACommunication`

Tools für PowerShell V5

Beispiel 1: Gibt alle Kommunikationen für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

Beispiel 2: Gibt alle Kommunikationen seit Mitternacht UTC am 1. Januar 2012 für den angegebenen Fall zurück.

```
Get-ASACommunication -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47" -AfterTime
 "2012-01-10T00:00Z"
```

- API-Details finden Sie DescribeCommunications in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASAService`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASAService`

Tools für PowerShell V5

Beispiel 1: Gibt alle verfügbaren Servicecodes, Namen und Kategorien zurück.

```
Get-ASAService
```

Beispiel 2: Gibt den Namen und die Kategorien für den Service mit dem angegebenen Code zurück.

```
Get-ASAService -ServiceCodeList "amazon-cloudfront"
```

Beispiel 3: Gibt den Namen und die Kategorien für die angegebenen Servicecodes zurück.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch")
```

Beispiel 4: Gibt den Namen und die Kategorien (in Japanisch) für die angegebenen Servicecodes zurück. Derzeit werden die Sprachcodes Englisch („en“) und Japanisch („ja“) unterstützt.

```
Get-ASAService -ServiceCodeList @("amazon-cloudfront", "amazon-cloudwatch") -
Language "ja"
```

- API-Details finden Sie DescribeServices in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASASeverityLevel`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASASeverityLevel`

Tools für PowerShell V5

Beispiel 1: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können.

```
Get-ASASeverityLevel
```

Beispiel 2: Gibt die Liste der Schweregrade zurück, die einem AWS Support-Fall zugewiesen werden können. Die Namen der Grade werden auf Japanisch zurückgegeben.

```
Get-ASASeverityLevel -Language "ja"
```

- API-Details finden Sie DescribeSeverityLevels in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASATrustedAdvisorCheck`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheck`

Tools für PowerShell V5

Beispiel 1: Gibt die Sammlung von Prüfungen von Trusted Advisor zurück. Sie müssen den Parameter „Language“ angeben, der entweder „en“ für englische Ausgabe oder „ja“ für japanische Ausgabe akzeptieren kann.

```
Get-ASATrustedAdvisorCheck -Language "en"
```

- API-Details finden Sie DescribeTrustedAdvisorChecks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASATrustedAdvisorCheckRefreshStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckRefreshStatus`

Tools für PowerShell V5

Beispiel 1: Gibt den aktuellen Status der Aktualisierungsanforderungen für die angegebenen Prüfungen zurück. Request-ASATrustedAdvisorCheckRefresh kann verwendet werden, um anzufordern, dass die Statusinformationen der Schecks aktualisiert werden.

```
Get-ASATrustedAdvisorCheckRefreshStatus -CheckId @("checkid1", "checkid2")
```

- API-Details finden Sie DescribeTrustedAdvisorCheckRefreshStatuses in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASATrustedAdvisorCheckResult`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckResult`

Tools für PowerShell V5

Beispiel 1: Gibt die Ergebnisse einer Prüfung von Trusted Advisor zurück. Die Liste der verfügbaren Trusted Advisor-Checks finden Sie unter Get-ASATrustedAdvisorChecks. Die

Ausgabe umfasst den Gesamtstatus der Prüfung, den Zeitstempel, zu dem die Prüfung zuletzt durchgeführt wurde, und die eindeutige Prüf-ID für die jeweilige Prüfung. Um die Ergebnisse auf Japanisch auszugeben, fügen Sie den Parameter -Language „ja“ hinzu.

```
Get-ASATrustedAdvisorCheckResult -CheckId "checkid1"
```

- API-Details finden Sie DescribeTrustedAdvisorCheckResult in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-ASATrustedAdvisorCheckSummary`

Das folgende Codebeispiel zeigt die Verwendung. `Get-ASATrustedAdvisorCheckSummary`

Tools für PowerShell V5

Beispiel 1: Gibt die neueste Zusammenfassung für die angegebene Prüfung von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId "checkid1"
```

Beispiel 2: Gibt die neuesten Zusammenfassungen für die angegebenen Prüfungen von Trusted Advisor zurück.

```
Get-ASATrustedAdvisorCheckSummary -CheckId @("checkid1", "checkid2")
```

- API-Details finden Sie DescribeTrustedAdvisorCheckSummaries in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `New-ASACase`

Tools für PowerShell V5

Beispiel 1: Erstellt einen neuen Fall im AWS Support Center. Werte für die CategoryCode Parameter - ServiceCode und - können mit dem Get-ASAService Cmdlet abgerufen werden. Der Wert für den SeverityCode Parameter - kann mit dem Cmdlet abgerufen werden. Get-ASASeverityLevel Der IssueType Parameterwert - kann entweder „Kundenservice“ oder „Technik“

sein. Bei Erfolg wird die AWS Support-Fallnummer ausgegeben. Standardmäßig wird der Fall auf Englisch bearbeitet. Um Japanisch zu verwenden, fügen Sie den Parameter -Language „ja“ hinzu. Die CommunicationBody Parameter -ServiceCode, -CategoryCode, -Subject und - sind obligatorisch.

```
New-ASACase -ServiceCode "amazon-cloudfront" -CategoryCode "APIs" -SeverityCode
 "low" -Subject "subject text" -CommunicationBody "description of the case" -
CcEmailAddress @("email1@domain.com", "email2@domain.com") -IssueType "technical"
```

- API-Details finden Sie CreateCase in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Request-ASATrustedAdvisorCheckRefresh`

Das folgende Codebeispiel zeigt die Verwendung. `Request-ASATrustedAdvisorCheckRefresh`

Tools für PowerShell V5

Beispiel 1: Fordert eine Aktualisierung für die angegebene Prüfung von Trusted Advisor an.

```
Request-ASATrustedAdvisorCheckRefresh -CheckId "checkid1"
```

- API-Details finden Sie RefreshTrustedAdvisorCheck in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Resolve-ASACase`

Das folgende Codebeispiel zeigt die Verwendung. `Resolve-ASACase`

Tools für PowerShell V5

Beispiel 1: Gibt den Anfangszustand des angegebenen Falls und den aktuellen Zustand nach Abschluss des Aufrufs zur Lösung zurück.

```
Resolve-ASACase -CaseId "case-12345678910-2013-c4c1d2bf33c5cf47"
```

- API-Details finden Sie ResolveCase in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Beispiele für Systems Manager zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Systems Manager verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Add-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung`Add-SSMResourceTag`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster mit neuen Tags aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$option1 = @{Key="Stack";Value=@("Production")}
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $option1
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Stack"
$tag1.Value = "Production"
```

```
Add-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -Tag $tag1
```

- API-Details finden Sie AddTagsToResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-SSMDocumentPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-SSMDocumentPermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden allen Konten für ein Dokument „Teilen“-Berechtigungen hinzugefügt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScript" -PermissionType "Share" -
AccountIdsToAdd all
```

Beispiel 2: Dieses Beispiel fügt einem bestimmten Konto die Berechtigung zum „Freigeben“ eines Dokuments hinzu. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Edit-SSMDocumentPermission -Name "RunShellScriptNew" -PermissionType "Share" -
AccountIdsToAdd "123456789012"
```

- API-Details finden Sie ModifyDocumentPermission in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMActivation`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel enthält Details zu den Aktivierungen in Ihrem Konto.

```
Get-SSMActivation
```

Ausgabe:

```
ActivationId        : 08e51e79-1e36-446c-8e63-9458569c1363
CreatedDate         : 3/1/2017 12:01:51 AM
DefaultInstanceName : MyWebServers
Description         :
ExpirationDate      : 3/2/2017 12:01:51 AM
Expired             : False
IamRole             : AutomationRole
RegistrationLimit   : 10
RegistrationsCount  : 0
```

- API-Details finden Sie DescribeActivations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociation`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die Zuordnung zwischen einer Instance und einem Dokument.

```
Get-SSMAssociation -InstanceId "i-0000293ffd8c57862" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temp_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- API-Details finden Sie DescribeAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAssociationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationExecution`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Ausführungen für die angegebene Zuordnungs-ID zurückgegeben

```
Get-SSMAssociationExecution -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId         : 123a45a0-c678-9012-3456-78901234db5e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 123a45a0-c678-9012-3456-78901234db5e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=4}
Status                : Success
```

- API-Details finden Sie DescribeAssociationExecutions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAssociationExecutionTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationExecutionTarget`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Ressourcen-ID und ihr Ausführungsstatus angezeigt, die Teil der Ausführungsziele der Zuordnung sind

```
Get-SSMAssociationExecutionTarget -AssociationId 123a45a0-
c678-9012-3456-78901234db5e -ExecutionId 123a45a0-c678-9012-3456-78901234db5e |
 Select-Object ResourceId, Status
```

Ausgabe:

```
ResourceId           Status
----------           ------
i-0b1b2a3456f7a890b  Success
i-01c12a45d6fc7a89f  Success
i-0a1caf234f56d7dc8  Success
```

```
i-012a3fd45af6dbcfe  Failed
i-0ddc1df23c4a5fb67  Success
```

Beispiel 2: Dieser Befehl überprüft die jeweilige Ausführung einer bestimmten Automatisierung seit gestern, der ein Befehlsdokument zugeordnet ist. Außerdem wird geprüft, ob die Ausführung der Zuordnung fehlgeschlagen ist, und wenn ja, werden die Details zum Befehlsaufruf für die Ausführung zusammen mit der Instance-ID angezeigt

```
$AssociationExecution= Get-SSMAssociationExecutionTarget -
AssociationId 1c234567-890f-1aca-a234-5a678d901cb0 -ExecutionId
 12345ca12-3456-2345-2b45-23456789012 |
    Where-Object {$_.LastExecutionDate -gt (Get-Date -Hour 00 -Minute
 00).AddDays(-1)}

foreach ($execution in $AssociationExecution) {
    if($execution.Status -ne 'Success'){
        Write-Output "There was an issue executing the association
 $($execution.AssociationId) on $($execution.ResourceId)"
        Get-SSMCommandInvocation -CommandId $execution.OutputSource.OutputSourceId -
Detail:$true | Select-Object -ExpandProperty CommandPlugins
    }
}
```

Ausgabe:

```
There was an issue executing the association 1c234567-890f-1aca-a234-5a678d901cb0 on
 i-0a1caf234f56d7dc8


Name                   : aws:runPowerShellScript
Output                 :
                         ----------ERROR-------
                         failed to run commands: exit status 1
OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 1
ResponseFinishDateTime : 5/29/2019 11:04:49 AM
ResponseStartDateTime  : 5/29/2019 11:04:49 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Failed
```

```
StatusDetails          : Failed
```

- API-Details finden Sie DescribeAssociationExecutionTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAssociationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet Details zu den Zuordnungen für eine Instance auf. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="InstanceId";Value=@("i-0000293ffd8c57862")}
Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 2: In diesem Beispiel werden alle Verknüpfungen für ein Konfigurationsdokument aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter2 = @{Key="Name";Value=@("AWS-UpdateSSMAgent")}
Get-SSMAssociationList -AssociationFilterList $filter2
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
```

```
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.AssociationFilter
$filter1.Key = "InstanceId"
$filter1.Value = "i-0000293ffd8c57862"

Get-SSMAssociationList -AssociationFilterList $filter1
```

Ausgabe:

```
AssociationId      : d8617c07-2079-4c18-9847-1655fc2698b0
DocumentVersion    :
InstanceId         : i-0000293ffd8c57862
LastExecutionDate  : 2/20/2015 8:31:11 AM
Name               : AWS-UpdateSSMAgent
Overview           : Amazon.SimpleSystemsManagement.Model.AssociationOverview
ScheduleExpression :
Targets            : {InstanceIds}
```

- API-Details finden Sie ListAssociations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAssociationVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAssociationVersionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Versionen der angegebenen Zuordnung abgerufen.

```
Get-SSMAssociationVersionList -AssociationId 123a45a0-c678-9012-3456-78901234db5e
```

Ausgabe:

```
AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
```

```
AssociationName    :
AssociationVersion : 2
ComplianceSeverity :
CreatedDate        : 3/12/2019 9:21:01 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression :
Targets            : {InstanceIds}

AssociationId      : 123a45a0-c678-9012-3456-78901234db5e
AssociationName    : test-case-1234567890
AssociationVersion : 1
ComplianceSeverity :
CreatedDate        : 3/2/2019 8:53:29 AM
DocumentVersion    :
MaxConcurrency     :
MaxErrors          :
Name               : AWS-GatherSoftwareInventory
OutputLocation     :
Parameters         : {}
ScheduleExpression : rate(30minutes)
Targets            : {InstanceIds}
```

- API-Details finden Sie ListAssociationVersions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung`Get-SSMAutomationExecution`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details einer Automation-Ausführung angezeigt.

```
Get-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
FailureMessage            : Step launchInstance failed maximum allowed times. You
 are not authorized to perform this operation. Encoded
                            authorization failure message:
 B_V2QyyN7NhSZQYpmVzpEc4oSnj2GLTNYnXUHsTbqJkNMoDgubmbtthLmZyaiUYekORIrA42-
fv1x-04q5Fjff6glh
                            Yb6TI5b0GQeeNrpwNvpDzmO-
PSR1swlAbg9fdM9BcNjyrznspUkWpuKu9EC1Ou6v3OXU1KC9nZ7mPlWMFZNkSioQqpwWEvMw-
GZktsQzm67qOhUhBNOLWYhbS
                            pkfiqzY-5nw3S0obx30fhd3EJa5O_-
GjV_a0nFXQJa70ik40bFOrEh3MtCSbrQT6--DvFy_FQ8TKvkIXadyVskeJI84XOF5WmA60f1pi5GI08i-
nRfZS6oDeU
                            gELBjjoFKD8s3L2aI0B6umWVxnQOjqhQRxwJ53b54sZJ2PW3v_mtg9-
q0CK0ezS3xfh_y0ilaUGOAZG-xjQFuvU_JZedWpla3xi-MZsmblAifBI
                            (Service: AmazonEC2; Status Code: 403; Error Code:
 UnauthorizedOperation; Request ID:
                            6a002f94-ba37-43fd-99e6-39517715fce5)
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
Parameters                : {[AutomationAssumeRole,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [InstanceIamRole,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]], [SourceAmiId,

 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
StepExecutions            : {launchInstance, updateOSSoftware, stopInstance,
 createImage...}
```

Beispiel 2: In diesem Beispiel werden die Schrittdetails für die angegebene Automatisierungsausführungs-ID aufgeführt

```
Get-SSMAutomationExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object -ExpandProperty StepExecutions | Select-Object
 StepName, Action, StepStatus, ValidNextSteps
```

Ausgabe:

```
StepName                  Action                   StepStatus ValidNextSteps
--------                  ------                   ---------- --------------
LaunchInstance            aws:runInstances         Success    {OSCompatibilityCheck}
OSCompatibilityCheck      aws:runCommand           Success    {RunPreUpdateScript}
RunPreUpdateScript        aws:runCommand           Success    {UpdateEC2Config}
UpdateEC2Config           aws:runCommand           Cancelled  {}
UpdateSSMAgent            aws:runCommand           Pending    {}
UpdateAWSPVDriver         aws:runCommand           Pending    {}
UpdateAWSEnaNetworkDriver aws:runCommand           Pending    {}
UpdateAWSNVMe             aws:runCommand           Pending    {}
InstallWindowsUpdates     aws:runCommand           Pending    {}
RunPostUpdateScript       aws:runCommand           Pending    {}
RunSysprepGeneralize      aws:runCommand           Pending    {}
StopInstance              aws:changeInstanceState  Pending    {}
CreateImage               aws:createImage          Pending    {}
TerminateInstance         aws:changeInstanceState  Pending    {}
```

- API-Details finden Sie GetAutomationExecution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAutomationExecutionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAutomationExecutionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle aktiven und beendeten Automation-Ausführungen beschrieben, die mit Ihrem Konto verknüpft sind.

```
Get-SSMAutomationExecutionList
```

Ausgabe:

```
AutomationExecutionId     : 4105a4fc-f944-11e6-9d32-8fb2db27a909
AutomationExecutionStatus : Failed
DocumentName              : AWS-UpdateLinuxAmi
DocumentVersion           : 1
ExecutedBy                : admin
ExecutionEndTime          : 2/22/2017 9:17:08 PM
ExecutionStartTime        : 2/22/2017 9:17:02 PM
LogFile                   :
```

```
Outputs                   : {[createImage.ImageId,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
```

Beispiel 2: In diesem Beispiel werden ExecutionID, Dokument und start/end Ausführungszeitstempel für Ausführungen mit einem AutomationExecutionStatus anderen Wert als 'Success' angezeigt

```
Get-SSMAutomationExecutionList | Where-Object AutomationExecutionStatus
 -ne "Success" | Select-Object AutomationExecutionId, DocumentName,
 AutomationExecutionStatus, ExecutionStartTime, ExecutionEndTime | Format-Table -
AutoSize
```

Ausgabe:

```
AutomationExecutionId                DocumentName
 AutomationExecutionStatus ExecutionStartTime   ExecutionEndTime
---------------------                ------------
 ------------------------- ------------------   ----------------
e1d2bad3-4567-8901-ae23-456c7c8901be AWS-UpdateWindowsAmi
 Cancelled                 4/16/2019 5:37:04 AM 4/16/2019 5:47:29 AM
61234567-a7f8-90e1-2b34-567b8bf9012c Fixed-UpdateAmi
 Cancelled                 4/16/2019 5:33:04 AM 4/16/2019 5:40:15 AM
91234d56-7e89-0ac1-2aee-34ea5d6a7c89 AWS-UpdateWindowsAmi                    Failed
                    4/16/2019 5:22:46 AM 4/16/2019 5:27:29 AM
```

- API-Details finden Sie DescribeAutomationExecutions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAutomationStepExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAutomationStepExecution`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen über alle aktiven und beendeten Schrittausführungen in einem Automation-Workflow angezeigt.

```
Get-SSMAutomationStepExecution -AutomationExecutionId e1d2bad3-4567-8901-
ae23-456c7c8901be | Select-Object StepName, Action, StepStatus
```

Ausgabe:

```
StepName                  Action                  StepStatus
--------                  ------                  ----------
LaunchInstance            aws:runInstances        Success
OSCompatibilityCheck      aws:runCommand          Success
RunPreUpdateScript        aws:runCommand          Success
UpdateEC2Config           aws:runCommand          Cancelled
UpdateSSMAgent            aws:runCommand          Pending
UpdateAWSPVDriver         aws:runCommand          Pending
UpdateAWSEnaNetworkDriver aws:runCommand          Pending
UpdateAWSNVMe             aws:runCommand          Pending
InstallWindowsUpdates     aws:runCommand          Pending
RunPostUpdateScript       aws:runCommand          Pending
RunSysprepGeneralize      aws:runCommand          Pending
StopInstance              aws:changeInstanceState Pending
CreateImage               aws:createImage         Pending
TerminateInstance         aws:changeInstanceState Pending
```

- API-Details finden Sie DescribeAutomationStepExecutions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMAvailablePatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMAvailablePatch`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel ruft alle verfügbaren Patches für Windows Server 2012 ab, die den MSRC-Schweregrad „Kritisch“ aufweisen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="PRODUCT";Values=@("WindowsServer2012")}
$filter2 = @{Key="MSRC_SEVERITY";Values=@("Critical")}

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Ausgabe:

```
Classification : SecurityUpdates
ContentUrl     : https://support.microsoft.com/en-us/kb/2727528
```

```
Description    : A security issue has been identified that could allow an
 unauthenticated remote attacker to compromise your system and gain control
                 over it. You can help protect your system by installing this update
 from Microsoft. After you install this update, you may have to
                 restart your system.
Id             : 1eb507be-2040-4eeb-803d-abc55700b715
KbNumber       : KB2727528
Language       : All
MsrcNumber     : MS12-072
MsrcSeverity   : Critical
Product        : WindowsServer2012
ProductFamily  : Windows
ReleaseDate    : 11/13/2012 6:00:00 PM
Title          : Security Update for Windows Server 2012 (KB2727528)
Vendor         : Microsoft
...
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie jeden Filter verwenden, New-Object um ihn zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "PRODUCT"
$filter1.Values = "WindowsServer2012"
$filter2 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter2.Key = "MSRC_SEVERITY"
$filter2.Values = "Critical"

Get-SSMAvailablePatch -Filter $filter1,$filter2
```

Beispiel 3: In diesem Beispiel werden alle Updates abgerufen, die in den letzten 20 Tagen veröffentlicht wurden und für Produkte gelten, die 2019 entsprechen WindowsServer

```
Get-SSMAvailablePatch | Where-Object ReleaseDate -ge (Get-Date).AddDays(-20) |
 Where-Object Product -eq "WindowsServer2019" | Select-Object ReleaseDate, Product,
 Title
```

Ausgabe:

```
ReleaseDate         Product           Title
-----------         -------           -----
4/9/2019 5:00:12 PM WindowsServer2019 2019-04 Security Update for Adobe Flash Player
 for Windows Server 2019 for x64-based Systems (KB4493478)
```

```
4/9/2019 5:00:06 PM WindowsServer2019 2019-04 Cumulative Update for Windows Server
 2019 for x64-based Systems (KB4493509)
4/2/2019 5:00:06 PM WindowsServer2019 2019-03 Servicing Stack Update for Windows
 Server 2019 for x64-based Systems (KB4493510)
```

- API-Details finden Sie DescribeAvailablePatches in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommand`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle angeforderten Befehle aufgeführt.

```
Get-SSMCommand
```

Ausgabe:

```
CommandId          : 4b75a163-d39a-4d97-87c9-98ae52c6be35
Comment            : Apply association with id at update time: 4cc73e42-
d5ae-4879-84f8-57e09c0efcd0
CompletedCount     : 1
DocumentName       : AWS-RefreshAssociation
ErrorCount         : 0
ExpiresAfter       : 2/24/2017 3:19:08 AM
InstanceIds        : {i-0cb2b964d3e14fd9f}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[associationIds,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 2/24/2017 3:18:08 AM
ServiceRole        :
Status             : Success
StatusDetails      : Success
TargetCount        : 1
Targets            : {}
```

Beispiel 2: In diesem Beispiel wird der Status eines bestimmten Befehls abgerufen.

```
Get-SSMCommand -CommandId "4b75a163-d39a-4d97-87c9-98ae52c6be35"
```

Beispiel 3: In diesem Beispiel werden alle SSM-Befehle abgerufen, die nach dem 01.04.2019 aufgerufen wurden T00:00:00Z

```
Get-SSMCommand -Filter @{Key="InvokedAfter";Value="2019-04-01T00:00:00Z"} | Select-
Object CommandId, DocumentName, Status, RequestedDateTime | Sort-Object -Property
 RequestedDateTime -Descending
```

Ausgabe:

```
CommandId                            DocumentName               Status
 RequestedDateTime
---------                            ------------               ------
 -----------------
edb1b23e-456a-7adb-aef8-90e-012ac34f AWS-RunPowerShellScript    Cancelled 4/16/2019
 5:45:23 AM
1a2dc3fb-4567-890d-a1ad-234b5d6bc7d9 AWS-ConfigureAWSPackage    Success   4/6/2019
 9:19:42 AM
12c3456c-7e90-4f12-1232-1234f5b67893 KT-Retrieve-Cloud-Type-Win Failed    4/2/2019
 4:13:07 AM
fe123b45-240c-4123-a2b3-234bdd567ecf AWS-RunInspecChecks        Failed    4/1/2019
 2:27:31 PM
1eb23aa4-567d-4123-12a3-4c1c2ab34561 AWS-RunPowerShellScript    Success   4/1/2019
 1:05:55 PM
1c2f3bb4-ee12-4bc1-1a23-12345eea123e AWS-RunInspecChecks        Failed    4/1/2019
 11:13:09 AM
```

- API-Details finden Sie ListCommands in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMCommandInvocation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommandInvocation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Aufrufe eines Befehls aufgeführt.

```
Get-SSMCommandInvocation -CommandId "b8eac879-0541-439d-94ec-47a80d554f44" -Detail
 $true
```

Ausgabe:

```
CommandId          : b8eac879-0541-439d-94ec-47a80d554f44
CommandPlugins     : {aws:runShellScript}
Comment            : IP config
DocumentName       : AWS-RunShellScript
InstanceId         : i-0cb2b964d3e14fd9f
InstanceName       :
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
RequestedDateTime  : 2/22/2017 8:13:16 PM
ServiceRole        :
StandardErrorUrl   :
StandardOutputUrl  :
Status             : Success
StatusDetails      : Success
TraceOutput        :
```

Beispiel 2: In diesem Beispiel wird die Befehls-ID CommandPlugins e1eb2e3c-ed4c-5123-45c1-234f5612345f aufgeführt

```
Get-SSMCommandInvocation -CommandId e1eb2e3c-ed4c-5123-45c1-234f5612345f -Detail:
$true | Select-Object -ExpandProperty CommandPlugins
```

Ausgabe:

```
Name                   : aws:runPowerShellScript
Output                 : Completed 17.7 KiB/17.7 KiB (40.1 KiB/s) with 1 file(s)
 remainingdownload: s3://dd-aess-r-ctmer/KUMO.png to ..\..\programdata\KUMO.png
                          kumo available

OutputS3BucketName     :
OutputS3KeyPrefix      :
OutputS3Region         : eu-west-1
ResponseCode           : 0
ResponseFinishDateTime : 4/3/2019 11:53:23 AM
ResponseStartDateTime  : 4/3/2019 11:53:21 AM
StandardErrorUrl       :
StandardOutputUrl      :
Status                 : Success
```

```
StatusDetails          : Success
```

- API-Details finden ListCommandInvocations Sie AWS -Tools für PowerShell in der Cmdlet-Referenz (V5).

`Get-SSMCommandInvocationDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMCommandInvocationDetail`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details eines Befehls angezeigt, der auf einer Instance ausgeführt wurde.

```
Get-SSMCommandInvocationDetail -InstanceId "i-0cb2b964d3e14fd9f" -CommandId
 "b8eac879-0541-439d-94ec-47a80d554f44"
```

Ausgabe:

```
CommandId              : b8eac879-0541-439d-94ec-47a80d554f44
Comment                : IP config
DocumentName           : AWS-RunShellScript
ExecutionElapsedTime   : PT0.004S
ExecutionEndDateTime   : 2017-02-22T20:13:16.651Z
ExecutionStartDateTime : 2017-02-22T20:13:16.651Z
InstanceId             : i-0cb2b964d3e14fd9f
PluginName             : aws:runShellScript
ResponseCode           : 0
StandardErrorContent   :
StandardErrorUrl       :
StandardOutputContent  :
StandardOutputUrl      :
Status                 : Success
StatusDetails          : Success
```

- API-Details finden Sie GetCommandInvocation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMComplianceItemList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMComplianceItemList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Liste der Compliance-Elemente für die angegebene Ressourcen-ID und den angegebenen Ressourcentyp aufgeführt, wobei nach dem Compliance-Typ „Association“ gefiltert wird

```
Get-SSMComplianceItemList -ResourceId i-1a2caf345f67d0dc2 -ResourceType
 ManagedInstance -Filter @{Key="ComplianceType";Values="Association"}
```

Ausgabe:

```
ComplianceType   : Association
Details          : {[DocumentName, AWS-GatherSoftwareInventory], [DocumentVersion,
 1]}
ExecutionSummary : Amazon.SimpleSystemsManagement.Model.ComplianceExecutionSummary
Id               : 123a45a1-c234-1234-1245-67891236db4e
ResourceId       : i-1a2caf345f67d0dc2
ResourceType     : ManagedInstance
Severity         : UNSPECIFIED
Status           : COMPLIANT
Title            :
```

- API-Details finden Sie ListComplianceItems in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMComplianceSummaryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMComplianceSummaryList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel gibt eine Sammelzahl konformer und nicht konformer Ressourcen für alle Konformitätstypen zurück.

```
Get-SSMComplianceSummaryList
```

Ausgabe:

```
ComplianceType CompliantSummary
 NonCompliantSummary
```

```
-------------- ----------------
 -------------------
FleetTotal     Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Association    Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Custom:InSpec  Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
Patch          Amazon.SimpleSystemsManagement.Model.CompliantSummary
 Amazon.SimpleSystemsManagement.Model.NonCompliantSummary
```

- API-Details finden Sie ListComplianceSummaries in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMConnectionStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMConnectionStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Session-Manager-Verbindungsstatus für eine Instance abgerufen, um festzustellen, ob sie verbunden und bereit ist, Session-Manager-Verbindungen zu empfangen.

```
Get-SSMConnectionStatus -Target i-0a1caf234f12d3dc4
```

Ausgabe:

```
Status    Target
------    ------
Connected i-0a1caf234f12d3dc4
```

- API-Details finden Sie GetConnectionStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDefaultPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDefaultPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Standard-Patch-Baseline angezeigt.

```
Get-SSMDefaultPatchBaseline
```

Ausgabe:

```
arn:aws:ssm:us-west-2:123456789012:patchbaseline/pb-04fb4ae6142167966
```

- API-Details finden Sie GetDefaultPatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDeployablePatchSnapshotForInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDeployablePatchSnapshotForInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der aktuelle Snapshot für die von einer Instance verwendete Patch-Baseline angezeigt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance ausgeführt werden. Um sicherzustellen, dass die Instance-Anmeldedaten verwendet werden, übergibt das Beispiel ein **`Amazon.Runtime.InstanceProfileAWSCredentials`**-Objekt an den Anmeldeinformationen-Parameter.

```
$credentials = [Amazon.Runtime.InstanceProfileAWSCredentials]::new()
Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f" -Credentials $credentials
```

Ausgabe:

```
InstanceId          SnapshotDownloadUrl
----------          -------------------
i-0cb2b964d3e14fd9f https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...1692/4681775b-098f-4435...
```

Beispiel 2: Dieses Beispiel zeigt, wie man das volle SnapshotDownloadUrl bekommt. Dieser Befehl muss von der Instance aus mit den Anmeldeinformationen der Instance

ausgeführt werden. Um sicherzustellen, dass die Instanzanmeldeinformationen verwendet werden, konfiguriert das Beispiel die PowerShell Sitzung so, dass sie ein **`Amazon.Runtime.InstanceProfileAWSCredentials`** Objekt verwendet.

```
Set-AWSCredential -Credential
 ([Amazon.Runtime.InstanceProfileAWSCredentials]::new())
(Get-SSMDeployablePatchSnapshotForInstance -SnapshotId "4681775b-098f-4435-
a956-0ef33373ac11" -InstanceId "i-0cb2b964d3e14fd9f").SnapshotDownloadUrl
```

Ausgabe:

```
https://patch-baseline-snapshot-us-west-2.s3-us-
west-2.amazonaws.com/853d0d3db0f0cafe...
```

- API-Details finden Sie GetDeployablePatchSnapshotForInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocument`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Inhalt eines Dokuments zurückgegeben.

```
Get-SSMDocument -Name "RunShellScript"
```

Ausgabe:

```
Content
-------
{...
```

Beispiel 2: In diesem Beispiel wird der vollständige Inhalt eines Dokuments angezeigt.

```
(Get-SSMDocument -Name "RunShellScript").Content
{
   "schemaVersion":"2.0",
   "description":"Run an updated script",
```

```
    "parameters":{
       "commands":{
          "type":"StringList",
          "description":"(Required) Specify a shell script or a command to run.",
          "minItems":1,
          "displayType":"textarea"
       }
    },
    "mainSteps":[
       {
          "action":"aws:runShellScript",
          "name":"runShellScript",
          "inputs":{
             "commands":"{{ commands }}"
          }
       },
       {
          "action":"aws:runPowerShellScript",
          "name":"runPowerShellScript",
          "inputs":{
             "commands":"{{ commands }}"
          }
       }
    ]
}
```

- API-Details finden Sie GetDocument in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDocumentDescription`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentDescription`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu einem Dokument zurückgegeben.

```
Get-SSMDocumentDescription -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate     : 2/24/2017 5:25:13 AM
DefaultVersion  : 1
```

```
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : f775e5df4904c6fa46686c4722fae9de1950dace25cd9608ff8d622046b68d9b
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 123456789012
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Active
```

- API-Details finden Sie DescribeDocument in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDocumentList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentList`

Tools für PowerShell V5

Beispiel 1: Listet alle Konfigurationsdokumente in Ihrem Konto auf.

```
Get-SSMDocumentList
```

Ausgabe:

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ApplyPatchBaseline
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2

DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureAWSPackage
Owner           : Amazon
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 2.0
```

```
DocumentType    : Command
DocumentVersion : 1
Name            : AWS-ConfigureCloudWatch
Owner           : Amazon
PlatformTypes   : {Windows}
SchemaVersion   : 1.2
...
```

Beispiel 2: In diesem Beispiel werden alle Automatisierungsdokumente abgerufen, deren Name mit „Platform“ übereinstimmt

```
Get-SSMDocumentList -DocumentFilterList @{Key="DocumentType";Value="Automation"} |
 Where-Object Name -Match "Platform"
```

Ausgabe:

```
DocumentFormat  : JSON
DocumentType    : Automation
DocumentVersion : 7
Name            : KT-Get-Platform
Owner           : 987654123456
PlatformTypes   : {Windows, Linux}
SchemaVersion   : 0.3
Tags            : {}
TargetType      :
VersionName     :
```

- API-Details finden Sie ListDocuments in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDocumentPermission`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentPermission`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "RunShellScript"
```

Ausgabe:

```
CreatedDate          DocumentVersion IsDefaultVersion Name
-----------          --------------- ---------------- ----
2/24/2017 5:25:13 AM 1               True             RunShellScript
```

- API-Details finden Sie DescribeDocumentPermission in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMDocumentVersionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMDocumentVersionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Versionen eines Dokuments aufgeführt.

```
Get-SSMDocumentVersionList -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
CreatedDate       : 6/1/2021 5:19:10 PM
DocumentFormat    : JSON
DocumentVersion   : 1
IsDefaultVersion  : True
Name              : AWS-UpdateSSMAgent
Status            : Active
```

- API-Details finden Sie ListDocumentVersions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMEffectiveInstanceAssociationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMEffectiveInstanceAssociationList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel beschreibt die effektiven Zuordnungen für eine Instance.

```
Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -MaxResult
 5
```

Ausgabe:

```
AssociationId                        Content
-------------                        -------
d8617c07-2079-4c18-9847-1655fc2698b0 {...
```

Beispiel 2: In diesem Beispiel wird der Inhalt der effektiven Verknüpfungen für eine Instance angezeigt.

```
(Get-SSMEffectiveInstanceAssociationList -InstanceId "i-0000293ffd8c57862" -
MaxResult 5).Content
```

Ausgabe:

```
{
    "schemaVersion": "1.2",
    "description": "Update the Amazon SSM Agent to the latest version or specified
 version.",
    "parameters": {
        "version": {
            "default": "",
            "description": "(Optional) A specific version of the Amazon SSM Agent to
 install. If not specified, the agen
t will be updated to the latest version.",
            "type": "String"
        },
        "allowDowngrade": {
            "default": "false",
            "description": "(Optional) Allow the Amazon SSM Agent service to be
 downgraded to an earlier version. If set
 to false, the service can be upgraded to newer versions only (default). If set to
 true, specify the earlier version.",
            "type": "String",
            "allowedValues": [
                "true",
                "false"
            ]
        }
```

```
    },
    "runtimeConfig": {
        "aws:updateSsmAgent": {
            "properties": [
                {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/
ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}",
                "targetVersion": "{{ version }}"
                }
            ]
        }
    }
}
```

- API-Details finden Sie DescribeEffectiveInstanceAssociations in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMEffectivePatchesForPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMEffectivePatchesForPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 aufgeführt.

```
Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1
```

Ausgabe:

```
Patch                                       PatchStatus
-----                                       -----------
Amazon.SimpleSystemsManagement.Model.Patch
 Amazon.SimpleSystemsManagement.Model.PatchStatus
```

Beispiel 2: In diesem Beispiel wird der Patchstatus für alle Patch-Baselines mit einer maximalen Ergebnisliste von 1 angezeigt.

```
(Get-SSMEffectivePatchesForPatchBaseline -BaselineId "pb-0a2f1059b670ebd31" -
MaxResult 1).PatchStatus
```

Ausgabe:

```
ApprovalDate          DeploymentStatus
------------          ----------------
12/21/2010 6:00:00 PM APPROVED
```

- API-Details finden Sie DescribeEffectivePatchesForPatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInstanceAssociationsStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstanceAssociationsStatus`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt Details zu den Zuordnungen für eine Instance.

```
Get-SSMInstanceAssociationsStatus -InstanceId "i-0000293ffd8c57862"
```

Ausgabe:

```
AssociationId    : d8617c07-2079-4c18-9847-1655fc2698b0
DetailedStatus   : Pending
DocumentVersion  : 1
ErrorCode        :
ExecutionDate    : 2/20/2015 8:31:11 AM
ExecutionSummary : temp_status_change
InstanceId       : i-0000293ffd8c57862
Name             : AWS-UpdateSSMAgent
OutputUrl        :
Status           : Pending
```

Beispiel 2: In diesem Beispiel wird der Status der Instance-Zuordnung für die angegebene Instance-ID überprüft und außerdem der Ausführungsstatus dieser Zuordnungen angezeigt

```
Get-SSMInstanceAssociationsStatus -InstanceId i-012e3cb4df567e8aa | ForEach-Object
 {Get-SSMAssociationExecution -AssociationId .AssociationId}
```

Ausgabe:

```
AssociationId         : 512a34a5-c678-1234-1234-12345678db9e
AssociationVersion    : 2
CreatedTime           : 3/2/2019 8:53:29 AM
DetailedStatus        :
ExecutionId           : 512a34a5-c678-1234-1234-12345678db9e
LastExecutionDate     : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=9}
Status                : Success
```

- API-Details finden Sie DescribeInstanceAssociationsStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInstanceInformation`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstanceInformation`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt Details zu jeder Ihrer Instances.

```
Get-SSMInstanceInformation
```

Ausgabe:

```
ActivationId                          :
AgentVersion                          : 2.0.672.0
AssociationOverview                   :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                     : Success
ComputerName                          : ip-172-31-44-222.us-west-2.compute.internal
IamRole                               :
InstanceId                            : i-0cb2b964d3e14fd9f
IPAddress                             : 172.31.44.222
IsLatestVersion                       : True
LastAssociationExecutionDate          : 2/24/2017 3:18:09 AM
LastPingDateTime                      : 2/24/2017 3:35:03 AM
LastSuccessfulAssociationExecutionDate : 2/24/2017 3:18:09 AM
Name                                  :
PingStatus                            : ConnectionLost
```

```
PlatformName                            : Amazon Linux AMI
PlatformType                            : Linux
PlatformVersion                         : 2016.09
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 2: Dieses Beispiel zeigt, wie der -Filter-Parameter verwendet wird, um Ergebnisse nur auf die AWS Systems Manager-Instanzen in der Region **`us-east-1`** mit einem **`AgentVersion`** von zu filtern. **`2.2.800.0`** Eine Liste gültiger -Filter-Schlüsselwerte finden Sie im InstanceInformation API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformation.html#systemsmanager -Type-InstanceInformation-ActivationId).

```
$Filters = @{
    Key="AgentVersion"
    Values="2.2.800.0"
}
Get-SSMInstanceInformation -Region us-east-1 -Filter $Filters
```

Ausgabe:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.01
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

ActivationId                            :
```

```
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.02
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 3: Dieses Beispiel zeigt, wie der InstanceInformationFilterList Parameter - verwendet wird, um Ergebnisse nur auf die AWS Systems Manager-Instanzen in **`PlatformTypes`** der Region **`us-east-1`** mit **`Windows`** oder zu filtern**`Linux`**. Eine Liste gültiger InstanceInformationFilterList Schlüsselwerte finden Sie im InstanceInformationFilter API-Referenzthema (https://docs.aws.amazon.com/systems-manager/latest/APIReference/API_InstanceInformationFilter.html).

```
$Filters = @{
   Key="PlatformTypes"
   ValueSet=("Windows","Linux")
}
Get-SSMInstanceInformation -Region us-east-1 -InstanceInformationFilterList $Filters
```

Ausgabe:

```
ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
```

```
InstanceId                              : i-EXAMPLEb0792d98ce
IPAddress                               : 10.0.0.27
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:02:50 AM
LastPingDateTime                        : 8/16/2018 7:40:27 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:02:50 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Ubuntu Server 18.04 LTS
PlatformType                            : Linux
PlatformVersion                         : 18.04
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance

ActivationId                            :
AgentVersion                            : 2.2.800.0
AssociationOverview                     :
 Amazon.SimpleSystemsManagement.Model.InstanceAggregatedAssociationOverview
AssociationStatus                       : Success
ComputerName                            : EXAMPLE-EXAMPLE.WORKGROUP
IamRole                                 :
InstanceId                              : i-EXAMPLEac7501d023
IPAddress                               : 10.0.0.100
IsLatestVersion                         : False
LastAssociationExecutionDate            : 8/16/2018 12:00:20 AM
LastPingDateTime                        : 8/16/2018 7:40:35 PM
LastSuccessfulAssociationExecutionDate : 8/16/2018 12:00:20 AM
Name                                    :
PingStatus                              : Online
PlatformName                            : Microsoft Windows Server 2016 Datacenter
PlatformType                            : Windows
PlatformVersion                         : 10.0.14393
RegistrationDate                        : 1/1/0001 12:00:00 AM
ResourceType                            : EC2Instance
```

Beispiel 4: In diesem Beispiel werden von SSM verwaltete Instanzen und Exporte InstanceId LastPingDateTime sowie PlatformName in eine CSV-Datei aufgeführt. PingStatus

```
Get-SSMInstanceInformation | Select-Object InstanceId, PingStatus, LastPingDateTime,
 PlatformName | Export-Csv Instance-details.csv -NoTypeInformation
```

- API-Details finden Sie DescribeInstanceInformation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInstancePatch`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details zur Patch-Konformität für eine Instance abgerufen.

```
Get-SSMInstancePatch -InstanceId "i-08ee91c0b17045407"
```

- API-Details finden Sie DescribeInstancePatches in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInstancePatchState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatchState`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Status der Patch-Zusammenfassung für eine Instance abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407"
```

Beispiel 2: In diesem Beispiel werden die Status der Patch-Zusammenfassung für zwei Instances abgerufen.

```
Get-SSMInstancePatchState -InstanceId "i-08ee91c0b17045407","i-09a618aec652973a9"
```

- API-Details finden Sie DescribeInstancePatchStates in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInstancePatchStatesForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInstancePatchStatesForPatchGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden für eine Patch-Gruppe eine Übersicht über den Patch-Zustand auf der Ebene einzelner Instances abgerufen.

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Production"
```

- API-Details finden Sie DescribeInstancePatchStatesForPatchGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInventory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die benutzerdefinierten Metadaten für Ihr Inventar abgerufen.

```
Get-SSMInventory
```

Ausgabe:

```
Data
  Id
----
  --
{[AWS:InstanceInformation,
 Amazon.SimpleSystemsManagement.Model.InventoryResultItem]} i-0cb2b964d3e14fd9f
```

- API-Details finden Sie GetInventory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInventoryEntriesList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventoryEntriesList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle benutzerdefinierten Inventareinträge für eine Instance aufgeführt.

```
Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo"
```

Ausgabe:

```
CaptureTime   : 2016-08-22T10:01:01Z
Entries       :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,System.String]}
InstanceId    : i-0cb2b964d3e14fd9f
NextToken     :
SchemaVersion : 1.0
TypeName      : Custom:RackInfo
```

Beispiel 2: In diesem Beispiel werden die Details aufgeführt.

```
(Get-SSMInventoryEntriesList -InstanceId "i-0cb2b964d3e14fd9f" -TypeName
 "Custom:RackInfo").Entries
```

Ausgabe:

```
Key          Value
---          -----
RackLocation Bay B/Row C/Rack D/Shelf E
```

- API-Details finden Sie ListInventoryEntries in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInventoryEntryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventoryEntryList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden **`AWS:Network`**-Typinventareinträge für die Instance abgerufen.

```
Get-SSMInventoryEntryList -InstanceId mi-088dcb0ecea37b076 -TypeName AWS:Network |
 Select-Object -ExpandProperty Entries
```

Ausgabe:

```
Key        Value
---        -----
DHCPServer 172.31.11.2
DNSServer  172.31.0.1
Gateway    172.31.11.2
IPV4       172.31.11.222
IPV6       fe12::3456:7da8:901a:12a3
MacAddress 1A:23:4E:5B:FB:67
Name       Amazon Elastic Network Adapter
SubnetMask 255.255.240.0
```

- API-Details finden Sie Get-SSMInventoryEntryList in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMInventorySchema`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMInventorySchema`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Liste von Inventartypnamen für das Konto zurückgegeben.

```
Get-SSMInventorySchema
```

- API-Details finden Sie GetInventorySchema in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMLatestEC2Image`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMLatestEC2Image`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die neuesten Windows-AMIs aufgeführt.

```
PS Get-SSMLatestEC2Image -Path ami-windows-latest
```

Ausgabe:

```
Name                                                                Value
----                                                                -----
Windows_Server-2008-R2_SP1-English-64Bit-SQL_2012_SP4_Express
 ami-0e5ddd288daff4fab
Windows_Server-2012-R2_RTM-Chinese_Simplified-64Bit-Base
 ami-0c5ea64e6bec1cb50
Windows_Server-2012-R2_RTM-Chinese_Traditional-64Bit-Base
 ami-09775eff0bf8c113d
Windows_Server-2012-R2_RTM-Dutch-64Bit-Base
 ami-025064b67e28cf5df
...
```

Beispiel 2: In diesem Beispiel wird die AMI-ID eines bestimmten Amazon-Linux-Images für die Region us-west-2 abgerufen.

```
PS Get-SSMLatestEC2Image -Path ami-amazon-linux-latest -ImageName amzn-ami-hvm-
x86_64-ebs -Region us-west-2
```

Ausgabe:

```
ami-09b92cd132204c704
```

Beispiel 3: In diesem Beispiel werden alle neuesten Windows-AMIs aufgeführt, die dem angegebenen Platzhalterausdruck entsprechen.

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *Windows*2019*English*
```

Ausgabe:

```
Name                                                Value
----                                                -----
Windows_Server-2019-English-Full-SQL_2017_Web       ami-085e9d27da5b73a42
Windows_Server-2019-English-STIG-Core               ami-0bfd85c29148c7f80
Windows_Server-2019-English-Full-SQL_2019_Web       ami-02099560d7fb11f20
Windows_Server-2019-English-Full-SQL_2016_SP2_Standard   ami-0d7ae2d81c07bd598
...
```

- API-Details finden Sie Get-SSMLatestEC2Image in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`Get-SSMMaintenanceWindow`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zu einem Wartungsfenster abgerufen.

```
Get-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
AllowUnassociatedTargets : False
CreatedDate              : 2/20/2017 6:14:05 PM
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
ModifiedDate             : 2/20/2017 6:14:05 PM
Name                     : TestMaintWin
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

- API-Details finden Sie GetMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecution`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet Informationen über eine Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecution -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
```

```
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
TaskIds           : {ac0c6ae1-daa3-4a89-832e-d384503b6586}
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- API-Details finden Sie GetMaintenanceWindowExecution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowExecutionList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d"
```

Ausgabe:

```
EndTime           : 2/20/2017 6:30:17 PM
StartTime         : 2/20/2017 6:30:16 PM
Status            : FAILED
StatusDetails     : One or more tasks in the orchestration failed.
WindowExecutionId : 6f3215cf-4101-4fa0-9b7b-9523269599c7
WindowId          : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster vor einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedBefore";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

Beispiel 3: In diesem Beispiel werden alle Ausführungen für ein Wartungsfenster nach einem bestimmten Datum aufgeführt.

```
$option1 = @{Key="ExecutedAfter";Values=@("2016-11-04T05:00:00Z")}
Get-SSMMaintenanceWindowExecutionList -WindowId "mw-03eb9db42890fb82d" -Filter
 $option1
```

- API-Details finden Sie DescribeMaintenanceWindowExecutions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowExecutionTask`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTask`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Informationen zu einer Aufgabe aufgeführt, die Teil einer Ausführung im Rahmen eines Wartungsfensters war.

```
Get-SSMMaintenanceWindowExecutionTask -TaskId "ac0c6ae1-daa3-4a89-832e-d384503b6586"
 -WindowExecutionId "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
MaxConcurrency    : 1
MaxErrors         : 1
Priority          : 10
ServiceRole       : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The maximum error count was exceeded.
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskParameters    :
 {Amazon.Runtime.Internal.Util.AlwaysSendDictionary`2[System.String,Amazon.SimpleSystemsMana
                    meterValueExpression]}
Type              : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- API-Details finden Sie GetMaintenanceWindowExecutionTask in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTaskInvocationList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet die Aufrufe einer Aufgabenausführungen als Teil einer Wartungsfenster-Ausführung auf.

```
Get-SSMMaintenanceWindowExecutionTaskInvocationList -TaskId "ac0c6ae1-
daa3-4a89-832e-d384503b6586" -WindowExecutionId "518d5565-5969-4cca-8f0e-
da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:34 PM
ExecutionId       :
InvocationId      : e274b6e1-fe56-4e32-bd2a-8073c6381d8b
OwnerInformation  :
Parameters        : {"documentName":"AWS-RunShellScript","instanceIds":
["i-0000293ffd8c57862"],"parameters":{"commands":["df"]},"maxConcurrency":"1",
                    "maxErrors":"1"}
StartTime         : 2/21/2017 4:00:34 PM
Status            : FAILED
StatusDetails     : The instance IDs list contains an invalid entry.
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
WindowTargetId    :
```

- API-Details finden Sie DescribeMaintenanceWindowExecutionTaskInvocations in der AWS - Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowExecutionTaskList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowExecutionTaskList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet die Aufgaben auf, die mit der Ausführung eines Wartungsfensters verbunden sind.

```
Get-SSMMaintenanceWindowExecutionTaskList -WindowExecutionId
 "518d5565-5969-4cca-8f0e-da3b2a638355"
```

Ausgabe:

```
EndTime           : 2/21/2017 4:00:35 PM
StartTime         : 2/21/2017 4:00:34 PM
Status            : SUCCESS
TaskArn           : AWS-RunShellScript
TaskExecutionId   : ac0c6ae1-daa3-4a89-832e-d384503b6586
TaskType          : RUN_COMMAND
WindowExecutionId : 518d5565-5969-4cca-8f0e-da3b2a638355
```

- API-Details finden Sie DescribeMaintenanceWindowExecutionTasks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Wartungsfenster in Ihrem Konto aufgeführt.

```
Get-SSMMaintenanceWindowList
```

Ausgabe:

```
Cutoff   : 1
Duration : 4
Enabled  : True
Name     : My-First-Maintenance-Window
WindowId : mw-06d59c1a07c022145
```

- API-Details finden Sie DescribeMaintenanceWindows in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowTarget`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowTarget`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Ziele für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTarget -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
OwnerInformation : Single instance
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : 350d44e6-28cc-44e2-951f-4b2c985838f6

OwnerInformation : Two instances in a list
ResourceType     : INSTANCE
Targets          : {InstanceIds}
WindowId         : mw-06cf17cbefcb4bf4f
WindowTargetId   : e078a987-2866-47be-bedd-d9cf49177d3a
```

- API-Details finden Sie DescribeMaintenanceWindowTargets in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMMaintenanceWindowTaskList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMMaintenanceWindowTaskList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden alle Aufgaben für ein Wartungsfenster aufgeführt.

```
Get-SSMMaintenanceWindowTaskList -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
LoggingInfo    :
MaxConcurrency : 1
MaxErrors      : 1
Priority       : 10
ServiceRoleArn : arn:aws:iam::123456789012:role/MaintenanceWindowsRole
Targets        : {InstanceIds}
TaskArn        : AWS-RunShellScript
TaskParameters : {[commands,
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression]}
```

```
Type           : RUN_COMMAND
WindowId       : mw-06cf17cbefcb4bf4f
WindowTaskId   : a23e338d-ff30-4398-8aa3-09cd052ebf17
```

- API-Details finden Sie DescribeMaintenanceWindowTasks in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMParameterHistory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterHistory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Werteverlauf für einen Parameter aufgeführt.

```
Get-SSMParameterHistory -Name "Welcome"
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:55:25 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
Value            : helloWorld
```

- API-Details finden Sie GetParameterHistory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMParameterList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterList`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet alle Parameter auf.

```
Get-SSMParameterList
```

Ausgabe:

```
Description      :
KeyId            :
LastModifiedDate : 3/3/2017 6:58:23 PM
LastModifiedUser : arn:aws:iam::123456789012:user/admin
Name             : Welcome
Type             : String
```

- API-Details finden Sie DescribeParameters in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMParameterValue`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMParameterValue`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Werte für einen Parameter aufgeführt.

```
Get-SSMParameterValue -Name "Welcome"
```

Ausgabe:

```
InvalidParameters Parameters
----------------- ----------
{}                {Welcome}
```

Beispiel 2: In diesem Beispiel werden die Details des Werts aufgeführt.

```
(Get-SSMParameterValue -Name "Welcome").Parameters
```

Ausgabe:

```
Name    Type   Value
----    ----   -----
Welcome String Good day, Sunshine!
```

- API-Details finden Sie GetParameters in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Get-SSMPatchBaseline

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaseline`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet alle Patch-Baselines auf.

```
Get-SSMPatchBaseline
```

Ausgabe:

```
BaselineDescription                                                BaselineId
                                                       BaselineName
-------------------                                                ----------
                                                       ------------
Default Patch Baseline Provided by AWS.                            arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultP...
Baseline containing all updates approved for production systems pb-045f10b4f382baeda
                                                       Production-B...
Baseline containing all updates approved for production systems pb-0a2f1059b670ebd31
                                                       Production-B...
```

Beispiel 2: Dieses Beispiel listet alle Patch-Baselines auf, die von bereitgestellt werden. AWS Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("AWS")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 3: In diesem Beispiel werden alle Patch-Baselines mit Ihnen als Eigentümer aufgeführt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$filter1 = @{Key="OWNER";Values=@("Self")}
```

Ausgabe:

```
Get-SSMPatchBaseline -Filter $filter1
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie jedes Tag verwenden, New-Object um es zu erstellen.

```
$filter1 = New-Object Amazon.SimpleSystemsManagement.Model.PatchOrchestratorFilter
$filter1.Key = "OWNER"
$filter1.Values = "AWS"

Get-SSMPatchBaseline -Filter $filter1
```

Ausgabe:

```
BaselineDescription                     BaselineId
                          BaselineName             DefaultBaselin

                                                   e
-------------------                     ----------
                          ------------             --------------
Default Patch Baseline Provided by AWS. arn:aws:ssm:us-
west-2:123456789012:patchbaseline/pb-04fb4ae6142167966 AWS-DefaultPatchBaseline True
```

- API-Details finden Sie DescribePatchBaselines in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMPatchBaselineDetail`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaselineDetail`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details für eine Patch-Baseline angezeigt.

```
Get-SSMPatchBaselineDetail -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
```

```
ModifiedDate    : 3/3/2017 5:02:19 PM
Name            : Production-Baseline
PatchGroups     : {}
RejectedPatches : {}
```

- API-Details finden Sie GetPatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchBaselineForPatchGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Patch-Baseline für eine Patchgruppe angezeigt.

```
Get-SSMPatchBaselineForPatchGroup -PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API-Details finden Sie GetPatchBaselineForPatchGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Registrierungen der Patchgruppen aufgeführt.

```
Get-SSMPatchGroup
```

Ausgabe:

```
BaselineIdentity                                          PatchGroup
----------------                                          ----------
```

```
Amazon.SimpleSystemsManagement.Model.PatchBaselineIdentity Production
```

- API-Details finden Sie DescribePatchGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMPatchGroupState`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMPatchGroupState`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die allgemeine Zusammenfassung der Patch-Konformität für eine Patch-Gruppe abgerufen.

```
Get-SSMPatchGroupState -PatchGroup "Production"
```

Ausgabe:

```
Instances                          : 4
InstancesWithFailedPatches         : 1
InstancesWithInstalledOtherPatches : 4
InstancesWithInstalledPatches      : 3
InstancesWithMissingPatches        : 0
InstancesWithNotApplicablePatches  : 0
```

- API-Details finden Sie DescribePatchGroupState in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMResourceComplianceSummaryList`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMResourceComplianceSummaryList`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Zusammenfassung der Anzahl auf Ressourcenebene abgerufen. Die Zusammenfassung enthält Informationen über den Status „konform“ und „nicht konform“ sowie detaillierte Angaben zum Schweregrad von Produkten, die „Windows10“ entsprechen. Da der MaxResult Standardwert 100 ist, wenn der Parameter nicht angegeben ist und dieser Wert nicht gültig ist, wird der MaxResult Parameter hinzugefügt und der Wert auf 50 gesetzt.

```
$FilterValues = @{
  "Key"="Product"
        "Type"="EQUAL"
        "Values"="Windows10"
}
        Get-SSMResourceComplianceSummaryList -Filter $FilterValues -MaxResult 50
```

- API-Details finden Sie ListResourceComplianceSummaries in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-SSMResourceTag`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel listet die Tags für ein Wartungsfenster auf.

```
Get-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow"
```

Ausgabe:

```
Key   Value
---   -----
Stack Production
```

- API-Details finden Sie ListTagsForResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMActivation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine verwaltete Instance erstellt.

```
New-SSMActivation -DefaultInstanceName "MyWebServers" -IamRole "SSMAutomationRole" -
RegistrationLimit 10
```

Ausgabe:

```
ActivationCode       ActivationId
--------------       ------------
KWChhOxBTiwDcKE9BlKC 08e51e79-1e36-446c-8e63-9458569c1363
```

- API-Details finden Sie CreateActivation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-SSMAssociation

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mithilfe von Instance-IDs einer Instance zugeordnet.

```
New-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Associated
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : Associated with AWS-UpdateSSMAgent
Status.AdditionalInfo :
```

Beispiel 2: In diesem Beispiel wird mithilfe von Zielen ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
New-SSMAssociation -Name "AWS-UpdateSSMAgent" -Target $target
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
```

```
Date                   : 3/1/2017 6:22:21 PM
Status.Name            :
Status.Date            :
Status.Message         :
Status.AdditionalInfo  :
```

Beispiel 3: In diesem Beispiel wird mithilfe von Zielen und Parametern ein Konfigurationsdokument einer Instance zugeordnet.

```
$target = @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
$params = @{
  "action"="configure"
  "mode"="ec2"
  "optionalConfigurationSource"="ssm"
  "optionalConfigurationLocation"=""
  "optionalRestart"="yes"
}
New-SSMAssociation -Name "Configure-CloudWatch" -AssociationName "CWConfiguration" -
Target $target -Parameter $params
```

Ausgabe:

```
Name                   : Configure-CloudWatch
InstanceId             :
Date                   : 5/17/2018 3:17:44 PM
Status.Name            :
Status.Date            :
Status.Message         :
Status.AdditionalInfo  :
```

Beispiel 4: In diesem Beispiel wird eine Zuordnung mit allen Instances in der Region erstellt, mit**`AWS-GatherSoftwareInventory`**. Außerdem werden benutzerdefinierte Dateien und Registrierungsverzeichnisse in den zu erfassenden Parametern bereitgestellt

```
$params =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$params["windowsRegistry"] ='[{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon
\MachineImage","Recursive":false,"ValueNames":["AMIName"]}]'
$params["files"] = '[{"Path":"C:\Program Files","Pattern":
["*.exe"],"Recursive":true}, {"Path":"C:\ProgramData","Pattern":
["*.log"],"Recursive":true}]'
```

```
New-SSMAssociation -AssociationName new-in-mum -Name AWS-GatherSoftwareInventory
 -Target @{Key="instanceids";Values="*"} -Parameter $params -region ap-south-1 -
ScheduleExpression "rate(720 minutes)"
```

Ausgabe:

```
Name                  : AWS-GatherSoftwareInventory
InstanceId            :
Date                  : 6/9/2019 8:57:56 AM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API-Details finden Sie CreateAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-SSMAssociationFromBatch`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMAssociationFromBatch`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Konfigurationsdokument mehreren Instances zugeordnet. Die Ausgabe gibt gegebenenfalls eine Liste mit erfolgreichen und fehlgeschlagenen Vorgängen zurück.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
New-SSMAssociationFromBatch -Entry $option1,$option2
```

Ausgabe:

```
Failed  Successful
------  ----------
{}      {Amazon.SimpleSystemsManagement.Model.FailedCreateAssociation,
 Amazon.SimpleSystemsManagement.Model.FailedCreateAsso...
```

Beispiel 2: In diesem Beispiel werden alle Details eines erfolgreichen Vorgangs angezeigt.

```
$option1 = @{InstanceId="i-0cb2b964d3e14fd9f";Name=@("AWS-UpdateSSMAgent")}
$option2 = @{InstanceId="i-0000293ffd8c57862";Name=@("AWS-UpdateSSMAgent")}
(New-SSMAssociationFromBatch -Entry $option1,$option2).Successful
```

- API-Details finden Sie CreateAssociationBatch in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMDocument`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Dokument in Ihrem Konto erstellt. Das Dokument muss im JSON-Format vorliegen. Weitere Informationen zum Schreiben eines Konfigurationsdokuments finden Sie unter Konfigurationsdokument in der SSM-API-Referenz.

```
New-SSMDocument -Content (Get-Content -Raw "c:\temp\RunShellScript.json") -Name
 "RunShellScript" -DocumentType "Command"
```

Ausgabe:

```
CreatedDate     : 3/1/2017 1:21:33 AM
DefaultVersion  : 1
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 1
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 1
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Creating
```

- API-Details finden Sie CreateDocument in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-SSMMaintenanceWindow

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein neues Wartungsfenster mit dem angegebenen Namen erstellt, das an jedem Dienstag um 16 Uhr für 4 Stunden läuft, mit einer Frist von 1 Stunde, und das Ziele ohne Zuordnung zulässt.

```
New-SSMMaintenanceWindow -Name "MyMaintenanceWindow" -Duration 4 -Cutoff 1 -
AllowUnassociatedTarget $true -Schedule "cron(0 16 ? * TUE *)"
```

Ausgabe:

```
mw-03eb53e1ea7383998
```

- API-Details finden Sie CreateMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-SSMPatchBaseline

Das folgende Codebeispiel zeigt die Verwendung. `New-SSMPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline erstellt, die Patches sieben Tage nach der Veröffentlichung durch Microsoft für verwaltete Instances genehmigt, auf denen Windows Server 2019 in einer Produktionsumgebung ausgeführt wird.

```
$rule = New-Object Amazon.SimpleSystemsManagement.Model.PatchRule
$rule.ApproveAfterDays = 7

$ruleFilters = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilterGroup

$patchFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$patchFilter.Key="PRODUCT"
$patchFilter.Values="WindowsServer2019"

$severityFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
```

```
$severityFilter.Key="MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$severityFilter.Values.Add("Moderate")

$classificationFilter = New-Object Amazon.SimpleSystemsManagement.Model.PatchFilter
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add( "SecurityUpdates" )
$classificationFilter.Values.Add( "Updates" )
$classificationFilter.Values.Add( "UpdateRollups" )
$classificationFilter.Values.Add( "CriticalUpdates" )

$ruleFilters.PatchFilters.Add($severityFilter)
$ruleFilters.PatchFilters.Add($classificationFilter)
$ruleFilters.PatchFilters.Add($patchFilter)
$rule.PatchFilterGroup = $ruleFilters

New-SSMPatchBaseline -Name "Production-Baseline-Windows2019" -Description "Baseline
 containing all updates approved for production systems" -ApprovalRules_PatchRule
 $rule
```

Ausgabe:

```
pb-0z4z6221c4296b23z
```

- API-Details finden Sie CreatePatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-SSMDefaultPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMDefaultPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline als Standard-Patch-Baseline registriert.

```
Register-SSMDefaultPatchBaseline -BaselineId "pb-03da896ca3b68b639"
```

Ausgabe:

```
pb-03da896ca3b68b639
```

- API-Details finden Sie RegisterDefaultPatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMPatchBaselineForPatchGroup`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel registriert eine Patch-Baseline für eine Patch-Gruppe.

```
Register-SSMPatchBaselineForPatchGroup -BaselineId "pb-03da896ca3b68b639" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-03da896ca3b68b639 Production
```

- API-Details finden Sie RegisterPatchBaselineForPatchGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-SSMTargetWithMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMTargetWithMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="InstanceIds";Values=@("i-0000293ffd8c57862")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
d8e47760-23ed-46a5-9f28-927337725398
```

Beispiel 2: In diesem Beispiel werden mehrere Instances mit einem Wartungsfenster registriert.

```
$option1 =
 @{Key="InstanceIds";Values=@("i-0000293ffd8c57862","i-0cb2b964d3e14fd9f")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Single instance" -ResourceType "INSTANCE"
```

Ausgabe:

```
6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

Beispiel 3: In diesem Beispiel wird mithilfe von EC2-Tags eine Instance mit einem Wartungsfenster registriert.

```
$option1 = @{Key="tag:Environment";Values=@("Production")}
Register-SSMTargetWithMaintenanceWindow -WindowId "mw-06cf17cbefcb4bf4f" -Target
 $option1 -OwnerInformation "Production Web Servers" -ResourceType "INSTANCE"
```

Ausgabe:

```
2994977e-aefb-4a71-beac-df620352f184
```

- API-Details finden Sie RegisterTargetWithMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-SSMTaskWithMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Register-SSMTaskWithMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Instance-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)
```

```
Register-SSMTaskWithMaintenanceWindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="InstanceIds";Values="i-0000293ffd8c57862" } -TaskType "RUN_COMMAND" -
Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 2: In diesem Beispiel wird eine Aufgabe mit einem Wartungsfenster unter Verwendung einer Ziel-ID registriert. Die Ausgabe ist die Aufgaben-ID.

```
$parameters = @{}
$parameterValues = New-Object
 Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression
$parameterValues.Values = @("Install")
$parameters.Add("Operation", $parameterValues)

register-ssmtaskwithmaintenancewindow -WindowId "mw-03a342e62c96d31b0"
 -ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
 -MaxConcurrency 1 -MaxError 1 -TaskArn "AWS-RunShellScript" -Target
 @{ Key="WindowTargetIds";Values="350d44e6-28cc-44e2-951f-4b2c985838f6" } -TaskType
 "RUN_COMMAND" -Priority 10 -TaskParameter $parameters
```

Ausgabe:

```
f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

Beispiel 3: In diesem Beispiel wird ein Parameterobjekt für das Run-Befehlsdokument **`AWS-RunPowerShellScript`** und eine Aufgabe mit einem bestimmten Wartungsfenster unter Verwendung der Ziel-ID erstellt. Die Rückgabeausgabe ist die Aufgaben-ID.

```
$parameters =
 [Collections.Generic.Dictionary[String,Collections.Generic.List[String]]]::new()
$parameters.Add("commands",@("ipconfig","dir env:\computername"))
$parameters.Add("executionTimeout",@(3600))

$props = @{
    WindowId = "mw-0123e4cce56ff78ae"
```

```
    ServiceRoleArn = "arn:aws:iam::123456789012:role/MaintenanceWindowsRole"
    MaxConcurrency = 1
    MaxError = 1
    TaskType = "RUN_COMMAND"
    TaskArn = "AWS-RunPowerShellScript"
    Target = @{Key="WindowTargetIds";Values="fe1234ea-56d7-890b-12f3-456b789bee0f"}
    Priority = 1
    RunCommand_Parameter = $parameters
    Name = "set-via-cmdlet"
}

Register-SSMTaskWithMaintenanceWindow @props
```

Ausgabe:

```
f1e2ef34-5678-12e3-456a-12334c5c6cbe
```

Beispiel 4: In diesem Beispiel wird eine AWS Systems Manager Automation-Aufgabe mithilfe eines Dokuments mit dem Namen **`Create-Snapshots`** registriert.

```
$automationParameters = @{}
$automationParameters.Add( "instanceId", @("{{ TARGET_ID }}") )
$automationParameters.Add( "AutomationAssumeRole",
 @("{arn:aws:iam::111111111111:role/AutomationRole}") )
$automationParameters.Add( "SnapshotTimeout", @("PT20M") )
Register-SSMTaskWithMaintenanceWindow -WindowId mw-123EXAMPLE456`
    -ServiceRoleArn "arn:aws:iam::123456789012:role/MW-Role"`
    -MaxConcurrency 1 -MaxError 1 -TaskArn "CreateVolumeSnapshots"`
    -Target @{ Key="WindowTargetIds";Values="4b5acdf4-946c-4355-
bd68-4329a43a5fd1" }`
    -TaskType "AUTOMATION"`
    -Priority 4`
    -Automation_DocumentVersion '$DEFAULT' -Automation_Parameter
 $automationParameters -Name "Create-Snapshots"
```

- API-Details finden Sie RegisterTaskWithMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMActivation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMActivation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Aktivierung gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMActivation -ActivationId "08e51e79-1e36-446c-8e63-9458569c1363"
```

- API-Details finden Sie DeleteActivation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Verknüpfung zwischen einer Instance und einem Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMAssociation -InstanceId "i-0cb2b964d3e14fd9f" -Name "AWS-UpdateSSMAgent"
```

- API-Details finden Sie DeleteAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMDocument`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Dokument gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMDocument -Name "RunShellScript"
```

- API-Details finden Sie DeleteDocument in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Wartungsfenster entfernt.

```
Remove-SSMMaintenanceWindow -WindowId "mw-06d59c1a07c022145"
```

Ausgabe:

```
mw-06d59c1a07c022145
```

- API-Details finden Sie DeleteMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMParameter`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMParameter`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Parameter gelöscht. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMParameter -Name "helloWorld"
```

- API-Details finden Sie DeleteParameter in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Patch-Baseline gelöscht.

```
Remove-SSMPatchBaseline -BaselineId "pb-045f10b4f382baeda"
```

Ausgabe:

```
pb-045f10b4f382baeda
```

- API-Details finden Sie DeletePatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-SSMResourceTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-SSMResourceTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Tag aus einem Wartungsfenster entfernt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Remove-SSMResourceTag -ResourceId "mw-03eb9db42890fb82d" -ResourceType
 "MaintenanceWindow" -TagKey "Production"
```

- API-Details finden Sie RemoveTagsFromResource in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Send-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Send-SSMCommand`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Echo-Befehl auf einer Ziel-Instance ausgeführt.

```
Send-SSMCommand -DocumentName "AWS-RunPowerShellScript" -Parameter @{commands =
 "echo helloWorld"} -Target @{Key="instanceids";Values=@("i-0cb2b964d3e14fd9f")}
```

Ausgabe:

```
CommandId          : d8d190fc-32c1-4d65-a0df-ff5ff3965524
Comment            :
CompletedCount     : 0
DocumentName       : AWS-RunPowerShellScript
ErrorCount         : 0
ExpiresAfter       : 3/7/2017 10:48:37 PM
```

```
InstanceIds        : {}
MaxConcurrency     : 50
MaxErrors          : 0
NotificationConfig : Amazon.SimpleSystemsManagement.Model.NotificationConfig
OutputS3BucketName :
OutputS3KeyPrefix  :
OutputS3Region     :
Parameters         : {[commands,
 Amazon.Runtime.Internal.Util.AlwaysSendList`1[System.String]]}
RequestedDateTime  : 3/7/2017 9:48:37 PM
ServiceRole        :
Status             : Pending
StatusDetails      : Pending
TargetCount        : 0
Targets            : {instanceids}
```

Beispiel 2: Dieses Beispiel zeigt, wie ein Befehl ausgeführt wird, der verschachtelte Parameter akzeptiert.

```
Send-SSMCommand -DocumentName "AWS-RunRemoteScript" -Parameter
 @{ sourceType="GitHub";sourceInfo='{"owner": "me","repository": "amazon-
ssm","path": "Examples/Install-Win32OpenSSH"}'; "commandLine"=".\Install-
Win32OpenSSH.ps1"} -InstanceId i-0cb2b964d3e14fd9f
```

- API-Details finden Sie SendCommand in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Start-SSMAutomationExecution`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Dokument ausgeführt, das eine Automation-Rolle, eine AMI-Quell-ID und eine Amazon-EC2-Instance-Rolle angibt.

```
Start-SSMAutomationExecution -DocumentName AWS-UpdateLinuxAmi -
Parameter @{'AutomationAssumeRole'='arn:aws:iam::123456789012:role/
SSMAutomationRole';'SourceAmiId'='ami-f173cc91';'InstanceIamRole'='EC2InstanceRole'}
```

Ausgabe:

```
3a532a4f-0382-11e7-9df7-6f11185f6dd1
```

- API-Details finden Sie StartAutomationExecution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Start-SSMSession`

Das folgende Codebeispiel zeigt die Verwendung. `Start-SSMSession`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Verbindung zu einem Ziel für eine Session-Manager-Sitzung initiiert, wodurch die Portweiterleitung aktiviert wird. Hinweis: Sie müssen den Regionsparameter hinzufügen, falls er nicht bereits mit AWSCredentials festgelegt wurde.

```
Start-SSMSession -Target 'i-064578e5e7454488f' -DocumentName 'AWS-
StartPortForwardingSession' -Parameter @{ localPortNumber = '8080'; portNumber =
 '80' } -Region 'us-west-1'
```

Ausgabe:

```
Starting session with SessionId: testUser-xi4glew849asyeryde34u4dfsdfy
Port 8080 opened for sessionId testUser-xi4glew849asyeryde34u4dfsdfy.
Waiting for connections...
```

Beispiel 2: Hier wird eine interaktive Sitzung mit einer angegebenen Instance für eine Session-Manager-Sitzung erstellt.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1'
```

Ausgabe:

```
Starting session with SessionId : testUser-xi4glew849asyeryde34u4dfsdfy
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.



Install the latest PowerShell for new features and improvements!
```

```
PS C:\Windows\system32> whoami
ec2amaz-fnsdrwv\ec2-test-user
PS C:\Windows\system32>
```

Beispiel 3: In diesem Beispiel wird eine Sitzung erstellt, ohne eine Verbindung zu ihr herzustellen, und es werden die TokenValue Eigenschaften SessionId StreamUrl, und zurückgegeben, die für die Verbindung mit der Sitzung erforderlich sind.

```
Start-SSMSession -Target 'i-1234567890abcdef0' -Region 'us-west-1' -
DisablePluginInvocation
```

Ausgabe:

```
SessionId        : testUser-xi4glew849asyeryde34u4dfsdfy
StreamUrl        : {StreamUrl value redacted}
TokenValue       : {Token value redacted}
ContentLength    : 1207
HttpStatusCode   : OK
```

- API-Details finden Sie StartSession in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-SSMAutomationExecution`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-SSMAutomationExecution`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Automation-Ausführung gestoppt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Stop-SSMAutomationExecution -AutomationExecutionId "4105a4fc-
f944-11e6-9d32-8fb2db27a909"
```

- API-Details finden Sie StopAutomationExecution in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-SSMCommand`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-SSMCommand`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird versucht, einen Befehl abzubrechen. Wenn der Vorgang erfolgreich ist, erfolgt keine Ausgabe.

```
Stop-SSMCommand -CommandId "9ded293e-e792-4440-8e3e-7b8ec5feaa38"
```

- API-Details finden Sie CancelCommand in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-SSMManagedInstance`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMManagedInstance`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung einer verwalteten Instance aufgehoben. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Unregister-SSMManagedInstance -InstanceId "mi-08ab247cdf1046573"
```

- API-Details finden Sie DeregisterManagedInstance in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-SSMPatchBaselineForPatchGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMPatchBaselineForPatchGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung einer Patchgruppe von einer Patch-Baseline aufgehoben.

```
Unregister-SSMPatchBaselineForPatchGroup -BaselineId "pb-045f10b4f382baeda" -
PatchGroup "Production"
```

Ausgabe:

```
BaselineId           PatchGroup
----------           ----------
pb-045f10b4f382baeda Production
```

- API-Details finden Sie DeregisterPatchBaselineForPatchGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-SSMTargetFromMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMTargetFromMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Ziel aus einem Wartungsfenster entfernt.

```
Unregister-SSMTargetFromMaintenanceWindow -WindowTargetId "6ab5c208-9fc4-4697-84b7-
b02a6cc25f7d" -WindowId "mw-06cf17cbefcb4bf4f"
```

Ausgabe:

```
WindowId             WindowTargetId
--------             --------------
mw-06cf17cbefcb4bf4f 6ab5c208-9fc4-4697-84b7-b02a6cc25f7d
```

- API-Details finden Sie DeregisterTargetFromMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-SSMTaskFromMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-SSMTaskFromMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Aufgabe aus einem Wartungsfenster entfernt.

```
Unregister-SSMTaskFromMaintenanceWindow -WindowTaskId "f34a2c47-ddfd-4c85-
a88d-72366b69af1b" -WindowId "mw-03a342e62c96d31b0"
```

Ausgabe:

```
WindowId             WindowTaskId
--------             ------------
mw-03a342e62c96d31b0 f34a2c47-ddfd-4c85-a88d-72366b69af1b
```

- API-Details finden Sie DeregisterTaskFromMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMAssociation`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird eine Verknüpfung mit einer neuen Dokumentversion aktualisiert.

```
Update-SSMAssociation -AssociationId "93285663-92df-44cb-9f26-2292d4ecc439" -
DocumentVersion "1"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            :
Date                  : 3/1/2017 6:22:21 PM
Status.Name           :
Status.Date           :
Status.Message        :
Status.AdditionalInfo :
```

- API-Details finden Sie UpdateAssociation in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMAssociationStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMAssociationStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Zuordnungsstatus der Zuordnung zwischen einer Instance und einem Konfigurationsdokument aktualisiert.

```
Update-SSMAssociationStatus -Name "AWS-UpdateSSMAgent" -InstanceId
 "i-0000293ffd8c57862" -AssociationStatus_Date "2015-02-20T08:31:11Z"
 -AssociationStatus_Name "Pending" -AssociationStatus_Message
 "temporary_status_change" -AssociationStatus_AdditionalInfo "Additional-Config-
Needed"
```

Ausgabe:

```
Name                  : AWS-UpdateSSMAgent
InstanceId            : i-0000293ffd8c57862
Date                  : 2/23/2017 6:55:22 PM
Status.Name           : Pending
Status.Date           : 2/20/2015 8:31:11 AM
Status.Message        : temporary_status_change
Status.AdditionalInfo : Additional-Config-Needed
```

- API-Details finden Sie UpdateAssociationStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMDocument`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMDocument`

Tools für PowerShell V5

Beispiel 1: Dadurch wird eine neue Version eines Dokuments mit dem aktualisierten Inhalt der von Ihnen angegebenen JSON-Datei erstellt. Das Dokument muss im JSON-Format vorliegen. Sie können die Dokumentversion mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocument -Name RunShellScript -DocumentVersion "1" -Content (Get-Content -
Raw "c:\temp\RunShellScript.json")
```

Ausgabe:

```
CreatedDate     : 3/1/2017 2:59:17 AM
DefaultVersion  : 1
```

```
Description     : Run an updated script
DocumentType    : Command
DocumentVersion : 2
Hash            : 1d5ce820e999ff051eb4841ed887593daf77120fd76cae0d18a53cc42e4e22c1
HashType        : Sha256
LatestVersion   : 2
Name            : RunShellScript
Owner           : 809632081692
Parameters      : {commands}
PlatformTypes   : {Linux}
SchemaVersion   : 2.0
Sha1            :
Status          : Updating
```

- API-Details finden Sie UpdateDocument in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMDocumentDefaultVersion`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMDocumentDefaultVersion`

Tools für PowerShell V5

Beispiel 1: Dadurch wird die Standardversion eines Dokuments aktualisiert. Sie können die verfügbaren Dokumentversionen mit dem Cmdlet "Get-SSMDocumentVersionList" abrufen.

```
Update-SSMDocumentDefaultVersion -Name "RunShellScript" -DocumentVersion "2"
```

Ausgabe:

```
DefaultVersion Name
-------------- ----
2              RunShellScript
```

- API-Details finden Sie UpdateDocumentDefaultVersion in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMMaintenanceWindow`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Name eines Wartungsfensters aktualisiert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Name "My-Renamed-MW"
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 2: In diesem Beispiel wird ein Wartungsfenster aktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $true
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : True
Name                     : My-Renamed-MW
Schedule                 : cron(0 */30 * * * ? *)
WindowId                 : mw-03eb9db42890fb82d
```

Beispiel 3: In diesem Beispiel wird ein Wartungsfenster deaktiviert.

```
Update-SSMMaintenanceWindow -WindowId "mw-03eb9db42890fb82d" -Enabled $false
```

Ausgabe:

```
AllowUnassociatedTargets : False
Cutoff                   : 1
Duration                 : 2
Enabled                  : False
Name                     : My-Renamed-MW
```

```
Schedule                  : cron(0 */30 * * * ? *)
WindowId                  : mw-03eb9db42890fb82d
```

- API-Details finden Sie UpdateMaintenanceWindow in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMManagedInstanceRole`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMManagedInstanceRole`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Rolle einer verwalteten Instance aktualisiert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Update-SSMManagedInstanceRole -InstanceId "mi-08ab247cdf1046573" -IamRole
 "AutomationRole"
```

- API-Details finden Sie UpdateManagedInstanceRole in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Update-SSMPatchBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `Update-SSMPatchBaseline`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden einer vorhandenen Patch-Baseline zwei Patches als abgelehnt und ein Patch als genehmigt hinzugefügt.

```
Update-SSMPatchBaseline -BaselineId "pb-03da896ca3b68b639" -RejectedPatch
 "KB2032276","MS10-048" -ApprovedPatch "KB2124261"
```

Ausgabe:

```
ApprovalRules   : Amazon.SimpleSystemsManagement.Model.PatchRuleGroup
ApprovedPatches : {KB2124261}
BaselineId      : pb-03da896ca3b68b639
CreatedDate     : 3/3/2017 5:02:19 PM
Description     : Baseline containing all updates approved for production systems
GlobalFilters   : Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
```

```
ModifiedDate    : 3/3/2017 5:22:10 PM
Name            : Production-Baseline
RejectedPatches : {KB2032276, MS10-048}
```

- API-Details finden Sie UpdatePatchBaseline in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-SSMComplianceItem`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMComplianceItem`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein benutzerdefiniertes Compliance-Element für die angegebene verwaltete Instance geschrieben

```
$item = [Amazon.SimpleSystemsManagement.Model.ComplianceItemEntry]::new()
$item.Id = "07Jun2019-3"
$item.Severity="LOW"
$item.Status="COMPLIANT"
$item.Title="Fin-test-1 - custom"
Write-SSMComplianceItem -ResourceId mi-012dcb3ecea45b678 -ComplianceType
 Custom:VSSCompliant2 -ResourceType ManagedInstance -Item $item -
ExecutionSummary_ExecutionTime "07-Jun-2019"
```

- API-Details finden Sie PutComplianceItems in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-SSMInventory`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMInventory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden einer Instance Informationen zum Rack-Standort zugewiesen. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
$data = New-Object
 "System.Collections.Generic.Dictionary[System.String,System.String]"
$data.Add("RackLocation", "Bay B/Row C/Rack D/Shelf F")
```

```
$items = New-Object
 "System.Collections.Generic.List[System.Collections.Generic.Dictionary[System.String,
 System.String]]"
$items.Add($data)

$customInventoryItem = New-Object Amazon.SimpleSystemsManagement.Model.InventoryItem
$customInventoryItem.CaptureTime = "2016-08-22T10:01:01Z"
$customInventoryItem.Content = $items
$customInventoryItem.TypeName = "Custom:TestRackInfo2"
$customInventoryItem.SchemaVersion = "1.0"

$inventoryItems = @($customInventoryItem)

Write-SSMInventory -InstanceId "i-0cb2b964d3e14fd9f" -Item $inventoryItems
```

- API-Details finden Sie PutInventory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Write-SSMParameter`

Das folgende Codebeispiel zeigt die Verwendung. `Write-SSMParameter`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird ein Parameter erstellt. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "helloWorld"
```

Beispiel 2: In diesem Beispiel wird ein Parameter geändert. Wenn der Befehl erfolgreich ausgeführt wurde, gibt es keine Ausgabe.

```
Write-SSMParameter -Name "Welcome" -Type "String" -Value "Good day, Sunshine!" -
Overwrite $true
```

- API-Details finden Sie PutParameter in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Amazon Translate-Beispiele mit Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit Amazon Translate verwenden.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ConvertTo-TRNTargetLanguage`

Das folgende Codebeispiel zeigt die Verwendung`ConvertTo-TRNTargetLanguage`.

Tools für PowerShell V5

Beispiel 1: Konvertiert den angegebenen englischen Text in Französisch. Der zu konvertierende Text kann auch als Parameter -Text übergeben werden.

```
"Hello World" | ConvertTo-TRNTargetLanguage -SourceLanguageCode en -
TargetLanguageCode fr
```

- API-Details finden Sie TranslateText in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

AWS WAFV2 Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. AWS WAFV2

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`New-WAF2WebACL`

Das folgende Codebeispiel zeigt die Verwendung`New-WAF2WebACL`.

Tools für PowerShell V5

Beispiel 1: Dieser Befehl erstellt eine neue Web-ACL mit dem Namen „waf-test“. Bitte beachten Sie, dass gemäß der Dokumentation zur Service-API 'DefaultAction' eine erforderliche Eigenschaft ist. Daher sollte entweder ein Wert für '- DefaultAction _Allow' oder and/or '- DefaultAction _Block 'angegeben werden. Da '- DefaultAction _Allow' und '- DefaultAction _Block 'nicht die erforderlichen Eigenschaften sind, könnte der Wert '@ {}' als Platzhalter verwendet werden, wie im obigen Beispiel gezeigt.

```
New-WAF2WebACL -Name "waf-test" -Scope REGIONAL -Region eu-
west-1 -VisibilityConfig_CloudWatchMetricsEnabled $true -
VisibilityConfig_SampledRequestsEnabled $true -VisibilityConfig_MetricName "waf-
test" -Description "Test" -DefaultAction_Allow @{}
```

Ausgabe:

```
ARN         : arn:aws:wafv2:eu-west-1:139480602983:regional/webacl/waf-
test/19460b3f-db14-4b9a-8e23-a417e1eb007f
Description : Test
Id          : 19460b3f-db14-4b9a-8e23-a417e1eb007f
LockToken   : 5a0cd5eb-d911-4341-b313-b429e6d6b6ab
Name        : waf-test
```

- API-Details finden Sie CreateWebAcl in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

WorkSpaces Beispiele zur Verwendung von Tools für V5 PowerShell

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und gängige Szenarien implementieren, indem Sie AWS -Tools für PowerShell V5 mit verwenden. WorkSpaces

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Approve-WKSIpRule`

Das folgende Codebeispiel zeigt die Verwendung`Approve-WKSIpRule`.

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Regeln zu einer vorhandenen IP-Gruppe hinzugefügt

```
$Rule = @(
@{IPRule = "10.1.0.0/0"; RuleDesc = "First Rule Added"},
@{IPRule = "10.2.0.0/0"; RuleDesc = "Second Rule Added"}
)

Approve-WKSIpRule -GroupId wsipg-abcnx2fcw -UserRule $Rule
```

- API-Details finden Sie AuthorizeIpRules in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Copy-WKSWorkspaceImage`

Das folgende Codebeispiel zeigt die Verwendung. `Copy-WKSWorkspaceImage`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel kopiert das Workspace-Image mit der angegebenen ID von us-west-2 in die aktuelle Region mit dem Namen "" CopiedImageTest

```
Copy-WKSWorkspaceImage -Name CopiedImageTest -SourceRegion us-west-2 -SourceImageId
 wsi-djfoedhw6
```

Ausgabe:

```
wsi-456abaqfe
```

- API-Details finden Sie CopyWorkspaceImage in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSClientProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSClientProperty`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird Reconnection für den Workspaces-Client aktiviert

```
Edit-WKSClientProperty -Region us-west-2 -ClientProperties_ReconnectEnabled
 "ENABLED" -ResourceId d-123414a369
```

- API-Details finden Sie ModifyClientProperties in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSSelfServicePermission`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSSelfServicePermission`

Tools für PowerShell V5

Beispiel1: In diesem Beispiel werden Self-Service-Berechtigungen zum Ändern des Rechentyps und zum Erhöhen der Volumengröße für das angegebene Verzeichnis aktiviert.

```
Edit-WKSSelfservicePermission -Region us-west-2 -ResourceId
 d-123454a369 -SelfservicePermissions_ChangeComputeType ENABLED -
SelfservicePermissions_IncreaseVolumeSize ENABLED
```

- API-Details finden Sie ModifySelfservicePermissions in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSWorkspaceAccessProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceAccessProperty`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Workspace-Zugriff auf Android und Chrome OS für das angegebene Verzeichnis aktiviert

```
Edit-WKSWorkspaceAccessProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceAccessProperties_DeviceTypeAndroid ALLOW -
WorkspaceAccessProperties_DeviceTypeChromeOs ALLOW
```

- API-Details finden Sie ModifyWorkspaceAccessProperties in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSWorkspaceCreationProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceCreationProperty`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden beim Erstellen eines Workspace die Standardwerte für Internetzugriff und Wartungsmodus auf „true“ gesetzt

```
Edit-WKSWorkspaceCreationProperty -Region us-west-2 -ResourceId
 d-123454a369 -WorkspaceCreationProperties_EnableInternetAccess $true -
WorkspaceCreationProperties_EnableMaintenanceMode $true
```

- API-Details finden Sie ModifyWorkspaceCreationProperties in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSWorkspaceProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceProperty`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Eigenschaft Workspace Running Mode für den angegebenen Workspace auf Auto Stop geändert

```
Edit-WKSWorkspaceProperty -WorkspaceId ws-w361s100v -Region us-west-2 -
WorkspaceProperties_RunningMode AUTO_STOP
```

- API-Details finden Sie ModifyWorkspaceProperties in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Edit-WKSWorkspaceState`

Das folgende Codebeispiel zeigt die Verwendung. `Edit-WKSWorkspaceState`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Status des angegebenen Workspace auf „Verfügbar“ geändert

```
Edit-WKSWorkspaceState -WorkspaceId ws-w361s100v -Region us-west-2 -WorkspaceState
 AVAILABLE
```

- API-Details finden Sie ModifyWorkspaceState in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSClientProperty`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSClientProperty`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Client-Eigenschaften des Workspace Client für das angegebene Verzeichnis abgerufen

```
Get-WKSClientProperty -ResourceId d-223562a123
```

- API-Details finden Sie DescribeClientProperties in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSIpGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Details der angegebenen IP-Gruppe in der angegebenen Region abgerufen

```
Get-WKSIpGroup -Region us-east-1 -GroupId wsipg-8m1234v45
```

Ausgabe:

```
GroupDesc GroupId         GroupName UserRules
--------- -------         --------- ---------
          wsipg-8m1234v45 TestGroup {Amazon.WorkSpaces.Model.IpRuleItem,
 Amazon.WorkSpaces.Model.IpRuleItem}
```

- API-Details finden Sie DescribeIpGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das Tag für den angegebenen Workspace abgerufen

```
Get-WKSTag -WorkspaceId ws-w361s234r -Region us-west-2
```

Ausgabe:

```
Key         Value
---         -----
auto-delete no
purpose     Workbench
```

- API-Details finden Sie DescribeTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Ruft Details aller Ihrer Daten WorkSpaces zur Pipeline ab.

```
Get-WKSWorkspace
```

Ausgabe:

```
BundleId                    : wsb-1a2b3c4d
ComputerName                :
DirectoryId                 : d-1a2b3c4d
ErrorCode                   :
ErrorMessage                :
IpAddress                   :
RootVolumeEncryptionEnabled : False
State                       : PENDING
SubnetId                    :
UserName                    : myuser
UserVolumeEncryptionEnabled : False
VolumeEncryptionKey         :
WorkspaceId                 : ws-1a2b3c4d
WorkspaceProperties         : Amazon.WorkSpaces.Model.WorkspaceProperties
```

Beispiel 2: Dieser Befehl zeigt die Werte der untergeordneten Eigenschaften von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Weitere Informationen zu den untergeordneten Eigenschaften von **`WorkSpaceProperties`** finden Sie unter https://docs.aws.amazon.com/workspaces/latest/api/API_WorkspaceProperties.html.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-xdaf7hc9s).WorkspaceProperties
```

Ausgabe:

```
        ComputeTypeName                     : STANDARD
        RootVolumeSizeGib                   : 80
        RunningMode                         : AUTO_STOP
        RunningModeAutoStopTimeoutInMinutes : 60
        UserVolumeSizeGib                   : 50
```

Beispiel 3: Dieser Befehl zeigt den Wert der untergeordneten Eigenschaft **`RootVolumeSizeGib`** von **`WorkSpaceProperties`** für einen Workspace in der **`us-west-2`**-Region an. Die Größe des Root-Volumes in GiB beträgt 80.

```
(Get-WKSWorkspace -Region us-west-2 -WorkSpaceId ws-
xdaf7hc9s).WorkspaceProperties.RootVolumeSizeGib
```

Ausgabe:

```
        80
```

- API-Details finden Sie DescribeWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspaceBundle`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceBundle`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zu allen Workspace-Bundles in der aktuellen Region abgerufen

```
Get-WKSWorkspaceBundle
```

Ausgabe:

```
BundleId        : wsb-sfhdgv342
ComputeType     : Amazon.WorkSpaces.Model.ComputeType
Description     : This bundle is custom
ImageId         : wsi-235aeqges
LastUpdatedTime : 12/26/2019 06:44:07
Name            : CustomBundleTest
Owner           : 233816212345
RootStorage     : Amazon.WorkSpaces.Model.RootStorage
UserStorage     : Amazon.WorkSpaces.Model.UserStorage
```

- API-Details finden Sie DescribeWorkspaceBundles in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspaceDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden die Verzeichnisdetails für registrierte Verzeichnisse aufgeführt

```
Get-WKSWorkspaceDirectory
```

Ausgabe:

```
Alias                       : TestWorkspace
CustomerUserName            : Administrator
DirectoryId                 : d-123414a369
DirectoryName               : TestDirectory.com
DirectoryType               : MicrosoftAD
DnsIpAddresses              : {172.31.43.45, 172.31.2.97}
IamRoleId                   : arn:aws:iam::761234567801:role/workspaces_RoleDefault
IpGroupIds                  : {}
RegistrationCode            : WSpdx+4RRT43
SelfservicePermissions      : Amazon.WorkSpaces.Model.SelfservicePermissions
State                       : REGISTERED
SubnetIds                   : {subnet-1m3m7b43, subnet-ard11aba}
Tenancy                     : SHARED
WorkspaceAccessProperties   : Amazon.WorkSpaces.Model.WorkspaceAccessProperties
WorkspaceCreationProperties :
 Amazon.WorkSpaces.Model.DefaultWorkspaceCreationProperties
WorkspaceSecurityGroupId    : sg-0ed2441234a123c43
```

- API-Details finden Sie DescribeWorkspaceDirectories in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspaceImage`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceImage`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel werden Details zu allen Images in der Region abgerufen

```
Get-WKSWorkspaceImage
```

Ausgabe:

```
Description     :This image is copied from another image
ErrorCode       :
ErrorMessage    :
```

```
ImageId         : wsi-345ahdjgo
Name            : CopiedImageTest
OperatingSystem : Amazon.WorkSpaces.Model.OperatingSystem
RequiredTenancy : DEFAULT
State           : AVAILABLE
```

- API-Details finden Sie DescribeWorkspaceImages in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspaceSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspaceSnapshot`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel zeigt den Zeitstempel des letzten Snapshots, der für den angegebenen Workspace erstellt wurde

```
Get-WKSWorkspaceSnapshot -WorkspaceId ws-w361s100v
```

Ausgabe:

```
RebuildSnapshots                   RestoreSnapshots
----------------                   ----------------
{Amazon.WorkSpaces.Model.Snapshot} {Amazon.WorkSpaces.Model.Snapshot}
```

- API-Details finden Sie DescribeWorkspaceSnapshots in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Get-WKSWorkspacesConnectionStatus`

Das folgende Codebeispiel zeigt die Verwendung. `Get-WKSWorkspacesConnectionStatus`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird der Verbindungsstatus für den angegebenen Workspace abgerufen

```
Get-WKSWorkspacesConnectionStatus -WorkspaceId ws-w123s234r
```

- API-Details finden Sie DescribeWorkspacesConnectionStatus in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-WKSIpGroup

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSIpGroup`

Tools für PowerShell V5

Beispiel 1: Dieses Beispiel erstellt eine leere IP-Gruppe mit dem Namen FreshEmptyIpGroup

```
New-WKSIpGroup -GroupName "FreshNewIPGroup"
```

Ausgabe:

```
wsipg-w45rty4ty
```

- API-Details finden Sie CreateIpGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

New-WKSTag

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** ein neues Tag hinzugefügt. Das Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag
```

Beispiel 1: In diesem Beispiel wird einem Workspace namens **`ws-wsname`** mehrere Tags hinzugefügt. Ein Tag hat den Schlüssel „Name“ und den Schlüsselwert **`AWS_Workspace`**; das andere Tag hat den Tag-Schlüssel „Stage“ und den Schlüsselwert „Test“.

```
$tag = New-Object Amazon.WorkSpaces.Model.Tag
```

```
$tag.Key = "Name"
$tag.Value = "AWS_Workspace"

$tag2 = New-Object Amazon.WorkSpaces.Model.Tag
$tag2.Key = "Stage"
$tag2.Value = "Test"
New-WKSTag -Region us-west-2 -WorkspaceId ws-wsname -Tag $tag,$tag2
```

- API-Details finden Sie CreateTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`New-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `New-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Erstellen Sie eine WorkSpace für das bereitgestellte Paket, das Verzeichnis und den Benutzer.

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId" =
 "d-1a2b3c4d"; "UserName" = "USERNAME"}
```

Beispiel 2: Dieses Beispiel erstellt mehrere WorkSpaces

```
New-WKSWorkspace -Workspace @{"BundleID" = "wsb-1a2b3c4d"; "DirectoryId"
 = "d-1a2b3c4d"; "UserName" = "USERNAME_1"},@{"BundleID" = "wsb-1a2b3c4d";
 "DirectoryId" = "d-1a2b3c4d"; "UserName" = "USERNAME_2"}
```

- API-Details finden Sie CreateWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Register-WKSIpGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene IP-Gruppe im angegebenen Verzeichnis registriert

```
Register-WKSIpGroup -GroupId wsipg-23ahsdres -DirectoryId d-123412e123
```

- API-Details finden Sie AssociateIpGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Register-WKSWorkspaceDirectory`

Das folgende Codebeispiel zeigt die Verwendung. `Register-WKSWorkspaceDirectory`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das angegebene Verzeichnis für Workspaces Service registriert

```
Register-WKSWorkspaceDirectory -DirectoryId d-123412a123 -EnableWorkDoc $false
```

- API-Details finden Sie RegisterWorkspaceDirectory in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSIpGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die angegebene IP-Gruppe gelöscht

```
Remove-WKSIpGroup -GroupId wsipg-32fhgtred
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSIpGroup (DeleteIpGroup)" on target
 "wsipg-32fhgtred".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteIpGroup in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-WKSTag`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSTag`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird das dem Workspace zugeordnete Tag entfernt

```
Remove-WKSTag -ResourceId ws-w10b3abcd -TagKey "Type"
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-WKSTag (DeleteTags)" on target "ws-w10b3abcd".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- API-Details finden Sie DeleteTags in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Remove-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Remove-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Beendet mehrere WorkSpaces. Die Verwendung der Option -Force verhindert, dass das Cmdlet zur Bestätigung auffordert.

```
Remove-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0" -Force
```

Beispiel 2: Ruft die Sammlung aller Ihrer IDs ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - von weiter Remove-WKSWorkspace, wodurch alle IDs beendet werden. WorkSpaces Das Cmdlet fordert Sie auf, bevor jedes Cmdlet beendet WorkSpace wird. Um die Bestätigungsaufforderung zu unterdrücken, fügen Sie den Schalter -Force hinzu.

```
Get-WKSWorkspaces | Remove-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie TerminateRequest Objekte übergeben werden, die definieren, dass beendet werden WorkSpaces soll. Das Cmdlet fordert Sie zur Bestätigung auf, bevor es fortfährt, es sei denn, der Schalterparameter -Force ist ebenfalls angegeben.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.TerminateRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Remove-WKSWorkspace -Request $arrRequest
```

- API-Details finden Sie TerminateWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Reset-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Reset-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Baut das angegebene neu auf. WorkSpace

```
Reset-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Ruft die Sammlung aller Ihrer IDs ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - von weiter Reset-WKSWorkspace, wodurch der WorkSpaces neu erstellt wird.

```
Get-WKSWorkspaces | Reset-WKSWorkspace
```

- API-Details finden Sie RebuildWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Restart-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Restart-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Startet das angegebene neu. WorkSpace

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d"
```

Beispiel 2: Startet mehrere WorkSpaces neu.

```
Restart-WKSWorkspace -WorkspaceId "ws-1a2b3c4d","ws-5a6b7c8d"
```

Beispiel 3: Ruft die Sammlung aller Ihrer IDs ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - von weiter Restart-WKSWorkspace, wodurch der neu WorkSpaces gestartet wird.

```
Get-WKSWorkspaces | Restart-WKSWorkspace
```

- API-Details finden Sie RebootWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Stop-WKSWorkspace`

Das folgende Codebeispiel zeigt die Verwendung. `Stop-WKSWorkspace`

Tools für PowerShell V5

Beispiel 1: Stoppt mehrere WorkSpaces.

```
Stop-WKSWorkspace -WorkspaceId "ws-1a2b3c4d5","ws-6a7b8c9d0"
```

Beispiel 2: Ruft die Sammlung all Ihrer Daten ab WorkSpaces und leitet die IDs an den WorkSpaceId Parameter - weiter, der Stop-WKSWorkspace bewirkt, dass WorkSpaces das gestoppt wird.

```
Get-WKSWorkspaces | Stop-WKSWorkspace
```

Beispiel 3: Dieses Beispiel zeigt, wie StopRequest Objekte übergeben werden, die definieren, dass gestoppt werden WorkSpaces soll.

```
$arrRequest = @()
$request1 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request1.WorkspaceId = 'ws-12345678'
$arrRequest += $request1
$request2 = New-Object Amazon.WorkSpaces.Model.StopRequest
$request2.WorkspaceId = 'ws-abcdefgh'
$arrRequest += $request2
Stop-WKSWorkspace -Request $arrRequest
```

- API-Details finden Sie StopWorkspaces in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

`Unregister-WKSIpGroup`

Das folgende Codebeispiel zeigt die Verwendung. `Unregister-WKSIpGroup`

Tools für PowerShell V5

Beispiel 1: In diesem Beispiel wird die Registrierung der angegebenen IP-Gruppe im angegebenen Verzeichnis aufgehoben

```
Unregister-WKSIpGroup -GroupId wsipg-12abcdphq -DirectoryId d-123454b123
```

- API-Details finden Sie DisassociateIpGroups in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Codebeispiele für SDK für Python

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Python with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for Python Developer Guide — Mehr über die Verwendung von Python mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- DynamoDB-Beispiele, die SDK für Python verwenden

DynamoDB-Beispiele, die SDK für Python verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für Python

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
This is a test output. Examples for AWS SDK for Python
for AWS DynamoDB will be available here soon.
```

- API-Details finden Sie ListTables in der AWS SDK for Python API-Referenz.

Codebeispiele für SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Python (Boto3) with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for Python (Boto3) -Entwicklerhandbuch — Weitere Informationen zur Verwendung von Python mit. AWS
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele unter Verwendung von SDK für Python (Boto3)
- API-Gateway-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Application Recovery Controller unter Verwendung von SDK für Python (Boto3)

- Beispiele für Audit Manager unter Verwendung von SDK für Python (Boto3)
- Aurora-Beispiele unter Verwendung von SDK für Python (Boto3)
- Auto-Scaling-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Bedrock unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Bedrock Agents unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Python (Boto3)
- CloudFormation Beispiele mit SDK für Python (Boto3)
- CloudFront Beispiele mit SDK für Python (Boto3)
- CloudWatch Beispiele zur Verwendung des SDK für Python (Boto3)
- CloudWatch Protokolliert Beispiele mithilfe des SDK für Python (Boto3)
- Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Python (Boto3)
- AWS Config Beispiele mit SDK für Python (Boto3)
- AWS Control Tower Beispiele zur Verwendung des SDK für Python (Boto3)
- Firehose-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Device Farm unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Python (Boto3)
- DynamoDB-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon EC2 unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon ECR unter Verwendung von SDK für Python (Boto3)
- Elastic Load Balancing – Beispiele für Version 2 unter Verwendung von SDK für Java 2.x
- Beispiele für Amazon EMR unter Verwendung von SDK für Python (Boto3)
- EventBridge Beispiele zur Verwendung des SDK für Python (Boto3)
- EventBridge Scheduler-Beispiele mit SDK für Python (Boto3)
- Amazon Glacier-Beispiele mit SDK für Python (Boto3)
- AWS Glue Beispiele mit SDK für Python (Boto3)
- HealthImaging Beispiele zur Verwendung des SDK für Python (Boto3)
- HealthLake Beispiele mit SDK für Python (Boto3)

- IAM-Beispiele unter Verwendung von SDK für Python (Boto3)
- AWS IoT Beispiele mit SDK für Python (Boto3)
- AWS IoT data Beispiele zur Verwendung des SDK für Python (Boto3)
- AWS IoT SiteWise Beispiele zur Verwendung des SDK für Python (Boto3)
- Beispiele für Amazon Keyspaces unter Verwendung von SDK für Python (Boto3)
- Kinesis-Beispiele unter Verwendung von SDK für Python (Boto3)
- AWS KMS Beispiele mit SDK für Python (Boto3)
- Lambda-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Managed Service für Apache Flink unter Verwendung von SDK für Python (Boto3)
- AWS Marketplace Katalog-API-Beispiele mit SDK für Python (Boto3)
- AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für Python (Boto3)
- Beispiele für Amazon MSK unter Verwendung von SDK für Python (Boto3)
- Neptune-Beispiele unter Verwendung von SDK für Python (Boto3)
- Organizations-Beispiele unter Verwendung von SDK für Python (Boto3)
- Beispiele für Partner Central unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Pinpoint unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Pinpoint SMS und Voice API unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Polly unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon RDS unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Redshift unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Rekognition unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon S3 unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon S3 Control mit SDK für Python (Boto3)
- Beispiele für S3 Directory Buckets unter Verwendung von SDK für Python (Boto3)
- Beispiele für Secrets Manager unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon SES unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon SES API v2 unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon SNS unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon SQS unter Verwendung von SDK für Python (Boto3)

- Beispiele für Step Functions unter Verwendung von SDK für Python (Boto3)
- AWS STS Beispiele mit SDK für Python (Boto3)
- Support Beispiele zur Verwendung des SDK für Python (Boto3)
- Beispiele für Systems Manager unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Textract unter Verwendung von SDK für Python (Boto3)
- Beispiele für Amazon Transcribe unter Verwendung von SDK für Python (Boto3)

ACM-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von ACM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Anfordern eines Zertifikats von ACM
- Importieren eines selbstsignierten Zertifikats
- Auflisten und Beschreiben von Zertifikaten
- Entfernen Sie Zertifikate.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die ACM umschließt.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html
```

```
        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
when
                                   the email domain differs from the primary domain
of
                                   the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn


    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.
```

```
        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in PEM
                            format.
        :return: The ARN of the imported certificate.
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
            return certificate_arn


    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None, all
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If None,
                          all key usages are included.
        :param extended_key_usage: Filters the results to the specified extended key
                                   usages. If None, all extended key usages are
                                   included.
        :param key_types: Filters the results to the specified key types. If None,
all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
```

```
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
        else:
            return certificates


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
            )
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
certificate["DomainName"]
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
certificate_arn)
            raise
        else:
            return certificate
```

```
    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in tags.items()],
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
certificate_arn)
            raise


    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.
```

```
        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
certificate_arn)
            raise
        else:
            return tags


    def remove_tags(self, certificate_arn, tags):
        """
        Removes tags from a certificate. If the value of a tag is specified, the tag
is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags), certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
```

```
            )
            raise


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

Verwenden Sie die Wrapper-Klasse, um Zertifikate für Ihr Konto zu verwalten.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the AWS Certificate Manager (ACM) demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    acm_certificate = AcmCertificate(boto3.client("acm"))
    domain = "example.com"
    sub_domains = [f"{sub}.{domain}" for sub in ["test", "dev"]]
    print(f"Request a certificate for {domain}.")
    certificate_arn = acm_certificate.request_validation(domain, sub_domains, "DNS")
    print(f"Started validation, got certificate ARN: {certificate_arn}.")

    import_cert_arn = None
    cert_file_name = input(
        "Enter the file name for a self-signed certificate in PEM format. "
        "This certificate will be imported to ACM. Press Enter to skip: "
    )
    if cert_file_name:
        pk_file_name = input(
```

```
            "Enter the file name for the unencrypted private key of the certificate.
"
            "This file must also be in PEM format: "
        )
        if pk_file_name:
            with open(cert_file_name, "rb") as cert_file:
                import_cert = cert_file.read()
            with open(pk_file_name, "rb") as pk_file:
                import_pk = pk_file.read()
            import_cert_arn = acm_certificate.import_certificate(import_cert,
import_pk)
            print(f"Certificate imported, got ARN: {import_cert_arn}")
        else:
            print("No private key file entered. Skipping certificate import.")
    else:
        print("Skipping self-signed certificate import.")

    print("Getting the first 10 issued certificates.")
    certificates = acm_certificate.list(10, statuses=["ISSUED"])
    print(f"Found {len(certificates)} issued certificates.")

    print(f"Getting metadata for certificate {certificate_arn}")
    cert_metadata = acm_certificate.describe(certificate_arn)
    pprint(cert_metadata)

    if import_cert_arn is not None:
        print(f"Getting certificate for imported certificate {import_cert_arn}")
        import_cert_data = acm_certificate.get(import_cert_arn)
        pprint(import_cert_data)

    print(f"Adding tags to certificate {certificate_arn}.")
    acm_certificate.add_tags(certificate_arn, {"purpose": "acm demo", "color":
"green"})
    tags = acm_certificate.list_tags(certificate_arn)
    print(f"Found tags: {tags}")
    acm_certificate.remove_tags(certificate_arn, {key: None for key in tags})
    print("Removed tags.")

    print("Removing certificates added during the demo.")
    acm_certificate.remove(certificate_arn)
    if import_cert_arn is not None:
        acm_certificate.remove(import_cert_arn)

    print("Thanks for watching!")
```

```
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AddTagsToCertificate
  - DeleteCertificate
  - DescribeCertificate
  - GetCertificate
  - ImportCertificate
  - ListCertificates
  - ListTagsForCertificate
  - RemoveTagsFromCertificate
  - RequestCertificate
  - ResendValidationEmail

Aktionen

`AddTagsToCertificate`

Das folgende Codebeispiel zeigt die Verwendung`AddTagsToCertificate`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
```

```
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{"Key": key, "Value": value} for key, value in tags.items()],
            )
            logger.info("Added %s tags to certificate %s.", len(tags),
certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.",
certificate_arn)
            raise
```

- API-Details finden Sie AddTagsToCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
            raise
```

- API-Details finden Sie DeleteCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn
            )
            certificate = response["Certificate"]
            logger.info(
                "Got metadata for certificate for domain %s.",
 certificate["DomainName"]
            )
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.",
 certificate_arn)
            raise
        else:
            return certificate
```

- API-Details finden Sie DescribeCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`GetCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `GetCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response =
 self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
```

```
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response
```

- API-Details finden Sie GetCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`ImportCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `ImportCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def import_certificate(self, certificate_body, private_key):
        """
        Imports a self-signed certificate to ACM.

        :param certificate_body: The body of the certificate, in PEM format.
        :param private_key: The unencrypted private key of the certificate, in PEM
                            format.
        :return: The ARN of the imported certificate.
```

```
        """
        try:
            response = self.acm_client.import_certificate(
                Certificate=certificate_body, PrivateKey=private_key
            )
            certificate_arn = response["CertificateArn"]
            logger.info("Imported certificate.")
        except ClientError:
            logger.exception("Couldn't import certificate.")
            raise
        else:
            return certificate_arn
```

- API-Details finden Sie ImportCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung. `ListCertificates`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client
```

```
    def list(
        self,
        max_items,
        statuses=None,
        key_usage=None,
        extended_key_usage=None,
        key_types=None,
    ):
        """
        Lists the certificates for the current account.

        :param max_items: The maximum number of certificates to list.
        :param statuses: Filters the results to the specified statuses. If None, all
                         certificates are included.
        :param key_usage: Filters the results to the specified key usages. If None,
                          all key usages are included.
        :param extended_key_usage: Filters the results to the specified extended key
                                   usages. If None, all extended key usages are
                                   included.
        :param key_types: Filters the results to the specified key types. If None,
all
                          key types are included.
        :return: The list of certificates.
        """
        try:
            kwargs = {"MaxItems": max_items}
            if statuses is not None:
                kwargs["CertificateStatuses"] = statuses
            includes = {}
            if key_usage is not None:
                includes["keyUsage"] = key_usage
            if extended_key_usage is not None:
                includes["extendedKeyUsage"] = extended_key_usage
            if key_types is not None:
                includes["keyTypes"] = key_types
            if includes:
                kwargs["Includes"] = includes
            response = self.acm_client.list_certificates(**kwargs)
            certificates = response["CertificateSummaryList"]
            logger.info("Got %s certificates.", len(certificates))
        except ClientError:
            logger.exception("Couldn't get certificates.")
            raise
```

```
        else:
            return certificates
```

- API-Details finden Sie ListCertificates in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTagsForCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `ListTagsForCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn
            )
```

```
            tags = {tag["Key"]: tag["Value"] for tag in response["Tags"]}
            logger.info("Got %s tags for certificates %s.", len(tags),
 certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
 certificate_arn)
            raise
        else:
            return tags
```

- API-Details finden Sie ListTagsForCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`RemoveTagsFromCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `RemoveTagsFromCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def remove_tags(self, certificate_arn, tags):
        """
```

```
        Removes tags from a certificate. If the value of a tag is specified, the tag
is
        removed only when the value matches the value of the certificate's tag.
        Otherwise, the tag is removed regardless of its value.

        :param certificate_arn: The ARN of the certificate.
        :param tags: The dictionary of tags to remove.
        """
        try:
            cert_tags = []
            for key, value in tags.items():
                tag = {"Key": key}
                if value is not None:
                    tag["Value"] = value
                cert_tags.append(tag)
            self.acm_client.remove_tags_from_certificate(
                CertificateArn=certificate_arn, Tags=cert_tags
            )
            logger.info(
                "Removed %s tags from certificate %s.", len(tags), certificate_arn
            )
        except ClientError:
            logger.exception(
                "Couldn't remove tags from certificate %s.", certificate_arn
            )
            raise
```

- API-Details finden Sie RemoveTagsFromCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`RequestCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `RequestCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None
    ):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the
certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
when
                                   the email domain differs from the primary domain
of
                                   the certificate.
```

```
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                "DomainName": domain,
                "ValidationMethod": method,
                "SubjectAlternativeNames": alternate_domains,
            }
            if validation_domains is not None:
                kwargs["DomainValidationOptions"] = [
                    {"DomainName": key, "ValidationDomain": value}
                    for key, value in validation_domains.items()
                ]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response["CertificateArn"]
            logger.info(
                "Requested %s validation for domain %s. Certificate ARN is %s.",
                method,
                domain,
                certificate_arn,
            )
        except ClientError:
            logger.exception(
                "Request for %s validation of domain %s failed.", method, domain
            )
            raise
        else:
            return certificate_arn
```

- API-Details finden Sie RequestCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`ResendValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung. `ResendValidationEmail`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """

    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client


    def resend_validation_email(self, certificate_arn, domain, validation_domain):
        """
        Request that validation email is sent again, for a certificate that was
        previously requested with email validation.

        :param certificate_arn: The ARN of the certificate.
        :param domain: The primary domain of the certificate.
        :param validation_domain: Alternate domain to use for determining email
                                  addresses to use for validation.
        """
        try:
            self.acm_client.resend_validation_email(
                CertificateArn=certificate_arn,
                Domain=domain,
                ValidationDomain=validation_domain,
            )
            logger.info(
                "Validation email resent to validation domain %s.",
 validation_domain
            )
        except ClientError:
            logger.exception(
```

```
            "Couldn't resend validation email to %s.", validation_domain
        )
        raise
```

- API-Details finden Sie ResendValidationEmail in der AWS API-Referenz zum SDK für Python (Boto3).

API-Gateway-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von API Gateway Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateDeployment`

Das folgende Codebeispiel zeigt die Verwendung`CreateDeployment`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
```

```
        else:
            return self.api_url()



    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

- API-Details finden Sie CreateDeployment in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateResource`

Das folgende Codebeispiel zeigt die Verwendung. `CreateResource`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
```

```
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id
```

- API-Details finden Sie CreateResource in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateRestApi`

Das folgende Codebeispiel zeigt die Verwendung. `CreateRestApi`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise
```

```
        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- API-Details finden Sie CreateRestApi in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteRestApi`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteRestApi`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
```

```
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def delete_rest_api(self):
        """
        Deletes a REST API, including all of its resources and configuration.
        """
        try:
            self.apig_client.delete_rest_api(restApiId=self.api_id)
            logger.info("Deleted REST API %s.", self.api_id)
            self.api_id = None
        except ClientError:
            logger.exception("Couldn't delete REST API %s.", self.api_id)
            raise
```

- API-Details finden Sie DeleteRestApi in der AWS API-Referenz zum SDK für Python (Boto3).

`GetResources`

Das folgende Codebeispiel zeigt die Verwendung. `GetResources`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """
```

```
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err

        return self.api_id
```

- API-Details finden Sie GetResources in der AWS API-Referenz zum SDK für Python (Boto3).

GetRestApis

Das folgende Codebeispiel zeigt die Verwendung. `GetRestApis`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def get_rest_api_id(self, api_name):
        """
        Gets the ID of a REST API from its name by searching the list of REST APIs
        for the current account. Because names need not be unique, this returns only
        the first API with the specified name.

        :param api_name: The name of the API to look up.
        :return: The ID of the specified API.
        """
        try:
            rest_api = None
            paginator = self.apig_client.get_paginator("get_rest_apis")
            for page in paginator.paginate():
                rest_api = next(
```

```
                    (item for item in page["items"] if item["name"] == api_name),
None
                )
                if rest_api is not None:
                    break
            self.api_id = rest_api["id"]
            logger.info("Found ID %s for API %s.", rest_api["id"], api_name)
        except ClientError:
            logger.exception("Couldn't find ID for API %s.", api_name)
            raise
        else:
            return rest_api["id"]
```

- API-Details finden Sie GetRestApis in der AWS API-Referenz zum SDK für Python (Boto3).

`PutIntegration`

Das folgende Codebeispiel zeigt die Verwendung. `PutIntegration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
```

```
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
```

```
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
```

```
                service_uri,
            )
            raise
```

- API-Details finden Sie PutIntegration in der AWS API-Referenz zum SDK für Python (Boto3).

`PutIntegrationResponse`

Das folgende Codebeispiel zeigt die Verwendung. `PutIntegrationResponse`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
```

```
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
```

```
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- API-Details finden Sie PutIntegrationResponse in der AWS API-Referenz zum SDK für Python (Boto3).

PutMethod

Das folgende Codebeispiel zeigt die Verwendung. PutMethod

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.
```

```
        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
```

```
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- API-Details finden Sie PutMethod in der AWS API-Referenz zum SDK für Python (Boto3).

`PutMethodResponse`

Das folgende Codebeispiel zeigt die Verwendung. `PutMethodResponse`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
 that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
```

```
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
```

```
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise
```

- API-Details finden Sie PutMethodResponse in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen Sie eine REST-API, um Daten zu verfolgen COVID-19

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die ein System simuliert, um tägliche Fälle COVID-19 in den Vereinigten Staaten anhand fiktiver Daten zu verfolgen.

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon DynamoDB AWS Lambda verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle COVID-19 in den Vereinigten Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über das API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

SDK für Python (Boto3)

Zeigt, wie Sie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) -API und AWS Chalice verwenden, um eine REST-API zu erstellen, die von einer Amazon Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt

eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldeinformationen verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen einer Websocket-Chat-Anwendung

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

SDK für Python (Boto3)

Zeigt, wie Sie das AWS SDK für Python (Boto3) mit Amazon API Gateway V2 verwenden, um eine Websocket-API zu erstellen, die in Amazon DynamoDB integriert ist. AWS Lambda

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.
- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Erstellen und Bereitstellen einer REST-API

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine REST-API, die von API Gateway bereitgestellt wird.
- Fügen Sie der REST-API Ressourcen hinzu, um ein Benutzerprofil darzustellen.
- Fügen Sie Integrationsmethoden hinzu, damit die REST-API eine DynamoDB-Tabelle verwendet, um Benutzerdatenprofile zu speichern.
- Senden Sie HTTP-Anfragen an die REST-API, um Benutzerprofile hinzuzufügen und abzurufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die API-Gateway-Operationen umschließt.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


class ApiGatewayToService:
    """
```

```
    Encapsulates Amazon API Gateway functions that are used to create a REST API
that
    integrates with another AWS service.
    """

    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None


    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result["id"]
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result["items"] if item["path"] == "/"
            )["id"]
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err
```

```
        return self.api_id


    def add_rest_resource(self, parent_id, resource_path):
        """
        Adds a resource to a REST API.

        :param parent_id: The ID of the parent resource.
        :param resource_path: The path of the new resource, relative to the parent.
        :return: The ID of the new resource.
        """
        try:
            result = self.apig_client.create_resource(
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path
            )
            resource_id = result["id"]
            logger.info("Created resource %s.", resource_path)
        except ClientError:
            logger.exception("Couldn't create resource %s.", resource_path)
            raise
        else:
            return resource_id


    def add_integration_method(
        self,
        resource_id,
        rest_method,
        service_endpoint_prefix,
        service_action,
        service_method,
        role_arn,
        mapping_template,
    ):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                        this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
```

```
                                'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                         Gateway permission to use the specified action with the
                         service.
        :param mapping_template: A mapping template that is used to translate REST
                                 elements, such as query parameters, to the request
                                 body format required by the service.
        """
        service_uri = (
            f"arn:aws:apigateway:{self.apig_client.meta.region_name}"
            f":{service_endpoint_prefix}:action/{service_action}"
        )
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType="NONE",
            )
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseModels={"application/json": "Empty"},
            )
            logger.info("Created %s method for resource %s.", rest_method,
resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id
            )
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type="AWS",
                integrationHttpMethod=service_method,
                credentials=role_arn,
```

```
                requestTemplates={"application/json": json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior="WHEN_NO_TEMPLATES",
            )
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode="200",
                responseTemplates={"application/json": ""},
            )
            logger.info(
                "Created integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id,
                service_uri,
            )
            raise


    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name
            )
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()
```

```
    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (
            f"https://{self.api_id}.execute-api.{self.apig_client.meta.region_name}"
            f".amazonaws.com/{self.stage}"
        )
        if resource is not None:
            url = f"{url}/{resource}"
        return url
```

Stellen Sie eine REST-API bereit und rufen Sie sie mit dem Requests-Paket auf.

```
def usage_demo(table_name, role_name, rest_api_name):
    """
    Demonstrates how to used API Gateway to create and deploy a REST API, and how
    to use the Requests package to call it.

    :param table_name: The name of the demo DynamoDB table.
    :param role_name: The name of the demo role that grants API Gateway permission
 to
                      call DynamoDB.
    :param rest_api_name: The name of the demo REST API created by the demo.
    """
    gateway = ApiGatewayToService(boto3.client("apigateway"))
    role = boto3.resource("iam").Role(role_name)

    print("Creating REST API in API Gateway.")
    gateway.create_rest_api(rest_api_name)

    print("Adding resources to the REST API.")
    profiles_id = gateway.add_rest_resource(gateway.root_id, "profiles")
    username_id = gateway.add_rest_resource(profiles_id, "{username}")
```

```
    # The DynamoDB service requires that all integration requests use POST.
    print("Adding integration methods to read and write profiles in Amazon
DynamoDB.")
    gateway.add_integration_method(
        profiles_id,
        "GET",
        "dynamodb",
        "Scan",
        "POST",
        role.arn,
        {"TableName": table_name},
    )
    gateway.add_integration_method(
        profiles_id,
        "POST",
        "dynamodb",
        "PutItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Item": {
                "username": {"S": "$input.path('$.username')"},
                "name": {"S": "$input.path('$.name')"},
                "title": {"S": "$input.path('$.title')"},
            },
        },
    )
    gateway.add_integration_method(
        username_id,
        "GET",
        "dynamodb",
        "GetItem",
        "POST",
        role.arn,
        {
            "TableName": table_name,
            "Key": {"username": {"S": "$method.request.path.username"}},
        },
    )

    stage = "test"
    print(f"Deploying the {stage} stage.")
    gateway.deploy_api(stage)
```

```
    profiles_url = gateway.api_url("profiles")
    print(
        f"Using the Requests package to post some people to the profiles REST API at
 "
        f"{profiles_url}."
    )
    requests.post(
        profiles_url,
        json={"username": "will", "name": "William Shakespeare", "title":
 "playwright"},
    )
    requests.post(
        profiles_url,
        json={
            "username": "ludwig",
            "name": "Ludwig van Beethoven",
            "title": "composer",
        },
    )
    requests.post(
        profiles_url,
        json={"username": "jane", "name": "Jane Austen", "title": "author"},
    )
    print("Getting the list of profiles from the REST API.")
    profiles = requests.get(profiles_url).json()
    pprint(profiles)
    print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/
jane).")
    jane = requests.get(f"{profiles_url}/jane").json()
    pprint(jane)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDeployment
  - CreateResource
  - CreateRestApi
  - DeleteRestApi

- GetResources
- GetRestApis
- PutIntegration
- PutIntegrationResponse
- PutMethod
- PutMethodResponse

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten unter GitHub ansehen. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Beispiele für Application Recovery Controller unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für Python (Boto3) Application Recovery Controllers Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung`GetRoutingControlState`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.
```

```
    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get or
 set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn
            )
            return response
        except Exception as error:
            print(error)
            raise error
```

- API-Details finden Sie GetRoutingControlState in der AWS API-Referenz für SDK für Python (Boto3).

`UpdateRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateRoutingControlState`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to update the state
 for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
```

```
    :return: The routing control update response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get or
 set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.update_routing_control_state(
                RoutingControlArn=routing_control_arn,
                RoutingControlState=routing_control_state,
            )
            return response
        except Exception as error:
            print(error)
```

- API-Details finden Sie UpdateRoutingControlState in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Audit Manager unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Audit Manager Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie ein benutzerdefiniertes Framework aus einem AWS Config Konformitätspaket

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Holen Sie sich eine Liste der AWS Config Konformitätspakete.
- Erstellen eines benutzerdefinierten Audit-Manager-Steuerelements für jede verwaltete Regel in einem Konformitätspaket
- Erstellen eines benutzerdefinierten Audit-Manager-Frameworks, das die Steuerelemente enthält

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print(
                "Number of conformance packs fetched: ",
```

```
                len(conformance_packs.get("ConformancePackDetails")),
            )
            print("Fetched the following conformance packs: ")
            all_cpack_names = {
                cp["ConformancePackName"]
                for cp in conformance_packs.get("ConformancePackDetails")
            }
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                "Provide ConformancePackName that you want to create a custom "
                "framework for: "
            )
            if cpack_name not in all_cpack_names:
                print(f"{cpack_name} is not in the list of conformance packs!")
                print(
                    "Provide a conformance pack name from the available list of "
                    "conformance packs."
                )
                raise Exception("Invalid conformance pack")
            print("-" * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
        """
        Create custom controls for all managed AWS Config rules in a conformance
pack.

        :param cpack_name: The name of the conformance pack to create controls for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack = self.config_client.describe_conformance_pack_compliance(
                ConformancePackName=cpack_name
            )
            print(
                "Number of rules in the conformance pack: ",
                len(rules_in_pack.get("ConformancePackRuleComplianceList")),
            )
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
```

```
                print(f"\t{rule.get('ConfigRuleName')}")
            print("-" * 88)
            print(
                "Creating a custom control for each rule and a custom framework "
                "consisting of these rules in Audit Manager."
            )
            am_controls = []
            for rule in rules_in_pack.get("ConformancePackRuleComplianceList"):
                config_rule = self.config_client.describe_config_rules(
                    ConfigRuleNames=[rule.get("ConfigRuleName")]
                )
                source_id = (
                    config_rule.get("ConfigRules")[0]
                    .get("Source", {})
                    .get("SourceIdentifier")
                )
                custom_control = self.auditmanager_client.create_control(
                    name="Config-" + rule.get("ConfigRuleName"),
                    controlMappingSources=[
                        {
                            "sourceName": "ConfigRule",
                            "sourceSetUpOption": "System_Controls_Mapping",
                            "sourceType": "AWS_Config",
                            "sourceKeyword": {
                                "keywordInputType": "SELECT_FROM_LIST",
                                "keywordValue": source_id,
                            },
                        }
                    ],
                ).get("control", {})
                am_controls.append({"id": custom_control.get("id")})
            print("Successfully created a control for each config rule.")
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom controls.")
            raise
        else:
            return am_controls

    def create_custom_framework(self, cpack_name, am_control_ids):
        """
        Create a custom Audit Manager framework from a selected AWS Config
conformance
        pack.
```

```
        :param cpack_name: The name of the conformance pack to create a framework
 from.
        :param am_control_ids: The IDs of the custom controls created from the
                               conformance pack.
        """
        try:
            print("Creating custom framework...")
            custom_framework = self.auditmanager_client.create_assessment_framework(
                name="Config-Conformance-pack-" + cpack_name,
                controlSets=[{"name": cpack_name, "controls": am_control_ids}],
            )
            print(
                f"Successfully created the custom framework: ",
                f"{custom_framework.get('framework').get('name')}: ",
                f"{custom_framework.get('framework').get('id')}",
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print("-" * 88)
    print(
        "You can use this sample to select a conformance pack from AWS Config and "
        "use AWS Audit Manager to create a custom control for all the managed "
        "rules under the conformance pack. A custom framework is also created "
        "with these controls."
    )
    print("-" * 88)
    conf_pack = ConformancePack(boto3.client("config"),
 boto3.client("auditmanager"))
    cpack_name = conf_pack.get_conformance_pack()
    am_controls = conf_pack.create_custom_controls(cpack_name)
    conf_pack.create_custom_framework(cpack_name, am_controls)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateAssessmentFramework
  - CreateControl

Erstellen Sie ein benutzerdefiniertes Framework, das Security Hub CSPM-Steuerelemente enthält

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie eine Liste aller Standardsteuerelemente ab, deren Datenquelle Security Hub CSPM ist.
- Erstellen eines benutzerdefinierten Audit-Manager-Frameworks, das die Steuerelemente enthält

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
        """
```

```
        print("-" * 88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", maxResults=100
                )
            else:
                control_list = self.auditmanager_client.list_controls(
                    controlType="Standard", nextToken=next_token, maxResults=100
                )
            print("Total controls found:",
len(control_list.get("controlMetadataList")))
            for control in control_list.get("controlMetadataList"):
                control_details = self.auditmanager_client.get_control(
                    controlId=control.get("id")
                ).get("control", {})
                if "AWS Security Hub" in control_details.get("controlSources"):
                    sechub_control_list.append({"id": control_details.get("id")})
            next_token = control_list.get("nextToken")
            if not next_token:
                break
            page += 1
        print("Number of Security Hub controls found: ", len(sechub_control_list))
        return sechub_control_list

    def create_custom_framework(self, am_controls):
        """
        Create a custom framework with a list of controls.

        :param am_controls: The list of controls to include in the framework.
        """
        try:
            print("Creating custom framework...")
            custom_framework = self.auditmanager_client.create_assessment_framework(
                name="All Security Hub Controls Framework",
                controlSets=[{"name": "Security-Hub", "controls": am_controls}],
            )
            print(
                f"Successfully created the custom framework: "
                f"{custom_framework.get('framework').get('name')}: "
```

```
                f"{custom_framework.get('framework').get('id')}"
            )
            print("-" * 88)
        except ClientError:
            logger.exception("Failed to create custom framework.")
            raise


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print("-" * 88)
    print(" This script creates a custom framework with all Security Hub controls.")
    print("-" * 88)
    sechub = SecurityHub(boto3.client("auditmanager"))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Python (Boto3).
  - CreateAssessmentFramework
  - GetControl
  - ListControls

Erstellen eines Bewertungberichts

Das folgende Codebeispiel zeigt, wie Sie einen Audit-Manager-Bewertungsbericht mit eintägigen Nachweisen erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
import boto3
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print("-" * 40)
        try:
            assessment_id = input("Provide assessment id [uuid]: ").lower()
            try:
                assessment_uuid = uuid.UUID(assessment_id)
            except ValueError:
                logger.error("Assessment Id is not a valid UUID: %s", assessment_id)
                raise
            evidence_folder = input("Provide evidence date [yyyy-mm-dd]: ")
            try:
                evidence_date = dateutil.parser.parse(evidence_folder).date()
            except ValueError:
                logger.error("Invalid date : %s", evidence_folder)
                raise
            try:
                self.auditmanager_client.get_assessment(
                    assessmentId=str(assessment_uuid)
                )
            except ClientError:
                logger.exception("Couldn't get assessment %s.", assessment_uuid)
                raise
        except (ValueError, ClientError):
            return None, None
        else:
            return assessment_uuid, evidence_date
```

```
    def clear_staging(self, assessment_uuid, evidence_date):
        """
        Find all the evidence in the report and clear it.
        """
        next_token = None
        page = 1
        interested_folder_id_list = []
        while True:
            print(f"Page [{page}]")
            if next_token is None:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid), maxResults=1000
                    )
                )
            else:
                folder_list = (
                    self.auditmanager_client.get_evidence_folders_by_assessment(
                        assessmentId=str(assessment_uuid),
                        nextToken=next_token,
                        maxResults=1000,
                    )
                )
            folders = folder_list.get("evidenceFolders")
            print(f"Got {len(folders)} folders.")
            for folder in folders:
                folder_id = folder.get("id")
                if folder.get("name") == str(evidence_date):
                    interested_folder_id_list.append(folder_id)
                if folder.get("assessmentReportSelectionCount") == folder.get(
                    "totalEvidence"
                ):
                    print(
                        f"Removing folder from report selection :
{folder.get('name')} "
                        f"{folder_id} {folder.get('controlId')}"
                    )

self.auditmanager_client.disassociate_assessment_report_evidence_folder(
                        assessmentId=str(assessment_uuid),
evidenceFolderId=folder_id
                    )
                elif folder.get("assessmentReportSelectionCount") > 0:
                    # Get all evidence in the folder and
```

```
                    # add selected evidence in the selected_evidence_list.
                    evidence_list = (
                        self.auditmanager_client.get_evidence_by_evidence_folder(
                            assessmentId=str(assessment_uuid),
                            controlSetId=folder_id,
                            evidenceFolderId=folder_id,
                            maxResults=1000,
                        )
                    )
                    selected_evidence_list = []
                    for evidence in evidence_list.get("evidence"):
                        if evidence.get("assessmentReportSelection") == "Yes":
                            selected_evidence_list.append(evidence.get("id"))
                    print(
                        f"Removing evidence report selection : {folder.get('name')}
"
                        f"{len(selected_evidence_list)}"
                    )

self.auditmanager_client.batch_disassociate_assessment_report_evidence(
                        assessmentId=str(assessment_uuid),
                        evidenceFolderId=folder_id,
                        evidenceIds=selected_evidence_list,
                    )
            next_token = folder_list.get("nextToken")
            if not next_token:
                break
            page += 1
        return interested_folder_id_list

    def add_folder_to_staging(self, assessment_uuid, folder_id_list):
        print(f"Adding folders to report : {folder_id_list}")
        for folder in folder_id_list:
            self.auditmanager_client.associate_assessment_report_evidence_folder(
                assessmentId=str(assessment_uuid), evidenceFolderId=folder
            )

    def get_report(self, assessment_uuid):
        report = self.auditmanager_client.create_assessment_report(
            name="ReportViaScript",
            description="testing",
            assessmentId=str(assessment_uuid),
        )
        if self._is_report_generated(report.get("assessmentReport").get("id")):
```

```
            report_url = self.auditmanager_client.get_assessment_report_url(
                assessmentReportId=report.get("assessmentReport").get("id"),
                assessmentId=str(assessment_uuid),
            )
            print(report_url.get("preSignedUrl"))
            urllib.request.urlretrieve(
                report_url.get("preSignedUrl").get("link"),
                report_url.get("preSignedUrl").get("hyperlinkName"),
            )
            print(
                f"Report saved as
 {report_url.get('preSignedUrl').get('hyperlinkName')}."
            )
        else:
            print("Report generation did not finish in 15 minutes.")
            print(
                "Failed to download report. Go to the console and manually download
 "
                "the report."
            )

    def _is_report_generated(self, assessment_report_id):
        max_wait_time = 0
        while max_wait_time < 900:
            print(f"Checking status of the report {assessment_report_id}")
            report_list =
 self.auditmanager_client.list_assessment_reports(maxResults=1)
            if (
                report_list.get("assessmentReports")[0].get("id")
                == assessment_report_id
                and report_list.get("assessmentReports")[0].get("status") ==
 "COMPLETE"
            ):
                return True
            print("Sleeping for 5 seconds...")
            time.sleep(5)
            max_wait_time += 5


def run_demo():
    print("-" * 88)
    print("Welcome to the AWS Audit Manager samples demo!")
    print("-" * 88)
    print(
```

```
        "This script creates an assessment report for an assessment with all the "
        "evidence collected on the provided date."
    )
    print("-" * 88)

    report = AuditReport(boto3.client("auditmanager"))
    assessment_uuid, evidence_date = report.get_input()
    if assessment_uuid is not None and evidence_date is not None:
        folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
        report.add_folder_to_staging(assessment_uuid, folder_id_list)
        report.get_report(assessment_uuid)


if __name__ == "__main__":
    run_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - AssociateAssessmentReportEvidenceFolder
  - BatchDisassociateAssessmentReportEvidence
  - CreateAssessmentReport
  - DisassociateAssessmentReportEvidenceFolder
  - GetAssessment
  - GetAssessmentReportUrl
  - GetEvidenceByEvidenceFolder
  - GetEvidenceFoldersByAssessment
  - ListAssessmentReports

Aurora-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")
```

```
# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database engine
 and
        create a DB cluster parameter group that is compatible with a selected
 engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created parameter
 group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
 {parameter_group_name}."
        )
        parameter_group =
 self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
 engine_versions})
            family_index = q.choose("Which family do you want to use? ", families)
```

```
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name, update_params)
        print(
            "You can get a list of parameters you've set by specifying a source of
'user'."
```

```
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster parameter
group.

        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the DB
cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ", q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters): ",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
```

```
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new DB
cluster
        contains no DB instances, so you must add one. The first DB instance that is
added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
```

```
                {
                    opt["DBInstanceClass"] + ", storage type: " + opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
 minutes."
            )
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on how
 to
        connect to it.

        :param cluster: The DB cluster to display.
        """
        print(
            "You can now connect to your database using your favorite MySql client.
\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password\n"
            "when prompted:\n"
```

```
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Co
        )
        print("-" * 88)

    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
 group.
        Before the DB cluster parameter group can be deleted, all associated DB
 instances and
        DB clusters must first be deleted.
```

```
        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting database instance {db_inst['DBInstanceIdentifier']}.")
            self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, cluster_name, db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
```

```
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
 parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Aurora-Aktionen zu verwalten.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
```

```
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
```

```
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
```

```
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why: %s:
%s",
                    cluster_name,
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created, such
as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
```

```
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
```

```
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
```

```
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
```

```
                                        this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
```

```
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions

- ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
```

```
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created, such
as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
```

```
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDBClusterSnapshot`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDBInstance`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
```

```
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteDBInstance`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBClusterParameters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
 filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
 retrieved.
                       For example, a source of 'user' retrieves only parameters
 that
                       were set by a user.
        :return: The list of requested parameters.
        """
```

```
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBClusterSnapshots`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- API-Details finden Sie DescribeDBClusterSnapshots in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBClusters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
```

```
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why: %s: 
%s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
```

```
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
```

```
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return inst_opts
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS API-Referenz zum SDK für Python (Boto3).

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung. `ModifyDBClusterParameterGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
 client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
```

```
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

SDK für Python (Boto3)

Zeigt, wie Sie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) -API und AWS Chalice verwenden, um eine REST-API zu erstellen, die von einer Amazon Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldeinformationen verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldeinformationen enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling client
 and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) -> None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")
```

```
    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) + 1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name, 1,
1)
    wait(10)
```

```
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1 to
3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
```

```
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities), group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)
```

```
    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value": group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
```

```
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
 demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Startvorlagen und Metriken zu verwalten. Diese Funktionen umfassen Amazon EC2 und CloudWatch Aktionen.

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""
```

```
    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
            logger.info("Launch template %s retrieved successfully.", template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.", template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
```

```
        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId": ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID %s.",
                template_name,
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:
            self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
```

```
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
```

```
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids else
[]
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
            f"Here are the details of the instance{'s' if len(instances) > 1 else
''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`AttachLoadBalancerTargetGroups`

Das folgende Codebeispiel zeigt die Verwendung`AttachLoadBalancerTargetGroups`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
```

```
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
```

```
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie AttachLoadBalancerTargetGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAutoScalingGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the configuration
of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )
```

```
            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
 exists. "
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch configurations.
 "
                    "Consider deleting unused resources or request a limit increase.
 "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-
auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAutoScalingGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie die Mindestgröße einer Auto-Scaling-Gruppe auf Null, beenden Sie alle Instances in der Gruppe und löschen Sie die Gruppe.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
```

```
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeAutoScalingGroups`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
```

```
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found, otherwise
None.
        :raises ClientError: If there is an error describing the Auto Scaling group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

```
        else:
            return groups[0] if len(groups) > 0 else None
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeAutoScalingInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
```

```
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeScalingActivities`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
```

```
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str, Any]]:
        """
        Gets information about scaling activities for the group. Scaling activities
        are things like instances stopping or starting in response to user requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator = paginator.paginate(AutoScalingGroupName=group_name)
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
```

```
            raise
        else:
            return activities
```

- API-Details finden Sie DescribeScalingActivities in der AWS API-Referenz zum SDK für Python (Boto3).

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. `DisableMetricsCollection`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.

        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
 collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
```

```
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
'{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- API-Details finden Sie DisableMetricsCollection in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung. `EnableMetricsCollection`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
```

```
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str, Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is modifying
the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
                logger.error(
```

```
                    f"The combination of parameters provided for enabling metrics on
'{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- API-Details finden Sie EnableMetricsCollection in der AWS API-Referenz zum SDK für Python (Boto3).

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung. `SetDesiredCapacity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
keep the
        number of running instances equal to the desired capacity.
```

```
        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto Scaling
group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting to
set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API-Details finden Sie SetDesiredCapacity in der AWS API-Referenz zum SDK für Python (Boto3).

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `TerminateInstanceInAutoScalingGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired capacity
                                  of the group. When passing True for this
 parameter,
                                  you can stop an instance without having a
 replacement
                                  instance start when the desired capacity threshold
 is
                                  crossed.
        :return: A dictionary containing details of the scaling activity that occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
 self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
 ShouldDecrementDesiredCapacity=decrease_capacity
```

```
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting to
terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateAutoScalingGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
                    "The Auto Scaling group '%s' is currently in use and cannot be
 modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting to
 update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die Parameter aktualisieren AWS Systems Manager .

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
```

```
    """
    Manages the deployment, demonstration, and destruction of resources for the
resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
```

```
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
```

```
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
 this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
 your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
```

```
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
```

```
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
```

```
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()
```

```
        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
to avoid unexpected charges."
```

```
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
 and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
```

```
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
```

```
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
```

```
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
```

```
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
```

```
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
```

```
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
```

```
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
```

```
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
```

```
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
```

```
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
```

```
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

```
    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.
```

```
        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)
```

```
            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
```

```
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
```

```
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
```

```
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
```

```
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
```

```
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
```

```
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
        lower thresholds. In production, you might want to decrease the sensitivity
 of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
```

```
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
```

```
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
```

```
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
```

```
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
```

```
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.
```

```
        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
```

```
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
```

```
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.
```

```
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
```

```
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
```

```
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate

- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Beispiele für Amazon Bedrock unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Aktionen
- Szenarien

Erste Schritte

Hallo Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock beginnen können.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Lists the available Amazon Bedrock models.
"""
import logging
import json
import boto3


from botocore.exceptions import ClientError


logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def list_foundation_models(bedrock_client):
    """
    Gets a list of available Amazon Bedrock foundation models.

    :return: The list of available bedrock foundation models.
    """

    try:
```

```
        response = bedrock_client.list_foundation_models()
        models = response["modelSummaries"]
        logger.info("Got %s foundation models.", len(models))
        return models

    except ClientError:
        logger.error("Couldn't list foundation models.")
        raise


def main():
    """Entry point for the example. Uses the AWS SDK for Python (Boto3)
    to create an Amazon Bedrock client. Then lists the available Bedrock models
    in the region set in the callers profile and credentials.
    """

    bedrock_client = boto3.client(service_name="bedrock")

    fm_models = list_foundation_models(bedrock_client)
    for model in fm_models:
        print(f"Model: {model['modelName']}")
        print(json.dumps(model, indent=2))
        print("---------------------------\n")

    logger.info("Done.")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ListFoundationModels in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`GetFoundationModel`

Das folgende Codebeispiel zeigt die Verwendung. `GetFoundationModel`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie Details über ein Basismodell ab.

```
    def get_foundation_model(self, model_identifier):
        """
        Get details about an Amazon Bedrock foundation model.

        :return: The foundation model's details.
        """

        try:
            return self.bedrock_client.get_foundation_model(
                modelIdentifier=model_identifier
            )["modelDetails"]
        except ClientError:
            logger.error(
                f"Couldn't get foundation models details for {model_identifier}"
            )
            raise
```

- API-Details finden Sie GetFoundationModel in der AWS API-Referenz zum SDK für Python (Boto3).

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung. `ListFoundationModels`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der verfügbaren Amazon-Bedrock-Basismodelle.

```
    def list_foundation_models(self):
        """
        List the available Amazon Bedrock foundation models.

        :return: The list of available bedrock foundation models.
        """

        try:
            response = self.bedrock_client.list_foundation_models()
            models = response["modelSummaries"]
            logger.info("Got %s foundation models.", len(models))
            return models

        except ClientError:
            logger.error("Couldn't list foundation models.")
            raise
```

- API-Details finden Sie ListFoundationModels in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Orchestrieren von Anwendungen mit generativer KI mit Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finde und fasse das aktuelle Repositorium mit den höchsten Trends zusammen. GitHub Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step-Funktionen

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien
- Amazon Nova
- Amazon Nova Canvas
- Amazon Nova Reel
- Amazon Titan Image Generator
- Amazon Titan Text
- Amazon Titan Text Embeddings
- Anthropic Claude
- Cohere Command
- DeepSeek
- Meta Llama
- Mistral AI
- Stable Image Core

Szenarien

Erstellen einer Playground-Anwendung für die Interaktion mit Amazon-Bedrock-Basismodellen

Das folgende Codebeispiel zeigt, wie Sie Playgrounds erstellen, um mit Amazon-Bedrock-Basismodellen über verschiedene Modalitäten zu interagieren.

SDK für Python (Boto3)

Der Python Foundation Model (FM) Playground ist eine Python/FastAPI Beispielanwendung, die zeigt, wie Amazon Bedrock mit Python verwendet wird. Dieses Beispiel zeigt, wie Python-Entwickler Amazon Bedrock verwenden können, um generative Anwendungen zu erstellen. AI-enabled Sie können die Amazon-Bedrock-Basismodelle in den folgenden drei Playgrounds testen und mit ihnen interagieren:

- Ein Text-Playground.
- Ein Chat-Playground.
- Ein Image-Playground.

In diesem Beispiel werden zudem die Basismodelle, auf die Sie Zugriff haben, sowie deren Eigenschaften aufgelistet und angezeigt. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHub Projekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Bedrock Runtime

Erstellen und Aufrufen verwalteter Prompts

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines verwalteten Prompts.
- Erstellen einer Version des Prompts.
- Aufrufen des Prompts mithilfe der Version.
- Bereinigen der Ressourcen (optional).

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines verwalteten Prompts.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None

    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of songs:
 {{number}}."""

        create_response = create_prompt(
```

```
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")
```

```
            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
```

```
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
 region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Orchestrieren von Anwendungen mit generativer KI mit Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finde und fasse das aktuelle Repositorium mit den höchsten Trends GitHub zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step-Funktionen

Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""
```

```
import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"


# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
```

```
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
 Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
```

```
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )

            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
```

```
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
```

```
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])

                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.
```

```
        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
```

```
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
```

```
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
 API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
# Use the Conversation API to send a text message to Amazon Nova.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
```

```
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Amazon Nova Text
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
```

```
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- API-Details finden Sie ConverseStream in der AWS API-Referenz zum SDK für Python (Boto3).

Verstehen von Dokumenten

In dem folgenden Codebeispiele wird gezeigt, wie ein Dokument mit Amazon Nova in Amazon Bedrock gesendet und verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Amazon Nova in Amazon Bedrock.

```
# Send and process a document with Amazon Nova on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Amazon Nova Lite.
model_id = "amazon.nova-lite-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
```

```
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Amazon Nova Canvas

InvokeModel

Das folgende Codebeispiel zeigt, wie Amazon Nova Canvas auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
# Use the native inference API to create an image with Amazon Nova Canvas

import base64
import json
import os
import random

import boto3

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID.
```

```
model_id = "amazon.nova-canvas-v1:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed between 0 and 858,993,459
seed = random.randint(0, 858993460)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "seed": seed,
        "quality": "standard",
        "height": 512,
        "width": 512,
        "numberOfImages": 1,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"nova_canvas_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"nova_canvas_{i}.png")
with open(image_path, "wb") as file:
```

```
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- API-Details finden Sie InvokeModel in der AWS API-Referenz zum SDK für Python (Boto3).

Amazon Nova Reel

Text-to-video

Das folgende Codebeispiel zeigt, wie Amazon Nova Reel zur Generierung eines Videos anhand eines Textprompts verwendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren.

```
"""
This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.

It shows how to:
- Set up the Amazon Bedrock runtime client
- Configure a text-to-video request
- Submit an asynchronous job for video generation
- Poll for job completion status
- Access the generated video from S3
"""

import random
import time

import boto3
```

```
# Replace with your own S3 bucket to store the generated video
# Format: s3://your-bucket-name
OUTPUT_S3_URI = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAME"


def start_text_to_video_generation_job(bedrock_runtime, prompt, output_s3_uri):
    """
    Starts an asynchronous text-to-video generation job using Amazon Nova Reel.

    :param bedrock_runtime: The Bedrock runtime client
    :param prompt: The text description of the video to generate
    :param output_s3_uri: S3 URI where the generated video will be stored

    :return: The invocation ARN of the async job
    """
    # Specify the model ID for text-to-video generation
    model_id = "amazon.nova-reel-v1:0"

    # Generate a random seed between 0 and 2,147,483,646
    # This helps ensure unique video generation results
    seed = random.randint(0, 2147483646)

    # Configure the video generation request with additional parameters
    model_input = {
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": {"text": prompt},
        "videoGenerationConfig": {
            "fps": 24,
            "durationSeconds": 6,
            "dimension": "1280x720",
            "seed": seed,
        },
    }

    # Specify the S3 location for the output video
    output_config = {"s3OutputDataConfig": {"s3Uri": output_s3_uri}}

    # Invoke the model asynchronously
    response = bedrock_runtime.start_async_invoke(
        modelId=model_id, modelInput=model_input, outputDataConfig=output_config
    )

    invocation_arn = response["invocationArn"]
```

```
    return invocation_arn


def query_job_status(bedrock_runtime, invocation_arn):
    """
    Queries the status of an asynchronous video generation job.

    :param bedrock_runtime: The Bedrock runtime client
    :param invocation_arn: The ARN of the async invocation to check

    :return: The runtime response containing the job status and details
    """
    return bedrock_runtime.get_async_invoke(invocationArn=invocation_arn)


def main():
    """
    Main function that demonstrates the complete workflow for generating
    a video from a text prompt using Amazon Nova Reel.
    """
    # Create a Bedrock Runtime client
    # Note: Credentials will be loaded from the environment or AWS CLI config
    bedrock_runtime = boto3.client("bedrock-runtime", region_name="us-east-1")

    # Configure the text prompt and output location
    prompt = "Closeup of a cute old steampunk robot. Camera zoom in."

    # Verify the S3 URI has been set to a valid bucket
    if "REPLACE-WITH-YOUR-S3-BUCKET-NAME" in OUTPUT_S3_URI:
        print("ERROR: You must replace the OUTPUT_S3_URI with your own S3 bucket
 URI")
        return

    print("Submitting video generation job...")
    invocation_arn = start_text_to_video_generation_job(
        bedrock_runtime, prompt, OUTPUT_S3_URI
    )
    print(f"Job started with invocation ARN: {invocation_arn}")

    # Poll for job completion
    while True:
        print("\nPolling job status...")
        job = query_job_status(bedrock_runtime, invocation_arn)
        status = job["status"]
```

```
        if status == "Completed":
            bucket_uri = job["outputDataConfig"]["s3OutputDataConfig"]["s3Uri"]
            print(f"\nSuccess! The video is available at: {bucket_uri}/output.mp4")
            break
        elif status == "Failed":
            print(
                f"\nVideo generation failed: {job.get('failureMessage', 'Unknown
 error')}"
            )
            break
        else:
            print("In progress. Waiting 15 seconds...")
            time.sleep(15)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - GetAsyncInvoke
  - StartAsyncInvoke

Amazon Titan Image Generator

InvokeModel

Die folgenden Codebeispiele zeigen, wie Sie Amazon Titan Image in Amazon Bedrock aufrufen, um ein Bild zu generieren.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit dem Amazon Titan Image Generator.

```
# Use the native inference API to create an image with Amazon Titan Image Generator

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Image Generator G1.
model_id = "amazon.titan-image-generator-v2:0"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 2147483647)

# Format the request payload using the model's native structure.
native_request = {
    "taskType": "TEXT_IMAGE",
    "textToImageParams": {"text": prompt},
    "imageGenerationConfig": {
        "numberOfImages": 1,
        "quality": "standard",
        "cfgScale": 8.0,
        "height": 512,
        "width": 512,
        "seed": seed,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())
```

```
# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"titan_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"titan_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- API-Details finden Sie InvokeModel in der AWS API-Referenz zum SDK für Python (Boto3).

Amazon Titan Text

InvokeModel

Das folgende Codebeispiel zeigt, wie mit der Invoke-Model-API eine Textnachricht an Amazon Titan Text gesendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden der API zum Aufrufen eines Modells zum Senden einer Textnachricht.

```
# Use the native inference API to send a text message to Amazon Titan Text.
```

```
import boto3
import json

from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Premier.
model_id = "amazon.titan-text-premier-v1:0"

# Define the prompt for the model.
prompt = "Describe the purpose of a 'hello world' program in one line."

# Format the request payload using the model's native structure.
native_request = {
    "inputText": prompt,
    "textGenerationConfig": {
        "maxTokenCount": 512,
        "temperature": 0.5,
    },
}

# Convert the native request to JSON.
request = json.dumps(native_request)

try:
    # Invoke the model with the request.
    response = client.invoke_model(modelId=model_id, body=request)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract and print the response text.
response_text = model_response["results"][0]["outputText"]
print(response_text)
```

- API-Details finden Sie InvokeModel in der AWS API-Referenz zum SDK für Python (Boto3).

Amazon Titan Text Embeddings

InvokeModel

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Beginnen Sie mit der Erstellung Ihrer ersten Einbettung.
- Erstellen Sie Einbettungen, indem Sie die Anzahl der Dimensionen und die Normalisierung konfigurieren (nur V2).

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Ihre erste Einbettung mit Amazon Titan Text Embeddings.

```
# Generate and print an embedding with Amazon Titan Text Embeddings V2.

import boto3
import json

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Titan Text Embeddings V2.
model_id = "amazon.titan-embed-text-v2:0"

# The text to convert to an embedding.
input_text = "Please recommend books with a theme similar to the movie 'Inception'."

# Create the request for the model.
native_request = {"inputText": input_text}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
```

```
response = client.invoke_model(modelId=model_id, body=request)

# Decode the model's native response body.
model_response = json.loads(response["body"].read())

# Extract and print the generated embedding and the input text token count.
embedding = model_response["embedding"]
input_token_count = model_response["inputTextTokenCount"]

print("\nYour input:")
print(input_text)
print(f"Number of input tokens: {input_token_count}")
print(f"Size of the generated embedding: {len(embedding)}")
print("Embedding:")
print(embedding)
```

- API-Details finden Sie InvokeModel in der AWS API-Referenz zum SDK für Python (Boto3).

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
# Use the Conversation API to send a text message to Anthropic Claude.

import boto3
from botocore.exceptions import ClientError
```

```
# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
# Use the Conversation API to send a text message to Anthropic Claude
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
```

```
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- API-Details finden Sie ConverseStream in der AWS API-Referenz zum SDK für Python (Boto3).

Verstehen von Dokumenten

Das folgende Codebeispiel zeigt, wie ein Dokument mit Anthropic Claude in Amazon Bedrock gesendet und verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und Verarbeiten eines Dokuments mit Anthropic Claude in Amazon Bedrock.

```
# Send and process a document with Anthropic Claude on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Claude 3 Haiku.
model_id = "anthropic.claude-3-haiku-20240307-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
```

```
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"
```

```
# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
```

```
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )
```

```
            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
```

```
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
```

```
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
 API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
```

```
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Cohere Command

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet wird.

SDK für Python (Boto3)

> **Note**
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command.

```
# Use the Conversation API to send a text message to Cohere Command.

import boto3
from botocore.exceptions import ClientError
```

```
# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Cohere Command gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Cohere Command und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Cohere Command
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Command R.
model_id = "cohere.command-r-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
```

```
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- API-Details finden Sie ConverseStream in der AWS API-Referenz zum SDK für Python (Boto3).

Verstehen von Dokumenten

Das folgende Codebeispiel zeigt, wie man ein Dokument mit Cohere-Command-Modellen in Amazon Bedrock sendet und verarbeitet.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Cohere-Command-Modellen in Amazon Bedrock.

```
# Send and process a document with Cohere Command models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Command R+.
model_id = "cohere.command-r-plus-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
```

```
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Ausführungsskript der Demo. Dieses Skript orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
"""
This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
The script interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
"""

import boto3
import logging
from enum import Enum

import utils.tool_use_print_utils as output
import weather_tool

logging.basicConfig(level=logging.INFO, format="%(message)s")

AWS_REGION = "us-east-1"
```

```
# For the most recent list of models supported by the Converse API's tool use
 functionality, visit:
# https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference.html
class SupportedModels(Enum):
    CLAUDE_OPUS = "anthropic.claude-3-opus-20240229-v1:0"
    CLAUDE_SONNET = "anthropic.claude-3-sonnet-20240229-v1:0"
    CLAUDE_HAIKU = "anthropic.claude-3-haiku-20240307-v1:0"
    COHERE_COMMAND_R = "cohere.command-r-v1:0"
    COHERE_COMMAND_R_PLUS = "cohere.command-r-plus-v1:0"


# Set the model ID, e.g., Claude 3 Haiku.
MODEL_ID = SupportedModels.CLAUDE_HAIKU.value

SYSTEM_PROMPT = """
You are a weather assistant that provides current weather data for user-specified
 locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
"""

# The maximum number of recursive calls allowed in the tool_use_demo function.
# This helps prevent infinite loops and potential performance issues.
MAX_RECURSIONS = 5


class ToolUseDemo:
```

```
    """
    Demonstrates the tool use feature with the Amazon Bedrock Converse API.
    """

    def __init__(self):
        # Prepare the system prompt
        self.system_prompt = [{"text": SYSTEM_PROMPT}]

        # Prepare the tool configuration with the weather tool's specification
        self.tool_config = {"tools": [weather_tool.get_tool_spec()]}

        # Create a Bedrock Runtime client in the specified AWS Region.
        self.bedrockRuntimeClient = boto3.client(
            "bedrock-runtime", region_name=AWS_REGION
        )

    def run(self):
        """
        Starts the conversation with the user and handles the interaction with
Bedrock.
        """
        # Print the greeting and a short user guide
        output.header()

        # Start with an emtpy conversation
        conversation = []

        # Get the first user input
        user_input = self._get_user_input()

        while user_input is not None:
            # Create a new message with the user input and append it to the
conversation
            message = {"role": "user", "content": [{"text": user_input}]}
            conversation.append(message)

            # Send the conversation to Amazon Bedrock
            bedrock_response = self._send_conversation_to_bedrock(conversation)

            # Recursively handle the model's response until the model has returned
            # its final response or the recursion counter has reached 0
            self._process_model_response(
                bedrock_response, conversation, max_recursion=MAX_RECURSIONS
            )
```

```
            # Repeat the loop until the user decides to exit the application
            user_input = self._get_user_input()

        output.footer()

    def _send_conversation_to_bedrock(self, conversation):
        """
        Sends the conversation, the system prompt, and the tool spec to Amazon
Bedrock, and returns the response.

        :param conversation: The conversation history including the next message to
send.
        :return: The response from Amazon Bedrock.
        """
        output.call_to_bedrock(conversation)

        # Send the conversation, system prompt, and tool configuration, and return
the response
        return self.bedrockRuntimeClient.converse(
            modelId=MODEL_ID,
            messages=conversation,
            system=self.system_prompt,
            toolConfig=self.tool_config,
        )

    def _process_model_response(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Processes the response received via Amazon Bedrock and performs the
necessary actions
        based on the stop reason.

        :param model_response: The model's response returned via Amazon Bedrock.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        if max_recursion <= 0:
            # Stop the process, the number of recursive calls could indicate an
infinite loop
            logging.warning(
                "Warning: Maximum number of recursions reached. Please try again."
```

```
            )
            exit(1)

        # Append the model's response to the ongoing conversation
        message = model_response["output"]["message"]
        conversation.append(message)

        if model_response["stopReason"] == "tool_use":
            # If the stop reason is "tool_use", forward everything to the tool use
handler
            self._handle_tool_use(message, conversation, max_recursion)

        if model_response["stopReason"] == "end_turn":
            # If the stop reason is "end_turn", print the model's response text, and
finish the process
            output.model_response(message["content"][0]["text"])
            return

    def _handle_tool_use(
        self, model_response, conversation, max_recursion=MAX_RECURSIONS
    ):
        """
        Handles the tool use case by invoking the specified tool and sending the
tool's response back to Bedrock.
        The tool response is appended to the conversation, and the conversation is
sent back to Amazon Bedrock for further processing.

        :param model_response: The model's response containing the tool use request.
        :param conversation: The conversation history.
        :param max_recursion: The maximum number of recursive calls allowed.
        """

        # Initialize an empty list of tool results
        tool_results = []

        # The model's response can consist of multiple content blocks
        for content_block in model_response["content"]:
            if "text" in content_block:
                # If the content block contains text, print it to the console
                output.model_response(content_block["text"])

            if "toolUse" in content_block:
                # If the content block is a tool use request, forward it to the tool
                tool_response = self._invoke_tool(content_block["toolUse"])
```

```
                # Add the tool use ID and the tool's response to the list of results
                tool_results.append(
                    {
                        "toolResult": {
                            "toolUseId": (tool_response["toolUseId"]),
                            "content": [{"json": tool_response["content"]}],
                        }
                    }
                )

        # Embed the tool results in a new user message
        message = {"role": "user", "content": tool_results}

        # Append the new message to the ongoing conversation
        conversation.append(message)

        # Send the conversation to Amazon Bedrock
        response = self._send_conversation_to_bedrock(conversation)

        # Recursively handle the model's response until the model has returned
        # its final response or the recursion counter has reached 0
        self._process_model_response(response, conversation, max_recursion - 1)

    def _invoke_tool(self, payload):
        """
        Invokes the specified tool with the given payload and returns the tool's
response.
        If the requested tool does not exist, an error message is returned.

        :param payload: The payload containing the tool name and input data.
        :return: The tool's response or an error message.
        """
        tool_name = payload["name"]

        if tool_name == "Weather_Tool":
            input_data = payload["input"]
            output.tool_use(tool_name, input_data)

            # Invoke the weather tool with the input data provided by
            response = weather_tool.fetch_weather_data(input_data)
        else:
            error_message = (
                f"The requested tool with name '{tool_name}' does not exist."
```

```
            )
            response = {"error": "true", "message": error_message}

        return {"toolUseId": payload["toolUseId"], "content": response}

    @staticmethod
    def _get_user_input(prompt="Your weather info request"):
        """
        Prompts the user for input and returns the user's response.
        Returns None if the user enters 'x' to exit.

        :param prompt: The prompt to display to the user.
        :return: The user's input or None if the user chooses to exit.
        """
        output.separator()
        user_input = input(f"{prompt} (x to exit): ")

        if user_input == "":
            prompt = "Please enter your weather info request, e.g. the name of a
 city"
            return ToolUseDemo._get_user_input(prompt)

        elif user_input.lower() == "x":
            return None

        else:
            return user_input


if __name__ == "__main__":
    tool_use_demo = ToolUseDemo()
    tool_use_demo.run()
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
import requests
from requests.exceptions import RequestException


def get_tool_spec():
```

```
    """
    Returns the JSON Schema specification for the Weather tool. The tool
 specification
    defines the input schema and describes the tool's functionality.
    For more information, see https://json-schema.org/understanding-json-schema/
reference.

    :return: The tool specification for the Weather tool.
    """
    return {
        "toolSpec": {
            "name": "Weather_Tool",
            "description": "Get the current weather for a given location, based on
 its WGS84 coordinates.",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "latitude": {
                            "type": "string",
                            "description": "Geographical WGS84 latitude of the
 location.",
                        },
                        "longitude": {
                            "type": "string",
                            "description": "Geographical WGS84 longitude of the
 location.",
                        },
                    },
                    "required": ["latitude", "longitude"],
                }
            },
        }
    }


def fetch_weather_data(input_data):
    """
    Fetches weather data for the given latitude and longitude using the Open-Meteo
 API.
    Returns the weather data or an error message if the request fails.

    :param input_data: The input data containing the latitude and longitude.
    :return: The weather data or an error message.
```

```
    """
    endpoint = "https://api.open-meteo.com/v1/forecast"
    latitude = input_data.get("latitude")
    longitude = input_data.get("longitude", "")
    params = {"latitude": latitude, "longitude": longitude, "current_weather": True}

    try:
        response = requests.get(endpoint, params=params)
        weather_data = {"weather_data": response.json()}
        response.raise_for_status()
        return weather_data
    except RequestException as e:
        return e.response.json()
    except Exception as e:
        return {"error": type(e), "message": str(e)}
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

DeepSeek

Verstehen von Dokumenten

Das folgende Codebeispiel zeigt, wie ein Dokument mit DeepSeek Amazon Bedrock gesendet und verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit DeepSeek Amazon Bedrock.

```
# Send and process a document with DeepSeek on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError
```

```
# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. DeepSeek-R1
model_id = "us.deepseek.r1-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 2000, "temperature": 0.3},
    )

    # Extract and print the reasoning and response text.
    reasoning, response_text = "", ""
    for item in response["output"]["message"]["content"]:
        for key, value in item.items():
            if key == "reasoningContent":
                reasoning = value["reasoningText"]["text"]
            elif key == "text":
```

```
                response_text = value

    print(f"\nReasoning:\n{reasoning}")
    print(f"\nResponse:\n{response_text}")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Meta Llama

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
# Use the Conversation API to send a text message to Meta Llama.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"
```

```
# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Details zur API finden Sie unter Converse in der API-Referenz zum AWS SDK für Python (Boto3).

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Meta Llama
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Llama 3 8b Instruct.
model_id = "meta.llama3-8b-instruct-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- API-Details finden Sie ConverseStream in der AWS API-Referenz zum SDK für Python (Boto3).

Verstehen von Dokumenten

Das folgende Codebeispiel zeigt, wie ein Dokument mit Llama in Amazon Bedrock gesendet und verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Llama in Amazon Bedrock.

```
# Send and process a document with Llama on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g. Llama 3.1 8B Instruct.
model_id = "us.meta.llama3-1-8b-instruct-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
```

```
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Mistral AI

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral.

```
# Use the Conversation API to send a text message to Mistral.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the response text.
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
```

```
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Mistral gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Mistral und verarbeiten Sie den Antwortstream in Echtzeit.

```
# Use the Conversation API to send a text message to Mistral
# and print the response stream.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Start a conversation with the user message.
user_message = "Describe the purpose of a 'hello world' program in one line."
conversation = [
    {
        "role": "user",
```

```
        "content": [{"text": user_message}],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    streaming_response = client.converse_stream(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 512, "temperature": 0.5, "topP": 0.9},
    )

    # Extract and print the streamed response text in real-time.
    for chunk in streaming_response["stream"]:
        if "contentBlockDelta" in chunk:
            text = chunk["contentBlockDelta"]["delta"]["text"]
            print(text, end="")

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- API-Details finden Sie ConverseStream in der AWS API-Referenz zum SDK für Python (Boto3).

Verstehen von Dokumenten

Das folgende Code-Beispiel zeigt, wie Sie ein Dokument mit Mistral-Modellen in Amazon Bedrock senden und verarbeiten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden und verarbeiten Sie ein Dokument mit Mistral-Modellen in Amazon Bedrock.

```
# Send and process a document with Mistral models on Amazon Bedrock.

import boto3
from botocore.exceptions import ClientError

# Create a Bedrock Runtime client in the AWS Region you want to use.
client = boto3.client("bedrock-runtime", region_name="us-east-1")

# Set the model ID, e.g., Mistral Large.
model_id = "mistral.mistral-large-2402-v1:0"

# Load the document
with open("example-data/amazon-nova-service-cards.pdf", "rb") as file:
    document_bytes = file.read()

# Start a conversation with a user message and the document
conversation = [
    {
        "role": "user",
        "content": [
            {"text": "Briefly compare the models described in this document"},
            {
                "document": {
                    # Available formats: html, md, pdf, doc/docx, xls/xlsx, csv, and
 txt
                    "format": "pdf",
                    "name": "Amazon Nova Service Cards",
                    "source": {"bytes": document_bytes},
                }
            },
        ],
    }
]

try:
    # Send the message to the model, using a basic inference configuration.
    response = client.converse(
        modelId=model_id,
        messages=conversation,
        inferenceConfig={"maxTokens": 500, "temperature": 0.3},
    )

    # Extract and print the response text.
```

```
    response_text = response["output"]["message"]["content"][0]["text"]
    print(response_text)

except (ClientError, Exception) as e:
    print(f"ERROR: Can't invoke '{model_id}'. Reason: {e}")
    exit(1)
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Python (Boto3).

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
# Use the native inference API to create an image with Stability AI Stable Image
 Core

import base64
import boto3
import json
import os
import random

# Create a Bedrock Runtime client in the AWS Region of your choice.
client = boto3.client("bedrock-runtime", region_name="us-west-2")

# Set the model ID, e.g., Stable Image Core.
```

```
model_id = "stability.stable-image-core-v1:1"

# Define the image generation prompt for the model.
prompt = "A stylized picture of a cute old steampunk robot."

# Generate a random seed.
seed = random.randint(0, 4294967295)

# Format the request payload using the model's native structure.
native_request = {
    "prompt": prompt,
    "aspect_ratio": "1:1",
    "seed": seed,
    "output_format": "png",
}

# Convert the native request to JSON.
request = json.dumps(native_request)

# Invoke the model with the request.
response = client.invoke_model(modelId=model_id, body=request)

# Decode the response body.
model_response = json.loads(response["body"].read())

# Extract the image data.
base64_image_data = model_response["images"][0]

# Save the generated image to a local folder.
i, output_dir = 1, "output"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
while os.path.exists(os.path.join(output_dir, f"stability_{i}.png")):
    i += 1

image_data = base64.b64decode(base64_image_data)

image_path = os.path.join(output_dir, f"stability_{i}.png")
with open(image_path, "wb") as file:
    file.write(image_data)

print(f"The generated image has been saved to {image_path}")
```

- API-Details finden Sie InvokeModel in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon Bedrock Agents unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Bedrock Agents Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateAgent`

Das folgende Codebeispiel zeigt die Verwendung`CreateAgent`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Agenten.

```
    def create_agent(self, agent_name, foundation_model, role_arn, instruction):
        """
        Creates an agent that orchestrates interactions between foundation models,
        data sources, software applications, user conversations, and APIs to carry
        out tasks to help customers.

        :param agent_name: A name for the agent.
        :param foundation_model: The foundation model to be used for orchestration
by the agent.
        :param role_arn: The ARN of the IAM role with permissions needed by the
agent.
        :param instruction: Instructions that tell the agent what it should do and
how it should
                            interact with users.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            response = self.client.create_agent(
                agentName=agent_name,
                foundationModel=foundation_model,
                agentResourceRoleArn=role_arn,
                instruction=instruction,
            )
        except ClientError as e:
            logger.error(f"Error: Couldn't create agent. Here's why: {e}")
            raise
        else:
            return response["agent"]
```

- API-Details finden Sie CreateAgent in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAgentActionGroup`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAgentActionGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Agentenaktionsgruppe.

```
    def create_agent_action_group(
            self, name, description, agent_id, agent_version, function_arn,
api_schema
    ):
        """
        Creates an action group for an agent. An action group defines a set of
actions that an
        agent should carry out for the customer.

        :param name: The name to give the action group.
        :param description: The description of the action group.
        :param agent_id: The unique identifier of the agent for which to create the
action group.
        :param agent_version: The version of the agent for which to create the
action group.
        :param function_arn: The ARN of the Lambda function containing the business
logic that is
                                carried out upon invoking the action.
        :param api_schema: Contains the OpenAPI schema for the action group.
        :return: Details about the action group that was created.
        """
        try:
            response = self.client.create_agent_action_group(
                actionGroupName=name,
                description=description,
                agentId=agent_id,
                agentVersion=agent_version,
                actionGroupExecutor={"lambda": function_arn},
                apiSchema={"payload": api_schema},
            )
            agent_action_group = response["agentActionGroup"]
        except ClientError as e:
```

```
            logger.error(f"Error: Couldn't create agent action group. Here's why:
{e}")
            raise
        else:
            return agent_action_group
```

- API-Details finden Sie CreateAgentActionGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAgentAlias`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAgentAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Alias für einen Agenten.

```
    def create_agent_alias(self, name, agent_id):
        """
        Creates an alias of an agent that can be used to deploy the agent.

        :param name: The name of the alias.
        :param agent_id: The unique identifier of the agent.
        :return: Details about the alias that was created.
        """
        try:
            response = self.client.create_agent_alias(
                agentAliasName=name, agentId=agent_id
            )
            agent_alias = response["agentAlias"]
        except ClientError as e:
            logger.error(f"Couldn't create agent alias. {e}")
            raise
```

```
        else:
            return agent_alias
```

- API-Details finden Sie CreateAgentAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateFlow`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Amazon-Bedrock-Flow.

```
def create_flow(client, flow_name, flow_description, role_arn, flow_def):
    """
    Creates an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_name (str): The name for the new flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: The response from CreateFlow.
    """
    try:

        logger.info("Creating flow: %s.", flow_name)

        response = client.create_flow(
            name=flow_name,
```

```
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully created flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response

    except ClientError as e:
        logger.exception("Client error creating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error creating flow: %s", {str(e)})
        raise
```

- API-Details finden Sie CreateFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateFlowAlias`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFlowAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines Alias für einen Amazon-Bedrock-Flow.

```
def create_flow_alias(client, flow_id, flow_version, name, description):
    """
    Creates an alias for an Amazon Bedrock flow.

    Args:
```

```
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The ID for the flow alias.
    """

    try:
        logger.info("Creating flow alias for flow: %s.", flow_id)

        response = client.create_flow_alias(
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
        logger.info("Successfully created flow alias for %s.", flow_id)

        return response['id']

    except ClientError as e:
        logging.exception("Client error creating alias for flow: %s - %s",
                flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating alias for flow : %s - %s",
                flow_id, str(e))
        raise
```

- API-Details finden Sie CreateFlowAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateFlowVersion`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFlowVersion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Version eines Amazon-Bedrock-Flows.

```
def create_flow_version(client, flow_id, description):
    """
    Creates a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        description (str) : A description for the flow.

    Returns:
        str: The version for the flow.
    """
    try:

        logger.info("Creating flow version for flow: %s.", flow_id)

        # Call CreateFlowVersion operation
        response = client.create_flow_version(
            flowIdentifier=flow_id,
            description=description
        )

        logging.info("Successfully created flow version %s for flow %s.",
            response['version'], flow_id)

        return response['version']

    except ClientError as e:
        logging.exception("Client error creating flow: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error creating flow : %s", str(e))
        raise
```

- API-Details finden Sie CreateFlowVersion in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateKnowledgeBase`

Das folgende Codebeispiel zeigt die Verwendung. `CreateKnowledgeBase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Amazon-Bedrock-Wissensdatenbank.

```
def create_knowledge_base(bedrock_agent_client, name, role_arn, description=None):
    """
    Creates a new knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        name (str): The name of the knowledge base.
        role_arn (str): The ARN of the IAM role that the knowledge base assumes to
 access resources.
        description (str, optional): A description of the knowledge base.

    Returns:
        dict: The details of the created knowledge base.
    """
    try:
        kwargs = {
            "name": name,
            "roleArn": role_arn,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
```

```
                }
            },
            "storageConfiguration": {
                "type": "OPENSEARCH_SERVERLESS",
                # Note: You will need to create an OpenSearch Serverless collection
 first and replace this ARN
                # with your actual collection ARN from the OpenSearch console. If
 you use the console instead,
                # you can use the quick-create flow to have Knowledge Bases create
 the collection for you.
                "opensearchServerlessConfiguration": {
                    "collectionArn": "arn:aws:aoss:us-
east-1::123456789012:collection/abcdefgh12345678defgh",
                        "fieldMapping": {
                        "metadataField": "metadata",
                        "textField": "text",
                        "vectorField": "vector"
                        },
                    "vectorIndexName": "test-uuid"
                    },
                },
            "clientToken": "test-client-token-" + str(uuid.uuid4())
        }

        if description:
            kwargs["description"] = description

        response = bedrock_agent_client.create_knowledge_base(**kwargs)

        logger.info("Created knowledge base with ID: %s", response["knowledgeBase"]
["knowledgeBaseId"])
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't create knowledge base. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie CreateKnowledgeBase in der AWS API-Referenz zum SDK für Python (Boto3).

`CreatePrompt`

Das folgende Codebeispiel zeigt die Verwendung. `CreatePrompt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen eines von Amazon Bedrock verwalteten Prompts.

```
def create_prompt(client, prompt_name, prompt_description, prompt_template,
 model_id=None):
    """
    Creates an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_name (str): The name for the new prompt.
    prompt_description (str): The description for the new prompt.
    prompt_template (str): The template for the prompt.
    model_id (str, optional): The model ID to associate with the prompt.

    Returns:
        dict: The response from CreatePrompt.
    """
    try:
        logger.info("Creating prompt: %s.", prompt_name)

        # Create a variant with the template
        variant = {
            "name": "default",
            "templateType": "TEXT",
            "templateConfiguration": {
                "text": {
                    "text": prompt_template,
```

```
                    "inputVariables": []
                }
            }
        }

        # Extract input variables from the template
        # Look for patterns like {{variable_name}}

        variables = re.findall(r'{{(.*?)}}', prompt_template)
        for var in variables:
            variant["templateConfiguration"]["text"]
["inputVariables"].append({"name": var.strip()})

        # Add model ID if provided
        if model_id:
            variant["modelId"] = model_id

        # Create the prompt with the variant
        create_params = {
            'name': prompt_name,
            'description': prompt_description,
            'variants': [variant]
        }

        response = client.create_prompt(**create_params)

        logger.info("Successfully created prompt: %s. ID: %s",
                    prompt_name,
                    response['id'])

        return response

    except ClientError as e:
        logger.exception("Client error creating prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt: %s", str(e))
        raise
```

- API-Details finden Sie CreatePrompt in der AWS API-Referenz zum SDK für Python (Boto3).

`CreatePromptVersion`

Das folgende Codebeispiel zeigt die Verwendung`CreatePromptVersion`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen einer Version eines von Amazon Bedrock verwalteten Prompts.

```
def create_prompt_version(client, prompt_id, description=None):
    """
    Creates a version of an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt to create a version for.
    description (str, optional): A description for the version.

    Returns:
        dict: The response from CreatePromptVersion.
    """
    try:
        logger.info("Creating version for prompt ID: %s.", prompt_id)

        create_params = {
            'promptIdentifier': prompt_id
        }

        if description:
            create_params['description'] = description

        response = client.create_prompt_version(**create_params)

        logger.info("Successfully created prompt version: %s", response['version'])
        logger.info("Prompt version ARN: %s", response['arn'])

        return response
```

```
    except ClientError as e:
        logger.exception("Client error creating prompt version: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error creating prompt version: %s", str(e))
        raise
```

- API-Details finden Sie CreatePromptVersion in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAgent`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAgent`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Agenten.

```
    def delete_agent(self, agent_id):
        """
        Deletes an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """

        try:
            response = self.client.delete_agent(
                agentId=agent_id, skipResourceInUseCheck=False
            )
        except ClientError as e:
```

```
            logger.error(f"Couldn't delete agent. {e}")
            raise
        else:
            return response
```

- API-Details finden Sie DeleteAgent in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAgentAlias`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAgentAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen des Alias eines Agenten.

```
    def delete_agent_alias(self, agent_id, agent_alias_id):
        """
        Deletes an alias of an Amazon Bedrock agent.

        :param agent_id: The unique identifier of the agent that the alias belongs
to.
        :param agent_alias_id: The unique identifier of the alias to delete.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """

        try:
            response = self.client.delete_agent_alias(
                agentId=agent_id, agentAliasId=agent_alias_id
            )
        except ClientError as e:
            logger.error(f"Couldn't delete agent alias. {e}")
            raise
```

```
        else:
            return response
```

- API-Details finden Sie DeleteAgentAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteFlow`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Amazon-Bedrock-Flows.

```
def delete_flow(client, flow_id):
    """
    Deletes an Amazon Bedrock flow.

    Args:
    client: Amazon Bedrock agent boto3 client.
    flow_id (str): The identifier of the flow that you want to delete.

    Returns:
        dict: The response from the DeleteFLow operation.
    """
    try:

        logger.info("Deleting flow ID: %s.",
                    flow_id)

        # Call DeleteFlow operation
        response = client.delete_flow(
            flowIdentifier=flow_id,
```

```
            skipResourceInUseCheck=True
        )

        logger.info("Finished deleting flow ID: %s", flow_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error deleting flow: %s", {str(e)})
        raise
```

- API-Details finden Sie DeleteFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteFlowAlias`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFlowAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines Alias für einen Amazon-Bedrock-Flow.

```
def delete_flow_alias(client, flow_id, flow_alias_id):
    """
    Deletes an Amazon Bedrock flow alias.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
```

```
    Returns:
        dict: The response from the call to DetectFLowAlias
    """
    try:

        logger.info("Deleting flow alias %s for flow: %s.", flow_alias_id, flow_id)

        # Delete the flow alias.
        response = client.delete_flow_alias(
            aliasIdentifier=flow_alias_id,
            flowIdentifier=flow_id
        )

        logging.info("Successfully deleted flow version for %s.", flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- API-Details finden Sie DeleteFlowAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteFlowVersion`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFlowVersion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen einer Version eines Amazon-Bedrock-Flows.

```
def delete_flow_version(client, flow_id, flow_version):
```

```
    """
    Deletes a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from DeleteFlowVersion.
    """
    try:

        logger.info("Deleting flow version %s for flow: %s.",flow_version, flow_id)

        # Call DeleteFlowVersion operation
        response = client.delete_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully deleted flow version %s for %s.",
                flow_version,
                flow_id)
        return response

    except ClientError as e:
        logging.exception("Client error deleting flow version: %s ", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected deleting flow version: %s", str(e))
        raise
```

- API-Details finden Sie DeleteFlowVersion in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteKnowledgeBase`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteKnowledgeBase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Amazon-Bedrock-Wissensdatenbank.

```
def delete_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
    Deletes a knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to delete.

    Returns:
        bool: True if the deletion was successful.
    """
    try:
        bedrock_agent_client.delete_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Deleted knowledge base: %s", knowledge_base_id)
        return True
    except ClientError as err:
        logger.error(
            "Couldn't delete knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie DeleteKnowledgeBase in der AWS API-Referenz zum SDK für Python (Boto3).

`DeletePrompt`

Das folgende Codebeispiel zeigt die Verwendung. `DeletePrompt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen eines von Amazon Bedrock verwalteten Prompts.

```
def delete_prompt(client, prompt_id):
    """
    Deletes an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to delete.

    Returns:
        dict: The response from the DeletePrompt operation.
    """
    try:
        logger.info("Deleting prompt ID: %s.", prompt_id)

        response = client.delete_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Finished deleting prompt ID: %s", prompt_id)

        return response

    except ClientError as e:
        logger.exception("Client error deleting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error deleting prompt: %s", str(e))
        raise
```

- API-Details finden Sie DeletePrompt in der AWS API-Referenz zum SDK für Python (Boto3).

GetAgent

Das folgende Codebeispiel zeigt die Verwendung. `GetAgent`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Agenten.

```
    def get_agent(self, agent_id, log_error=True):
        """
        Gets information about an agent.

        :param agent_id: The unique identifier of the agent.
        :param log_error: Whether to log any errors that occur when getting the
agent.
                          If True, errors will be logged to the logger. If False,
errors
                          will still be raised, but not logged.
        :return: The information about the requested agent.
        """

        try:
            response = self.client.get_agent(agentId=agent_id)
            agent = response["agent"]
        except ClientError as e:
            if log_error:
                logger.error(f"Couldn't get agent {agent_id}. {e}")
            raise
        else:
            return agent
```

- API-Details finden Sie GetAgent in der AWS API-Referenz zum SDK für Python (Boto3).

GetFlow

Das folgende Codebeispiel zeigt die Verwendung. `GetFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines Amazon-Bedrock-Flows.

```
def get_flow(client, flow_id):
    """
    Gets an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to get.

    Returns:
        dict: The response from the GetFlow operation.
    """
    try:

        logger.info("Getting flow ID: %s.",
                    flow_id)

        # Call GetFlow operation.
        response = client.get_flow(
            flowIdentifier=flow_id
        )

        logger.info("Retrieved flow ID: %s. Name: %s", flow_id,
                    response['name'])
```

```
        return response

    except ClientError as e:
        logger.exception("Client error getting flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error getting flow: %s", {str(e)})
        raise
```

- API-Details finden Sie GetFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`GetFlowVersion`

Das folgende Codebeispiel zeigt die Verwendung. `GetFlowVersion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen einer Version eines Amazon-Bedrock-Flows.

```
def get_flow_version(client, flow_id, flow_version):
    """
    Gets information about a version of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.
        flow_version (str): The flow version of the flow.

    Returns:
        dict: The response from the call to GetFlowVersion.
    """
```

```
    try:

        logger.info("Deleting flow version for flow: %s.", flow_id)

        # Call GetFlowVersion operation
        response = client.get_flow_version(
            flowIdentifier=flow_id,
            flowVersion=flow_version
        )

        logging.info("Successfully got flow version %s information for flow %s.",
                    flow_version,
                    flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error getting flow version: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error getting flow version: %s", str(e))
        raise
```

- API-Details finden Sie GetFlowVersion in der AWS API-Referenz zum SDK für Python (Boto3).

`GetKnowledgeBase`

Das folgende Codebeispiel zeigt die Verwendung. `GetKnowledgeBase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Amazon-Bedrock-Wissensdatenbank ab.

```
def get_knowledge_base(bedrock_agent_client, knowledge_base_id):
    """
```

```
    Gets details about a specific knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base.

    Returns:
        dict: The details of the knowledge base.
    """
    try:
        response = bedrock_agent_client.get_knowledge_base(
            knowledgeBaseId=knowledge_base_id
        )

        logger.info("Retrieved knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]
    except ClientError as err:
        logger.error(
            "Couldn't get knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie GetKnowledgeBase in der AWS API-Referenz zum SDK für Python (Boto3).

`GetPrompt`

Das folgende Codebeispiel zeigt die Verwendung. `GetPrompt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abrufen eines von Amazon Bedrock verwalteten Prompts.

```
def get_prompt(client, prompt_id):
    """
    Gets an Amazon Bedrock managed prompt.

    Args:
    client: Amazon Bedrock Agent boto3 client.
    prompt_id (str): The identifier of the prompt that you want to get.

    Returns:
        dict: The response from the GetPrompt operation.
    """
    try:
        logger.info("Getting prompt ID: %s.", prompt_id)

        response = client.get_prompt(
            promptIdentifier=prompt_id
        )

        logger.info("Retrieved prompt ID: %s. Name: %s",
                    prompt_id,
                    response['name'])

        return response

    except ClientError as e:
        logger.exception("Client error getting prompt: %s", str(e))
        raise

    except Exception as e:
        logger.exception("Unexpected error getting prompt: %s", str(e))
        raise
```

- API-Details finden Sie GetPrompt in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAgentActionGroups`

Das folgende Codebeispiel zeigt die Verwendung. `ListAgentActionGroups`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aktionsgruppen für einen Agenten.

```
    def list_agent_action_groups(self, agent_id, agent_version):
        """
        List the action groups for a version of an Amazon Bedrock Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of action group summaries for the version of the agent.
        """

        try:
            action_groups = []

            paginator = self.client.get_paginator("list_agent_action_groups")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                action_groups.extend(page["actionGroupSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list action groups. {e}")
            raise
        else:
            return action_groups
```

- API-Details finden Sie ListAgentActionGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAgentKnowledgeBases`

Das folgende Codebeispiel zeigt die Verwendung. `ListAgentKnowledgeBases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der einem Agenten zugeordneten Wissensdatenbanken.

```
    def list_agent_knowledge_bases(self, agent_id, agent_version):
        """
        List the knowledge bases associated with a version of an Amazon Bedrock
Agent.

        :param agent_id: The unique identifier of the agent.
        :param agent_version: The version of the agent.
        :return: The list of knowledge base summaries for the version of the agent.
        """

        try:
            knowledge_bases = []

            paginator = self.client.get_paginator("list_agent_knowledge_bases")
            for page in paginator.paginate(
                    agentId=agent_id,
                    agentVersion=agent_version,
                    PaginationConfig={"PageSize": 10},
            ):
                knowledge_bases.extend(page["agentKnowledgeBaseSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list knowledge bases. {e}")
            raise
        else:
            return knowledge_bases
```

- API-Details finden Sie ListAgentKnowledgeBases in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAgents`

Das folgende Codebeispiel zeigt die Verwendung. `ListAgents`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Agenten, die zu einem Konto gehören.

```
    def list_agents(self):
        """
        List the available Amazon Bedrock Agents.

        :return: The list of available bedrock agents.
        """

        try:
            all_agents = []

            paginator = self.client.get_paginator("list_agents")
            for page in paginator.paginate(PaginationConfig={"PageSize": 10}):
                all_agents.extend(page["agentSummaries"])

        except ClientError as e:
            logger.error(f"Couldn't list agents. {e}")
            raise
        else:
            return all_agents
```

- API-Details finden Sie ListAgents in der AWS API-Referenz zum SDK für Python (Boto3).

`ListFlowAliases`

Das folgende Codebeispiel zeigt die Verwendung. `ListFlowAliases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Aliasse für einen Amazon-Bedrock-Flow.

```
def list_flow_aliases(client, flow_id):
    """
    Lists the aliases of an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowAliases.
    """
    try:

        finished = False

        logger.info("Listing flow aliases for flow: %s.", flow_id)

        print(f"Aliases for flow: {flow_id}")

        response = client.list_flow_aliases(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            for alias in response['flowAliasSummaries']:
                print(f"Alias Name: {alias['name']}")
                print(f"ID: {alias['id']}")
```

```
                print(f"Description: {alias.get('description', 'No
 description')}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_aliases(maxResults=10,
                                                        nextToken=next_token)
                else:
                    finished = True

        logging.info("Successfully listed flow aliases for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow aliases: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow aliases: %s", str(e))
        raise
```

- API-Details finden Sie ListFlowAliases in der AWS API-Referenz zum SDK für Python (Boto3).

`ListFlowVersions`

Das folgende Codebeispiel zeigt die Verwendung. `ListFlowVersions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Versionen eines Amazon-Bedrock-Flows.

```
def list_flow_versions(client, flow_id):
    """
    Lists the versions of an Amazon Bedrock flow.
```

```
    Args:
        client: Amazon bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        dict: The response from ListFlowVersions.
    """
    try:

        finished = False

        logger.info("Listing flow versions for flow: %s.", flow_id)

        response = client.list_flow_versions(
            flowIdentifier=flow_id,
            maxResults=10)

        while finished is False:

            print(f"Versions for flow:{flow_id}")
            for version in response['flowVersionSummaries']:
                print(f"Version: {version['version']}")
                print(f"Status: {version['status']}\n")

                if 'nextToken' in response:
                    next_token = response['nextToken']
                    response = client.list_flow_versions(maxResults=10,
                                                         nextToken=next_token)
                else:
                    finished = True


        logging.info("Successfully listed flow versions for flow %s.",
                flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- API-Details finden Sie ListFlowVersions in der AWS API-Referenz zum SDK für Python (Boto3).

`ListFlows`

Das folgende Codebeispiel zeigt die Verwendung. `ListFlows`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der Amazon-Bedrock-Flows.

```
def list_flows(client):
    """
    Lists versions of an Amazon Bedrock flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        Nothing.
    """
    try:
        finished = False

        logger.info("Listing flows:")

        response = client.list_flows(maxResults=10)

        while finished is False:

            for flow in response['flowSummaries']:
                print(f"ID: {flow['id']}")
```

```
                print(f"Name: {flow['name']}")
                print(
                    f"Description: {flow.get('description', 'No description')}")
                print(f"Latest version: {flow['version']}")
                print(f"Status: {flow['status']}\n")

            if 'nextToken' in response:
                next_token = response['nextToken']
                response = client.list_flows(maxResults=10,
                                             nextToken=next_token)
            else:
                finished = True

        logging.info("Successfully listed flows.")


    except ClientError as e:
        logging.exception("Client error listing flow versions: %s", str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error listing flow versions: %s", str(e))
        raise
```

- API-Details finden Sie ListFlows in der AWS API-Referenz zum SDK für Python (Boto3).

`ListKnowledgeBases`

Das folgende Codebeispiel zeigt die Verwendung. `ListKnowledgeBases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listet die Amazon-Bedrock-Wissensdatenbanken auf.

```
def list_knowledge_bases(bedrock_agent_client, max_results=None):
    """
```

```
    Lists the knowledge bases in your AWS account.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        max_results (int, optional): The maximum number of knowledge bases to
return.

    Returns:
        list: A list of knowledge base details.
    """
    try:
        kwargs = {}
        if max_results is not None:
            kwargs["maxResults"] = max_results

        # Initialize an empty list to store all knowledge bases
        all_knowledge_bases = []

        # Use paginator to handle pagination automatically
        paginator = bedrock_agent_client.get_paginator('list_knowledge_bases')
        page_iterator = paginator.paginate(**kwargs)

        # Iterate through each page of results
        for page in page_iterator:
            all_knowledge_bases.extend(page.get('knowledgeBaseSummaries', []))

        logger.info("Found %s knowledge bases.", len(all_knowledge_bases))
        return all_knowledge_bases
    except ClientError as err:
        logger.error(
            "Couldn't list knowledge bases. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie ListKnowledgeBases in der AWS API-Referenz zum SDK für Python (Boto3).

`ListPrompts`

Das folgende Codebeispiel zeigt die Verwendung. `ListPrompts`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Auflisten der von Amazon Bedrock verwalteten Prompts.

```
def list_prompts(client, max_results=10):
    """
    Lists Amazon Bedrock managed prompts.

    Args:
        client: Amazon Bedrock Agent boto3 client.
        max_results (int): Maximum number of results to return per page.

    Returns:
        list: A list of prompt summaries.
    """
    try:
        logger.info("Listing prompts:")

        # Create a paginator for the list_prompts operation
        paginator = client.get_paginator('list_prompts')

        # Create the pagination parameters
        pagination_config = {
            'maxResults': max_results
        }

        # Initialize an empty list to store all prompts
        all_prompts = []

        # Iterate through all pages
        for page in paginator.paginate(**pagination_config):
            all_prompts.extend(page.get('promptSummaries', []))

        logger.info("Successfully listed %s prompts.", len(all_prompts))
        return all_prompts
```

```
    except ClientError as e:
        logger.exception("Client error listing prompts: %s", str(e))
        raise
    except Exception as e:
        logger.exception("Unexpected error listing prompts: %s", str(e))
        raise
```

- API-Details finden Sie ListPrompts in der AWS API-Referenz zum SDK für Python (Boto3).

`PrepareAgent`

Das folgende Codebeispiel zeigt die Verwendung. `PrepareAgent`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Vorbereiten eines Agenten auf interne Tests.

```
    def prepare_agent(self, agent_id):
        """
        Creates a DRAFT version of the agent that can be used for internal testing.

        :param agent_id: The unique identifier of the agent to prepare.
        :return: The response from Amazon Bedrock Agents if successful, otherwise
raises an exception.
        """
        try:
            prepared_agent_details = self.client.prepare_agent(agentId=agent_id)
        except ClientError as e:
            logger.error(f"Couldn't prepare agent. {e}")
            raise
        else:
            return prepared_agent_details
```

- API-Details finden Sie PrepareAgent in der AWS API-Referenz zum SDK für Python (Boto3).

`PrepareFlow`

Das folgende Codebeispiel zeigt die Verwendung. `PrepareFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Vorbereiten eines Amazon-Bedrock-Flows.

```
def prepare_flow(client, flow_id):
    """
    Prepares an Amazon Bedrock Flow.

    Args:
        client: Amazon Bedrock agent boto3 client.
        flow_id (str): The identifier of the flow that you want to prepare.

    Returns:
        str: The status of the flow preparation
    """
    try:

        # Prepare the flow.
        logger.info("Preparing flow ID: %s",
                    flow_id)

        response = client.prepare_flow(
            flowIdentifier=flow_id
        )

        status = response.get('status')
```

```
        while status == "Preparing":
            logger.info("Preparing flow ID: %s. Status %s",
                        flow_id, status)

            sleep(5)
            response = client.get_flow(
                flowIdentifier=flow_id
            )
            status = response.get('status')
            print(f"Flow Status: {status}")

        if status == "Prepared":
            logger.info("Finished preparing flow ID: %s. Status %s",
                        flow_id, status)
        else:
            logger.warning("flow ID: %s not prepared. Status %s",
                           flow_id, status)

        return status

    except ClientError as e:
        logger.exception("Client error preparing flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error preparing flow: %s", {str(e)})
        raise
```

- API-Details finden Sie PrepareFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateFlow`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren eines Amazon-Bedrock-Flows.

```
def update_flow(client, flow_id, flow_name, flow_description, role_arn, flow_def):
    """
    Updates an Amazon Bedrock flow.

    Args:
    client: bedrock agent boto3 client.
    flow_id (str): The ID for the flow that you want to update.
    flow_name (str): The name for the flow.
    role_arn (str):  The ARN for the IAM role that use flow uses.
    flow_def (json): The JSON definition of the flow that you want to create.

    Returns:
        dict: Flow information if successful.
    """
    try:

        logger.info("Updating flow: %s.", flow_id)

        response = client.update_flow(
            flowIdentifier=flow_id,
            name=flow_name,
            description=flow_description,
            executionRoleArn=role_arn,
            definition=flow_def
        )

        logger.info("Successfully updated flow: %s. ID: %s",
                    flow_name,
                    {response['id']})

        return response

    except ClientError as e:
        logger.exception("Client error updating flow: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexepcted error updating flow: %s", {str(e)})
        raise
```

- API-Details finden Sie UpdateFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateFlowAlias`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateFlowAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren des Alias eines Amazon-Bedrock-Flows.

```
def update_flow_alias(client, flow_id, alias_id, flow_version, name, description):
    """
    Updates an alias for an Amazon Bedrock flow.

    Args:
        client: bedrock agent boto3 client.
        flow_id (str): The identifier of the flow.

    Returns:
        str: The response from UpdateFlowAlias.
    """

    try:
        logger.info("Updating flow alias %s for flow: %s.", alias_id, flow_id)

        response = client.update_flow_alias(
            aliasIdentifier=alias_id,
            flowIdentifier=flow_id,
            name=name,
            description=description,
            routingConfiguration=[
                {
                    "flowVersion": flow_version
                }
            ]
        )
```

```
        logger.info("Successfully updated flow alias %s for %s.", alias_id, flow_id)

        return response

    except ClientError as e:
        logging.exception("Client error updating alias %s for flow: %s - %s",
                alias_id, flow_id, str(e))
        raise
    except Exception as e:
        logging.exception("Unexpected error updating alias %s for flow : %s - %s",
                alias_id, flow_id, str(e))
        raise
```

- API-Details finden Sie UpdateFlowAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateKnowledgeBase`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateKnowledgeBase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren einer Amazon-Bedrock-Wissensdatenbank.

```
def update_knowledge_base(bedrock_agent_client, knowledge_base_id, name=None,
 description=None, role_arn=None):
    """
    Updates an existing knowledge base.

    Args:
        bedrock_agent_client: The Boto3 Bedrock Agent client.
        knowledge_base_id (str): The ID of the knowledge base to update.
        name (str, optional): The new name for the knowledge base.
        description (str, optional): The new description for the knowledge base.
```

```
        role_arn (str, optional): The new IAM role ARN for the knowledge base.

    Returns:
        dict: The details of the updated knowledge base.
    """
    try:
        kwargs = {
            "knowledgeBaseId": knowledge_base_id,
            "knowledgeBaseConfiguration": {
                "type": "VECTOR",
                "vectorKnowledgeBaseConfiguration": {
                    "embeddingModelArn": "arn:aws:bedrock:us-east-1::foundation-
model/amazon.titan-embed-text-v1"
                }
            }
        }

        if name:
            kwargs["name"] = name
        if description:
            kwargs["description"] = description
        if role_arn:
            kwargs["roleArn"] = role_arn

        response = bedrock_agent_client.update_knowledge_base(**kwargs)

        logger.info("Updated knowledge base: %s", knowledge_base_id)
        return response["knowledgeBase"]

    except ClientError as err:
        logger.error(
            "Couldn't update knowledge base %s. Here's why: %s: %s",
            knowledge_base_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie UpdateKnowledgeBase in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Aufrufen eines Flows

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Flow.
- Erstellen Sie den Flow.
- Stellen Sie den vollständig konfigurierten Flow bereit.
- Rufen Sie den Flow mit vom Benutzer bereitgestellten Prompts auf.
- Löschen aller erstellten Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellt eine Musik-Playlist basierend auf dem vom Benutzenden angegebenen Genre und der Anzahl der Songs.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)
```

```
def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music playlist
 based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
 prompt.
```

```
    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting of
the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
```

```
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }




def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
```

```
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the input
of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
```

```
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn

    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise
```

```
def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
```

```
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None
```

```
        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)

                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
```

```
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.

    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None
```

```
    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }




def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
```

```
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """


    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
```

```
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """


    # Trust relationship policy - allows Amazon Bedrock service to assume this role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
```

```
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """
```

```
    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
    """




    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
```

```
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Erstellen und Aufrufen verwalteter Prompts

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines verwalteten Prompts.
- Erstellen einer Version des Prompts.
- Aufrufen des Prompts mithilfe der Version.
- Bereinigen der Ressourcen (optional).

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines verwalteten Prompts.

```
import argparse
import boto3
import logging
import time

# Now import the modules
from prompt import create_prompt, create_prompt_version, delete_prompt
from run_prompt import invoke_prompt

logging.basicConfig(
    level=logging.INFO,
    format='%(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)



def run_scenario(bedrock_client, bedrock_runtime_client, model_id, cleanup=True):
    """
    Runs the Amazon Bedrock managed prompt scenario.

    Args:
        bedrock_client: The Amazon Bedrock Agent client.
        bedrock_runtime_client: The Amazon Bedrock Runtime client.
        model_id (str): The model ID to use for the prompt.
        cleanup (bool): Whether to clean up resources at the end of the scenario.

    Returns:
        dict: A dictionary containing the created resources.
    """
    prompt_id = None
```

```
    try:
        # Step 1: Create a prompt
        print("\n=== Step 1: Creating a prompt ===")
        prompt_name = f"PlaylistGenerator-{int(time.time())}"
        prompt_description = "Playlist generator"
        prompt_template = """
          Make me a {{genre}} playlist consisting of the following number of songs:
{{number}}."""

        create_response = create_prompt(
            bedrock_client,
            prompt_name,
            prompt_description,
            prompt_template,
            model_id
        )

        prompt_id = create_response['id']
        print(f"Created prompt: {prompt_name} with ID: {prompt_id}")

        # Create a version of the prompt
        print("\n=== Creating a version of the prompt ===")
        version_response = create_prompt_version(
            bedrock_client,
            prompt_id,
            description="Initial version of the product description generator"
        )

        prompt_version_arn = version_response['arn']
        prompt_version = version_response['version']

        print(f"Created prompt version: {prompt_version}")
        print(f"Prompt version ARN: {prompt_version_arn}")

        # Step 2: Invoke the prompt directly
        print("\n=== Step 2: Invoking the prompt ===")
        input_variables = {
            "genre": "pop",
            "number": "2",
           }

        # Use the ARN from the create_prompt_version response
        result = invoke_prompt(
            bedrock_runtime_client,
```

```
            prompt_version_arn,
            input_variables
        )
        # Display the playlist
        print(f"\n{result}")


        # Step 3: Clean up resources (optional)
        if cleanup:
            print("\n=== Step 3: Cleaning up resources ===")

            # Delete the prompt
            print(f"Deleting prompt {prompt_id}...")
            delete_prompt(bedrock_client, prompt_id)

            print("Cleanup complete")
        else:
            print("\n=== Resources were not cleaned up ===")
            print(f"Prompt ID: {prompt_id}")



    except Exception as e:
        logger.exception("Error in scenario: %s", str(e))

        # Attempt to clean up if an error occurred and cleanup was requested
        if cleanup and prompt_id:
            try:
                print("\nCleaning up resources after error...")

                # Delete the prompt
                try:
                    delete_prompt(bedrock_client, prompt_id)
                    print("Cleanup after error complete")
                except Exception as cleanup_error:
                    logger.error("Error during cleanup: %s", str(cleanup_error))
            except Exception as final_error:
                logger.error("Final error during cleanup: %s", str(final_error))

        # Re-raise the original exception
        raise

def main():
    """
```

```
    Entry point for the Amazon Bedrock managed prompt scenario.
    """
    parser = argparse.ArgumentParser(
        description="Run the Amazon Bedrock managed prompt scenario."
    )
    parser.add_argument(
        '--region',
        default='us-east-1',
        help="The AWS Region to use."
    )
    parser.add_argument(
        '--model-id',
        default='anthropic.claude-v2',
        help="The model ID to use for the prompt."
    )
    parser.add_argument(
        '--cleanup',
        action='store_true',
        default=True,
        help="Clean up resources at the end of the scenario."
    )
    parser.add_argument(
        '--no-cleanup',
        action='store_false',
        dest='cleanup',
        help="Don't clean up resources at the end of the scenario."
    )
    args = parser.parse_args()

    bedrock_client = boto3.client('bedrock-agent', region_name=args.region)
    bedrock_runtime_client = boto3.client('bedrock-runtime',
region_name=args.region)

    print("=== Amazon Bedrock Managed Prompt Scenario ===")
    print(f"Region: {args.region}")
    print(f"Model ID: {args.model_id}")
    print(f"Cleanup resources: {args.cleanup}")

    try:
        run_scenario(
            bedrock_client,
            bedrock_runtime_client,
            args.model_id,
            args.cleanup
```

```
        )

    except Exception as e:
        logger.exception("Error running scenario: %s", str(e))

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Converse
  - CreatePrompt
  - CreatePromptVersion
  - DeletePrompt

Erstellen und Ausführen von Agenten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Agenten.
- Erstellen einer Lambda-Funktion, die die Funktionen des Agenten implementiert.
- Erstellen einer Lambda-Funktion, die die Funktionen des Agenten implementiert.
- Erstellen einer Aktionsgruppe, die den Agenten mit der Lambda-Funktion verbindet.
- Bereitstellen des vollständig konfigurierten Agenten.
- Aufrufen des Agenten mit vom Benutzer bereitgestellten Prompts.
- Löschen aller erstellten Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und Aufrufen eines Agenten.

```
REGION = "us-east-1"
ROLE_POLICY_NAME = "agent_permissions"


class BedrockAgentScenarioWrapper:
    """Runs a scenario that shows how to get started using Amazon Bedrock Agents."""

    def __init__(
            self, bedrock_agent_client, runtime_client, lambda_client, iam_resource,
 postfix
    ):
        self.iam_resource = iam_resource
        self.lambda_client = lambda_client
        self.bedrock_agent_runtime_client = runtime_client
        self.postfix = postfix

        self.bedrock_wrapper = BedrockAgentWrapper(bedrock_agent_client)

        self.agent = None
        self.agent_alias = None
        self.agent_role = None
        self.prepared_agent_details = None
        self.lambda_role = None
        self.lambda_function = None

    def run_scenario(self):
        print("=" * 88)
        print("Welcome to the Amazon Bedrock Agents demo.")
        print("=" * 88)

        # Query input from user
        print("Let's start with creating an agent:")
        print("-" * 40)
        name, foundation_model = self._request_name_and_model_from_user()
        print("-" * 40)

        # Create an execution role for the agent
        self.agent_role = self._create_agent_role(foundation_model)

        # Create the agent
        self.agent = self._create_agent(name, foundation_model)
```

```
        # Prepare a DRAFT version of the agent
        self.prepared_agent_details = self._prepare_agent()

        # Create the agent's Lambda function
        self.lambda_function = self._create_lambda_function()

        # Configure permissions for the agent to invoke the Lambda function
        self._allow_agent_to_invoke_function()
        self._let_function_accept_invocations_from_agent()

        # Create an action group to connect the agent with the Lambda function
        self._create_agent_action_group()

        # If the agent has been modified or any components have been added, prepare
the agent again
        components = [self._get_agent()]
        components += self._get_agent_action_groups()
        components += self._get_agent_knowledge_bases()

        latest_update = max(component["updatedAt"] for component in components)
        if latest_update > self.prepared_agent_details["preparedAt"]:
            self.prepared_agent_details = self._prepare_agent()

        # Create an agent alias
        self.agent_alias = self._create_agent_alias()

        # Test the agent
        self._chat_with_agent(self.agent_alias)

        print("=" * 88)
        print("Thanks for running the demo!\n")

        if q.ask("Do you want to delete the created resources? [y/N] ", q.is_yesno):
            self._delete_resources()
            print("=" * 88)
            print(
                "All demo resources have been deleted. Thanks again for running the
demo!"
            )
        else:
            self._list_resources()
            print("=" * 88)
            print("Thanks again for running the demo!")
```

```
    def _request_name_and_model_from_user(self):
        existing_agent_names = [
            agent["agentName"] for agent in self.bedrock_wrapper.list_agents()
        ]

        while True:
            name = q.ask("Enter an agent name: ", self.is_valid_agent_name)
            if name.lower() not in [n.lower() for n in existing_agent_names]:
                break
            print(
                f"Agent {name} conflicts with an existing agent. Please use a
different name."
            )

        models = ["anthropic.claude-instant-v1", "anthropic.claude-v2"]
        model_id = models[
            q.choose("Which foundation model would you like to use? ", models)
        ]

        return name, model_id

    def _create_agent_role(self, model_id):
        role_name = f"AmazonBedrockExecutionRoleForAgents_{self.postfix}"
        model_arn = f"arn:aws:bedrock:{REGION}::foundation-model/{model_id}*"

        print("Creating an an execution role for the agent...")

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service": "bedrock.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )
```

```
            role.Policy(ROLE_POLICY_NAME).put(
                PolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": "bedrock:InvokeModel",
                                "Resource": model_arn,
                            }
                        ],
                    }
                )
            )
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        return role

    def _create_agent(self, name, model_id):
        print("Creating the agent...")

        instruction = """
            You are a friendly chat bot. You have access to a function called that
returns
            information about the current date and time. When responding with date
or time,
            please make sure to add the timezone UTC.
            """
        agent = self.bedrock_wrapper.create_agent(
            agent_name=name,
            foundation_model=model_id,
            instruction=instruction,
            role_arn=self.agent_role.arn,
        )
        self._wait_for_agent_status(agent["agentId"], "NOT_PREPARED")

        return agent

    def _prepare_agent(self):
        print("Preparing the agent...")

        agent_id = self.agent["agentId"]
```

```
        prepared_agent_details = self.bedrock_wrapper.prepare_agent(agent_id)
        self._wait_for_agent_status(agent_id, "PREPARED")

        return prepared_agent_details

    def _create_lambda_function(self):
        print("Creating the Lambda function...")

        function_name = f"AmazonBedrockExampleFunction_{self.postfix}"

        self.lambda_role = self._create_lambda_role()

        try:
            deployment_package = self._create_deployment_package(function_name)

            lambda_function = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="Lambda function for Amazon Bedrock example",
                Runtime="python3.11",
                Role=self.lambda_role.arn,
                Handler=f"{function_name}.lambda_handler",
                Code={"ZipFile": deployment_package},
                Publish=True,
            )

            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)

        except ClientError as e:
            logger.error(
                f"Couldn't create Lambda function {function_name}. Here's why: {e}"
            )
            raise

        return lambda_function

    def _create_lambda_role(self):
        print("Creating an execution role for the Lambda function...")

        role_name = f"AmazonBedrockExecutionRoleForLambda_{self.postfix}"

        try:
            role = self.iam_resource.create_role(
                RoleName=role_name,
```

```
                AssumeRolePolicyDocument=json.dumps(
                    {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {"Service": "lambda.amazonaws.com"},
                                "Action": "sts:AssumeRole",
                            }
                        ],
                    }
                ),
            )
            role.attach_policy(
                PolicyArn="arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"
            )
            print(f"Created role {role_name}")
        except ClientError as e:
            logger.error(f"Couldn't create role {role_name}. Here's why: {e}")
            raise

        print("Waiting for the execution role to be fully propagated...")
        wait(10)

        return role

    def _allow_agent_to_invoke_function(self):
        policy = self.iam_resource.RolePolicy(
            self.agent_role.role_name, ROLE_POLICY_NAME
        )
        doc = policy.policy_document
        doc["Statement"].append(
            {
                "Effect": "Allow",
                "Action": "lambda:InvokeFunction",
                "Resource": self.lambda_function["FunctionArn"],
            }
        )
        self.agent_role.Policy(ROLE_POLICY_NAME).put(PolicyDocument=json.dumps(doc))

    def _let_function_accept_invocations_from_agent(self):
        try:
            self.lambda_client.add_permission(
```

```
                FunctionName=self.lambda_function["FunctionName"],
                SourceArn=self.agent["agentArn"],
                StatementId="BedrockAccess",
                Action="lambda:InvokeFunction",
                Principal="bedrock.amazonaws.com",
            )
        except ClientError as e:
            logger.error(
                f"Couldn't grant Bedrock permission to invoke the Lambda function.
Here's why: {e}"
            )
            raise

    def _create_agent_action_group(self):
        print("Creating an action group for the agent...")

        try:
            with open("./scenario_resources/api_schema.yaml") as file:
                self.bedrock_wrapper.create_agent_action_group(
                    name="current_date_and_time",
                    description="Gets the current date and time.",
                    agent_id=self.agent["agentId"],
                    agent_version=self.prepared_agent_details["agentVersion"],
                    function_arn=self.lambda_function["FunctionArn"],
                    api_schema=json.dumps(yaml.safe_load(file)),
                )
        except ClientError as e:
            logger.error(f"Couldn't create agent action group. Here's why: {e}")
            raise

    def _get_agent(self):
        return self.bedrock_wrapper.get_agent(self.agent["agentId"])

    def _get_agent_action_groups(self):
        return self.bedrock_wrapper.list_agent_action_groups(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _get_agent_knowledge_bases(self):
        return self.bedrock_wrapper.list_agent_knowledge_bases(
            self.agent["agentId"], self.prepared_agent_details["agentVersion"]
        )

    def _create_agent_alias(self):
```

```
        print("Creating an agent alias...")

        agent_alias_name = "test_agent_alias"
        agent_alias = self.bedrock_wrapper.create_agent_alias(
            agent_alias_name, self.agent["agentId"]
        )

        self._wait_for_agent_status(self.agent["agentId"], "PREPARED")

        return agent_alias

    def _wait_for_agent_status(self, agent_id, status):
        while self.bedrock_wrapper.get_agent(agent_id)["agentStatus"] != status:
            wait(2)

    def _chat_with_agent(self, agent_alias):
        print("-" * 88)
        print("The agent is ready to chat.")
        print("Try asking for the date or time. Type 'exit' to quit.")

        # Create a unique session ID for the conversation
        session_id = uuid.uuid4().hex

        while True:
            prompt = q.ask("Prompt: ", q.non_empty)

            if prompt == "exit":
                break

            response = asyncio.run(self._invoke_agent(agent_alias, prompt,
session_id))

            print(f"Agent: {response}")

    async def _invoke_agent(self, agent_alias, prompt, session_id):
        response = self.bedrock_agent_runtime_client.invoke_agent(
            agentId=self.agent["agentId"],
            agentAliasId=agent_alias["agentAliasId"],
            sessionId=session_id,
            inputText=prompt,
        )

        completion = ""
```

```
        for event in response.get("completion"):
            chunk = event["chunk"]
            completion += chunk["bytes"].decode()

        return completion

    def _delete_resources(self):
        if self.agent:
            agent_id = self.agent["agentId"]

            if self.agent_alias:
                agent_alias_id = self.agent_alias["agentAliasId"]
                print("Deleting agent alias...")
                self.bedrock_wrapper.delete_agent_alias(agent_id, agent_alias_id)

            print("Deleting agent...")
            agent_status = self.bedrock_wrapper.delete_agent(agent_id)
["agentStatus"]
            while agent_status == "DELETING":
                wait(5)
                try:
                    agent_status = self.bedrock_wrapper.get_agent(
                        agent_id, log_error=False
                    )["agentStatus"]
                except ClientError as err:
                    if err.response["Error"]["Code"] == "ResourceNotFoundException":
                        agent_status = "DELETED"

        if self.lambda_function:
            name = self.lambda_function["FunctionName"]
            print(f"Deleting function '{name}'...")
            self.lambda_client.delete_function(FunctionName=name)

        if self.agent_role:
            print(f"Deleting role '{self.agent_role.role_name}'...")
            self.agent_role.Policy(ROLE_POLICY_NAME).delete()
            self.agent_role.delete()

        if self.lambda_role:
            print(f"Deleting role '{self.lambda_role.role_name}'...")
            for policy in self.lambda_role.attached_policies.all():
                policy.detach_role(RoleName=self.lambda_role.role_name)
            self.lambda_role.delete()
```

```
    def _list_resources(self):
        print("-" * 40)
        print(f"Here is the list of created resources in '{REGION}'.")
        print("Make sure you delete them once you're done to avoid unnecessary
 costs.")
        if self.agent:
            print(f"Bedrock Agent:   {self.agent['agentName']}")
        if self.lambda_function:
            print(f"Lambda function: {self.lambda_function['FunctionName']}")
        if self.agent_role:
            print(f"IAM role:        {self.agent_role.role_name}")
        if self.lambda_role:
            print(f"IAM role:        {self.lambda_role.role_name}")

    @staticmethod
    def is_valid_agent_name(answer):
        valid_regex = r"^[a-zA-Z0-9_-]{1,100}$"
        return (
            answer
            if answer and len(answer) <= 100 and re.match(valid_regex, answer)
            else None,
            "I need a name for the agent, please. Valid characters are a-z, A-Z,
 0-9, _ (underscore) and - (hyphen).",
        )

    @staticmethod
    def _create_deployment_package(function_name):
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
            zipped.write(
                "./scenario_resources/lambda_function.py", f"{function_name}.py"
            )
        buffer.seek(0)
        return buffer.read()


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    postfix = "".join(
        random.choice(string.ascii_lowercase + "0123456789") for _ in range(8)
    )
    scenario = BedrockAgentScenarioWrapper(
        bedrock_agent_client=boto3.client(
```

```
            service_name="bedrock-agent", region_name=REGION
        ),
        runtime_client=boto3.client(
            service_name="bedrock-agent-runtime", region_name=REGION
        ),
        lambda_client=boto3.client(service_name="lambda", region_name=REGION),
        iam_resource=boto3.resource("iam"),
        postfix=postfix,
    )
    try:
        scenario.run_scenario()
    except Exception as e:
        logging.exception(f"Something went wrong with the demo. Here's what: {e}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAgent
  - CreateAgentActionGroup
  - CreateAgentAlias
  - DeleteAgent
  - DeleteAgentAlias
  - GetAgent
  - ListAgentActionGroups
  - ListAgentKnowledgeBases
  - ListAgents
  - PrepareAgent

Orchestrieren von Anwendungen mit generativer KI mit Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum

Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finde und fasse das aktuelle Repositorium mit den höchsten Trends GitHub zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock
- Step-Funktionen

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Runtime AWS SDK für Python (Boto3) mit Amazon Bedrock Agents Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel zeigt, wie Sie InvokeFlow mit einem Amazon Bedrock-Flow kommunizieren können, der einen Agentenknoten enthält.

Weitere Informationen finden Sie unter Konversation mit einem Amazon-Bedrock-Flow.

SDK für Python (Boto3)

> **Note**
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Shows how to run an Amazon Bedrock flow with InvokeFlow and handle muli-turn
 interaction
```

```
for a single conversation.
For more information, see https://docs.aws.amazon.com/bedrock/latest/userguide/
flows-multi-turn-invocation.html.

"""
import logging
import boto3
import botocore

import botocore.exceptions

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


def invoke_flow(client, flow_id, flow_alias_id, input_data, execution_id):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
        execution_id: Execution ID for continuing a flow. Use the value None on
 first run.

    Returns:
        Dict containing flow_complete status, input_required info, and execution_id
    """

    response = None
    request_params = None

    if execution_id is None:
        # Don't pass execution ID for first run.
        request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }
    else:
        request_params = {
```

```
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "executionId": execution_id,
            "inputs": [input_data],
            "enableTrace": True
        }

    response = client.invoke_flow(**request_params)

    if "executionId" not in request_params:
        execution_id = response['executionId']

    input_required = None
    flow_status = ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Check if more input us needed from user.
        elif 'flowMultiTurnInputRequestEvent' in event:
            input_required = event

        # Print the model output.
        elif 'flowOutputEvent' in event:
            print(event['flowOutputEvent']['content']['document'])

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "input_required": input_required,
        "execution_id": execution_id
    }


def converse_with_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Run a conversation with the supplied flow.
```

```
    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """

    flow_execution_id = None
    finished = False

    # Get the intial prompt from the user.
    user_input = input("Enter input: ")

    # Use prompt to create input data.
    flow_input_data = {
        "content": {
            "document": user_input
        },
        "nodeName": "FlowInputNode",
        "nodeOutputName": "document"
    }

    try:
        while not finished:
            # Invoke the flow until successfully finished.

            result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data,
 flow_execution_id)

            status = result['flow_status']
            flow_execution_id = result['execution_id']
            more_input = result['input_required']
            if status == "INPUT_REQUIRED":
                # The flow needs more information from the user.
                logger.info("The flow %s requires more input", flow_id)
                user_input = input(
                    more_input['flowMultiTurnInputRequestEvent']['content']
['document'] + ": ")
                flow_input_data = {
                    "content": {
                        "document": user_input
                    },
```

```
                    "nodeName": more_input['flowMultiTurnInputRequestEvent']
['nodeName'],
                    "nodeInputName": "agentInputText"

                }
            elif status == "SUCCESS":
                # The flow completed successfully.
                finished = True
                logger.info("The flow %s successfully completed.", flow_id)

    except botocore.exceptions.ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})

    except Exception as e:
        print(f"An error occurred: {str(e)}")
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})


def main():
    """
    Main entry point for the script.
    """

    # Replace these with your actual flow ID and flow alias ID.
    FLOW_ID = 'YOUR_FLOW_ID'
    FLOW_ALIAS_ID = 'YOUR_FLOW_ALIAS_ID'

    logger.info("Starting conversation with FLOW: %s ID: %s",
                FLOW_ID, FLOW_ALIAS_ID)

    # Get the Bedrock agent runtime client.
    session = boto3.Session(profile_name='default')
    bedrock_agent_client = session.client('bedrock-agent-runtime')

    # Start the conversation.
    converse_with_flow(bedrock_agent_client, FLOW_ID, FLOW_ALIAS_ID)

    logger.info("Conversation with FLOW: %s ID: %s finished",
                FLOW_ID, FLOW_ALIAS_ID)


if __name__ == "__main__":
```

```
    main()
```

- API-Details finden Sie InvokeFlow in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`InvokeAgent`

Das folgende Codebeispiel zeigt die Verwendung. `InvokeAgent`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aufrufen eines Agenten.

```
    def invoke_agent(self, agent_id, agent_alias_id, session_id, prompt):
        """
        Sends a prompt for the agent to process and respond to.

        :param agent_id: The unique identifier of the agent to use.
        :param agent_alias_id: The alias of the agent to use.
        :param session_id: The unique identifier of the session. Use the same value
across requests
                           to continue the same conversation.
        :param prompt: The prompt that you want Claude to complete.
        :return: Inference response from the model.
        """

        try:
            # Note: The execution time depends on the foundation model, complexity
of the agent,
            # and the length of the prompt. In some cases, it can take up to a
minute or more to
            # generate a response.
            response = self.agents_runtime_client.invoke_agent(
```

```
                agentId=agent_id,
                agentAliasId=agent_alias_id,
                sessionId=session_id,
                inputText=prompt,
            )

            completion = ""

            for event in response.get("completion"):
                chunk = event["chunk"]
                completion = completion + chunk["bytes"].decode()

        except ClientError as e:
            logger.error(f"Couldn't invoke agent. {e}")
            raise

        return completion
```

- API-Details finden Sie InvokeAgent in der AWS API-Referenz zum SDK für Python (Boto3).

`InvokeFlow`

Das folgende Codebeispiel zeigt die Verwendung. `InvokeFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aufrufen eines Flows.

```
    def invoke_flow(self, flow_id, flow_alias_id, input_data, execution_id):
        """
        Invoke an Amazon Bedrock flow and handle the response stream.

        Args:
            param flow_id: The ID of the flow to invoke.
```

```
            param flow_alias_id: The alias ID of the flow.
            param input_data: Input data for the flow.
            param execution_id: Execution ID for continuing a flow. Use the value
None on first run.

        Return: Response from the flow.
        """
        try:

            request_params = None

            if execution_id is None:
                # Don't pass execution ID for first run.
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "inputs": input_data,
                    "enableTrace": True
                }
            else:
                request_params = {
                    "flowIdentifier": flow_id,
                    "flowAliasIdentifier": flow_alias_id,
                    "executionId": execution_id,
                    "inputs": input_data,
                    "enableTrace": True
                }

            response = self.agents_runtime_client.invoke_flow(**request_params)

            if "executionId" not in request_params:
                execution_id = response['executionId']

            result = ""

            # Get the streaming response
            for event in response['responseStream']:
                result = result + str(event) + '\n'
            print(result)

        except ClientError as e:
            logger.error("Couldn't invoke flow %s.", {e})
            raise
```

```
        return result
```

- API-Details finden Sie InvokeFlow in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Aufrufen eines Flows

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Ausführungsrolle für den Flow.
- Erstellen Sie den Flow.
- Stellen Sie den vollständig konfigurierten Flow bereit.
- Rufen Sie den Flow mit vom Benutzer bereitgestellten Prompts auf.
- Löschen aller erstellten Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellt eine Musik-Playlist basierend auf dem vom Benutzenden angegebenen Genre und der Anzahl der Songs.

```
from datetime import datetime
import logging
import boto3

from botocore.exceptions import ClientError

from roles import create_flow_role, delete_flow_role, update_role_policy
from flow import create_flow, prepare_flow, delete_flow
from run_flow import run_playlist_flow
from flow_version import create_flow_version, delete_flow_version
```

```
from flow_alias import create_flow_alias, delete_flow_alias

logging.basicConfig(
    level=logging.INFO
)
logger = logging.getLogger(__name__)

def create_input_node(name):
    """
    Creates an input node configuration for an Amazon Bedrock flow.

    The input node serves as the entry point for the flow and defines
    the initial document structure that will be passed to subsequent nodes.

    Args:
        name (str): The name of the input node.

    Returns:
        dict: The input node configuration.

    """
    return {
        "type": "Input",
        "name": name,
        "outputs": [
            {
                "name": "document",
                "type": "Object"
            }
        ]
    }


def create_prompt_node(name, model_id):
    """
    Creates a prompt node configuration for a Bedrock flow that generates music
 playlists.

    The prompt node defines an inline prompt template that creates a music playlist
 based on
    a specified genre and number of songs. The prompt uses two variables that are
 mapped from
    the input JSON object:
    - {{genre}}: The genre of music to create a playlist for
```

```
    - {{number}}: The number of songs to include in the playlist

    Args:
        name (str): The name of the prompt node.
        model_id (str): The identifier of the foundation model to use for the
prompt.

    Returns:
        dict: The prompt node.

    """

    return {
        "type": "Prompt",
        "name": name,
        "configuration": {
            "prompt": {
                "sourceConfiguration": {
                    "inline": {
                        "modelId": model_id,
                        "templateType": "TEXT",
                        "inferenceConfiguration": {
                            "text": {
                                "temperature": 0.8
                            }
                        },
                        "templateConfiguration": {
                            "text": {
                                "text": "Make me a {{genre}} playlist consisting of
the following number of songs: {{number}}."
                            }
                        }
                    }
                }
            }
        },
        "inputs": [
            {
                "name": "genre",
                "type": "String",
                "expression": "$.data.genre"
            },
            {
                "name": "number",
```

```
                "type": "Number",
                "expression": "$.data.number"
            }
        ],
        "outputs": [
            {
                "name": "modelCompletion",
                "type": "String"
            }
        ]
    }


def create_output_node(name):
    """
    Creates an output node configuration for a Bedrock flow.

    The output node validates that the output from the last node is a string
    and returns it unmodified. The input name must be "document".

    Args:
        name (str): The name of the output node.

    Returns:
        dict: The output node configuration containing the output node:

    """

    return {
        "type": "Output",
        "name": name,
        "inputs": [
            {
                "name": "document",
                "type": "String",
                "expression": "$.data"
            }
        ]
    }
```

```
def create_playlist_flow(client, flow_name, flow_description, role_arn,
 prompt_model_id):
    """
    Creates the playlist generator flow.
    Args:
        client: bedrock agent boto3 client.
        role_arn (str): Name for the new IAM role.
        prompt_model_id (str): The id of the model to use in the prompt node.
    Returns:
        dict: The response from the create_flow operation.
    """

    input_node = create_input_node("FlowInput")
    prompt_node = create_prompt_node("MakePlaylist", prompt_model_id)
    output_node = create_output_node("FlowOutput")

    # Create connections between the nodes
    connections = []

    #  First, create connections between the output of the flow
    # input node and each input of the prompt node.
    for prompt_node_input in prompt_node["inputs"]:
        connections.append(
            {
                "name": "_".join([input_node["name"], prompt_node["name"],
                                  prompt_node_input["name"]]),
                "source": input_node["name"],
                "target": prompt_node["name"],
                "type": "Data",
                "configuration": {
                    "data": {
                        "sourceOutput": input_node["outputs"][0]["name"],
                        "targetInput": prompt_node_input["name"]
                    }
                }
            }
        )

    # Then, create a connection between the output of the prompt node and the input
 of the flow output node
    connections.append(
        {
            "name": "_".join([prompt_node["name"], output_node["name"]]),
            "source": prompt_node["name"],
```

```
            "target": output_node["name"],
            "type": "Data",
            "configuration": {
                "data": {
                    "sourceOutput": prompt_node["outputs"][0]["name"],
                    "targetInput": output_node["inputs"][0]["name"]
                }
            }
        }
    )

    flow_def = {
        "nodes": [input_node, prompt_node, output_node],
        "connections": connections
    }

    # Create the flow.

    response = create_flow(
        client, flow_name, flow_description, role_arn, flow_def)

    return response



def get_model_arn(client, model_id):
    """
    Gets the Amazon Resource Name (ARN) for a model.
    Args:
        client (str): Amazon Bedrock boto3 client.
        model_id (str): The id of the model.
    Returns:
        str: The ARN of the model.
    """

    try:
        # Call GetFoundationModelDetails operation
        response = client.get_foundation_model(modelIdentifier=model_id)

        # Extract model ARN from the response
        model_arn = response['modelDetails']['modelArn']

        return model_arn
```

```
    except ClientError as e:
        logger.exception("Client error getting model ARN: %s", {str(e)})
        raise

    except Exception as e:
        logger.exception("Unexpected error getting model ARN: %s", {str(e)})
        raise


def prepare_flow_version_and_alias(bedrock_agent_client,
                                   flow_id):
    """
    Prepares the flow and then creates a flow version and flow alias.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        flowd_id (str): The ID of the flow that you want to prepare.
    Returns: The flow_version and flow_alias.

    """

    status = prepare_flow(bedrock_agent_client, flow_id)

    flow_version = None
    flow_alias = None

    if status == 'Prepared':

        # Create the flow version and alias.
        flow_version = create_flow_version(bedrock_agent_client,
                                           flow_id,
                                           f"flow version for flow {flow_id}.")

        flow_alias = create_flow_alias(bedrock_agent_client,
                                       flow_id,
                                       flow_version,
                                       "latest",
                                       f"Alias for flow {flow_id}, version
 {flow_version}")

    return flow_version, flow_alias



def delete_role_resources(bedrock_agent_client,
```

```
                          iam_client,
                          role_name,
                          flow_id,
                          flow_version,
                          flow_alias):
    """
    Deletes the flow, flow alias, flow version, and IAM roles.
    Args:
        bedrock_agent_client: Amazon Bedrock Agent boto3 client.
        iam_client: Amazon IAM boto3 client.
        role_name (str): The name of the IAM role.
        flow_id (str): The id of the flow.
        flow_version (str): The version of the flow.
        flow_alias (str): The alias of the flow.
    """

    if flow_id is not None:
        if flow_alias is not None:
            delete_flow_alias(bedrock_agent_client, flow_id, flow_alias)
        if flow_version is not None:
            delete_flow_version(bedrock_agent_client,
                        flow_id, flow_version)
        delete_flow(bedrock_agent_client, flow_id)

    if role_name is not None:
        delete_flow_role(iam_client, role_name)



def main():
    """
    Creates, runs, and optionally deletes a Bedrock flow for generating music
 playlists.

    Note:
        Requires valid AWS credentials in the default profile
    """

    delete_choice = "y"
    try:

        # Get various boto3 clients.
        session = boto3.Session(profile_name='default')
        bedrock_agent_runtime_client = session.client('bedrock-agent-runtime')
```

```
        bedrock_agent_client = session.client('bedrock-agent')
        bedrock_client = session.client('bedrock')
        iam_client = session.client('iam')

        role_name = None
        flow_id = None
        flow_version = None
        flow_alias = None

        #Change the model as needed.
        prompt_model_id = "amazon.nova-pro-v1:0"

        # Base the flow name on the current date and time
        current_time = datetime.now()
        timestamp = current_time.strftime("%Y-%m-%d-%H-%M-%S")
        flow_name = f"FlowPlayList_{timestamp}"
        flow_description = "A flow to generate a music playlist."

        # Create a role for the flow.
        role_name = f"BedrockFlowRole-{flow_name}"
        role = create_flow_role(iam_client, role_name)
        role_arn = role['Arn']

        # Create the flow.
        response = create_playlist_flow(
            bedrock_agent_client, flow_name, flow_description, role_arn,
prompt_model_id)
        flow_id = response.get('id')

        if flow_id:
            # Update accessible resources in the role.
            model_arn = get_model_arn(bedrock_client, prompt_model_id)
            update_role_policy(iam_client, role_name, [
                               response.get('arn'), model_arn])

            # Prepare the flow and flow version.
            flow_version, flow_alias = prepare_flow_version_and_alias(
                bedrock_agent_client, flow_id)

            # Run the flow.
            if flow_version and flow_alias:
                run_playlist_flow(bedrock_agent_runtime_client,
                                  flow_id, flow_alias)
```

```
                delete_choice = input("Delete flow? y or n : ").lower()


            else:
                print("Couldn't run. Deleting flow and role.")
                delete_flow(bedrock_agent_client, flow_id)
                delete_flow_role(iam_client, role_name)
        else:
            print("Couldn't create flow.")


    except Exception as e:
        print(f"Fatal error: {str(e)}")

    finally:
        if delete_choice == 'y':
                delete_role_resources(bedrock_agent_client,
                                      iam_client,
                                      role_name,
                                      flow_id,
                                      flow_version,
                                      flow_alias)
        else:
            print("Flow not deleted. ")
            print(f"\tFlow ID: {flow_id}")
            print(f"\tFlow version: {flow_version}")
            print(f"\tFlow alias: {flow_alias}")
            print(f"\tRole ARN: {role_arn}")

        print("Done!")

if __name__ == "__main__":
    main()


def invoke_flow(client, flow_id, flow_alias_id, input_data):
    """
    Invoke an Amazon Bedrock flow and handle the response stream.

    Args:
        client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to invoke.
        flow_alias_id: The alias ID of the flow.
        input_data: Input data for the flow.
```

```
    Returns:
        Dict containing flow status and flow output.
    """

    response = None
    request_params = None

    request_params = {
            "flowIdentifier": flow_id,
            "flowAliasIdentifier": flow_alias_id,
            "inputs": [input_data],
            "enableTrace": True
        }


    response = client.invoke_flow(**request_params)

    flow_status = ""
    output= ""

    # Process the streaming response
    for event in response['responseStream']:

        # Check if flow is complete.
        if 'flowCompletionEvent' in event:
            flow_status = event['flowCompletionEvent']['completionReason']

        # Save the model output.
        elif 'flowOutputEvent' in event:
            output = event['flowOutputEvent']['content']['document']
            logger.info("Output : %s", output)

        # Log trace events.
        elif 'flowTraceEvent' in event:
            logger.info("Flow trace:  %s", event['flowTraceEvent'])

    return {
        "flow_status": flow_status,
        "output": output

    }
```

```
def run_playlist_flow(bedrock_agent_client, flow_id, flow_alias_id):
    """
    Runs the playlist generator flow.

    Args:
        bedrock_agent_client: Boto3 client for Amazon Bedrock agent runtime.
        flow_id: The ID of the flow to run.
        flow_alias_id: The alias ID of the flow.

    """


    print ("Welcome to the playlist generator flow.")
    # Get the initial prompt from the user.
    genre = input("Enter genre: ")
    number_of_songs = int(input("Enter number of songs: "))


    # Use prompt to create input data for the input node.
    flow_input_data = {
        "content": {
            "document": {
                "genre" : genre,
                "number" : number_of_songs
            }
        },
        "nodeName": "FlowInput",
        "nodeOutputName": "document"
    }

    try:

        result = invoke_flow(
                bedrock_agent_client, flow_id, flow_alias_id, flow_input_data)

        status = result['flow_status']

        if status == "SUCCESS":
                # The flow completed successfully.
                logger.info("The flow %s successfully completed.", flow_id)
                print(result['output'])
        else:
```

```
            logger.warning("Flow status: %s",status)

    except ClientError as e:
        print(f"Client error: {str(e)}")
        logger.error("Client error: %s", {str(e)})
        raise

    except Exception as e:
        logger.error("An error occurred: %s", {str(e)})
        logger.error("Error type: %s", {type(e)})
        raise



def create_flow_role(client, role_name):
    """
    Creates an IAM role for Amazon Bedrock with permissions to run a flow.

    Args:
        role_name (str): Name for the new IAM role.
    Returns:
        str: The role Amazon Resource Name.
    """


    # Trust relationship policy - allows Amazon Bedrock service to assume this role.
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {
                "Service": "bedrock.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }]
    }

    # Basic inline policy for for running a flow.

    resources = "*"

    bedrock_policy = {
        "Version":"2012-10-17",
        "Statement": [
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                # Using * as placeholder - Later you update with specific ARNs.
                "Resource": resources
            }
        ]
    }



    try:
        # Create the IAM role with trust policy
        logging.info("Creating role: %s",role_name)
        role = client.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy),
            Description="Role for Amazon Bedrock operations"
        )

        # Attach inline policy to the role
        print("Attaching inline policy")
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(bedrock_policy)
        )

        logging.info("Create Role ARN: %s", role['Role']['Arn'])
        return role['Role']

    except ClientError as e:
        logging.warning("Error creating role: %s", str(e))
        raise
    except Exception as e:
        logging.warning("Unexpected error: %s", str(e))
        raise


def update_role_policy(client, role_name, resource_arns):
```

```
    """
    Updates an IAM role's inline policy with specific resource ARNs.

    Args:
        role_name (str): Name of the existing role.
        resource_arns (list): List of resource ARNs to allow access to.
    """


    updated_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "bedrock:GetFlow",
                    "bedrock:InvokeModel",
                    "bedrock:Retrieve",
                    "bedrock:RetrieveAndGenerate"
                ],
                "Resource": resource_arns
            }
        ]
    }

    try:
        client.put_role_policy(
            RoleName=role_name,
            PolicyName=f"{role_name}-policy",
            PolicyDocument=json.dumps(updated_policy)
        )
        logging.info("Updated policy for role: %s",role_name)

    except ClientError as e:
        logging.warning("Error updating role policy: %s", str(e))
        raise


def delete_flow_role(client, role_name):
    """
    Deletes an IAM role.

    Args:
        role_name (str): Name of the role to delete.
```

```
    """



    try:
        # Detach and delete inline policies
        policies = client.list_role_policies(RoleName=role_name)['PolicyNames']
        for policy_name in policies:
            client.delete_role_policy(RoleName=role_name, PolicyName=policy_name)

        # Delete the role
        client.delete_role(RoleName=role_name)
        logging.info("Deleted role: %s", role_name)


    except ClientError as e:
        logging.info("Error Deleting role: %s", str(e))
        raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFlow
  - CreateFlowAlias
  - CreateFlowVersion
  - DeleteFlow
  - DeleteFlowAlias
  - DeleteFlowVersion
  - GetFlow
  - GetFlowAlias
  - GetFlowVersion
  - InvokeFlow
  - PrepareFlow

Orchestrieren von Anwendungen mit generativer KI mit Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finde und fasse das aktuelle Repositorium mit den höchsten Trends GitHub zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock
- Runtime der Agenten für Amazon Bedrock

- Step-Funktionen

CloudFormation Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren CloudFormation.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine REST-API, um COVID-19 Daten zu verfolgen

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die ein System simuliert, um tägliche Fälle COVID-19 in den Vereinigten Staaten anhand fiktiver Daten zu verfolgen.

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon DynamoDB AWS Lambda verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle COVID-19 in den Vereinigten Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über das API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.

- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

CloudFront Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren CloudFront.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetDistributionConfig`

Das folgende Codebeispiel zeigt die Verwendung`GetDistributionConfig`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- API-Details finden Sie GetDistributionConfig in der AWS API-Referenz zum SDK für Python (Boto3).

`ListDistributions`

Das folgende Codebeispiel zeigt die Verwendung. `ListDistributions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] == "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- API-Details finden Sie ListDistributions in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateDistribution`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateDistribution`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
```

```
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- API-Details finden Sie UpdateDistribution in der AWS API-Referenz zum SDK für Python (Boto3).

CloudWatch Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- API-Details finden Sie DeleteAlarms in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeAlarmsForMetric`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
        return alarm_iter
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS API-Referenz zum SDK für Python (Boto3).

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung. `DisableAlarmActions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- API-Details finden Sie DisableAlarmActions in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung. `EnableAlarmActions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.
```

```
        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- API-Details finden Sie EnableAlarmActions in der AWS API-Referenz zum SDK für Python (Boto3).

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung. `GetMetricStatistics`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
```

```
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period, Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
            raise
        else:
            return stats
```

- API-Details finden Sie GetMetricStatistics in der AWS API-Referenz zum SDK für Python (Boto3).

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung. `ListMetrics`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
```

```
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- API-Details finden Sie ListMetrics in der AWS API-Referenz zum SDK für Python (Boto3).

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung. `PutMetricAlarm`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
```

```
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                             alarm threshold before the alarm is set into an alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
```

```
                metric_name,
            )
            raise
        else:
            return alarm
```

- API-Details finden Sie PutMetricAlarm in der AWS API-Referenz zum SDK für Python (Boto3).

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung. `PutMetricData`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
```

```
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)
            raise
```

Fügen Sie eine Reihe von Daten in eine CloudWatch Metrik ein.

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
```

```
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
name)
            raise
```

- API-Details finden Sie PutMetricData in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Metriken und Alarme verwalten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Alarm, um eine CloudWatch Metrik zu beobachten.
- Geben Sie Daten in eine Metrik ein und lösen Sie den Alarm aus.
- Rufen Sie Daten aus dem Alarm ab.
- Löschen Sie den Alarm.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die CloudWatch Operationen umschließt.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
```

```
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
name)
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                             alarm threshold before the alarm is set into an alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
```

```
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
```

```
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)
            raise


    def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average value
                           or maximum value.
        :return: The retrieved statistics for the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period, Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
            raise
        else:
            return stats


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.
```

```
        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

Verwenden Sie die Wrapper-Klasse, um Daten in eine Metrik einzugeben, einen Alarm auszulösen, der die Metrik überwacht, und Daten aus dem Alarm abzurufen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the "
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")
```

```
    print(
        f"Sending data to trigger the alarm. This requires data over the threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))

    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")
```

```
    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteAlarms
  - DescribeAlarmsForMetric
  - DisableAlarmActions
  - EnableAlarmActions
  - GetMetricStatistics
  - ListMetrics
  - PutMetricAlarm
  - PutMetricData

CloudWatch Protokolliert Beispiele mithilfe des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

- Szenarien

Aktionen

`GetQueryResults`

Das folgende Codebeispiel zeigt die Verwendung`GetQueryResults`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- API-Details finden Sie GetQueryResults in der AWS API-Referenz zum SDK für Python (Boto3).

StartLiveTail

Das folgende Codebeispiel zeigt die Verwendung. `StartLiveTail`

SDK für Python (Boto3)

Binden Sie die erforderlichen Dateien ein.

```
import boto3
import time
from datetime import datetime
```

Starten Sie die Live-Tail-Sitzung.

```
    # Initialize the client
    client = boto3.client('logs')

    start_time = time.time()

    try:
        response = client.start_live_tail(
            logGroupIdentifiers=log_group_identifiers,
            logStreamNames=log_streams,
            logEventFilterPattern=filter_pattern
        )
        event_stream = response['responseStream']
        # Handle the events streamed back in the response
        for event in event_stream:
            # Set a timeout to close the stream.
            # This will end the Live Tail session.
            if (time.time() - start_time >= 10):
                event_stream.close()
                break
            # Handle when session is started
            if 'sessionStart' in event:
                session_start_event = event['sessionStart']
                print(session_start_event)
            # Handle when log event is given in a session update
            elif 'sessionUpdate' in event:
                log_events = event['sessionUpdate']['sessionResults']
                for log_event in log_events:
```

```
                    print('[{date}]
{log}'.format(date=datetime.fromtimestamp(log_event['timestamp']/1000),log=log_event['messa
            else:
                # On-stream exceptions are captured here
                raise RuntimeError(str(event))
    except Exception as e:
        print(e)
```

- API-Details finden Sie StartLiveTail in der AWS API-Referenz für SDK für Python (Boto3).

`StartQuery`

Das folgende Codebeispiel zeigt die Verwendung. `StartQuery`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(
```

```
self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
```

```
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")
```

- API-Details finden Sie StartQuery in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Ausführen einer umfangreichen Abfrage

Das folgende Codebeispiel zeigt, wie CloudWatch Logs verwendet werden, um mehr als 10.000 Datensätze abzufragen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Diese Datei ruft ein Beispielmodul zur Verwaltung von CloudWatch Abfragen mit mehr als 10.000 Ergebnissen auf.

```
import logging
import os
import sys

import boto3
from botocore.config import Config
```

```
from cloudwatch_query import CloudWatchQuery
from date_utilities import DateUtilities

# Configure logging at the module level.
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(filename)s:%(lineno)d - %(message)s",
)

DEFAULT_QUERY_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"


class CloudWatchLogsQueryRunner:
    def __init__(self):
        """
        Initializes the CloudWatchLogsQueryRunner class by setting up date utilities
        and creating a CloudWatch Logs client with retry configuration.
        """
        self.date_utilities = DateUtilities()
        self.cloudwatch_logs_client = self.create_cloudwatch_logs_client()

    def create_cloudwatch_logs_client(self):
        """
        Creates and returns a CloudWatch Logs client with a specified retry
 configuration.

        :return: A CloudWatch Logs client instance.
        :rtype: boto3.client
        """
        try:
            return boto3.client("logs", config=Config(retries={"max_attempts": 10}))
        except Exception as e:
            logging.error(f"Failed to create CloudWatch Logs client: {e}")
            sys.exit(1)

    def fetch_environment_variables(self):
        """
        Fetches and validates required environment variables for query start and end
 dates.
        Fetches the environment variable for log group, returning the default value
 if it
        does not exist.
```

```
        :return: Tuple of query start date and end date as integers and the log
group.
        :rtype: tuple
        :raises SystemExit: If required environment variables are missing or
invalid.
        """
        try:
            query_start_date = int(os.environ["QUERY_START_DATE"])
            query_end_date = int(os.environ["QUERY_END_DATE"])
        except KeyError:
            logging.error(
                "Both QUERY_START_DATE and QUERY_END_DATE environment variables are
required."
            )
            sys.exit(1)
        except ValueError as e:
            logging.error(f"Error parsing date environment variables: {e}")
            sys.exit(1)

        try:
            log_group = os.environ["QUERY_LOG_GROUP"]
        except KeyError:
            logging.warning("No QUERY_LOG_GROUP environment variable, using default
value")
            log_group = DEFAULT_QUERY_LOG_GROUP

        return query_start_date, query_end_date, log_group

    def convert_dates_to_iso8601(self, start_date, end_date):
        """
        Converts UNIX timestamp dates to ISO 8601 format using DateUtilities.

        :param start_date: The start date in UNIX timestamp.
        :type start_date: int
        :param end_date: The end date in UNIX timestamp.
        :type end_date: int
        :return: Start and end dates in ISO 8601 format.
        :rtype: tuple
        """
        start_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            start_date
        )
        end_date_iso8601 = self.date_utilities.convert_unix_timestamp_to_iso8601(
            end_date
```

```
        )
        return start_date_iso8601, end_date_iso8601

    def execute_query(
        self,
        start_date_iso8601,
        end_date_iso8601,
        log_group="/workflows/cloudwatch-logs/large-query",
        query="fields @timestamp, @message | sort @timestamp asc"
    ):
        """
        Creates a CloudWatchQuery instance and executes the query with provided date
 range.

        :param start_date_iso8601: The start date in ISO 8601 format.
        :type start_date_iso8601: str
        :param end_date_iso8601: The end date in ISO 8601 format.
        :type end_date_iso8601: str
        :param log_group: Log group to search: "/workflows/cloudwatch-logs/large-
query"
        :type log_group: str
        :param query: Query string to pass to the CloudWatchQuery instance
        :type query: str
        """
        cloudwatch_query = CloudWatchQuery(
            log_group=log_group,
            query_string=query
        )
        cloudwatch_query.query_logs((start_date_iso8601, end_date_iso8601))
        logging.info("Query executed successfully.")
        logging.info(
            f"Queries completed in {cloudwatch_query.query_duration} seconds. Total
 logs found: {len(cloudwatch_query.query_results)}"
        )


def main():
    """
    Main function to start a recursive CloudWatch logs query.
    Fetches required environment variables, converts dates, and executes the query.
    """
    logging.info("Starting a recursive CloudWatch logs query...")
    runner = CloudWatchLogsQueryRunner()
```

```
    query_start_date, query_end_date, log_group =
 runner.fetch_environment_variables()
    start_date_iso8601 = DateUtilities.convert_unix_timestamp_to_iso8601(
        query_start_date
    )
    end_date_iso8601 =
 DateUtilities.convert_unix_timestamp_to_iso8601(query_end_date)
    runner.execute_query(start_date_iso8601, end_date_iso8601, log_group=log_group)


if __name__ == "__main__":
    main()
```

Dieses Modul verarbeitet CloudWatch Abfragen mit mehr als 10.000 Ergebnissen.

```
import logging
import time
from datetime import datetime
import threading
import boto3

from date_utilities import DateUtilities

DEFAULT_QUERY = "fields @timestamp, @message | sort @timestamp asc"
DEFAULT_LOG_GROUP = "/workflows/cloudwatch-logs/large-query"

class DateOutOfBoundsError(Exception):
    """Exception raised when the date range for a query is out of bounds."""

    pass


class CloudWatchQuery:
    """
    A class to query AWS CloudWatch logs within a specified date range.

    :vartype date_range: tuple
    :ivar limit: Maximum number of log entries to return.
    :vartype limit: int
    :log_group str: Name of the log group to query
    :query_string str: query
    """
```

```
    def __init__(self, log_group: str = DEFAULT_LOG_GROUP, query_string:
str=DEFAULT_QUERY) -> None:
        self.lock = threading.Lock()
        self.log_group = log_group
        self.query_string = query_string
        self.query_results = []
        self.query_duration = None
        self.datetime_format = "%Y-%m-%d %H:%M:%S.%f"
        self.date_utilities = DateUtilities()
        self.limit = 10000

    def query_logs(self, date_range):
        """
        Executes a CloudWatch logs query for a specified date range and calculates
the execution time of the query.

        :return: A batch of logs retrieved from the CloudWatch logs query.
        :rtype: list
        """
        start_time = datetime.now()

        start_date, end_date = self.date_utilities.normalize_date_range_format(
            date_range, from_format="unix_timestamp", to_format="datetime"
        )

        logging.info(
            f"Original query:"
            f"\n       START:     {start_date}"
            f"\n       END:       {end_date}"
            f"\n       LOG GROUP: {self.log_group}"
        )
        self.recursive_query((start_date, end_date))
        end_time = datetime.now()
        self.query_duration = (end_time - start_time).total_seconds()

    def recursive_query(self, date_range):
        """
        Processes logs within a given date range, fetching batches of logs
recursively if necessary.

        :param date_range: The date range to fetch logs for, specified as a tuple
(start_timestamp, end_timestamp).
        :type date_range: tuple
```

```
        :return: None if the recursive fetching is continued or stops when the final
batch of logs is processed.
                 Although it doesn't explicitly return the query results, this
method accumulates all fetched logs
                 in the `self.query_results` attribute.
        :rtype: None
        """
        batch_of_logs = self.perform_query(date_range)
        # Add the batch to the accumulated logs
        with self.lock:
            self.query_results.extend(batch_of_logs)
        if len(batch_of_logs) == self.limit:
            logging.info(f"Fetched {self.limit}, checking for more...")
            most_recent_log = self.find_most_recent_log(batch_of_logs)
            most_recent_log_timestamp = next(
                item["value"]
                for item in most_recent_log
                if item["field"] == "@timestamp"
            )
            new_range = (most_recent_log_timestamp, date_range[1])
            midpoint = self.date_utilities.find_middle_time(new_range)

            first_half_thread = threading.Thread(
                target=self.recursive_query,
                args=((most_recent_log_timestamp, midpoint),),
            )
            second_half_thread = threading.Thread(
                target=self.recursive_query, args=((midpoint, date_range[1]),)
            )

            first_half_thread.start()
            second_half_thread.start()

            first_half_thread.join()
            second_half_thread.join()

    def find_most_recent_log(self, logs):
        """
        Search a list of log items and return most recent log entry.
        :param logs: A list of logs to analyze.
        :return: log
        :type :return List containing log item details
        """
        most_recent_log = None
```

```
        most_recent_date = "1970-01-01 00:00:00.000"

        for log in logs:
            for item in log:
                if item["field"] == "@timestamp":
                    logging.debug(f"Compared: {item['value']} to
{most_recent_date}")
                    if (
                        self.date_utilities.compare_dates(
                            item["value"], most_recent_date
                        )
                        == item["value"]
                    ):
                        logging.debug(f"New most recent: {item['value']}")
                        most_recent_date = item["value"]
                        most_recent_log = log
        logging.info(f"Most recent log date of batch: {most_recent_date}")
        return most_recent_log

    def perform_query(self, date_range):
        """
        Performs the actual CloudWatch log query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :return: A list containing the query results.
        :rtype: list
        """
        client = boto3.client("logs")
        try:
            try:
                start_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
                )
                end_time = round(

self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
                )
                response = client.start_query(
                    logGroupName=self.log_group,
                    startTime=start_time,
                    endTime=end_time,
```

```
                    queryString=self.query_string,
                    limit=self.limit,
                )
                query_id = response["queryId"]
            except client.exceptions.ResourceNotFoundException as e:
                raise DateOutOfBoundsError(f"Resource not found: {e}")
            while True:
                time.sleep(1)
                results = client.get_query_results(queryId=query_id)
                if results["status"] in [
                    "Complete",
                    "Failed",
                    "Cancelled",
                    "Timeout",
                    "Unknown",
                ]:
                    return results.get("results", [])
        except DateOutOfBoundsError:
            return []

    def _initiate_query(self, client, date_range, max_logs):
        """
        Initiates the CloudWatch logs query.

        :param date_range: A tuple representing the start and end datetime for the
query.
        :type date_range: tuple
        :param max_logs: The maximum number of logs to retrieve.
        :type max_logs: int
        :return: The query ID as a string.
        :rtype: str
        """
        try:
            start_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[0])
            )
            end_time = round(
                self.date_utilities.convert_iso8601_to_unix_timestamp(date_range[1])
            )
            response = client.start_query(
                logGroupName=self.log_group,
                startTime=start_time,
                endTime=end_time,
                queryString=self.query_string,
```

```
                limit=max_logs,
            )
            return response["queryId"]
        except client.exceptions.ResourceNotFoundException as e:
            raise DateOutOfBoundsError(f"Resource not found: {e}")


    def _wait_for_query_results(self, client, query_id):
        """
        Waits for the query to complete and retrieves the results.

        :param query_id: The ID of the initiated query.
        :type query_id: str
        :return: A list containing the results of the query.
        :rtype: list
        """
        while True:
            time.sleep(1)
            results = client.get_query_results(queryId=query_id)
            if results["status"] in [
                "Complete",
                "Failed",
                "Cancelled",
                "Timeout",
                "Unknown",
            ]:
                return results.get("results", [])
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - GetQueryResults
  - StartQuery

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel eines geplanten EventBridge Amazon-Ereignisses registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten zum späteren Abrufen in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erstellt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für Python (Boto3)

Zeigt, wie Sie das AWS SDK für Python (Boto3) zusammen mit Amazon Textract verwenden, um Text-, Formular- und Tabellenelemente in einem Dokumentbild zu erkennen. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)
```

```
# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- API-Details finden Sie ListUserPools in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung. `AdminGetUser`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
```

```
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
```

```
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API-Details finden Sie AdminGetUser in der AWS API-Referenz zum SDK für Python (Boto3).

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung. `AdminInitiateAuth`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
```

```
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful, this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
```

```
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is not
 "
                        "configured for TOTP MFA. This example requires TOTP MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API-Details finden Sie AdminInitiateAuth in der AWS API-Referenz zum SDK für Python (Boto3).

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung. `AdminRespondToAuthChallenge`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Reagieren Sie auf eine MFA-Herausforderung, indem Sie einen Code bereitstellen, der von einer zugehörigen MFA-Anwendung generiert wurde.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
```

```
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step of
        a two-factor sign-in. When sign-in is successful, it returns an access token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
```

```
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You might
 have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS API-Referenz zum SDK für Python (Boto3).

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung. `AssociateSoftwareToken`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
```

```
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API-Details finden Sie AssociateSoftwareToken in der AWS API-Referenz zum SDK für Python (Boto3).

`ConfirmDevice`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmDevice`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a new
        MFA code from the MFA application.
```

```
        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False. When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value, srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- API-Details finden Sie ConfirmDevice in der AWS API-Referenz zum SDK für Python (Boto3).

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung. `ConfirmSignUp`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- API-Details finden Sie ConfirmSignUp in der AWS API-Referenz zum SDK für Python (Boto3).

`InitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung. `InitiateAuth`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird veranschaulicht, wie die Authentifizierung mit einem nachverfolgten Gerät gestartet wird. Um die Anmeldung abzuschließen, muss der Client korrekt auf SRP-Abfragen (Secure Remote Password) reagieren.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.
```

```
        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
```

```
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API-Details finden Sie InitiateAuth in der AWS API-Referenz zum SDK für Python (Boto3).

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung. `ListUsers`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return users
```

- API-Details finden Sie ListUsers in der AWS API-Referenz zum SDK für Python (Boto3).

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung. `ResendConfirmationCode`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.
```

```
        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- API-Details finden Sie ResendConfirmationCode in der AWS API-Referenz zum SDK für Python (Boto3).

`RespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung. `RespondToAuthChallenge`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Melden Sie sich mit einem nachverfolgten Gerät an. Um die Anmeldung abzuschließen, muss der Client korrekt auf SRP-Abfragen (Secure Remote Password) reagieren.

```
class CognitoIdentityProviderWrapper:
```

```
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
```

```
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
```

```
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- API-Details finden Sie RespondToAuthChallenge in der AWS API-Referenz zum SDK für Python (Boto3).

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung. `SignUp`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
```

```
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- API-Details finden Sie SignUp in der AWS API-Referenz zum SDK für Python (Boto3).

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung. `VerifySoftwareToken`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
```

```
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- API-Details finden Sie VerifySoftwareToken in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon-Cognito-Funktionen einschließt, die im Szenario verwendet werden.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name, response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

```
    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful, this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
```

```
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is not
"
                        "configured for TOTP MFA. This example requires TOTP MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
```

```
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step of
        a two-factor sign-in. When sign-in is successful, it returns an access token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
```

```
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You might
have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
```

```
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False. When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value, srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
```

```
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
                 access token for the user.
        """
        try:
```

```
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
```

```
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

Erstellen Sie eine Klasse, die das Szenario ausführt. Dieses Beispiel registriert auch ein MFA-Gerät für die Nachverfolgung durch Amazon Cognito und zeigt Ihnen, wie Sie sich mithilfe eines Passworts und der Informationen des nachverfolgten Geräts anmelden. Dadurch wird die Eingabe eines neuen MFA-Codes unnötig.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ", q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
```

```
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ", q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA "
                "application, such as Google Authenticator."
            )
```

```
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to appear
in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens for
it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
```

```
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device {device_key}.
\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password, aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id, args.client_id)
    except Exception:
```

```
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Verwenden Sie Amazon Cognito-Identitätspools

Das folgende Codebeispiel zeigt, wie eine webbasierte Demoanwendung erstellt wird, die den Authentifizierungsablauf von Identitätspools demonstriert.

SDK für Python (Boto3)

Zeigt eine webbasierte Demoanwendung, die die Authentifizierungsabläufe von Amazon Cognito-Identitätspools demonstriert und es Benutzern ermöglicht, sowohl erweiterte als auch grundlegende Authentifizierungsabläufe mit verschiedenen Identitätsanbietern interaktiv zu erkunden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity Provider

Beispiele für Amazon Comprehend unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDocumentClassifier`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
```

```
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it immediately
        starts training on the data found in the specified Amazon S3 bucket.
Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
data.
        :param training_key: The prefix used to find training data in the training
                             bucket. If multiple objects have the same prefix, all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/{training_key}"},
```

```
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn
```

- API-Details finden Sie CreateDocumentClassifier in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteDocumentClassifier`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def delete(self):
```

```
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
            raise
```

- API-Details finden Sie DeleteDocumentClassifier in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDocumentClassificationJob`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDocumentClassificationJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None
```

```
    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job
```

- API-Details finden Sie DescribeDocumentClassificationJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDocumentClassifier`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
```

```
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier
```

- API-Details finden Sie DescribeDocumentClassifier in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeTopicsDetectionJob`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeTopicsDetectionJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- API-Details finden Sie DescribeTopicsDetectionJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectDominantLanguage`

Das folgende Codebeispiel zeigt die Verwendung. `DetectDominantLanguage`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- API-Details finden Sie DetectDominantLanguage in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectEntities`

Das folgende Codebeispiel zeigt die Verwendung. `DetectEntities`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
 places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
```

```
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities
```

- API-Details finden Sie DetectEntities in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectKeyPhrases`

Das folgende Codebeispiel zeigt die Verwendung. `DetectKeyPhrases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
```

```
                Text=text, LanguageCode=language_code
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases
```

- API-Details finden Sie DetectKeyPhrases in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectPiiEntities`

Das folgende Codebeispiel zeigt die Verwendung. `DetectPiiEntities`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.
```

```
        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- API-Details finden Sie DetectPiiEntities in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectSentiment`

Das folgende Codebeispiel zeigt die Verwendung. `DetectSentiment`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
```

```
        self.comprehend_client = comprehend_client


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- API-Details finden Sie DetectSentiment in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectSyntax`

Das folgende Codebeispiel zeigt die Verwendung. `DetectSyntax`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendDetect:
```

```
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

- API-Details finden Sie DetectSyntax in der AWS API-Referenz zum SDK für Python (Boto3).

`ListDocumentClassificationJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListDocumentClassificationJobs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

- API-Details finden Sie ListDocumentClassificationJobs in der AWS API-Referenz zum SDK für Python (Boto3).

`ListDocumentClassifiers`

Das folgende Codebeispiel zeigt die Verwendung. `ListDocumentClassifiers`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers
```

- API-Details finden Sie ListDocumentClassifiers in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTopicsDetectionJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListTopicsDetectionJobs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

- API-Details finden Sie ListTopicsDetectionJobs in der AWS API-Referenz zum SDK für Python (Boto3).

StartDocumentClassificationJob

Das folgende Codebeispiel zeigt die Verwendung. `StartDocumentClassificationJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
```

```
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- API-Details finden Sie StartDocumentClassificationJob in der AWS API-Referenz zum SDK für Python (Boto3).

`StartTopicsDetectionJob`

Das folgende Codebeispiel zeigt die Verwendung. `StartTopicsDetectionJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
 stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so
 you
        can call `describe_topics_detection_job` to get job status until it
```

```
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix, all
                            of them are used.
        :param input_format: The format of the input data, either one document per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- API-Details finden Sie StartTopicsDetectionJob in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erkennen von Dokumentenelementen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erkennen von Sprachen, Entitäten und Schlüsselbegriffen in einem Dokument
- Erkennen von persönlich identifizierbaren Informationen (PII) in einem Dokument
- Erkennen der Stimmung eines Dokuments
- Erkennen von Syntaxelementen in einem Dokument

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon-Comprehend-Aktionen umschließt.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
```

```
            languages = response["Languages"]
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages


    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities


    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code
```

```
            )
            phrases = response["KeyPhrases"]
            logger.info("Detected %s phrases.", len(phrases))
        except ClientError:
            logger.exception("Couldn't detect phrases.")
            raise
        else:
            return phrases


    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code
            )
            entities = response["Entities"]
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities


    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code
```

```
            )
            logger.info("Detected primary sentiment %s.", response["Sentiment"])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response


    def detect_syntax(self, text, language_code):
        """
        Detects syntactical elements of a document. Syntax tokens are portions of
        text along with their use as parts of speech, such as nouns, verbs, and
        interjections.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of syntax tokens along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_syntax(
                Text=text, LanguageCode=language_code
            )
            tokens = response["SyntaxTokens"]
            logger.info("Detected %s syntax tokens.", len(tokens))
        except ClientError:
            logger.exception("Couldn't detect syntax.")
            raise
        else:
            return tokens
```

Rufen Sie Funktionen der Wrapper-Klasse auf, um Entitäten, Formulierungen und mehr in einem Dokument zu erkennen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    comp_detect = ComprehendDetect(boto3.client("comprehend"))
    with open("detect_sample.txt") as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print("-" * 88)
    print(sample_text)
    print("-" * 88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]["LanguageCode"]

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment["SentimentScore"])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

    print("Thanks for watching!")
```

```
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DetectDominantLanguage
  - DetectEntities
  - DetectKeyPhrases
  - DetectPiiEntities
  - DetectSentiment
  - DetectSyntax

Entitäten in Text erkennen, der aus einem Bild extrahiert wurde

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter. TextractAndComprehendNotebook.ipynb

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Ausführen eines Amazon-Comprehend-Themenmodellierungsauftrags für Beispieldaten
- Rufen Sie von Informationen über den Auftrag ab.
- Extrahieren von Auftragsausgabedaten aus Amazon S3

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Wrapper-Klasse, um Amazon-Comprehend-Themenmodellierungsaktionen aufzurufen.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
```

```
        input bucket and written to the specified output bucket. Output data is
stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so
you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                            bucket. If multiple objects have the same prefix, all
                            of them are used.
        :param input_format: The format of the input data, either one document per
                            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                    grants Comprehend permission to read from the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
            )
            logger.info("Started topic modeling job %s.", response["JobId"])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
```

```
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id
            )
            job = response["TopicsDetectionJobProperties"]
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists topic modeling jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_topics_detection_jobs()
            jobs = response["TopicsDetectionJobPropertiesList"]
            logger.info("Got %s topic detection jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get topic detection jobs.")
            raise
        else:
            return jobs
```

Verwenden Sie die Wrapper-Klasse, um einen Themenmodellierungsauftrag auszuführen und Auftragsdaten abzurufen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    input_prefix = "input/"
    output_prefix = "output/"
    demo_resources = ComprehendDemoResources(
        boto3.resource("s3"), boto3.resource("iam")
    )
    topic_modeler = ComprehendTopicModeler(boto3.client("comprehend"))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup("comprehend-topic-modeler-demo")
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy(
        {"Bucket": "public-sample-us-west-2", "Key": "TopicModeling/Sample.txt"},
        f"{input_prefix}sample.txt",
    )

    print("Starting topic modeling job on sample data.")
    job_info = topic_modeler.start_job(
        "demo-topic-modeling-job",
        demo_resources.bucket.name,
        input_prefix,
        JobInputFormat.per_line,
        demo_resources.bucket.name,
        output_prefix,
        demo_resources.data_access_role.arn,
    )

    print(
        f"Waiting for job {job_info['JobId']} to complete. This typically takes "
        f"20 - 30 minutes."
    )
    job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = topic_modeler.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from the output Amazon S3 bucket: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = demo_resources.extract_job_output(job)
    lines = 10
```

```
    print(f"First {lines} lines of document topics output:")
    pprint(job_output["doc-topics.csv"]["data"][:lines])
    print(f"First {lines} lines of terms output:")
    pprint(job_output["topic-terms.csv"]["data"][:lines])

    print("Cleaning up resources created for the demo.")
    demo_resources.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeTopicsDetectionJob
  - ListTopicsDetectionJobs
  - StartTopicsDetectionJob

Trainieren eines benutzerdefinierten Klassifikators und klassifizieren von Dokumenten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Amazon-Comprehend-Klassifikators mit mehreren Labels
- Trainieren des Klassifikators anhand von Beispieldaten
- Ausführen eines Klassifizierungsauftrags für einen zweiten Datensatz
- Extrahieren der Auftragsausgabedaten aus Amazon S3

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Wrapper-Klasse, um die Aktionen des Amazon-Comprehend-Dokumentenklassifikators aufzurufen.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""

    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None


    def create(
        self,
        name,
        language_code,
        training_bucket,
        training_key,
        data_access_role_arn,
        mode,
    ):
        """
        Creates a custom classifier. After the classifier is created, it immediately
        starts training on the data found in the specified Amazon S3 bucket.
 Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
 data.
        :param training_key: The prefix used to find training data in the training
                             bucket. If multiple objects have the same prefix, all
                             of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     training bucket.
        :return: The ARN of the newly created classifier.
        """
```

```
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={"S3Uri": f"s3://{training_bucket}/{training_key}"},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value,
            )
            self.classifier_arn = response["DocumentClassifierArn"]
            logger.info("Started classifier creation. Arn is: %s.",
self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn


    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            classifier = response["DocumentClassifierProperties"]
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't get classifier %s.", self.classifier_arn)
            raise
        else:
            return classifier


    def list(self):
        """
        Lists custom classifiers for the current account.
```

```
        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response["DocumentClassifierPropertiesList"]
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception(
                "Couldn't get classifiers.",
            )
            raise
        else:
            return classifiers


    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn
            )
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
            raise


    def start_job(
        self,
        job_name,
        input_bucket,
        input_key,
        input_format,
        output_bucket,
        output_key,
        data_access_role_arn,
    ):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
```

```
        archive compressed in gzip format. The job runs asynchronously, so you can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document per
                             file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                     grants Comprehend permission to read from the
                                     input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    "S3Uri": f"s3://{input_bucket}/{input_key}",
                    "InputFormat": input_format.value,
                },
                OutputDataConfig={"S3Uri": f"s3://{output_bucket}/{output_key}"},
                DataAccessRoleArn=data_access_role_arn,
            )
            logger.info(
                "Document classification job %s is %s.", job_name,
response["JobStatus"]
            )
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response


    def describe_job(self, job_id):
        """
        Gets metadata about a classification job.
```

```
        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_document_classification_job(
                JobId=job_id
            )
            job = response["DocumentClassificationJobProperties"]
            logger.info("Got classification job %s.", job["JobName"])
        except ClientError:
            logger.exception("Couldn't get classification job %s.", job_id)
            raise
        else:
            return job


    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response["DocumentClassificationJobPropertiesList"]
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception(
                "Couldn't get document classification jobs.",
            )
            raise
        else:
            return jobs
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """
```

```
    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages
 resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = "training/"
        self.input_prefix = "input/"
        self.input_format = JobInputFormat.per_line
        self.output_prefix = "output/"

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup("comprehend-classifier-demo")

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace("\r", " ").replace("\n", " ").replace(",", ";")

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """
        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(f"{GITHUB_SEARCH_URL}?
{query}&per_page={issue_count}")
        if response.status_code == 200:
            issue_page = response.json()["items"]
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
```

```
                    "title": ClassifierDemo._sanitize_text(issue["title"]),
                    "body": ClassifierDemo._sanitize_text(issue["body"]),
                    "labels": {label["name"] for label in issue["labels"]},
                }
                for issue in issue_page
            ]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code,
                response.json(),
            )
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """
        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:closed+label:{label}",
                per_label_count,
            )
            for issue in issues:
                issue["labels"] = issue["labels"].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice
        any issue could be submitted to the classifier for labeling.

        :param training_labels: The set of labels to query for.
        :return: The set of issues used for input.
        """
```

```
        issues = []
        per_label_count = 5
        for label in training_labels:
            issues += self._get_issues(
                f"q=type:issue+repo:boto/boto3+state:open+label:{label}",
                per_label_count,
            )
        return issues

    def upload_issue_data(self, issues, training=False):
        """
        Uploads issue data to an Amazon S3 bucket, either for training or for input.
        The data is first put into the format expected by Comprehend. For training,
        the set of pipe-delimited labels is prepended to each document. For
        input, labels are not sent.

        :param issues: The set of issues to upload to Amazon S3.
        :param training: Indicates whether the issue data is used for training or
                         input.
        """
        try:
            obj_key = (
                self.training_prefix if training else self.input_prefix
            ) + "issues.txt"
            if training:
                issue_strings = [
                    f"{'|'.join(issue['labels'])},{issue['title']} {issue['body']}"
                    for issue in issues
                ]
            else:
                issue_strings = [
                    f"{issue['title']} {issue['body']}" for issue in issues
                ]
            issue_bytes = BytesIO("\n".join(issue_strings).encode("utf-8"))
            self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
            logger.info(
                "Uploaded data as %s to bucket %s.",
                obj_key,
                self.demo_resources.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload data to bucket %s.",
self.demo_resources.bucket.name
```

```
            )
            raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive["data"]:
                in_line = int(line["Line"])
                in_labels = input_issues[in_line]["labels"]
                out_labels = {
                    label["Name"]
                    for label in line["Labels"]
                    if float(label["Score"]) > 0.3
                }
                reconciled.append(
                    f"{line['File']}, line {in_line} has labels {in_labels}.\n"
                    f"\tClassifier assigned {out_labels}."
                )
        logger.info("Reconciled input and output labels.")
        return reconciled
```

Trainieren Sie einen Klassifikator anhand einer Reihe von GitHub Problemen mit bekannten Bezeichnungen und senden Sie dann eine zweite Gruppe von GitHub Problemen an den Klassifikator, damit sie gekennzeichnet werden können.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    comp_demo = ClassifierDemo(
        ComprehendDemoResources(boto3.resource("s3"), boto3.resource("iam"))
    )
    comp_classifier = ComprehendClassifier(boto3.client("comprehend"))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client
    )
    training_labels = {"bug", "feature-request", "dynamodb", "s3"}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = "doc-example-classifier"
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name,
        "en",
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label,
    )
    print(
        f"Waiting until {classifier_name} is trained. This typically takes "
        f"30-40 minutes."
    )
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
```

```
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        "issue_classification_job",
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn,
    )
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info["JobId"])

    job = comp_classifier.describe_job(job_info["JobId"])
    print(f"Job {job['JobId']} complete:")
    pprint(job)

    print(
        f"Getting job output data from Amazon S3: "
        f"{job['OutputDataConfig']['S3Uri']}."
    )
    job_output = comp_demo.extract_job_output(job)
    print("Job output:")
    pprint(job_output)

    print("Reconciling job output with labels from GitHub:")
    reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
    print(*reconciled_output, sep="\n")

    answer = input(f"Do you want to delete the classifier {classifier_name} (y/n)?
")
    if answer.lower() == "y":
        print(f"Deleting {classifier_name}.")
        comp_classifier.delete()

    print("Cleaning up resources created for the demo.")
    comp_demo.cleanup()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDocumentClassifier
  - DeleteDocumentClassifier
  - DescribeDocumentClassificationJob
  - DescribeDocumentClassifier
  - ListDocumentClassificationJobs
  - ListDocumentClassifiers
  - StartDocumentClassificationJob

AWS Config Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS Config.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteConfigRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteConfigRule`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def delete_config_rule(self, rule_name):
        """
        Delete the specified rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.config_client.delete_config_rule(ConfigRuleName=rule_name)
            logger.info("Deleted rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't delete rule %s.", rule_name)
            raise
```

- API-Details finden Sie DeleteConfigRule in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeConfigRules`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeConfigRules`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def describe_config_rule(self, rule_name):
        """
        Gets data for the specified rule.

        :param rule_name: The name of the rule to retrieve.
        :return: The rule data.
        """
        try:
            response = self.config_client.describe_config_rules(
                ConfigRuleNames=[rule_name]
            )
            rule = response["ConfigRules"]
            logger.info("Got data for rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't get data for rule %s.", rule_name)
            raise
        else:
            return rule
```

- API-Details finden Sie DescribeConfigRules in der AWS API-Referenz zum SDK für Python (Boto3).

`PutConfigRule`

Das folgende Codebeispiel zeigt die Verwendung. `PutConfigRule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """

    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client


    def put_config_rule(self, rule_name):
        """
        Sets a configuration rule that prohibits making Amazon S3 buckets publicly
        readable.

        :param rule_name: The name to give the rule.
        """
        try:
            self.config_client.put_config_rule(
                ConfigRule={
                    "ConfigRuleName": rule_name,
                    "Description": "S3 Public Read Prohibited Bucket Rule",
```

```
                    "Scope": {
                        "ComplianceResourceTypes": [
                            "AWS::S3::Bucket",
                        ],
                    },
                    "Source": {
                        "Owner": "AWS",
                        "SourceIdentifier": "S3_BUCKET_PUBLIC_READ_PROHIBITED",
                    },
                    "InputParameters": "{}",
                    "ConfigRuleState": "ACTIVE",
                }
            )
            logger.info("Created configuration rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't create configuration rule %s.", rule_name)
            raise
```

- API-Details finden Sie PutConfigRule in der AWS API-Referenz zum SDK für Python (Boto3).

AWS Control Tower Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS Control Tower.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Control Tower

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Control Tower beginnen.

SDK für Python (Boto3)

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from typing import Any, List


def hello_controltower(controltower_client: Any) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Control Tower client
    and list all available baselines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param controltower_client: A Boto3 AWS Control Tower Client object. This object
 wraps
                                the low-level AWS Control Tower service API.
    """
    print("Hello, AWS Control Tower! Let's list available baselines:\n")
    paginator = controltower_client.get_paginator("list_baselines")
    page_iterator = paginator.paginate()

    baseline_names: List[str] = []
    try:
        for page in page_iterator:
            for baseline in page["baselines"]:
                baseline_names.append(baseline["name"])

        print(f"{len(baseline_names)} baseline(s) retrieved.")
        for baseline_name in baseline_names:
```

```
            print(f"\t{baseline_name}")

    except controltower_client.exceptions.AccessDeniedException:
        print("Access denied. Please ensure you have the necessary permissions.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")


if __name__ == "__main__":
    hello_controltower(boto3.client("controltower"))
```

- API-Details finden Sie ListBaselines in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listen Sie Landing Zones auf.
- Listen Sie Baselines auf, aktivieren Sie sie, rufen Sie sie ab, setzen Sie sie zurück und deaktivieren Sie sie.
- Listen Sie Steuerelemente auf, aktivieren Sie sie, rufen Sie sie ab und deaktivieren Sie sie.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die AWS Control Tower Funktionen demonstriert.

```
class ControlTowerScenario:
    IDENTITY_CENTER_BASELINE = "baseline/LN25R72TTG6IGPTQ"
    stack_name = ""

    def __init__(
```

```
        self, controltower_wrapper: ControlTowerWrapper, org_client: boto3.client
    ):
        """
        :param controltower_wrapper: An instance of the ControlTowerWrapper class.
        :param org_client: A Boto3 Organization client.
        """
        self.controltower_wrapper = controltower_wrapper
        self.org_client = org_client
        self.stack = None
        self.ou_id = None
        self.ou_arn = None
        self.account_id = None
        self.landing_zone_id = None
        self.use_landing_zone = False

    def run_scenario(self) -> None:
        print("-" * 88)
        print(
            "\tWelcome to the AWS Control Tower with ControlCatalog example
 scenario."
        )
        print("-" * 88)

        print(
            "This demo will walk you through working with AWS Control Tower for
 landing zones,\n"
            "managing baselines, and working with controls."
        )

        self.account_id = boto3.client("sts").get_caller_identity()["Account"]

        print(
            "Some demo operations require the use of a landing zone. "
            "\nYou can use an existing landing zone or opt out of these operations
 in the demo."
            "\nFor instructions on how to set up a landing zone, "
            "\nsee https://docs.aws.amazon.com/controltower/latest/userguide/
getting-started-from-console.html"
        )
        # List available landing zones
        landing_zones = self.controltower_wrapper.list_landing_zones()
        if landing_zones:
            print("\nAvailable Landing Zones:")
            for i, lz in enumerate(landing_zones, 1):
```

```
                print(f"{i} {lz['arn']})")

            # Ask if user wants to use the first landing zone in the list
            if q.ask(
                f"Do you want to use the first landing zone in the list
({landing_zones[0]['arn']})? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = landing_zones[0]["arn"]
                print(f"Using landing zone ID: {self.landing_zone_id})")
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id
            elif q.ask(
                f"Do you want to use a different existing Landing Zone for this
demo? (y/n) ",
                q.is_yesno,
            ):
                self.use_landing_zone = True
                self.landing_zone_id = q.ask("Enter landing zone id: ", q.non_empty)
                # Set up organization and get Sandbox OU ID.
                sandbox_ou_id = self.setup_organization()
                # Store the OU ID for use in the CloudFormation template.
                self.ou_id = sandbox_ou_id

        # List and Enable Baseline.
        print("\nManaging Baselines:")
        control_tower_baseline = None
        identity_center_baseline = None
        baselines = self.controltower_wrapper.list_baselines()
        print("\nListing available Baselines:")
        for baseline in baselines:
            if baseline["name"] == "AWSControlTowerBaseline":
                control_tower_baseline = baseline
            print(f"{baseline['name']}")

        if self.use_landing_zone:
            print("\nListing enabled baselines:")
            enabled_baselines = self.controltower_wrapper.list_enabled_baselines()
            for baseline in enabled_baselines:
                # If the Identity Center baseline is enabled, the identifier must be
used for other baselines.
```

```
                if self.IDENTITY_CENTER_BASELINE in baseline["baselineIdentifier"]:
                    identity_center_baseline = baseline
                print(f"{baseline['baselineIdentifier']}")

            if q.ask(
                f"Do you want to enable the Control Tower Baseline? (y/n) ",
                q.is_yesno,
            ):
                print("\nEnabling Control Tower Baseline.")
                ic_baseline_arn = (
                    identity_center_baseline["arn"]
                    if identity_center_baseline
                    else None
                )
                baseline_arn = self.controltower_wrapper.enable_baseline(
                    self.ou_arn, ic_baseline_arn, control_tower_baseline["arn"],
"5.0"
                )
                if baseline_arn:
                    print(f"Enabled baseline ARN: {baseline_arn}")
                else:
                    # Find the enabled baseline so we can reset it.
                    for enabled_baseline in enabled_baselines:
                        if (
                            enabled_baseline["baselineIdentifier"]
                            == control_tower_baseline["arn"]
                        ):
                            baseline_arn = enabled_baseline["arn"]
                    if baseline_arn:
                        print("No change, the selected baseline was already
enabled.")

                if baseline_arn and q.ask(
                    f"Do you want to reset the Control Tower Baseline? (y/n) ",
                    q.is_yesno,
                ):
                    print(f"\nResetting Control Tower Baseline. {baseline_arn}")
                    operation_id = self.controltower_wrapper.reset_enabled_baseline(
                        baseline_arn
                    )
                    print(f"\nReset baseline operation id {operation_id}.")

                if baseline_arn and q.ask(
                    f"Do you want to disable the Control Tower Baseline? (y/n) ",
```

```
                    q.is_yesno,
                ):
                    print(f"Disabling baseline ARN: {baseline_arn}")
                    operation_id = self.controltower_wrapper.disable_baseline(
                        baseline_arn
                    )
                    print(f"\nDisabled baseline operation id {operation_id}.")

                    # Re-enable the baseline for the next step.
                    print("\nEnabling Control Tower Baseline.")
                    self.controltower_wrapper.enable_baseline(
                        self.ou_arn,
                        ic_baseline_arn,
                        control_tower_baseline["arn"],
                        "5.0",
                    )

        # List and Enable Controls.
        print("\nManaging Controls:")
        controls = self.controltower_wrapper.list_controls()
        print("\nListing first 5 available Controls:")
        for i, control in enumerate(controls[:5], 1):
            print(f"{i}. {control['Name']} - {control['Arn']}")

        if self.use_landing_zone:
            target_ou = self.ou_arn
            enabled_controls = self.controltower_wrapper.list_enabled_controls(
                target_ou
            )
            print("\nListing enabled controls:")
            for i, control in enumerate(enabled_controls, 1):
                print(f"{i}. {control['controlIdentifier']}")

            # Enable first non-enabled control as an example.
            enabled_control_arns = [control["arn"] for control in enabled_controls]
            control_arn = next(
                control["Arn"]
                for control in controls
                if control["Arn"] not in enabled_control_arns
            )

            if control_arn and q.ask(
                f"Do you want to enable the control {control_arn}? (y/n) ",
                q.is_yesno,
```

```
            ):
                print(f"\nEnabling control: {control_arn}")
                operation_id = self.controltower_wrapper.enable_control(
                    control_arn, target_ou
                )

                if operation_id:
                    print(f"Enabled control with operation id {operation_id}")

            if control_arn and q.ask(
                f"Do you want to disable the control? (y/n) ",
                q.is_yesno,
            ):
                print("\nDisabling the control...")
                operation_id = self.controltower_wrapper.disable_control(
                    control_arn, target_ou
                )
                print(f"Disable operation ID: {operation_id}")

        print("\nThis concludes the example scenario.")

        print("Thanks for watching!")
        print("-" * 88)

    def setup_organization(self):
        """
        Checks if the current account is part of an organization and creates one if
needed.
        Also ensures a Sandbox OU exists and returns its ID.

        :return: The ID of the Sandbox OU
        """
        print("\nChecking organization status...")

        try:
            # Check if account is part of an organization
            org_response = self.org_client.describe_organization()
            org_id = org_response["Organization"]["Id"]
            print(f"Account is part of organization: {org_id}")

        except ClientError as error:
            if error.response["Error"]["Code"] ==
"AWSOrganizationsNotInUseException":
                print("No organization found. Creating a new organization...")
```

```
            try:
                create_response = self.org_client.create_organization(
                    FeatureSet="ALL"
                )
                org_id = create_response["Organization"]["Id"]
                print(f"Created new organization: {org_id}")

                # Wait for organization to be available.
                waiter = self.org_client.get_waiter("organization_active")
                waiter.wait(
                    Organization=org_id,
                    WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                )

            except ClientError as create_error:
                logger.error(
                    "Couldn't create organization. Here's why: %s: %s",
                    create_error.response["Error"]["Code"],
                    create_error.response["Error"]["Message"],
                )
                raise
        else:
            logger.error(
                "Couldn't describe organization. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

    # Look for Sandbox OU.
    sandbox_ou_id = None
    paginator = self.org_client.get_paginator(
        "list_organizational_units_for_parent"
    )

    try:
        # Get root ID first.
        roots = self.org_client.list_roots()["Roots"]
        if not roots:
            raise ValueError("No root found in organization")
        root_id = roots[0]["Id"]

        # Search for existing Sandbox OU.
        print("Checking for Sandbox OU...")
```

```
            for page in paginator.paginate(ParentId=root_id):
                for ou in page["OrganizationalUnits"]:
                    if ou["Name"] == "Sandbox":
                        sandbox_ou_id = ou["Id"]
                        self.ou_arn = ou["Arn"]
                        print(f"Found existing Sandbox OU: {sandbox_ou_id}")
                        break
                if sandbox_ou_id:
                    break

            # Create Sandbox OU if it doesn't exist.
            if not sandbox_ou_id:
                print("Creating Sandbox OU...")
                create_ou_response = self.org_client.create_organizational_unit(
                    ParentId=root_id, Name="Sandbox"
                )
                sandbox_ou_id = create_ou_response["OrganizationalUnit"]["Id"]
                print(f"Created new Sandbox OU: {sandbox_ou_id}")

                # Wait for OU to be available.
                waiter = self.org_client.get_waiter("organizational_unit_active")
                waiter.wait(
                    OrganizationalUnitId=sandbox_ou_id,
                    WaiterConfig={"Delay": 5, "MaxAttempts": 12},
                )

        except ClientError as error:
            logger.error(
                "Couldn't set up Sandbox OU. Here's why: %s: %s",
                error.response["Error"]["Code"],
                error.response["Error"]["Message"],
            )
            raise

        return sandbox_ou_id


if __name__ == "__main__":
    try:
        org = boto3.client("organizations")
        control_tower_wrapper = ControlTowerWrapper.from_client()

        scenario = ControlTowerScenario(control_tower_wrapper, org)
        scenario.run_scenario()
```

```
    except Exception:
        logging.exception("Something went wrong with the scenario.")



class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
```

```
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center baseline
if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other than
it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
```

```
                parameters=parameters,
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception: %s:
%s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None


    def list_controls(self):
        """
        Lists all controls in the Control Tower control catalog.

        :return: List of controls.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controlcatalog_client.get_paginator("list_controls")
            controls = []
            for page in paginator.paginate():
                controls.extend(page["Controls"])
            return controls
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
```

```
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
```

```
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled baseline
to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
```

```
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
```

```
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
        try:
            paginator =
 self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
```

```
            return enabled_controls
        else:
            logger.error(
                "Couldn't list enabled controls. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
        raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

Aktionen

`DisableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`DisableBaseline`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_baseline(self, enabled_baseline_identifier: str):
        """
        Disables a baseline for a specific target and waits for the operation to
 complete.

        :param enabled_baseline_identifier: The identifier of the baseline to
 disable.
        :return: The operation ID.
        :raises ClientError: If disabling the baseline fails.
        """
        try:
            response = self.controltower_client.disable_baseline(
```

```
                enabledBaselineIdentifier=enabled_baseline_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["operationIdentifier"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                print(
                    f"Conflict disabling baseline: {err.response['Error']
['Message']}. Skipping disable step."
                )
                return None
            else:
                logger.error(
                    "Couldn't disable baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie DisableBaseline in der AWS API-Referenz zum SDK für Python (Boto3).

`DisableControl`

Das folgende Codebeispiel zeigt die Verwendung. `DisableControl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def disable_control(self, control_arn: str, target_identifier: str):
        """
        Disables a control for a specified target.

        :param control_arn: The ARN of the control to disable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If disabling the control fails.
        """
        try:
            response = self.controltower_client.disable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
```

```
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Control not found.")
            else:
                logger.error(
                    "Couldn't disable control. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie DisableControl in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `EnableBaseline`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
```

```
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_baseline(
        self,
        target_identifier: str,
        identity_center_baseline: str,
        baseline_identifier: str,
        baseline_version: str,
    ):
        """
        Enables a baseline for the specified target if it's not already enabled.

        :param target_identifier: The ARN of the target.
        :param baseline_identifier: The identifier of baseline to enable.
        :param identity_center_baseline: The identifier of identity center baseline
if it is enabled.
        :param baseline_version: The version of baseline to enable.
        :return: The enabled baseline ARN or None if already enabled.
        :raises ClientError: If enabling the baseline fails for reasons other than
it being already enabled.
        """
        try:
            # Only include parameters if identity_center_baseline is not empty
            parameters = []
            if identity_center_baseline:
                parameters = [
                    {
                        "key": "IdentityCenterEnabledBaselineArn",
                        "value": identity_center_baseline,
                    }
                ]

            response = self.controltower_client.enable_baseline(
                baselineIdentifier=baseline_identifier,
                baselineVersion=baseline_version,
                targetIdentifier=target_identifier,
                parameters=parameters,
```

```
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return response["arn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                if "already enabled" in err.response["Error"]["Message"]:
                    print("Baseline is already enabled for this target")
                else:
                    print(
                        "Unable to enable baseline due to validation exception: %s:
%s",
                        err.response["Error"]["Code"],
                        err.response["Error"]["Message"],
                    )
            logger.error(
                "Couldn't enable baseline. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            return None
```

- API-Details finden Sie EnableBaseline in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableControl`

Das folgende Codebeispiel zeigt die Verwendung. `EnableControl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def enable_control(self, control_arn: str, target_identifier: str):
        """
        Enables a control for a specified target.

        :param control_arn: The ARN of the control to enable.
        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: The operation ID.
        :raises ClientError: If enabling the control fails.
        """
        try:
            print(control_arn)
            print(target_identifier)
```

```
            response = self.controltower_client.enable_control(
                controlIdentifier=control_arn, targetIdentifier=target_identifier
            )

            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_control_operation(operation_id)
                print(f"Control operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)

            return operation_id

        except ClientError as err:
            if (
                err.response["Error"]["Code"] == "ValidationException"
                and "already enabled" in err.response["Error"]["Message"]
            ):
                logger.info("Control is already enabled for this target")
                return None
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return None
            logger.error(
                "Couldn't enable control. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie EnableControl in der AWS API-Referenz zum SDK für Python (Boto3).

`GetBaselineOperation`

Das folgende Codebeispiel zeigt die Verwendung. `GetBaselineOperation`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_baseline_operation(self, operation_id: str):
        """
        Gets the status of a baseline operation.

        :param operation_id: The ID of the baseline operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_baseline_operation(
                operationIdentifier=operation_id
            )
```

```
            return response["baselineOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get baseline operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie GetBaselineOperation in der AWS API-Referenz zum SDK für Python (Boto3).

`GetControlOperation`

Das folgende Codebeispiel zeigt die Verwendung. `GetControlOperation`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
```

```
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def get_control_operation(self, operation_id: str):
        """
        Gets the status of a control operation.

        :param operation_id: The ID of the control operation.
        :return: The operation status.
        :raises ClientError: If getting the operation status fails.
        """
        try:
            response = self.controltower_client.get_control_operation(
                operationIdentifier=operation_id
            )
            return response["controlOperation"]["status"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Operation not found.")
            else:
                logger.error(
                    "Couldn't get control operation status. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie GetControlOperation in der AWS API-Referenz zum SDK für Python (Boto3).

`ListBaselines`

Das folgende Codebeispiel zeigt die Verwendung. `ListBaselines`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_baselines(self):
        """
        Lists all baselines.

        :return: List of baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_baselines")
            baselines = []
            for page in paginator.paginate():
                baselines.extend(page["baselines"])
```

```
            return baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
                )
            else:
                logger.error(
                    "Couldn't list baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListBaselines in der AWS API-Referenz zum SDK für Python (Boto3).

`ListEnabledBaselines`

Das folgende Codebeispiel zeigt die Verwendung. `ListEnabledBaselines`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
```

```
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_baselines(self):
        """
        Lists all enabled baselines.

        :return: List of enabled baselines.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_baselines")
            enabled_baselines = []
            for page in paginator.paginate():
                enabled_baselines.extend(page["enabledBaselines"])
            return enabled_baselines

        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't list enabled baselines. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListEnabledBaselines in der AWS API-Referenz zum SDK für Python (Boto3).

`ListEnabledControls`

Das folgende Codebeispiel zeigt die Verwendung. `ListEnabledControls`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_enabled_controls(self, target_identifier: str):
        """
        Lists all enabled controls for a specific target.

        :param target_identifier: The identifier of the target (e.g., OU ARN).
        :return: List of enabled controls.
        :raises ClientError: If the listing operation fails.
        """
        enabled_controls = []
```

```
        try:
            paginator =
self.controltower_client.get_paginator("list_enabled_controls")

            for page in paginator.paginate(targetIdentifier=target_identifier):
                enabled_controls.extend(page["enabledControls"])
            return enabled_controls

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
permissions."
                )
                return enabled_controls
            elif (
                err.response["Error"]["Code"] == "ResourceNotFoundException"
                and "not registered with AWS Control Tower"
                in err.response["Error"]["Message"]
            ):
                logger.error("Control Tower must be enabled to work with controls.")
                return enabled_controls
            else:
                logger.error(
                    "Couldn't list enabled controls. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListEnabledControls in der AWS API-Referenz zum SDK für Python (Boto3).

`ListLandingZones`

Das folgende Codebeispiel zeigt die Verwendung. `ListLandingZones`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def list_landing_zones(self):
        """
        Lists all landing zones.

        :return: List of landing zones.
        :raises ClientError: If the listing operation fails.
        """
        try:
            paginator = self.controltower_client.get_paginator("list_landing_zones")
            landing_zones = []
            for page in paginator.paginate():
                landing_zones.extend(page["landingZones"])
```

```
            return landing_zones

        except ClientError as err:
            if err.response["Error"]["Code"] == "AccessDeniedException":
                logger.error(
                    "Access denied. Please ensure you have the necessary
 permissions."
                )
            else:
                logger.error(
                    "Couldn't list landing zones. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListLandingZones in der AWS API-Referenz zum SDK für Python (Boto3).

`ResetEnabledBaseline`

Das folgende Codebeispiel zeigt die Verwendung. `ResetEnabledBaseline`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ControlTowerWrapper:
    """Encapsulates AWS Control Tower and Control Catalog functionality."""

    def __init__(
        self, controltower_client: boto3.client, controlcatalog_client: boto3.client
    ):
```

```
        """
        :param controltower_client: A Boto3 Amazon ControlTower client.
        :param controlcatalog_client: A Boto3 Amazon ControlCatalog client.
        """
        self.controltower_client = controltower_client
        self.controlcatalog_client = controlcatalog_client

    @classmethod
    def from_client(cls):
        controltower_client = boto3.client("controltower")
        controlcatalog_client = boto3.client("controlcatalog")
        return cls(controltower_client, controlcatalog_client)


    def reset_enabled_baseline(self, enabled_baseline_identifier: str):
        """
        Resets an enabled baseline for a specific target.

        :param enabled_baseline_identifier: The identifier of the enabled baseline
 to reset.
        :return: The operation ID.
        :raises ClientError: If resetting the baseline fails.
        """
        try:
            response = self.controltower_client.reset_enabled_baseline(
                enabledBaselineIdentifier=enabled_baseline_identifier
            )
            operation_id = response["operationIdentifier"]
            while True:
                status = self.get_baseline_operation(operation_id)
                print(f"Baseline operation status: {status}")
                if status in ["SUCCEEDED", "FAILED"]:
                    break
                time.sleep(30)
            return operation_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Target not found.")
            else:
                logger.error(
                    "Couldn't reset enabled baseline. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
```

- API-Details finden Sie ResetEnabledBaseline in der AWS API-Referenz zum SDK für Python (Boto3).

Firehose-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Firehose Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung`PutRecord`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose delivery
 stream.
        It retries with exponential backoff in case of exceptions.
        """
        try:
            entry = self._create_record_entry(record)
```

```
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise
```

- API-Details finden Sie PutRecord in der AWS API-Referenz zum SDK für Python (Boto3).

`PutRecordBatch`

Das folgende Codebeispiel zeigt die Verwendung. `PutRecordBatch`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
```

```
            config (object): Configuration object with delivery stream name and
region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records. Error:
{e}")
```

- API-Details finden Sie PutRecordBatch in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Einfügen von Datensätzen in Firehose

Das folgende Codebeispiel zeigt, wie Firehose zur Verarbeitung einzelner Datensätze und Batch-Datensätze verwendet werden kann.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Skript werden Einzel- und Batch-Datensätze an Firehose übermittelt.

```
import json
import logging
import random
from datetime import datetime, timedelta

import backoff
import boto3

from config import get_config


def load_sample_data(path: str) -> dict:
    """
    Load sample data from a JSON file.

    Args:
        path (str): The file path to the JSON file containing sample data.

    Returns:
        dict: The loaded sample data as a dictionary.
    """
    with open(path, "r") as f:
        return json.load(f)


# Configure logging
```

```
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class FirehoseClient:
    """
    AWS Firehose client to send records and monitor metrics.

    Attributes:
        config (object): Configuration object with delivery stream name and region.
        delivery_stream_name (str): Name of the Firehose delivery stream.
        region (str): AWS region for Firehose and CloudWatch clients.
        firehose (boto3.client): Boto3 Firehose client.
        cloudwatch (boto3.client): Boto3 CloudWatch client.
    """

    def __init__(self, config):
        """
        Initialize the FirehoseClient.

        Args:
            config (object): Configuration object with delivery stream name and
 region.
        """
        self.config = config
        self.delivery_stream_name = config.delivery_stream_name
        self.region = config.region
        self.firehose = boto3.client("firehose", region_name=self.region)
        self.cloudwatch = boto3.client("cloudwatch", region_name=self.region)


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record(self, record: dict):
        """
        Put individual records to Firehose with backoff and retry.

        Args:
            record (dict): The data record to be sent to Firehose.

        This method attempts to send an individual record to the Firehose delivery
 stream.
        It retries with exponential backoff in case of exceptions.
```

```
        """
        try:
            entry = self._create_record_entry(record)
            response = self.firehose.put_record(
                DeliveryStreamName=self.delivery_stream_name, Record=entry
            )
            self._log_response(response, entry)
        except Exception:
            logger.info(f"Fail record: {record}.")
            raise


    @backoff.on_exception(
        backoff.expo, Exception, max_tries=5, jitter=backoff.full_jitter
    )
    def put_record_batch(self, data: list, batch_size: int = 500):
        """
        Put records in batches to Firehose with backoff and retry.

        Args:
            data (list): List of data records to be sent to Firehose.
            batch_size (int): Number of records to send in each batch. Default is
500.

        This method attempts to send records in batches to the Firehose delivery
stream.
        It retries with exponential backoff in case of exceptions.
        """
        for i in range(0, len(data), batch_size):
            batch = data[i : i + batch_size]
            record_dicts = [{"Data": json.dumps(record)} for record in batch]
            try:
                response = self.firehose.put_record_batch(
                    DeliveryStreamName=self.delivery_stream_name,
Records=record_dicts
                )
                self._log_batch_response(response, len(batch))
            except Exception as e:
                logger.info(f"Failed to send batch of {len(batch)} records. Error:
{e}")


    def get_metric_statistics(
        self,
```

```
        metric_name: str,
        start_time: datetime,
        end_time: datetime,
        period: int,
        statistics: list = ["Sum"],
    ) -> list:
        """
        Retrieve metric statistics from CloudWatch.

        Args:
            metric_name (str): The name of the metric.
            start_time (datetime): The start time for the metric statistics.
            end_time (datetime): The end time for the metric statistics.
            period (int): The granularity, in seconds, of the returned data points.
            statistics (list): A list of statistics to retrieve. Default is ['Sum'].

        Returns:
            list: List of datapoints containing the metric statistics.
        """
        response = self.cloudwatch.get_metric_statistics(
            Namespace="AWS/Firehose",
            MetricName=metric_name,
            Dimensions=[
                {"Name": "DeliveryStreamName", "Value": self.delivery_stream_name},
            ],
            StartTime=start_time,
            EndTime=end_time,
            Period=period,
            Statistics=statistics,
        )
        return response["Datapoints"]

    def monitor_metrics(self):
        """
        Monitor Firehose metrics for the last 5 minutes.

        This method retrieves and logs the 'IncomingBytes', 'IncomingRecords', and
'FailedPutCount' metrics
        from CloudWatch for the last 5 minutes.
        """
        end_time = datetime.utcnow()
        start_time = end_time - timedelta(minutes=10)
        period = int((end_time - start_time).total_seconds())
```

```
        metrics = {
            "IncomingBytes": self.get_metric_statistics(
                "IncomingBytes", start_time, end_time, period
            ),
            "IncomingRecords": self.get_metric_statistics(
                "IncomingRecords", start_time, end_time, period
            ),
            "FailedPutCount": self.get_metric_statistics(
                "FailedPutCount", start_time, end_time, period
            ),
        }

        for metric, datapoints in metrics.items():
            if datapoints:
                total_sum = sum(datapoint["Sum"] for datapoint in datapoints)
                if metric == "IncomingBytes":
                    logger.info(
                        f"{metric}: {round(total_sum)} ({total_sum / (1024 *
1024):.2f} MB)"
                    )
                else:
                    logger.info(f"{metric}: {round(total_sum)}")
            else:
                logger.info(f"No data found for {metric} over the last 5 minutes")


    def _create_record_entry(self, record: dict) -> dict:
        """
        Create a record entry for Firehose.

        Args:
            record (dict): The data record to be sent.

        Returns:
            dict: The record entry formatted for Firehose.

        Raises:
            Exception: If a simulated network error occurs.
        """
        if random.random() < 0.2:
            raise Exception("Simulated network error")
        elif random.random() < 0.1:
            return {"Data": '{"malformed": "data"'}
        else:
```

```
            return {"Data": json.dumps(record)}

    def _log_response(self, response: dict, entry: dict):
        """
        Log the response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record API call.
            entry (dict): The record entry that was sent.
        """
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            logger.info(f"Sent record: {entry}")
        else:
            logger.info(f"Fail record: {entry}")

    def _log_batch_response(self, response: dict, batch_size: int):
        """
        Log the batch response from Firehose.

        Args:
            response (dict): The response from the Firehose put_record_batch API
 call.
            batch_size (int): The number of records in the batch.
        """
        if response.get("FailedPutCount", 0) > 0:
            logger.info(
                f'Failed to send {response["FailedPutCount"]} records in batch of
 {batch_size}'
            )
        else:
            logger.info(f"Successfully sent batch of {batch_size} records")


if __name__ == "__main__":
    config = get_config()
    data = load_sample_data(config.sample_data_file)
    client = FirehoseClient(config)

    # Process the first 100 sample network records
    for record in data[:100]:
        try:
            client.put_record(record)
        except Exception as e:
            logger.info(f"Put record failed after retries and backoff: {e}")
```

```
    client.monitor_metrics()

    # Process remaining records using the batch method
    try:
        client.put_record_batch(data[100:])
    except Exception as e:
        logger.info(f"Put record batch failed after retries and backoff: {e}")
    client.monitor_metrics()
```

Diese Datei enthält die Konfiguration für das oben genannte Skript.

```
class Config:
    def __init__(self):
        self.delivery_stream_name = "ENTER YOUR DELIVERY STREAM NAME HERE"
        self.region = "us-east-1"
        self.sample_data_file = (
            "../../../../../scenarios/features/firehose/resources/
sample_records.json"
        )


def get_config():
    return Config()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - PutRecord
  - PutRecordBatch

Beispiele für Device Farm unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Device Farm Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Ausführen von Browsertests und Erstellen von Screenshots

Das folgende Codebeispiel zeigt, wie Sie mit Device-Farm-Browsertests durchführen und Screenshots erstellen können.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie PyTest und Selenium, um bestimmte Websites aufzurufen, Screenshots zu machen und den tatsächlichen Inhalt der Website mit dem erwarteten Inhalt zu vergleichen.

```
import datetime
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait


def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return (
            subprocess.check_output(["git", "rev-parse", "--short", "HEAD"])
```

```
            .decode("utf-8")
            .strip()
        )
    except:
        return "norepo"


class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.

    """

    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))

    def setup_method(self, method):
        """
        Set up a test.

        This makes sure that the session for an individual test is ready.

        The AWS credentials are read from the default ~/.aws/credentials or from the
        command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY environment
        variables.

        The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
        environment variable.
        """
        devicefarm_client = boto3.client("devicefarm")
        project_arn = os.environ.get("PROJECT_ARN", None)
        if project_arn is None:
            raise ValueError("Must set PROJECT_ARN")
        # Request a driver hub URL for the Selenium client
        testgrid_url_response = devicefarm_client.create_test_grid_url(
            projectArn=project_arn, expiresInSeconds=300
        )

        # We want a directory to save our files into. We're going to make a
 directory
        # in the current directory that holds our results.
        self.screenshot_path = os.path.join(
```

```
            ".", "results", get_git_hash() + "-" +
(datetime.date.today().isoformat())
        )
        if not os.path.exists(self.screenshot_path):
            os.makedirs(self.screenshot_path, exist_ok=True)

        # We want a Firefox instance on Windows
        desired_cap = DesiredCapabilities.FIREFOX
        desired_cap["platform"] = "windows"
        desired_cap["BrowserVersion"] = "latest"

        # Configure the webdriver with the appropriate remote endpoint.
        self.driver = webdriver.Remote(testgrid_url_response["url"], desired_cap)

        #
        # Auto-Tagging
        #

        # In order to get the Session ARN, we need to look up the session by the
        # Project ARN and session ID (from the driver).
        testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
            projectArn=project_arn, sessionId=self.driver.session_id
        )

        # Save the session's ARN so we can tag the session.
        self.session_arn = testgrid_session_arn_response["testGridSession"]["arn"]

        # In order to tag it, we're going to use the resourcegroupstaggingapi client
to
        # add a tag to the session ARN that we just got.
        tag_client = boto3.client("resourcegroupstaggingapi")
        tag_client.tag_resources(
            ResourceARNList=[self.session_arn],
            Tags={"TestSuite": f"testsuite {method.__name__}", "GitId":
get_git_hash()},
        )

    def teardown_method(self, method):
        """
        Clean up resources used by each method.
        """
        # End the Selenium session so we're off the clock.
        self.driver.quit()
```

```
    @pytest.mark.parametrize(
        "query,leading",
        [
            pytest.param(
                "Seattle",
                "Seattle (/si#ætəl/ (listen) see-AT-əl) is a seaport city on the
West Coast of the United States.",
            ),
            pytest.param(
                "Selenium",
                "Selenium is a chemical element with the symbol Se and atomic number
34.",
            ),
            pytest.param(
                "Amazon Locker",
                "Amazon Locker is a self-service package delivery service offered by
online retailer Amazon.",
            ),
            pytest.param(
                "Kootenai Falls",
                "Kootenai Falls is a waterfall on the Kootenay River located in
Lincoln County, Montana, just off U.S. Route 2.",
            ),
            pytest.param(
                "Dorayaki",
                "Dorayaki (####, ####, ####, ####) is a type of Japanese
confection.",
            ),
            pytest.param("Robot Face", "<|°_°|> (also known as Robot Face or
Robot)"),
        ],
    )
    def test_first_paragraph_text(self, query, leading):
        """
        This test looks at the first paragraph of a page on Wikipedia, comparing it
to
        a known leading sentence.

        If the leading sentence matches, the test passes. A screenshot is taken
before
        the final assertion is made, letting us debug if something isn't right.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
```

```
        # Find the search box, enter a query, and press enter
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # Wait for the search box to go stale -- This means we've navigated fully.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the leading paragraph of the article.
        lead = leading.lower()
        # Find the element...
        lead_para = self.driver.find_element(
            By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]"
        )
        # ... and copy out its text.
        our_text = lead_para.text.lower()
        our_text = our_text[: len(lead)]
        # Take a screenshot and compare the strings.
        self.save_screenshot(f"leadingpara_{query}.png")
        assert our_text.startswith(lead)

    @pytest.mark.parametrize(
        "query,expected",
        [
            pytest.param("Automation Testing", "Test Automation"),
            pytest.param("DevOps", "DevOps"),
            pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
            pytest.param("EarthBound", "EarthBound"),
            pytest.param("Covered Bridges Today", "Covered Bridges Today"),
            pytest.param("Kurt Godel", "Kurt Gödel"),
            pytest.param("N//ng language", "N#ng language"),
            pytest.param(
                "Who the Frick Is Jackson Pollock?", "Who the $&% Is Jackson
Pollock?"
            ),
        ],
    )
    def test_redirect_titles(self, query, expected):
        """
        A test comparing pages we expect to (or not to) redirect on Wikipedia.

        This test checks to see that the page ("query") redirects (or doesn't) to
the
```

```
        "expected" page title. Several of these are common synonyms ("Jackdaws...")
        while others are because of characters untypable by most keyboards ("N#ng
language")

        A screenshot is taken just before the final assertion is made to aid in
        debugging and verification.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter some text into it, and send an enter key.
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # wait until the page has rolled over -- once the search input handle is
stale,
        # the browser has navigated.
        WebDriverWait(self.driver, 5).until(
            expected_conditions.staleness_of(search_input)
        )
        # Get the first heading & take a screenshot
        our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
        self.save_screenshot(f"redirect_{query}.png")
        # did it match?
        assert our_text == expected.lower()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTestGridUrl
  - GetTestGridSession

Hochladen und Testen von Gerätepaketen

Das folgende Codebeispiel zeigt, wie Sie Pakete für mobile Geräte mit Device Farm hochladen und testen können.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie die kompilierte Android-Anwendung und die Testpakete auf Device Farm hoch, starten Sie einen Test, warten Sie auf den Abschluss des Tests und melden Sie die Ergebnisse.

```
import boto3
import os
import requests
import string
import random
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
    # This is our app under test.
    "appFilePath": "app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
    "poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
    "namePrefix": "MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage": "tests.zip",
}

client = boto3.client("devicefarm")

unique = (
    config["namePrefix"]
    + "-"
    + (datetime.date.today().isoformat())
    + ("".join(random.sample(string.ascii_letters, 8)))
```

```
)

print(
    f"The unique identifier for this run is '{unique}'. All uploads will be prefixed
 "
    f"with this."
)


def upload_df_file(filename, type_, mime="application/octet-stream"):
    upload_response = client.create_upload(
        projectArn=config["projectArn"],
        name=unique + "_" + os.path.basename(filename),
        type=type_,
        contentType=mime,
    )
    upload_arn = upload_response["upload"]["arn"]
    # Extract the URL of the upload and use Requests to upload it.
    upload_url = upload_response["upload"]["url"]
    with open(filename, "rb") as file_stream:
        print(
            f"Uploading {filename} to Device Farm as "
            f"{upload_response['upload']['name']}... ",
            end="",
        )
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime}
        )
        print(" done")
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(
            f"Upload of {filename} in state {upload_response['upload']['status']} "
            f"after " + str(datetime.datetime.now() - started)
        )
        if upload_response["upload"]["status"] == "FAILED":
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is: \n"
                f"{+upload_response['upload']['message']}"
            )
        if upload_response["upload"]["status"] == "SUCCEEDED":
            break
```

```
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn


our_upload_arn = upload_df_file(config["appFilePath"], "ANDROID_APP")
our_test_package_arn = upload_df_file(
    config["testPackage"], "APPIUM_PYTHON_TEST_PACKAGE"
)
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],
    appArn=our_upload_arn,
    devicePoolArn=config["poolArn"],
    name=unique,
    test={
        "type": "APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn,
    },
)
run_arn = response["run"]["arn"]
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = "UNKNOWN"
try:
    while True:
        response = client.get_run(arn=run_arn)
        state = response["run"]["status"]
        if state == "COMPLETED" or state == "ERRORED":
            break
        else:
            print(
                f" Run {unique} in state {state}, total "
                f"time {datetime.datetime.now() - start_time}"
            )
            time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)
```

```
print(f"Tests finished in state {state} after {datetime.datetime.now() -
 start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), "results", unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response["jobs"]:
    job_name = job["name"]
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job["arn"])["suites"]
    for suite in suites:
        for test in client.list_tests(arn=suite["arn"])["tests"]:
            # Get the artifacts.
            for artifact_type in ["FILE", "SCREENSHOT", "LOG"]:
                artifacts = client.list_artifacts(type=artifact_type,
 arn=test["arn"])[
                    "artifacts"
                ]
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows &
 macOS.
                    path_to = os.path.join(
                        save_path,
                        job_name,
                        suite["name"],
                        test["name"].replace(":", "_"),
                    )
                    os.makedirs(path_to, exist_ok=True)
                    filename = (
                        artifact["type"]
                        + "_"
                        + artifact["name"]
                        + "."
                        + artifact["extension"]
                    )
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, "wb") as fn:
                        with requests.get(
                            artifact["url"], allow_redirects=True
                        ) as request:
```

```
                                fn.write(request.content)
print("Finished")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateUpload
  - GetRun
  - GetUpload
  - ListArtifacts
  - ListJobs
  - ListSuites
  - ListTests
  - ScheduleRun
  - StopRun

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'

def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

DynamoDB-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3

# Create a DynamoDB client using the default credentials and region
dynamodb = boto3.client("dynamodb")

# Initialize a paginator for the list_tables operation
paginator = dynamodb.get_paginator("list_tables")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(Limit=10)
```

```
# List the tables in the current AWS account
print("Here are the DynamoDB tables in your account:")

# Use pagination to list all tables
table_names = []

for page in page_iterator:
    for table_name in page.get("TableNames", []):
        print(f"- {table_name}")
        table_names.append(table_name)

if not table_names:
    print("You don't have any DynamoDB tables in your account.")
else:
    print(f"\nFound {len(table_names)} tables.")
```

- API-Details finden Sie ListTables in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die eine DynamoDB-Tabelle enthält.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
```

```
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists


    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
```

```
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table


    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return tables


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at least
                       the keys required by the schema that was specified when the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
```

```
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
```

```
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p": plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response = self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]
```

```
    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

        return movies


    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
```

```
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
def get_sample_movie_data(movie_file_name):
    """
    Gets sample movie data, either from a local file or by first downloading it from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in
 JSON format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
```

```
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(
            "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip"
        )
        movie_zip = ZipFile(BytesIO(movie_content.content))
        movie_zip.extractall()

    try:
        with open(movie_file_name) as movie_file:
            movie_data = json.load(movie_file, parse_float=Decimal)
    except FileNotFoundError:
        print(
            f"File {movie_file_name} not found. You must first download the file to
 "
            "run this demo. See the README for instructions."
        )
        raise
    else:
        # The sample file lists over 4000 movies, return only the first 250.
        return movie_data[:250]
```

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print("-" * 88)

    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")

    my_movie = Question.ask_questions(
```

```
        [
            Question(
                "title", "Enter the title of a movie you want to add to the table: "
            ),
            Question("year", "What year was it released? ", Question.is_int),
            Question(
                "rating",
                "On a scale of 1 - 10, how do you rate it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question("plot", "Summarize the plot for me: "),
        ]
    )
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'.")
    print("-" * 88)

    movie_update = Question.ask_questions(
        [
            Question(
                "rating",
                f"\nLet's update your movie.\nYou rated it {my_movie['rating']},
what new "
                f"rating would you give it? ",
                Question.is_float,
                Question.in_range(1, 10),
            ),
            Question(
                "plot",
                f"You summarized the plot as '{my_movie['plot']}'.\nWhat would you
say now? ",
            ),
        ]
    )
    my_movie.update(movie_update)
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print("-" * 88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
```

```
        movies.write_batch(movie_data)
        print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
    print("-" * 88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
    if Question.ask_question(
        f"Let's move on...do you want to get info about '{title}'? (y/n) ",
        Question.is_yesno,
    ):
        movie = movies.get_movie(title, 2001)
        print("\nHere's what I found:")
        pprint(movie)
    print("-" * 88)

    ask_for_year = True
    while ask_for_year:
        release_year = Question.ask_question(
            f"\nLet's get a list of movies released in a given year. Enter a year
between "
            f"1972 and 2018: ",
            Question.is_int,
            Question.in_range(1972, 2018),
        )
        releases = movies.query_movies(release_year)
        if releases:
            print(f"There were {len(releases)} movies released in {release_year}:")
            for release in releases:
                print(f"\t{release['title']}")
            ask_for_year = False
        else:
            print(f"I don't know about any movies released in {release_year}!")
            ask_for_year = Question.ask_question(
                "Try another year? (y/n) ", Question.is_yesno
            )
    print("-" * 88)

    years = Question.ask_questions(
        [
            Question(
                "first",
                f"\nNow let's scan for movies released in a range of years. Enter a
year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
```

```
            ),
            Question(
                "second",
                "Now enter another year: ",
                Question.is_int,
                Question.in_range(1972, 2018),
            ),
        ]
    )
    releases = movies.scan_movies(years)
    if releases:
        count = Question.ask_question(
            f"\nFound {len(releases)} movies. How many do you want to see? ",
            Question.is_int,
            Question.in_range(1, len(releases)),
        )
        print(f"\nHere are your {count} movies:\n")
        pprint(releases[:count])
    else:
        print(
            f"I don't know about any movies released between {years['first']} "
            f"and {years['second']}."
        )
    print("-" * 88)

    if Question.ask_question(
        f"\nLet's remove your movie from the table. Do you want to remove "
        f"'{my_movie['title']}'? (y/n)",
        Question.is_yesno,
    ):
        movies.delete_movie(my_movie["title"], my_movie["year"])
        print(f"\nRemoved '{my_movie['title']}' from the table.")
    print("-" * 88)

    if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
        movies.delete_table()
        print(f"Deleted {table_name}.")
    else:
        print(
            "Don't forget to delete the table when you're done or you might incur "
            "charges on your account."
        )

    print("\nThanks for watching!")
```

```
    print("-" * 88)


if __name__ == "__main__":
    try:
        run_scenario(
            "doc-example-table-movies", "moviedata.json", boto3.resource("dynamodb")
        )
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

In diesem Szenario wird die folgende Helper-Klasse verwendet, um Fragen an einer Eingabeaufforderung zu stellen.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """

    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
                    multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators until
                           one fails or they all pass. Validators may also convert
 the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
```

```
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators
            )
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
                if answer is None:
                    print(complaint)
                    break
        return answer

    @staticmethod
    def non_empty(answer):
        """
        Validates that the answer is not empty.
        :return: The non-empty answer, or None.
        """
        return answer if answer != "" else None, "I need an answer. Please?"

    @staticmethod
    def is_yesno(answer):
        """
        Validates a yes/no answer.
        :return: True when the answer is 'y'; otherwise, False.
        """
        return answer.lower() == "y", ""
```

```
    @staticmethod
    def is_int(answer):
        """
        Validates that the answer can be converted to an int.
        :return: The int answer; otherwise, None.
        """
        try:
            int_answer = int(answer)
        except ValueError:
            int_answer = None
        return int_answer, f"{answer} must be a valid integer."

    @staticmethod
    def is_letter(answer):
        """
        Validates that the answer is a letter.
        :return The letter answer, converted to uppercase; otherwise, None.
        """
        return (
            answer.upper() if answer.isalpha() else None,
            f"{answer} must be a single letter.",
        )

    @staticmethod
    def is_float(answer):
        """
        Validate that the answer can be converted to a float.
        :return The float answer; otherwise, None.
        """
        try:
            float_answer = float(answer)
        except ValueError:
            float_answer = None
        return float_answer, f"{answer} must be a valid float."

    @staticmethod
    def in_range(lower, upper):
        """
        Validate that the answer is within a range. The answer must be of a type
that can
        be compared to the lower and upper bounds.
        :return: The answer, if it is within the range; otherwise, None.
        """
```

```
        def _validate(answer):
            return (
                answer if lower <= answer <= upper else None,
                f"{answer} must be between {lower} and {upper}.",
            )

        return _validate
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated with
                           each statement. This list must be in the same order as
 the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table
 "
                    "does not exist."
                )
            else:
```

```
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s:
%s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

- API-Details finden Sie BatchExecuteStatement in der AWS API-Referenz zum SDK für Python (Boto3).

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung. `BatchGetItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import decimal
import json
import logging
import os
import pprint
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource("dynamodb")

MAX_GET_SIZE = 100  # Amazon DynamoDB rejects a get batch larger than 100 items.
```

```
def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.

    When Amazon DynamoDB cannot process all items in a batch, a set of unprocessed
    keys is returned. This function uses an exponential backoff algorithm to retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most 100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1  # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get("Responses", []):
            retrieved[key] += response["Responses"][key]
        unprocessed = response["UnprocessedKeys"]
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key["Keys"]) for batch_key in batch_keys.values()]
            )
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.",
 unprocessed_count
            )
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
        else:
            break

    return retrieved
```

- API-Details finden Sie BatchGetItem in der AWS API-Referenz zum SDK für Python (Boto3).

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung. `BatchWriteItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """
```

```
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at least
                       the keys required by the schema that was specified when the
                       table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie BatchWriteItem in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Tabelle zum Speichern von Filmdaten.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def create_table(self, table_name):
        """
        Creates an Amazon DynamoDB table that can be used to store movie data.
        The table uses the release year of the movie as the partition key and the
        title as the sort key.

        :param table_name: The name of the table to create.
        :return: The newly created table.
        """
        try:
            self.table = self.dyn_resource.create_table(
                TableName=table_name,
                KeySchema=[
                    {"AttributeName": "year", "KeyType": "HASH"},  # Partition key
                    {"AttributeName": "title", "KeyType": "RANGE"},  # Sort key
                ],
                AttributeDefinitions=[
                    {"AttributeName": "year", "AttributeType": "N"},
                    {"AttributeName": "title", "AttributeType": "S"},
                ],
                BillingMode='PAY_PER_REQUEST',
            )
            self.table.wait_until_exists()
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.table
```

- API-Details finden Sie CreateTable in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_movie(self, title, year):
```

```
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Sie können eine Bedingung angeben, damit ein Element nur gelöscht wird, wenn es bestimmte Kriterien erfüllt.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item is
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={"year": year, "title": title},
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))},
            )
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.",
                    title,
                    rating,
                )
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie DeleteItem in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
```

```
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteTable in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """
        try:
            table = self.dyn_resource.Table(table_name)
            table.load()
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                exists = False
            else:
                logger.error(
                    "Couldn't check for existence of %s. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            self.table = table
        return exists
```

- API-Details finden Sie DescribeTable in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeTimeToLive`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeTimeToLive`

SDK für Python (Boto3)

Beschreiben Sie die TTL-Konfiguration für eine bestehende DynamoDB-Tabelle mithilfe von AWS SDK für Python (Boto3).

```
import boto3
```

```
def describe_ttl(table_name, region):
    """
    Describes TTL on an existing table, as well as a region.

    :param table_name: String representing the name of the table
    :param region: AWS Region of the table - example `us-east-1`
    :return: Time to live description.
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        ttl_description = dynamodb.describe_time_to_live(TableName=table_name)
        print(
            f"TimeToLive for table {table_name} is status
 {ttl_description['TimeToLiveDescription']['TimeToLiveStatus']}"
        )

        return ttl_description
    except Exception as e:
        print(f"Error describing table: {e}")
        raise


# Enter your own table name and AWS region
describe_ttl("your-table-name", "us-east-1")
```

- API-Details finden Sie DescribeTimeToLive in der AWS API-Referenz für SDK für Python (Boto3).

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung. `ExecuteStatement`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
 exist.",
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
            raise
        else:
            return output
```

- API-Details finden Sie ExecuteStatement in der AWS API-Referenz zum SDK für Python (Boto3).

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung. `GetItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
```

```
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={"year": year, "title": title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Item"]
```

- API-Details finden Sie GetItem in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung. `ListTables`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None
```

```
    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tables
```

- API-Details finden Sie ListTables in der AWS API-Referenz zum SDK für Python (Boto3).

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung. `PutItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.
```

```
    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    "year": year,
                    "title": title,
```

```
                    "info": {"plot": plot, "rating": Decimal(str(rating))},
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie PutItem in der AWS API-Referenz zum SDK für Python (Boto3).

`Query`

Das folgende Codebeispiel zeigt die Verwendung. `Query`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fragen Sie Elemente mithilfe eines Schlüsselbedingungsausdrucks ab.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
```

```
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response = self.table.query(KeyConditionExpression=Key("year").eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Items"]
```

Fragen Sie Elemente ab und projizieren Sie sie, um eine Teilmenge von Daten zurückzugeben.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def query_and_project_movies(self, year, title_bounds):
        """
        Query for movies that were released in a specified year and that have titles
        that start within a range of letters. A projection expression is used
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
        """
        try:
            response = self.table.query(
                ProjectionExpression="#yr, title, info.genres, info.actors[0]",
                ExpressionAttributeNames={"#yr": "year"},
                KeyConditionExpression=(
                    Key("year").eq(year)
                    & Key("title").between(
                        title_bounds["first"], title_bounds["second"]
                    )
                ),
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ValidationException":
                logger.warning(
                    "There's a validation error. Here's the message: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't query for movies. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return response["Items"]
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
    """
```

```
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            "FilterExpression": Key("year").between(
                year_range["first"], year_range["second"]
            ),
            "ProjectionExpression": "#yr, title, info.rating",
            "ExpressionAttributeNames": {"#yr": "year"},
        }
        try:
            done = False
            start_key = None
            while not done:
                if start_key:
                    scan_kwargs["ExclusiveStartKey"] = start_key
                response = self.table.scan(**scan_kwargs)
                movies.extend(response.get("Items", []))
                start_key = response.get("LastEvaluatedKey", None)
                done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        return movies
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS SDK für Python (Boto3).

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren Sie ein Element mithilfe eines Aktualisierungsausdrucks.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data.

    Example data structure for a movie record in this table:
        {
            "year": 1999,
            "title": "For Love of the Game",
            "info": {
                "directors": ["Sam Raimi"],
                "release_date": "1999-09-15T00:00:00Z",
                "rating": 6.3,
                "plot": "A washed up pitcher flashes through his career.",
                "rank": 4987,
                "running_time_secs": 8220,
                "actors": [
                    "Kevin Costner",
                    "Kelly Preston",
                    "John C. Reilly"
                ]
            }
        }
```

```
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        # The table variable is set during the scenario in the call to
        # 'exists' if the table exists. Otherwise, it is set by 'create_table'.
        self.table = None


    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to the give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={":r": Decimal(str(rating)), ":p": plot},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Aktualisieren Sie ein Element mithilfe eines Aktualisierungsausdrucks, der eine arithmetische Operation enthält.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic operation,
        you can adjust a value in a single request, rather than first getting its
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the movie.
        :return: The updated rating.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={":val": Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW",
            )
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title,
                self.table.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Attributes"]
```

Aktualisieren Sie ein Element nur, wenn es bestimmte Bedingungen erfüllt.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table


    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is greater
        than a specified threshold. If the movie does not list more than the
 threshold,
        no actors are removed.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param actor_threshold: The threshold of actors to check.
        :return: The movie data after the update.
        """
        try:
            response = self.table.update_item(
                Key={"year": year, "title": title},
                UpdateExpression="remove info.actors[0]",
                ConditionExpression="size(info.actors) > :num",
                ExpressionAttributeValues={":num": actor_threshold},
                ReturnValues="ALL_NEW",
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't update %s because it has fewer than %s actors.",
                    title,
                    actor_threshold + 1,
                )
            else:
                logger.error(
                    "Couldn't update movie %s. Here's why: %s: %s",
                    title,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["Attributes"]
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz zum SDK für Python (Boto3).

UpdateTimeToLive

Das folgende Codebeispiel zeigt die Verwendung. `UpdateTimeToLive`

SDK für Python (Boto3)

Aktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import boto3


def enable_ttl(table_name, ttl_attribute_name):
    """
    Enables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table
    :param ttl_attribute_name: The name of the TTL attribute being provided to the
 table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": True, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been enabled successfully.")
        else:
            print(
                f"Failed to enable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
```

```
        return response
    except Exception as ex:
        print("Couldn't enable TTL in table %s. Here's why: %s" % (table_name, ex))
        raise


# your values
enable_ttl("your-table-name", "expireAt")
```

Deaktivieren Sie TTL für eine vorhandene DynamoDB-Tabelle.

```
import boto3


def disable_ttl(table_name, ttl_attribute_name):
    """
    Disables TTL on DynamoDB table for a given attribute name
        on success, returns a status code of 200
        on error, throws an exception

    :param table_name: Name of the DynamoDB table being modified
    :param ttl_attribute_name: The name of the TTL attribute being provided to the
 table.
    """
    try:
        dynamodb = boto3.client("dynamodb")

        # Enable TTL on an existing DynamoDB table
        response = dynamodb.update_time_to_live(
            TableName=table_name,
            TimeToLiveSpecification={"Enabled": False, "AttributeName":
 ttl_attribute_name},
        )

        # In the returned response, check for a successful status code.
        if response["ResponseMetadata"]["HTTPStatusCode"] == 200:
            print("TTL has been disabled successfully.")
        else:
            print(
                f"Failed to disable TTL, status code {response['ResponseMetadata']
['HTTPStatusCode']}"
            )
```

```
    except Exception as ex:
        print("Couldn't disable TTL in table %s. Here's why: %s" % (table_name, ex))
        raise


# your values
disable_ttl("your-table-name", "expireAt")
```

- API-Details finden Sie UpdateTimeToLive in der AWS API-Referenz für SDK für Python (Boto3).

Szenarien

Beschleunigen von Lesevorgängen mit DAX

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen und Schreiben von Daten in eine Tabelle mit sowohl den DAX- als auch den SDK-Clients.
- Abrufen, Abfragen und Scannen der Tabelle mit beiden Clients und Vergleichen ihrer Leistung.

Weitere Informationen finden Sie unter Entwickeln mit dem DynamoDB-Accelerator-Client.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Tabelle mit dem DAX- oder Boto3-Client.

```
import boto3


def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.
```

```
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table_name = "TryDaxTable"
    params = {
        "TableName": table_name,
        "KeySchema": [
            {"AttributeName": "partition_key", "KeyType": "HASH"},
            {"AttributeName": "sort_key", "KeyType": "RANGE"},
        ],
        "AttributeDefinitions": [
            {"AttributeName": "partition_key", "AttributeType": "N"},
            {"AttributeName": "sort_key", "AttributeType": "N"},
        ],
        "BillingMode": "PAY_PER_REQUEST",
    }
    table = dyn_resource.create_table(**params)
    print(f"Creating {table_name}...")
    table.wait_until_exists()
    return table


if __name__ == "__main__":
    dax_table = create_dax_table()
    print(f"Created table.")
```

Schreiben Sie Testdaten in die Tabelle.

```
import boto3


def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
```

```
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    some_data = "X" * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(
                Item={
                    "partition_key": partition_key,
                    "sort_key": sort_key,
                    "some_data": some_data,
                }
            )
            print(f"Put item ({partition_key}, {sort_key}) succeeded.")


if __name__ == "__main__":
    write_key_count = 10
    write_item_size = 1000
    print(
        f"Writing {write_key_count*write_key_count} items to the table. "
        f"Each item is {write_item_size} characters."
    )
    write_data_to_dax_table(write_key_count, write_item_size)
```

Rufen Sie Elemente für eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client ab und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import sys
import time
import amazondax
import boto3


def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
```

```
    and reported.

    :param key_count: The number of items to get from the table in each iteration.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        for partition_key in range(1, key_count + 1):
            for sort_key in range(1, key_count + 1):
                table.get_item(
                    Key={"partition_key": partition_key, "sort_key": sort_key}
                )
                print(".", end="")
                sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the DAX client."
        )
```

```
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax
            )
    else:
        print(
            f"Getting each item from the table {test_iterations} times, "
            f"using the Boto3 client."
        )
        test_start, test_end = get_item_test(test_key_count, test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/ test_iterations}."
    )
```

Fragen Sie die Tabelle im Hinblick auf eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client ab und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import time
import sys
import amazondax
import boto3
from boto3.dynamodb.conditions import Key


def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
    Queries the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param partition_key: The partition key value to use in the query. The query
                          returns items that have partition keys equal to this
 value.
    :param sort_keys: The range of sort key values for the query. The query returns
                      items that have sort key values between these two values.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
```

```
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    key_condition_expression = Key("partition_key").eq(partition_key) & Key(
        "sort_key"
    ).between(*sort_keys)

    start = time.perf_counter()
    for _ in range(iterations):
        table.query(KeyConditionExpression=key_condition_expression)
        print(".", end="")
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations,
 dyn_resource=dax
            )
```

```
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations
        )

    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Scannen Sie die Tabelle auf eine Reihe von Iterationen sowohl für den DAX-Client als auch für den Boto3-Client und melden Sie die jeweils aufgewendete Zeit.

```
import argparse
import time
import sys
import amazondax
import boto3


def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print(".", end="")
        sys.stdout.flush()
    print()
```

```
    end = time.perf_counter()
    return start, end


if __name__ == "__main__":
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "endpoint_url",
        nargs="?",
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
 used.",
    )
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
 completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
 dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3
 client.")
        test_start, test_end = scan_test(test_iterations)
    print(
        f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{(test_end - test_start)/test_iterations}."
    )
```

Löschen Sie die Tabelle.

```
import boto3


def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
```

```
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource("dynamodb")

    table = dyn_resource.Table("TryDaxTable")
    table.delete()

    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()


if __name__ == "__main__":
    delete_dax_table()
    print("Table deleted!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTable
  - DeleteTable
  - GetItem
  - PutItem
  - Query
  - Scan

Vergleichen mehrerer Werte mit einem einzigen Attribut

Im folgenden Codebeispiel wird gezeigt, wie Sie mehrere Werte mit einem einzigen Attribut in DynamoDB vergleichen.

- Verwenden des IN-Operators, um mehrere Werte mit einem einzigen Attribut zu vergleichen
- Vergleichen des IN-Operators mit mehreren OR-Bedingungen
- Verständnis der Vorteilen von IN hinsichtlich Leistung und Ausdruckskomplexität

SDK für Python (Boto3)

Vergleichen Sie mehrere Werte mit einem einzigen Attribut mithilfe von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key
from typing import Any, Dict, List, Optional


def compare_multiple_values(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Query or scan a DynamoDB table to find items where an attribute matches any
 value from a list.

    This function demonstrates the use of the IN operator to compare a single
 attribute
    against multiple possible values, which is more efficient than using multiple OR
 conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
 values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key attribute
 for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
 query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create the filter expression using the is_in method
    filter_expression = Attr(attribute_name).is_in(values_list)

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value:
```

```
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition, FilterExpression=filter_expression
        )
    else:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}


def compare_with_or_conditions(
    table_name: str,
    attribute_name: str,
    values_list: List[Any],
    partition_key_name: Optional[str] = None,
    partition_key_value: Optional[str] = None,
) -> Dict[str, Any]:
    """
    Alternative implementation using multiple OR conditions instead of the IN
 operator.

    This function is provided for comparison to show why using the IN operator is
 preferable.
    With many values, this approach becomes verbose and less efficient.

    Args:
```

```
        table_name (str): The name of the DynamoDB table.
        attribute_name (str): The name of the attribute to compare against the
values list.
        values_list (List[Any]): List of values to compare the attribute against.
        partition_key_name (Optional[str]): The name of the partition key attribute
for query operations.
        partition_key_value (Optional[str]): The value of the partition key to
query.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the matching items.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create a filter expression with multiple OR conditions
    filter_expression = None
    for value in values_list:
        condition = Attr(attribute_name).eq(value)
        if filter_expression is None:
            filter_expression = condition
        else:
            filter_expression = filter_expression | condition

    # If partition key is provided, perform a query operation
    if partition_key_name and partition_key_value and filter_expression:
        key_condition = Key(partition_key_name).eq(partition_key_value)
        response = table.query(
            KeyConditionExpression=key_condition, FilterExpression=filter_expression
        )
    elif filter_expression:
        # Otherwise, perform a scan operation
        response = table.scan(FilterExpression=filter_expression)
    else:
        # Return empty response if no values provided
        return {"Items": [], "Count": 0}

    # Handle pagination if there are more results
    items = response.get("Items", [])
    while "LastEvaluatedKey" in response:
        if partition_key_name and partition_key_value:
            response = table.query(
                KeyConditionExpression=key_condition,
```

```
                FilterExpression=filter_expression,
                ExclusiveStartKey=response["LastEvaluatedKey"],
            )
        else:
            response = table.scan(
                FilterExpression=filter_expression,
 ExclusiveStartKey=response["LastEvaluatedKey"]
            )
        items.extend(response.get("Items", []))

    # Return the complete result
    return {"Items": items, "Count": len(items)}
```

Anwendungsbeispiel für den Vergleich mehrerer Werte mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the compare_multiple_values function."""
    # Example parameters
    table_name = "Products"
    attribute_name = "Category"
    values_list = ["Electronics", "Computers", "Accessories"]

    print(f"Searching for products in any of these categories: {values_list}")

    # Using the IN operator (recommended approach)
    print("\nApproach 1: Using the IN operator")
    response = compare_multiple_values(
        table_name=table_name, attribute_name=attribute_name,
 values_list=values_list
    )

    print(f"Found {response['Count']} products in the specified categories")

    # Using multiple OR conditions (alternative approach)
    print("\nApproach 2: Using multiple OR conditions")
    response2 = compare_with_or_conditions(
        table_name=table_name, attribute_name=attribute_name,
 values_list=values_list
    )
```

```
    print(f"Found {response2['Count']} products in the specified categories")

    # Example with a query operation
    print("\nQuerying a specific manufacturer's products in multiple categories")
    partition_key_name = "Manufacturer"
    partition_key_value = "Acme"

    response3 = compare_multiple_values(
        table_name=table_name,
        attribute_name=attribute_name,
        values_list=values_list,
        partition_key_name=partition_key_name,
        partition_key_value=partition_key_value,
    )

    print(f"Found {response3['Count']} Acme products in the specified categories")

    # Explain the benefits of using the IN operator
    print("\nBenefits of using the IN operator:")
    print("1. More concise expression compared to multiple OR conditions")
    print("2. Better readability and maintainability")
    print("3. Potentially better performance with large value lists")
    print("4. Simpler code that's less prone to errors")
    print("5. Easier to modify when adding or removing values")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Query
  - Scan

Bedingtes Aktualisieren der TTL eines Elements

Im folgenden Codebeispiel wird gezeigt, wie die TTL eines Elements bedingungsabhängig aktualisiert wird.

SDK für Python (Boto3)

Aktualisieren Sie die TTL für ein vorhandenes DynamoDB-Element in einer Tabelle mit einer Bedingung.

```
from datetime import datetime, timedelta

import boto3
from botocore.exceptions import ClientError


def update_dynamodb_item_ttl(table_name, region, primary_key, sort_key,
 ttl_attribute):
    """
    Updates an existing record in a DynamoDB table with a new or updated TTL
 attribute.

    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :param ttl_attribute: name of the TTL attribute in the target DynamoDB table
    :return:
    """
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Generate updated TTL in epoch second format
        updated_expiration_time = int((datetime.now() +
 timedelta(days=90)).timestamp())

        # Define the update expression for adding/updating a new attribute
        update_expression = "SET newAttribute = :val1"

        # Define the condition expression for checking if 'expireAt' is not expired
        condition_expression = "expireAt > :val2"

        # Define the expression attribute values
        expression_attribute_values = {":val1": ttl_attribute, ":val2":
 updated_expiration_time}

        response = table.update_item(
            Key={"primaryKey": primary_key, "sortKey": sort_key},
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeValues=expression_attribute_values,
        )
```

```
        print("Item updated successfully.")
        return response["ResponseMetadata"]["HTTPStatusCode"]  # Ideally a 200 OK
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            print("Condition check failed: Item's 'expireAt' is expired.")
        else:
            print(f"Error updating item: {e}")
    except Exception as e:
        print(f"Error updating item: {e}")


# replace with your values
update_dynamodb_item_ttl(
    "your-table-name",
    "us-east-1",
    "your-partition-key-value",
    "your-sort-key-value",
    "your-ttl-attribute-value",
)
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Zählen von Ausdrucksoperatoren

Im folgenden Codebeispiel wird gezeigt, wie Ausdrucksoperatoren in DynamoDB gezählt werden.

- Machen Sie sich mit der Obergrenze von 300 Operatoren in DynamoDB vertraut.
- Zählen Sie Operatoren in komplexen Ausdrücken.
- Optimieren Sie Ausdrücke, um innerhalb der Grenzwerte zu bleiben.

SDK für Python (Boto3)

Demonstrieren Sie das Zählen von Ausdrucksoperatoren mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List, Optional, Tuple
```

```
def create_complex_filter_expression(
    attribute_name: str, values: List[Any], use_or: bool = True
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a complex filter expression with multiple conditions.

    This function demonstrates how to build a complex filter expression
    and count the number of operators used.

    Args:
        attribute_name (str): The name of the attribute to filter on.
        values (List[Any]): List of values to compare against.
        use_or (bool, optional): Whether to use OR between conditions. Defaults to
 True.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if not values:
        return "", {}, {}, 0

    # Initialize expression components
    filter_expression = ""
    expression_attribute_values = {}
    expression_attribute_names = {"#attr": attribute_name}
    operator_count = 0

    # Build the filter expression
    for i, value in enumerate(values):
        value_placeholder = f":val{i}"
        expression_attribute_values[value_placeholder] = value

        if i > 0:
            # Add OR or AND operator between conditions
            filter_expression += " OR " if use_or else " AND "
            operator_count += 1  # Count the OR/AND operator

        # Add the condition
        filter_expression += f"#attr = {value_placeholder}"
```

```
        operator_count += 1  # Count the = operator

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def create_nested_filter_expression(
    depth: int, conditions_per_level: int
) -> Tuple[str, Dict[str, Any], Dict[str, str], int]:
    """
    Create a deeply nested filter expression with multiple conditions.

    This function demonstrates how to build a complex nested filter expression
    and count the number of operators used.

    Args:
        depth (int): The depth of nesting.
        conditions_per_level (int): Number of conditions at each level.

    Returns:
        Tuple[str, Dict[str, Any], Dict[str, str], int]: A tuple containing:
            - The filter expression string
            - Expression attribute values
            - Expression attribute names
            - The number of operators used
    """
    if depth <= 0 or conditions_per_level <= 0:
        return "", {}, {}, 0

    # Initialize expression components
    expression_attribute_values = {}
    expression_attribute_names = {}
    operator_count = 0

    def build_nested_expression(current_depth: int, prefix: str) -> str:
        nonlocal operator_count

        if current_depth <= 0:
            return ""
```

```
        # Build conditions at this level
        conditions = []
        for i in range(conditions_per_level):
            attr_name = f"attr{prefix}_{i}"
            attr_placeholder = f"#attr{prefix}_{i}"
            val_placeholder = f":val{prefix}_{i}"

            expression_attribute_names[attr_placeholder] = attr_name
            expression_attribute_values[val_placeholder] = i

            conditions.append(f"{attr_placeholder} = {val_placeholder}")
            operator_count += 1  # Count the = operator

        # Join conditions with AND
        level_expression = " AND ".join(conditions)
        operator_count += max(0, len(conditions) - 1)  # Count the AND operators

        # If not at the deepest level, add nested expressions
        if current_depth > 1:
            nested_expr = build_nested_expression(current_depth - 1,
 f"{prefix}_{current_depth}")
            if nested_expr:
                level_expression = f"({level_expression}) OR ({nested_expr})"
                operator_count += 1  # Count the OR operator

        return level_expression

    # Build the expression starting from the top level
    filter_expression = build_nested_expression(depth, "1")

    return (
        filter_expression,
        expression_attribute_values,
        expression_attribute_names,
        operator_count,
    )


def count_operators_in_update_expression(update_expression: str) -> int:
    """
    Count the number of operators in an update expression.

    This function demonstrates how to count operators in an update expression
    based on DynamoDB's rules.
```

```
    Args:
        update_expression (str): The update expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count SET operations
    if "SET" in update_expression:
        set_section = (
            update_expression.split("SET")[1].split("REMOVE")[0].split("ADD")
[0].split("DELETE")[0]
        )

        # Count assignment operators (=)
        operator_count += set_section.count("=")

        # Count arithmetic operators (+, -)
        operator_count += set_section.count("+")
        operator_count += set_section.count("-")

        # Count list_append function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("list_append")

        # Count if_not_exists function calls (each counts as 1 operator)
        operator_count += set_section.lower().count("if_not_exists")

    # Count REMOVE operations (no additional operators)

    # Count ADD operations (each ADD counts as 1 operator)
    if "ADD" in update_expression:
        add_section = (
            update_expression.split("ADD")[1].split("DELETE")[0].split("SET")
[0].split("REMOVE")[0]
        )
        operator_count += add_section.count(",") + 1

    # Count DELETE operations (each DELETE counts as 1 operator)
    if "DELETE" in update_expression:
        delete_section = (
            update_expression.split("DELETE")[1].split("SET")[0].split("ADD")
[0].split("REMOVE")[0]
```

```
        )
        operator_count += delete_section.count(",") + 1

    return operator_count


def count_operators_in_condition_expression(condition_expression: str) -> int:
    """
    Count the number of operators in a condition expression.

    This function demonstrates how to count operators in a condition expression
    based on DynamoDB's rules.

    Args:
        condition_expression (str): The condition expression to analyze.

    Returns:
        int: The number of operators in the expression.
    """
    operator_count = 0

    # Count comparison operators
    comparison_operators = ["=", "<>", "<", "<=", ">", ">="]
    for op in comparison_operators:
        operator_count += condition_expression.count(op)

    # Count logical operators
    operator_count += condition_expression.upper().count(" AND ")
    operator_count += condition_expression.upper().count(" OR ")
    operator_count += condition_expression.upper().count("NOT ")

    # Count BETWEEN operator (counts as 2: BETWEEN + AND)
    between_count = condition_expression.upper().count(" BETWEEN ")
    operator_count += between_count * 2

    # Count IN operator (counts as 1 regardless of number of values)
    operator_count += condition_expression.upper().count(" IN ")

    # Count functions (each counts as 1 operator)
    functions = [
        "attribute_exists",
        "attribute_not_exists",
        "attribute_type",
        "begins_with",
```

```
        "contains",
        "size",
    ]
    for func in functions:
        operator_count += condition_expression.lower().count(func)

    return operator_count


# Note: This function is for demonstration purposes only and should be called from
 example_usage()
# It's not meant to be used directly as a test function
def _test_expression_limit(
    table_name: str, key: Dict[str, Any], operator_count: int, attribute_name: str =
 "TestAttribute"
) -> Tuple[bool, Optional[str]]:
    """
    Test if an expression with a specific number of operators exceeds the limit.

    This function demonstrates how to test the 300 operator limit by creating
    an expression with a specified number of operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        operator_count (int): The number of operators to include in the expression.
        attribute_name (str, optional): The name of the attribute to update.
 Defaults to "TestAttribute".

    Returns:
        Tuple[bool, Optional[str]]: A tuple containing:
            - A boolean indicating if the operation succeeded
            - The error message if it failed, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Create an update expression with the specified number of operators
    update_expression = f"SET #{attribute_name} = :val0"
    expression_attribute_names = {f"#{attribute_name}": attribute_name}
    expression_attribute_values = {":val0": 0}

    # Add additional SET operations to reach the desired operator count
```

```
    # Each assignment adds 1 operator
    for i in range(1, operator_count):
        attr_name = f"{attribute_name}{i}"
        attr_placeholder = f"#attr{i}"
        val_placeholder = f":val{i}"

        update_expression += f", {attr_placeholder} = {val_placeholder}"
        expression_attribute_names[attr_placeholder] = attr_name
        expression_attribute_values[val_placeholder] = i

    try:
        # Attempt the update operation
        table.update_item(
            Key=key,
            UpdateExpression=update_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues=expression_attribute_values,
        )
        return True, None
    except ClientError as e:
        error_message = e.response["Error"]["Message"]

        if "expression contains too many operators" in error_message.lower():
            return False, error_message
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung des Ausdrucksoperators „Zählen mit“. AWS SDK für Python (Boto3)

```
def example_usage():
    """Example of how to use the expression operator counting functions."""

    print("Example 1: Creating a complex filter expression with multiple
 conditions")
    attribute_name = "Status"
    values = ["Active", "Pending", "Processing", "Shipped", "Delivered"]

    filter_expr, expr_attr_vals, expr_attr_names, op_count =
 create_complex_filter_expression(
```

```
        attribute_name=attribute_name, values=values, use_or=True
    )

    print(f"Filter Expression: {filter_expr}")
    print(f"Expression Attribute Values: {expr_attr_vals}")
    print(f"Expression Attribute Names: {expr_attr_names}")
    print(f"Operator Count: {op_count}")

    print("\nExample 2: Creating a nested filter expression")
    nested_expr, nested_vals, nested_names, nested_count =
create_nested_filter_expression(
        depth=3, conditions_per_level=2
    )

    print(f"Nested Filter Expression: {nested_expr}")
    print(f"Operator Count: {nested_count}")

    print("\nExample 3: Counting operators in an update expression")
    update_expression = "SET #name = :name, #age = :age + :increment, #address.#city
= :city, #status = if_not_exists(#status, :default_status) REMOVE #old_field ADD
#counter :value DELETE #set_attr :set_val"
    update_op_count = count_operators_in_update_expression(update_expression)

    print(f"Update Expression: {update_expression}")
    print(f"Operator Count: {update_op_count}")

    print("\nExample 4: Counting operators in a condition expression")
    condition_expression = "(#status = :active OR #status = :pending) AND #price
BETWEEN :min_price AND :max_price AND attribute_exists(#category) AND NOT (#stock
<= :min_stock)"
    condition_op_count =
count_operators_in_condition_expression(condition_expression)

    print(f"Condition Expression: {condition_expression}")
    print(f"Operator Count: {condition_op_count}")

    print("\nExample 5: Testing the 300 operator limit")

    # This is just for demonstration - in a real application, you would use your
actual table
    # Note: This function is renamed to _test_expression_limit to avoid pytest
trying to run it
    print("In a real application, you would test with _test_expression_limit
function")
```

```
    print("Expression with 250 operators would be under the limit")
    print("Expression with 350 operators would exceed the 300 operator limit")

    print("\nOperator Counting Rules in DynamoDB:")
    print("1. Comparison Operators (=, <>, <, <=, >, >=): 1 operator each")
    print("2. Logical Operators (AND, OR, NOT): 1 operator each")
    print("3. BETWEEN: 2 operators (BETWEEN + AND)")
    print("4. IN: 1 operator (regardless of number of values)")
    print("5. Functions (attribute_exists, begins_with, etc.): 1 operator each")
    print("6. Arithmetic Operators (+, -): 1 operator each")
    print("7. SET assignments (=): 1 operator each")
    print("8. ADD and DELETE operations: 1 operator each")

    print("\nStrategies for Working Within the 300 Operator Limit:")
    print("1. Break operations into multiple requests")
    print("2. Use DynamoDB Transactions for complex operations")
    print("3. Optimize data model to reduce query complexity")
    print("4. Use application-side filtering for less critical filters")
    print("5. Consider using IN operator instead of multiple OR conditions")
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Erstellen Sie eine REST-API, um Daten zu verfolgen COVID-19

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die ein System simuliert, um tägliche Fälle COVID-19 in den Vereinigten Staaten anhand fiktiver Daten zu verfolgen.

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon DynamoDB AWS Lambda verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle COVID-19 in den Vereinigten Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über das API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.

- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.
- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Erstellen einer Messenger-Anwendung

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

SDK für Python (Boto3)

Zeigt, wie Sie with verwenden, AWS Step Functions um eine Messenger-Anwendung zu erstellen, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. AWS SDK für Python (Boto3) Die State-Machine ist in eine AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht wird.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step-Funktionen

Erstellen einer Tabelle mit aktiviertem Warmdurchsatz

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit aktiviertem Warmdurchsatz erstellt wird.

SDK für Python (Boto3)

Erstellen Sie eine DynamoDB-Tabelle mit Warmdurchsatz-Einstellung mithilfe von AWS SDK für Python (Boto3).

```
from boto3 import client
from botocore.exceptions import ClientError


def create_dynamodb_table_warm_throughput(
    table_name,
    partition_key,
    sort_key,
    misc_key_attr,
    non_key_attr,
    table_provisioned_read_units,
    table_provisioned_write_units,
    table_warm_reads,
    table_warm_writes,
    gsi_name,
    gsi_provisioned_read_units,
    gsi_provisioned_write_units,
    gsi_warm_reads,
    gsi_warm_writes,
    region_name="us-east-1",
):
    """
    Creates a DynamoDB table with a warm throughput setting configured.

    :param table_name: The name of the table to be created.
```

```
    :param partition_key: The partition key for the table being created.
    :param sort_key: The sort key for the table being created.
    :param misc_key_attr: A miscellaneous key attribute for the table being created.
    :param non_key_attr: A non-key attribute for the table being created.
    :param table_provisioned_read_units: The newly created table's provisioned read
capacity units.
    :param table_provisioned_write_units: The newly created table's provisioned
write capacity units.
    :param table_warm_reads: The read units per second setting for the table's warm
throughput.
    :param table_warm_writes: The write units per second setting for the table's
warm throughput.
    :param gsi_name: The name of the Global Secondary Index (GSI) to be created on
the table.
    :param gsi_provisioned_read_units: The configured Global Secondary Index (GSI)
provisioned read capacity units.
    :param gsi_provisioned_write_units: The configured Global Secondary Index (GSI)
provisioned write capacity units.
    :param gsi_warm_reads: The read units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param gsi_warm_writes: The write units per second setting for the Global
Secondary Index (GSI)'s warm throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Define the table attributes
        attribute_definitions = [
            {"AttributeName": partition_key, "AttributeType": "S"},
            {"AttributeName": sort_key, "AttributeType": "S"},
            {"AttributeName": misc_key_attr, "AttributeType": "N"},
        ]

        # Define the table key schema
        key_schema = [
            {"AttributeName": partition_key, "KeyType": "HASH"},
            {"AttributeName": sort_key, "KeyType": "RANGE"},
        ]

        # Define the provisioned throughput for the table
        provisioned_throughput = {
            "ReadCapacityUnits": table_provisioned_read_units,
            "WriteCapacityUnits": table_provisioned_write_units,
```

```
        }

        # Define the global secondary index
        gsi_key_schema = [
            {"AttributeName": sort_key, "KeyType": "HASH"},
            {"AttributeName": misc_key_attr, "KeyType": "RANGE"},
        ]
        gsi_projection = {"ProjectionType": "INCLUDE", "NonKeyAttributes":
[non_key_attr]}
        gsi_provisioned_throughput = {
            "ReadCapacityUnits": gsi_provisioned_read_units,
            "WriteCapacityUnits": gsi_provisioned_write_units,
        }
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_warm_reads,
            "WriteUnitsPerSecond": gsi_warm_writes,
        }
        global_secondary_indexes = [
            {
                "IndexName": gsi_name,
                "KeySchema": gsi_key_schema,
                "Projection": gsi_projection,
                "ProvisionedThroughput": gsi_provisioned_throughput,
                "WarmThroughput": gsi_warm_throughput,
            }
        ]

        # Define the warm throughput for the table
        warm_throughput = {
            "ReadUnitsPerSecond": table_warm_reads,
            "WriteUnitsPerSecond": table_warm_writes,
        }

        # Create the DynamoDB client and create the table
        response = ddb.create_table(
            TableName=table_name,
            AttributeDefinitions=attribute_definitions,
            KeySchema=key_schema,
            ProvisionedThroughput=provisioned_throughput,
            GlobalSecondaryIndexes=global_secondary_indexes,
            WarmThroughput=warm_throughput,
        )

        print(response)
```

```
        return response
    except ClientError as e:
        print(f"Error creating table: {e}")
        raise e
```

- API-Details finden Sie CreateTable in der AWS API-Referenz für SDK für Python (Boto3).

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in Amazon DynamoDB verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer DynamoDB-Tabelle gespeichert sind.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel im AWS Code Examples Repository unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen einer Websocket-Chat-Anwendung

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

SDK für Python (Boto3)

Zeigt, wie Sie das AWS SDK für Python (Boto3) mit Amazon API Gateway V2 verwenden, um eine Websocket-API zu erstellen, die in Amazon DynamoDB integriert ist. AWS Lambda

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.
- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Erstellen eines Elements mit einer TTL

Das folgende Codebeispiel zeigt, wie Sie ein Element mit TTL erstellen.

SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def create_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Creates a DynamoDB item with an attached expiry attribute.

    :param table_name: Table name for the boto3 resource to target when creating an
 item
    :param region: string representing the AWS region. Example: `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
```

```
    try:
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expiration time (90 days from now) in epoch second format
        expiration_time = int((datetime.now() + timedelta(days=90)).timestamp())

        item = {
            "primaryKey": primary_key,
            "sortKey": sort_key,
            "creationDate": current_time,
            "expireAt": expiration_time,
        }
        response = table.put_item(Item=item)

        print("Item created successfully.")
        return response
    except Exception as e:
        print(f"Error creating item: {e}")
        raise e


# Use your own values
create_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- API-Details finden Sie PutItem in der AWS API-Referenz zum SDK für Python (Boto3).

Ausführen erweiterter Abfrageoperationen

Im folgenden Codebeispiel wird gezeigt, wie Sie erweiterte Abfrageoperationen in DynamoDB ausführen.

- Abfragen von Tabellen mithilfe verschiedener Filter- und Bedingungstechniken
- Implementieren der Paginierung für große Ergebnismengen
- Verwenden globaler sekundärer Indizes für alternative Zugriffsmuster

- Anwenden von Konsistenzkontrollen auf Grundlage der Anwendungsanforderungen

SDK für Python (Boto3)

Abfrage mit stark konsistenten Lesevorgängen mit. AWS SDK für Python (Boto3)

```
import time

import boto3
from boto3.dynamodb.conditions import Key


def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent reads.
 Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
```

```
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
 ConsistentRead=consistent_read)

    return response
```

Abfrage unter Verwendung eines globalen sekundären Index mit AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

Abfrage mit Paginierung unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        page_size (int, optional): The number of items to return per page. Defaults
to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If None,
retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1
```

```
        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large result
 sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
```

```
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

Abfrage mit komplexen Filtern unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
```

```
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)
```

```
        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
```

```
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

Abfrage mit einem dynamisch erstellten Filterausdruck unter Verwendung von AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions where
            keys are attribute names and values are dictionaries with 'operator' and
 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
```

```
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
```

```
                    filter_expression = filter_expression & current_condition

    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Abfrage mit einem Filterausdruck und Verwendungslimit AWS SDK für Python (Boto3).

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults to
 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] = Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value": filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Ausführen von Listenoperationen

Das folgende Codebeispiel zeigt, wie Listenoperationen für DynamoDB-Tabellen ausgeführt werden.

- Hinzufügen von Elementen zu einem Listenattribut
- Entfernen von Elementen aus einem Listenattribut
- Aktualisieren von bestimmten Elementen in einer Liste nach Index
- Verwenden von Funktionen zum Anfügen und zum Auflisten von Indizes

SDK für Python (Boto3)

Demonstrieren Sie Listenoperationen mit AWS SDK für Python (Boto3).

```
import boto3
import json
from typing import Any, Dict, List, Optional, Union
```

```
def create_list_attribute(
    table_name: str, key: Dict[str, Any], list_name: str, list_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a new list attribute or replace an existing one.

    This function demonstrates how to create a new list attribute or replace
    an existing list with new values.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        list_values (List[Any]): The values to set in the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation to create or replace the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = :list_values",
        ExpressionAttributeValues={":list_values": list_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_append:
 List[Any]
) -> Dict[str, Any]:
    """
    Append values to the end of a list attribute.

    This function demonstrates how to use the list_append function to add elements
```

```
    to the end of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append to add values to the end of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append({list_name}, :values)",
        ExpressionAttributeValues={":values": values_to_append},
        ReturnValues="UPDATED_NEW",
    )

    return response


def prepend_to_list(
    table_name: str, key: Dict[str, Any], list_name: str, values_to_prepend:
 List[Any]
) -> Dict[str, Any]:
    """
    Prepend values to the beginning of a list attribute.

    This function demonstrates how to use the list_append function to add elements
    to the beginning of a list attribute.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        values_to_prepend (List[Any]): The values to prepend to the list.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use list_append with reversed order to add values to the beginning of the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = list_append(:values, {list_name})",
        ExpressionAttributeValues={":values": values_to_prepend},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int, new_value: Any
) -> Dict[str, Any]:
    """
    Update a specific element in a list attribute.

    This function demonstrates how to update a specific element in a list attribute
    using the index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to update.
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the index notation to update a specific element
    response = table.update_item(
```

```
        Key=key,
        UpdateExpression=f"SET {list_name}[{index}] = :value",
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_list_element(
    table_name: str, key: Dict[str, Any], list_name: str, index: int
) -> Dict[str, Any]:
    """
    Remove a specific element from a list attribute.

    This function demonstrates how to remove a specific element from a list
 attribute
    using the REMOVE action with index notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        index (int): The zero-based index of the element to remove.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the REMOVE action with index notation to remove a specific element
    response = table.update_item(
        Key=key, UpdateExpression=f"REMOVE {list_name}[{index}]",
 ReturnValues="UPDATED_NEW"
    )

    return response


def update_nested_list_element(
    table_name: str, key: Dict[str, Any], path: str, new_value: Any
```

```
) -> Dict[str, Any]:
    """
    Update an element in a nested list structure.

    This function demonstrates how to update an element in a nested list structure
    using expression attribute names for the path components.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested element (e.g., "parent[0].child[1]").
        new_value (Any): The new value for the element.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Define a type for path parts
    path_part = Dict[str, Union[str, int]]
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Parse the path to extract attribute names and indices
    path_parts: List[path_part] = []
    current_part = ""
    in_bracket = False

    for char in path:
        if char == "[":
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
            in_bracket = True
        elif char == "]":
            if current_part:
                # Fix for mypy: Use a properly typed dictionary with Union type
                path_parts.append({"type": "index", "value": int(current_part)})
                current_part = ""
            in_bracket = False
        elif char == "." and not in_bracket:
            if current_part:
                path_parts.append({"type": "attribute", "value": current_part})
                current_part = ""
```

```
        else:
            current_part += char

    if current_part:
        path_parts.append({"type": "attribute", "value": current_part})

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        if part["type"] == "attribute":
            name_placeholder = f"#attr{i}"
            expression_attribute_names[name_placeholder] = part["value"]

            if path_expression:
                path_expression += "."
            path_expression += name_placeholder
        elif part["type"] == "index":
            path_expression += f"[{part['value']}]"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": new_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def create_list_if_not_exists(
    table_name: str, key: Dict[str, Any], list_name: str, default_values: List[Any]
) -> Dict[str, Any]:
    """
    Create a list attribute if it doesn't exist.
```

```
    This function demonstrates how to use if_not_exists to create a list attribute
    with default values if it doesn't already exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        list_name (str): The name of the list attribute.
        default_values (List[Any]): The default values for the list.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to create the list if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} = if_not_exists({list_name}, :default)",
        ExpressionAttributeValues={":default": default_values},
        ReturnValues="UPDATED_NEW",
    )

    return response


def append_to_list_safely(
    table_name: str,
    key: Dict[str, Any],
    list_name: str,
    values_to_append: List[Any],
    default_values: Optional[List[Any]] = None,
) -> Dict[str, Any]:
    """
    Append values to a list, creating it if it doesn't exist.

    This function demonstrates how to safely append values to a list attribute,
    creating the list with default values if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
```

```
        list_name (str): The name of the list attribute.
        values_to_append (List[Any]): The values to append to the list.
        default_values (Optional[List[Any]]): The default values if the list doesn't
exist.
            If not provided, values_to_append will be used as the default.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # If default_values is not provided, use values_to_append
    if default_values is None:
        default_values = values_to_append

    # Use if_not_exists with list_append to safely append to the list
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {list_name} =
list_append(if_not_exists({list_name}, :default), :values)",
        ExpressionAttributeValues={
            ":default": default_values if default_values else [],
            ":values": values_to_append,
        },
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Beispiel für die Verwendung von Listenoperationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use list operations in DynamoDB."""
    # Example parameters
    table_name = "UserData"
    key = {"UserId": "user123"}
```

```
    print("Example 1: Creating a list attribute")
    try:
        response = create_list_attribute(
            table_name=table_name,
            key=key,
            list_name="Interests",
            list_values=["Reading", "Hiking", "Photography"],
        )
        print(
            f"List attribute created successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list attribute: {e}")

    print("\nExample 2: Appending values to a list")
    try:
        response = append_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_append=["Cooking", "Gardening"],
        )
        print(
            f"Values appended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error appending to list: {e}")

    print("\nExample 3: Prepending values to a list")
    try:
        response = prepend_to_list(
            table_name=table_name,
            key=key,
            list_name="Interests",
            values_to_prepend=["Travel", "Music"],
        )
        print(
            f"Values prepended to list successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error prepending to list: {e}")
```

```
    print("\nExample 4: Updating a specific list element")
    try:
        response = update_list_element(
            table_name=table_name,
            key=key,
            list_name="Interests",
            index=2,
            new_value="Mountain Hiking",
        )
        print(
            f"List element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating list element: {e}")

    print("\nExample 5: Removing a list element")
    try:
        response = remove_list_element(
            table_name=table_name, key=key, list_name="Interests", index=0
        )
        print(
            f"List element removed successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error removing list element: {e}")

    print("\nExample 6: Working with nested lists")
    try:
        # First, create an item with a nested structure
        dynamodb = boto3.resource("dynamodb")
        table = dynamodb.Table(table_name)

        table.update_item(
            Key={"UserId": "user456"},
            UpdateExpression="SET #skills = :skills",
            ExpressionAttributeNames={"#skills": "Skills"},
            ExpressionAttributeValues={
                ":skills": [
                    {"Category": "Programming", "Languages": ["Python", "Java",
"JavaScript"]},
```

```
                    {"Category": "Database", "Systems": ["DynamoDB", "MongoDB",
"PostgreSQL"]},
                ]
            },
        )

        # Now update a nested element
        response = update_nested_list_element(
            table_name=table_name,
            key={"UserId": "user456"},
            path="Skills[0].Languages[1]",
            new_value="TypeScript",
        )
        print(
            f"Nested list element updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error working with nested lists: {e}")

    print("\nExample 7: Creating a list if it doesn't exist")
    try:
        response = create_list_if_not_exists(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Preferences",
            default_values=["Default1", "Default2", "Default3"],
        )
        print(
            f"List created with default values:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error creating list with default values: {e}")

    print("\nExample 8: Safely appending to a list")
    try:
        response = append_to_list_safely(
            table_name=table_name,
            key={"UserId": "user789"},
            list_name="Notifications",
            values_to_append=["New message received"],
            default_values=[],
        )
```

```
        print(f"Safely appended to list: {json.dumps(response.get('Attributes', {}),
default=str)}")
    except Exception as e:
        print(f"Error safely appending to list: {e}")

    print("\nKey Points About Working with Lists in DynamoDB:")
    print("1. Lists are ordered collections of elements that can be of different
types")
    print("2. Use the SET operation with direct assignment to create or replace a
list")
    print("3. Use list_append() to add elements to a list without replacing the
entire list")
    print("4. To append to the end: list_append(list_name, :values)")
    print("5. To prepend to the beginning: list_append(:values, list_name)")
    print("6. Use index notation list_name[index] to access or update specific
elements")
    print("7. Use the REMOVE action with index notation to remove specific
elements")
    print("8. Lists can contain nested structures like maps and other lists")
    print("9. Use if_not_exists() to create a list with default values if it doesn't
exist")
    print("10. List indices are zero-based (the first element is at index 0)")
    print("11. Attempting to access an index beyond the list bounds will result in
an error")
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Ausführen von Map-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Map-Operationen in DynamoDB ausgeführt werden.

- Fügen Sie verschachtelte Attribute in Map-Strukturen hinzu und aktualisieren Sie sie.
- Entfernen bestimmter Felder aus Maps
- Arbeiten mit tief verschachtelten Map-Attributen

SDK für Python (Boto3)

Demonstrieren Sie Kartenoperationen mit. AWS SDK für Python (Boto3)

```
"""
Example of updating map attributes in DynamoDB.

This module demonstrates how to update map attributes in DynamoDB, including
handling cases where the map attribute might not exist yet.
"""


import boto3
from typing import Any, Dict, Optional



def update_map_attribute_safe(
    table_name: str, key: Dict[str, Any], map_name: str, map_key: str, value: Any
) -> Dict[str, Any]:
    """
    Update a specific key in a map attribute, creating the map if it doesn't exist.

    This function demonstrates how to safely update a key within a map attribute,
    even if the map doesn't exist yet in the item.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with attribute_not_exists to safely update the map
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map.#key = :value",
        ExpressionAttributeNames={"#map": map_name, "#key": map_key},
```

```
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_nested_map(
    table_name: str, key: Dict[str, Any], path: str, value: Any
) -> Dict[str, Any]:
    """
    Add or update a value in a deeply nested map structure.

    This function demonstrates how to update a value at a specific path in a
    nested map structure, creating any intermediate maps as needed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path (str): The path to the nested attribute (e.g.,
 "user.preferences.theme").
        value (Any): The value to set at the specified path.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part
```

```
        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_map_with_if_not_exists(
    table_name: str,
    key: Dict[str, Any],
    map_name: str,
    map_key: str,
    value: Any,
    default_map: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update a key in a map, creating the map with default values if it doesn't exist.

    This function demonstrates how to use if_not_exists to initialize a map with
    default values if it doesn't exist yet, and then update a specific key.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        map_key (str): The key within the map to update.
        value (Any): The value to set for the map key.
        default_map (Optional[Dict[str, Any]]): Default map values if the map
 doesn't exist.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Set default map if not provided
    if default_map is None:
        default_map = {}

    # Create a map with the new key-value pair
    updated_map = default_map.copy()
    updated_map[map_key] = value

    # Use if_not_exists to initialize the map if it doesn't exist
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #map = if_not_exists(#map, :default_map)",
        ExpressionAttributeNames={"#map": map_name},
        ExpressionAttributeValues={":default_map": updated_map},
        ReturnValues="UPDATED_NEW",
    )

    return response


def merge_into_map(
    table_name: str, key: Dict[str, Any], map_name: str, values_to_merge: Dict[str,
 Any]
) -> Dict[str, Any]:
    """
    Merge multiple key-value pairs into a map attribute.

    This function demonstrates how to update multiple keys in a map attribute
    in a single operation, without overwriting the entire map.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        map_name (str): The name of the map attribute.
        values_to_merge (Dict[str, Any]): Key-value pairs to merge into the map.

    Returns:
```

```
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression for each key-value pair
    update_expression = "SET "
    expression_attribute_names = {"#map": map_name}
    expression_attribute_values = {}

    # Add each key-value pair to the update expression
    for i, (k, v) in enumerate(values_to_merge.items()):
        key_placeholder = f"#key{i}"
        value_placeholder = f":value{i}"

        expression_attribute_names[key_placeholder] = k
        expression_attribute_values[value_placeholder] = v

        if i > 0:
            update_expression += ", "
        update_expression += f"#map.{key_placeholder} = {value_placeholder}"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response




def example_usage():
    """Example of how to use the map attribute update functions."""
    # Example parameters
    table_name = "UserProfiles"
    key = {"UserId": "user123"}
```

```
    print("Example 1: Updating a specific key in a map attribute")
    try:
        response = update_map_attribute_safe(
            table_name=table_name, key=key, map_name="Preferences", map_key="Theme",
value="Dark"
        )
        print(f"Map attribute updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating map attribute: {e}")

    print("\nExample 2: Adding a value to a deeply nested map")
    try:
        response = add_to_nested_map(
            table_name=table_name, key=key, path="Settings.Notifications.Email",
value=True
        )
        print(f"Nested map updated successfully: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating nested map: {e}")

    print("\nExample 3: Initializing a map with default values if it doesn't exist")
    try:
        default_map = {"Language": "English", "Currency": "USD", "TimeZone": "UTC"}

        response = update_map_with_if_not_exists(
            table_name=table_name,
            key={"UserId": "newuser456"},
            map_name="Preferences",
            map_key="Theme",
            value="Light",
            default_map=default_map,
        )
        print(f"Map initialized with defaults: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error initializing map: {e}")

    print("\nExample 4: Merging multiple values into a map")
    try:
        values_to_merge = {
            "NotificationsEnabled": True,
            "EmailFrequency": "Daily",
            "PushNotifications": False,
        }
```

```
        response = merge_into_map(
            table_name=table_name,
            key=key,
            map_name="NotificationSettings",
            values_to_merge=values_to_merge,
        )
        print(f"Multiple values merged into map: {response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error merging values into map: {e}")

    print("\nBest practices for working with map attributes in DynamoDB:")
    print("1. Use dot notation to access and update nested attributes")
    print("2. Use ExpressionAttributeNames to handle reserved words and special
 characters")
    print("3. Use if_not_exists() to handle cases where attributes might not exist")
    print("4. Update specific map keys rather than overwriting the entire map")
    print("5. Use a single update operation to modify multiple map keys for better
 performance")
    print("6. Consider your data model carefully to minimize the need for deeply
 nested attributes")



if __name__ == "__main__":
    example_usage()
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Ausführen von Set-Operationen

Im folgenden Codebeispiel wird gezeigt, wie Set-Operationen in DynamoDB ausgeführt werden.

- Hinzufügen von Elementen zu einem Set-Attribut
- Entfernen von Elementen aus einem Set-Attribut
- Verwenden der Operationen ADD und DELETE mit Sätzen

SDK für Python (Boto3)

Demonstrieren Sie Satzoperationen mit. AWS SDK für Python (Boto3)

```
import boto3
from typing import Any, Dict, List


def create_set_attribute(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    set_values: List[Any],
    set_type: str = "string",
) -> Dict[str, Any]:
    """
    Create a new set attribute or add elements to an existing set.

    This function demonstrates how to use the ADD operation to create a new set
    or add elements to an existing set.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        set_values (List[Any]): The values to add to the set.
        set_type (str, optional): The type of set to create: "string", "number", or
 "binary".
            Defaults to "string".

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a DynamoDB set based on the specified type
    if set_type == "string":
        dynamo_set = set(str(value) for value in set_values)
    elif set_type == "number":
        # We need to use actual float values for the DynamoDB API
        # but mypy expects strings in sets, so we need to use type: ignore
        dynamo_set = set(float(value) for value in set_values)  # type: ignore
    else:  # binary set is not directly supported in high-level API, handled
 differently
```

```
        raise ValueError("Binary sets are not supported in this example")

    # Use the ADD operation to create or update the set
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #set_attr :set_values",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":set_values": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def add_to_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_add: List[Any]
) -> Dict[str, Any]:
    """
    Add elements to an existing set attribute.

    This function demonstrates how to use the ADD operation to add elements to an
 existing set.
    If the set doesn't exist, it will be created.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_add (List[Any]): The values to add to the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_add)

    # Use the ADD operation to add values to the set
    response = table.update_item(
        Key=key,
```

```
        UpdateExpression="ADD #set_attr :values_to_add",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_add": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )

    return response


def remove_from_set(
    table_name: str, key: Dict[str, Any], set_name: str, values_to_remove: List[Any]
) -> Dict[str, Any]:
    """
    Remove elements from a set attribute.

    This function demonstrates how to use the DELETE operation to remove elements
 from a set.
    If the last element is removed, the attribute will be deleted entirely.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        values_to_remove (List[Any]): The values to remove from the set.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Convert the list to a set (assuming string set for simplicity)
    dynamo_set = set(str(value) for value in values_to_remove)

    # Use the DELETE operation to remove values from the set
    response = table.update_item(
        Key=key,
        UpdateExpression="DELETE #set_attr :values_to_remove",
        ExpressionAttributeNames={"#set_attr": set_name},
        ExpressionAttributeValues={":values_to_remove": dynamo_set},
        ReturnValues="UPDATED_NEW",
    )
```

```
    return response


def check_if_set_exists(table_name: str, key: Dict[str, Any], set_name: str) ->
 bool:
    """
    Check if a set attribute exists in an item.

    This function demonstrates how to check if a set attribute exists after
    potentially removing all elements from it.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to check.
        set_name (str): The name of the set attribute.

    Returns:
        bool: True if the set attribute exists, False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Get the item
    response = table.get_item(
        Key=key, ProjectionExpression="#set_attr",
 ExpressionAttributeNames={"#set_attr": set_name}
    )

    # Check if the item exists and has the set attribute
    return "Item" in response and set_name in response["Item"]


def demonstrate_last_element_removal(
    table_name: str, key: Dict[str, Any], set_name: str
) -> Dict[str, Any]:
    """
    Demonstrate what happens when you remove the last element from a set.

    This function creates a set with a single element, then removes that element,
    showing that the attribute is completely removed when the last element is
 deleted.
```

```
    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Step 1: Create a set with a single element
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=["last_element"],
        set_type="string",
    )

    # Step 2: Check that the set exists
    exists_before = check_if_set_exists(table_name, key, set_name)

    # Step 3: Remove the last element
    delete_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
 values_to_remove=["last_element"]
    )

    # Step 4: Check if the set still exists
    exists_after = check_if_set_exists(table_name, key, set_name)

    # Return the results
    return {
        "create_response": create_response,
        "exists_before": exists_before,
        "delete_response": delete_response,
        "exists_after": exists_after,
    }


def work_with_number_set(
    table_name: str,
    key: Dict[str, Any],
    set_name: str,
    initial_values: List[float],
    values_to_add: List[float],
```

```
    values_to_remove: List[float],
) -> Dict[str, Any]:
    """
    Demonstrate working with a number set in DynamoDB.

    This function shows how to create and manipulate a set of numbers.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        set_name (str): The name of the set attribute.
        initial_values (List[float]): The initial values for the set.
        values_to_add (List[float]): Values to add to the set.
        values_to_remove (List[float]): Values to remove from the set.

    Returns:
        Dict[str, Any]: A dictionary containing the responses from each operation.
    """
    # Step 1: Create the number set
    create_response = create_set_attribute(
        table_name=table_name,
        key=key,
        set_name=set_name,
        set_values=initial_values,
        set_type="number",
    )

    # Step 2: Add more numbers to the set
    add_response = add_to_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_add=values_to_add
    )

    # Step 3: Remove some numbers from the set
    remove_response = remove_from_set(
        table_name=table_name, key=key, set_name=set_name,
values_to_remove=values_to_remove
    )

    # Step 4: Get the final state
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    get_response = table.get_item(
```

```
        Key=key,
        ProjectionExpression=f"#{set_name}",
        ExpressionAttributeNames={f"#{set_name}": set_name},
    )

    # Return all responses
    return {
        "create_response": create_response,
        "add_response": add_response,
        "remove_response": remove_response,
        "final_state": get_response.get("Item", {}),
    }
```

Beispiel für die Verwendung von Set-Operationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the set operations functions."""
    # Example parameters
    table_name = "UserPreferences"
    key = {"UserId": "user123"}

    print("Example 1: Creating a string set attribute")
    try:
        response = create_set_attribute(
            table_name=table_name,
            key=key,
            set_name="FavoriteTags",
            set_values=["AWS", "DynamoDB", "NoSQL"],
            set_type="string",
        )
        print(f"Set attribute created successfully: {response.get('Attributes',
 {})}")
    except Exception as e:
        print(f"Error creating set attribute: {e}")

    print("\nExample 2: Adding elements to an existing set")
    try:
        response = add_to_set(
            table_name=table_name,
            key=key,
```

```
            set_name="FavoriteTags",
            values_to_add=["Database", "Serverless"],
        )
        print(f"Elements added to set successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error adding to set: {e}")

    print("\nExample 3: Removing elements from a set")
    try:
        response = remove_from_set(
            table_name=table_name, key=key, set_name="FavoriteTags",
values_to_remove=["NoSQL"]
        )
        print(f"Elements removed from set successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error removing from set: {e}")

    print("\nExample 4: Demonstrating what happens when you remove the last element
from a set")
    try:
        results = demonstrate_last_element_removal(
            table_name=table_name, key={"UserId": "tempUser"},
set_name="SingleElementSet"
        )

        print(f"Set exists before removal: {results['exists_before']}")
        print(f"Set exists after removal: {results['exists_after']}")

        if not results["exists_after"]:
            print("The set attribute was completely removed when the last element
was deleted.")
        else:
            print("The set attribute still exists after removing the last element.")
    except Exception as e:
        print(f"Error in last element removal demonstration: {e}")

    print("\nExample 5: Working with a number set")
    try:
        results = work_with_number_set(
            table_name=table_name,
            key={"UserId": "user456"},
            set_name="LuckyNumbers",
```

```
            initial_values=[7, 13, 42],
            values_to_add=[99, 100],
            values_to_remove=[13],
        )

        print(f"Initial number set: {results['create_response'].get('Attributes',
{})}")
        print(f"After adding numbers: {results['add_response'].get('Attributes',
{})}")
        print(f"After removing numbers:
{results['remove_response'].get('Attributes', {})}")
        print(f"Final state: {results['final_state']}")
    except Exception as e:
        print(f"Error working with number set: {e}")

    print("\nKey Points About DynamoDB Sets:")
    print("1. Sets can only contain elements of the same type (string, number, or
binary)")
    print("2. Sets automatically eliminate duplicate values")
    print("3. The ADD operation creates a set if it doesn't exist")
    print("4. The DELETE operation removes specified elements from a set")
    print("5. When the last element is removed from a set, the entire attribute is
deleted")
    print("6. Empty sets are not allowed in DynamoDB")
    print("7. Sets are unordered collections")
    print("8. The ADD operation is atomic for sets")
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Stapel von PartiQL-Anweisungen ausführen kann.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.
```

```
        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated with
                           each statement. This list must be in the same order as
the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[
                    {"Statement": statement, "Parameters": params}
                    for statement, params in zip(statements, param_list)
                ]
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table
"
                    "does not exist."
                )
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s:
%s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen in Stapeln ausführt.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
```

```
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    movie_data = [
        {
            "title": f"House PartiQL",
            "year": datetime.now().year - 5,
            "info": {
                "plot": "Wacky high jinks result from querying a mysterious
database.",
                "rating": Decimal("8.5"),
            },
        },
        {
            "title": f"House PartiQL 2",
            "year": datetime.now().year - 3,
            "info": {
                "plot": "Moderate high jinks result from querying another mysterious
database.",
                "rating": Decimal("6.5"),
            },
        },
        {
            "title": f"House PartiQL 3",
            "year": datetime.now().year - 1,
            "info": {
                "plot": "Tepid high jinks result from querying yet another
mysterious database.",
                "rating": Decimal("2.5"),
            },
        },
    ]

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
        f'INSERT INTO "{table_name}" ' f"VALUE {{'title': ?, 'year': ?, 'info': ?}}"
    ] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)
```

```
    print(f"Getting data for a batch of movies.")
    statements = [f'SELECT * FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    output = wrapper.run_partiql(statements, params)
    for item in output["Responses"]:
        print(f"\n{item['Item']['title']}, {item['Item']['year']}")
        pprint(item["Item"])
    print("-" * 88)

    ratings = [Decimal("7.7"), Decimal("5.5"), Decimal("1.3")]
    print(f"Updating a batch of movies with new ratings.")
    statements = [
        f'UPDATE "{table_name}" SET info.rating=? ' f"WHERE title=? AND year=?"
    ] * len(movie_data)
    params = [
        [rating, movie["title"], movie["year"]]
        for rating, movie in zip(ratings, movie_data)
    ]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Getting projected data from the table to verify our update.")
    output = wrapper.dyn_resource.meta.client.execute_statement(
        Statement=f'SELECT title, info.rating FROM "{table_name}"'
    )
    pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting a batch of movies from the table.")
    statements = [f'DELETE FROM "{table_name}" WHERE title=? AND year=?'] * len(
        movie_data
    )
    params = [[movie["title"], movie["year"]] for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)
```

```
    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- API-Details finden Sie BatchExecuteStatement in der AWS API-Referenz zum SDK für Python (Boto3).

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die PartiQL-Anweisungen ausführen kann.

```
from datetime import datetime
from decimal import Decimal
import logging
```

```
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """

    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource


    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
 to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                       statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not
 exist.",
```

```
                    statement,
                )
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return output
```

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print("-" * 88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print("-" * 88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't stop
 using."
    rating = Decimal("9.9")

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"{table_name}\" VALUE {{'title': ?, 'year': ?, 'info': ?}}",
        [title, year, {"plot": plot, "rating": rating}],
    )
    print("Success!")
    print("-" * 88)
```

```
    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    rating = Decimal("2.4")
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f'UPDATE "{table_name}" SET info.rating=? WHERE title=? AND year=?',
        [rating, title, year],
    )
    print("Success!")
    print("-" * 88)

    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f'SELECT * FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    for item in output["Items"]:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output["Items"])
    print("-" * 88)

    print(f"Deleting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f'DELETE FROM "{table_name}" WHERE title=? AND year=?', [title, year]
    )
    print("Success!")
    print("-" * 88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print("-" * 88)

    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
```

```
        dyn_res = boto3.resource("dynamodb")
        scaffold = Scaffold(dyn_res)
        movies = PartiQLWrapper(dyn_res)
        run_scenario(scaffold, movies, "doc-example-table-partiql-movies")
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- API-Details finden Sie ExecuteStatement in der AWS API-Referenz zum SDK für Python (Boto3).

Abfragen einer Tabelle mit einem globalen sekundären Index

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem globalen sekundären Index abgefragt wird.

- Abfragen einer DynamoDB-Tabelle mit ihrem Primärschlüssel
- Abfragen eines globalen sekundären Index (GSI) für alternative Zugriffsmuster
- Vergleichen von Tabellenabfragen und GSI-Abfragen

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit ihrem Primärschlüssel und einem Global Secondary Index (GSI) mit ab. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key


def query_table(table_name, partition_key_name, partition_key_value):
    """
    Query a DynamoDB table using its primary key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
```

```
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the table's primary key
    response =
 table.query(KeyConditionExpression=Key(partition_key_name).eq(partition_key_value))

    return response


def query_gsi(table_name, index_name, partition_key_name, partition_key_value):
    """
    Query a Global Secondary Index (GSI) on a DynamoDB table.

    Args:
        table_name (str): The name of the DynamoDB table.
        index_name (str): The name of the Global Secondary Index.
        partition_key_name (str): The name of the GSI's partition key attribute.
        partition_key_value (str): The value of the GSI's partition key to query.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query on the GSI
    response = table.query(
        IndexName=index_name,
 KeyConditionExpression=Key(partition_key_name).eq(partition_key_value)
    )

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mithilfe einer „begins_with“-Bedingung

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einer „begins_with“-Bedingung abgefragt wird.

- Verwenden der Funktion „begins_with“ in einem Schlüsselbedingungsausdruck
- Filtern der Elemente auf Grundlage eines Präfixmusters für den Sortierschlüssel

SDK für Python (Boto3)

Fragen Sie mit AWS SDK für Python (Boto3) eine DynamoDB-Tabelle mithilfe einer „begins_with“-Bedingung im Sortierschlüssel ab.

```
import boto3
from boto3.dynamodb.conditions import Key


def query_with_begins_with(
    table_name, partition_key_name, partition_key_value, sort_key_name, prefix
):
    """
    Query a DynamoDB table with a begins_with condition on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute.
        prefix (str): The prefix to match at the beginning of the sort key.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Perform the query with a begins_with condition on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) & Key(
        sort_key_name
    ).begins_with(prefix)
    response = table.query(KeyConditionExpression=key_condition)
```

```
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit einem Datumsbereich

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mithilfe eines Datumsbereichs im Sortierschlüssel abgefragt wird.

- Abfragen von Elementen innerhalb eines bestimmten Datumsbereichs
- Verwenden von Vergleichsoperatoren für Sortierschlüssel im Datumsformat

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle nach Elementen innerhalb eines Datumsbereichs mit ab. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
```

```
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year, month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
```

```
        sort_key_name (str): The name of the sort key attribute (containing date
values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit einem komplexen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem komplexen Filterausdruck abgefragt wird.

- Anwenden von komplexen Filterausdrücken auf Abfrageergebnisse
- Kombinieren von mehreren Bedingungen über logische Operatoren
- Filtern von Elementen auf der Grundlage von Nicht-Schlüsselattributen.

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem komplexen Filterausdruck ab mit. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_complex_filter(
    table_name,
    partition_key_name,
    partition_key_value,
    min_rating=None,
    status_list=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        min_rating (float, optional): Minimum rating value for filtering.
        status_list (list, optional): List of status values to include.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize the filter expression and expression attribute values
    filter_expression = None
    expression_attribute_values = {}

    # Build the filter expression based on provided parameters
    if min_rating is not None:
        filter_expression = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if status_list and len(status_list) > 0:
        status_condition = None
        for i, status in enumerate(status_list):
            status_value_name = f":status{i}"
            expression_attribute_values[status_value_name] = status

            if status_condition is None:
                status_condition = Attr("status").eq(status)
            else:
                status_condition = status_condition | Attr("status").eq(status)

        if filter_expression is None:
            filter_expression = status_condition
        else:
            filter_expression = filter_expression & status_condition

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if filter_expression is None:
            filter_expression = price_condition
        else:
            filter_expression = filter_expression & price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
```

```
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response


def query_with_complex_filter_and_or(
    table_name,
    partition_key_name,
    partition_key_value,
    category=None,
    min_rating=None,
    max_price=None,
):
    """
    Query a DynamoDB table with a complex filter expression using AND and OR
 operators.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        category (str, optional): Category value for filtering.
        min_rating (float, optional): Minimum rating value for filtering.
        max_price (float, optional): Maximum price value for filtering.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build a complex filter expression with AND and OR operators
    filter_expression = None
    expression_attribute_values = {}

    # Build the category condition
    if category:
```

```
        filter_expression = Attr("category").eq(category)
        expression_attribute_values[":category"] = category

    # Build the rating and price condition (rating >= min_rating OR price <=
max_price)
    rating_price_condition = None

    if min_rating is not None:
        rating_price_condition = Attr("rating").gte(min_rating)
        expression_attribute_values[":min_rating"] = min_rating

    if max_price is not None:
        price_condition = Attr("price").lte(max_price)
        expression_attribute_values[":max_price"] = max_price

        if rating_price_condition is None:
            rating_price_condition = price_condition
        else:
            rating_price_condition = rating_price_condition | price_condition

    # Combine the conditions
    if rating_price_condition:
        if filter_expression is None:
            filter_expression = rating_price_condition
        else:
            filter_expression = filter_expression & rating_price_condition

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression
        if expression_attribute_values:
            query_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the query
    response = table.query(**query_params)
    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit einem dynamischen Filterausdruck

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem dynamischen Filterausdruck abgefragt wird.

- Erstellen von Filterausdrücken dynamisch zur Laufzeit
- Konstruieren von Filterbedingungen auf der Grundlage von Benutzereingaben oder Anwendungsstatus
- Hinzufügen oder Entfernen von Filterkriterien unter bestimmten Bedingungen

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem dynamisch erstellten Filterausdruck ab. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_dynamic_filter(
    table_name, partition_key_name, partition_key_value, filter_conditions=None
):
    """
    Query a DynamoDB table with a dynamically constructed filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_conditions (dict, optional): A dictionary of filter conditions where
            keys are attribute names and values are dictionaries with 'operator' and
 'value'.
            Example: {'rating': {'operator': '>=', 'value': 4}, 'status':
 {'operator': '=', 'value': 'active'}}

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Start with the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Initialize variables for the filter expression and attribute values
    filter_expression = None
    expression_attribute_values = {":pk_val": partition_key_value}

    # Dynamically build the filter expression if filter conditions are provided
    if filter_conditions:
        for attr_name, condition in filter_conditions.items():
            operator = condition.get("operator")
            value = condition.get("value")
            attr_value_name = f":{attr_name}"
            expression_attribute_values[attr_value_name] = value

            # Create the appropriate filter expression based on the operator
            current_condition = None
            if operator == "=":
                current_condition = Attr(attr_name).eq(value)
            elif operator == "!=":
                current_condition = Attr(attr_name).ne(value)
            elif operator == ">":
                current_condition = Attr(attr_name).gt(value)
            elif operator == ">=":
                current_condition = Attr(attr_name).gte(value)
            elif operator == "<":
                current_condition = Attr(attr_name).lt(value)
            elif operator == "<=":
                current_condition = Attr(attr_name).lte(value)
            elif operator == "contains":
                current_condition = Attr(attr_name).contains(value)
            elif operator == "begins_with":
                current_condition = Attr(attr_name).begins_with(value)

            # Combine with existing filter expression using AND
            if current_condition:
                if filter_expression is None:
                    filter_expression = current_condition
                else:
                    filter_expression = filter_expression & current_condition
```

```
    # Perform the query with the dynamically built filter expression
    query_params = {"KeyConditionExpression": key_condition}

    if filter_expression:
        query_params["FilterExpression"] = filter_expression

    response = table.query(**query_params)
    return response
```

Veranschaulicht, wie dynamische Filterausdrücke mit verwendet werden. AWS SDK für Python (Boto3)

```
def example_usage():
    """Example of how to use the query_with_dynamic_filter function."""
    # Example parameters
    table_name = "Products"
    partition_key_name = "Category"
    partition_key_value = "Electronics"

    # Define dynamic filter conditions based on user input or runtime conditions
    user_min_rating = 4  # This could come from user input
    user_status_filter = "active"  # This could come from user input

    filter_conditions = {}

    # Only add conditions that are actually specified
    if user_min_rating is not None:
        filter_conditions["rating"] = {"operator": ">=", "value": user_min_rating}

    if user_status_filter:
        filter_conditions["status"] = {"operator": "=", "value": user_status_filter}

    print(
        f"Querying products in category '{partition_key_value}' with filter
 conditions: {filter_conditions}"
    )

    # Execute the query with dynamic filter
    response = query_with_dynamic_filter(
```

```
        table_name, partition_key_name, partition_key_value, filter_conditions
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Product: {item}")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit einem Filterausdruck und einem Limit

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit einem Filterausdruck und Limit abgefragt wird.

- Anwenden von Filterausdrücken auf Abfrageergebnisse mit einem Limit für die ausgewerteten Elemente
- Verstehen, wie sich Limits auf gefilterte Abfrageergebnisse auswirken
- Steuern der maximalen Anzahl der in einer Abfrage verarbeiteten Elemente

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit einem Filterausdruck ab und beschränken Sie die Verwendung. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_filter_and_limit(
    table_name,
    partition_key_name,
    partition_key_value,
    filter_attribute=None,
    filter_value=None,
```

```
    limit=10,
):
    """
    Query a DynamoDB table with a filter expression and limit the number of results.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute (str, optional): The attribute name to filter on.
        filter_value (any, optional): The value to compare against in the filter.
        limit (int, optional): The maximum number of items to evaluate. Defaults to
 10.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Prepare the query parameters
    query_params = {"KeyConditionExpression": key_condition, "Limit": limit}

    # Add the filter expression if filter attributes are provided
    if filter_attribute and filter_value is not None:
        query_params["FilterExpression"] = Attr(filter_attribute).gt(filter_value)
        query_params["ExpressionAttributeValues"] = {":filter_value": filter_value}

    # Execute the query
    response = table.query(**query_params)
    return response
```

Veranschaulicht, wie Filterausdrücke mit Grenzwerten verwendet werden. AWS SDK für Python (Boto3)

```
def example_usage():
```

```
    """Example of how to use the query_with_filter_and_limit function."""
    # Example parameters
    table_name = "ProductReviews"
    partition_key_name = "ProductId"
    partition_key_value = "P123456"
    filter_attribute = "Rating"
    filter_value = 3  # Filter for ratings > 3
    limit = 5

    print(f"Querying reviews for product '{partition_key_value}' with rating >
{filter_value}")
    print(f"Limiting to {limit} evaluated items")

    # Execute the query with filter and limit
    response = query_with_filter_and_limit(
        table_name, partition_key_name, partition_key_value, filter_attribute,
filter_value, limit
    )

    # Process the results
    items = response.get("Items", [])
    print(f"\nReturned {len(items)} items that passed the filter")

    for item in items:
        print(f"Review: {item}")

    # Explain the difference between Limit and actual results
    explain_limit_vs_results(response)

    # Check if there are more results
    if "LastEvaluatedKey" in response:
        print("\nThere are more results available. Use the LastEvaluatedKey for
pagination.")
    else:
        print("\nAll matching results have been retrieved.")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit verschachtelten Attributen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit verschachtelten Attributen abgefragt wird.

- Zugreifen und Filtern nach verschachtelten Attributen in DynamoDB-Elementen
- Verwenden von Dokumentpfadausdrücken, um auf verschachtelte Elemente zu verweisen

SDK für Python (Boto3)

Abfragen einer DynamoDB-Tabelle mit verschachtelten Attributen mithilfe von. AWS SDK für Python (Boto3)

```
from typing import Any, Dict, List

import boto3
from boto3.dynamodb.conditions import Attr, Key


def query_with_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_path: str,
    comparison_operator: str,
    comparison_value: Any,
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by nested attributes.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_path (str): The path to the nested attribute (e.g., 'specs.weight').
        comparison_operator (str): The comparison operator to use ('=', '!=', '<',
 '<=', '>', '>=').
        comparison_value (any): The value to compare against.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the filter expression based on the nested attribute path and comparison
 operator
    filter_expression = None
    if comparison_operator == "=":
        filter_expression = Attr(nested_path).eq(comparison_value)
    elif comparison_operator == "!=":
        filter_expression = Attr(nested_path).ne(comparison_value)
    elif comparison_operator == "<":
        filter_expression = Attr(nested_path).lt(comparison_value)
    elif comparison_operator == "<=":
        filter_expression = Attr(nested_path).lte(comparison_value)
    elif comparison_operator == ">":
        filter_expression = Attr(nested_path).gt(comparison_value)
    elif comparison_operator == ">=":
        filter_expression = Attr(nested_path).gte(comparison_value)
    elif comparison_operator == "contains":
        filter_expression = Attr(nested_path).contains(comparison_value)
    elif comparison_operator == "begins_with":
        filter_expression = Attr(nested_path).begins_with(comparison_value)

    # Execute the query with the filter expression
    response = table.query(KeyConditionExpression=key_condition,
 FilterExpression=filter_expression)

    return response


def query_with_multiple_nested_attributes(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    nested_conditions: List[Dict[str, Any]],
) -> Dict[str, Any]:
    """
    Query a DynamoDB table and filter by multiple nested attributes.

    Args:
```

```
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        nested_conditions (list): A list of dictionaries, each containing:
            - path (str): The path to the nested attribute
            - operator (str): The comparison operator
            - value (any): The value to compare against

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    # Build the combined filter expression for all nested attributes
    combined_filter = None

    for condition in nested_conditions:
        if not isinstance(condition, dict):
            continue
        path = condition.get("path", "")
        operator = condition.get("operator", "")
        value = condition.get("value")

        if not path or not operator:
            continue

        # Build the individual filter expression
        current_filter = None
        if operator == "=":
            current_filter = Attr(path).eq(value)
        elif operator == "!=":
            current_filter = Attr(path).ne(value)
        elif operator == "<":
            current_filter = Attr(path).lt(value)
        elif operator == "<=":
            current_filter = Attr(path).lte(value)
        elif operator == ">":
            current_filter = Attr(path).gt(value)
        elif operator == ">=":
```

```
            current_filter = Attr(path).gte(value)
        elif operator == "contains":
            current_filter = Attr(path).contains(value)
        elif operator == "begins_with":
            current_filter = Attr(path).begins_with(value)

        # Combine with the existing filter using AND
        if current_filter:
            if combined_filter is None:
                combined_filter = current_filter
            else:
                combined_filter = combined_filter & current_filter

    # Execute the query with the combined filter expression
    response = table.query(KeyConditionExpression=key_condition,
 FilterExpression=combined_filter)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit Paginierung

Das folgende Codebeispiel zeigt, wie eine Tabelle mit Paginierung abgefragt wird.

- Implementieren der Paginierung für DynamoDB-Abfrageergebnisse
- Verwenden Sie den LastEvaluatedKey , um nachfolgende Seiten abzurufen.
- Steuern der Anzahl der Elemente pro Seite mit dem Parameter „Limit“

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit Paginierung unter Verwendung von ab. AWS SDK für Python (Boto3)

```
import boto3
from boto3.dynamodb.conditions import Key
```

```
def query_with_pagination(
    table_name, partition_key_name, partition_key_value, page_size=25,
 max_pages=None
):
    """
    Query a DynamoDB table with pagination to handle large result sets.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.
        max_pages (int, optional): The maximum number of pages to retrieve. If None,
 retrieves all pages.

    Returns:
        list: All items retrieved from the query across all pages.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_count = 0
    all_items = []

    # Paginate through the results
    while True:
        # Check if we've reached the maximum number of pages
        if max_pages is not None and page_count >= max_pages:
            break

        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
 Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
 query
```

```
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Process the current page of results
        items = response.get("Items", [])
        all_items.extend(items)

        # Update pagination tracking
        page_count += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if not last_evaluated_key:
            break

    return all_items


def query_with_pagination_generator(
    table_name, partition_key_name, partition_key_value, page_size=25
):
    """
    Query a DynamoDB table with pagination using a generator to handle large result
 sets.
    This approach is memory-efficient as it yields one page at a time.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        page_size (int, optional): The number of items to return per page. Defaults
 to 25.

    Yields:
        tuple: A tuple containing (items, page_number, last_page) where:
            - items is a list of items for the current page
            - page_number is the current page number (starting from 1)
            - last_page is a boolean indicating if this is the last page
    """
```

```
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Initialize variables for pagination
    last_evaluated_key = None
    page_number = 0

    # Paginate through the results
    while True:
        # Prepare the query parameters
        query_params = {
            "KeyConditionExpression":
Key(partition_key_name).eq(partition_key_value),
            "Limit": page_size,
        }

        # Add the ExclusiveStartKey if we have a LastEvaluatedKey from a previous
query
        if last_evaluated_key:
            query_params["ExclusiveStartKey"] = last_evaluated_key

        # Execute the query
        response = table.query(**query_params)

        # Get the current page of results
        items = response.get("Items", [])
        page_number += 1

        # Get the LastEvaluatedKey for the next page, if any
        last_evaluated_key = response.get("LastEvaluatedKey")

        # Determine if this is the last page
        is_last_page = last_evaluated_key is None

        # Yield the current page of results
        yield (items, page_number, is_last_page)

        # If there's no LastEvaluatedKey, we've reached the end of the results
        if is_last_page:
            break
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen einer Tabelle mit strikt konsistenten Lesevorgängen

Im folgenden Codebeispiel wird gezeigt, wie eine Tabelle mit strikt konsistenten Lesevorgängen abgefragt wird.

- Konfigurieren des Konsistenzniveaus für DynamoDB-Abfragen
- Verwenden der strikt konsistenten Lesevorgänge, um die aktuellen Daten zu erhalten
- Erkennen der Kompromisse zwischen letztendlicher Konsistenz und starker Konsistenz

SDK für Python (Boto3)

Fragen Sie eine DynamoDB-Tabelle mit der Option für stark konsistente Lesevorgänge mit. AWS SDK für Python (Boto3)

```
import time

import boto3
from boto3.dynamodb.conditions import Key


def query_with_consistent_read(
    table_name,
    partition_key_name,
    partition_key_value,
    sort_key_name=None,
    sort_key_value=None,
    consistent_read=True,
):
    """
    Query a DynamoDB table with the option for strongly consistent reads.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str, optional): The name of the sort key attribute.
```

```
        sort_key_value (str, optional): The value of the sort key to query.
        consistent_read (bool, optional): Whether to use strongly consistent reads.
Defaults to True.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the key condition expression
    key_condition = Key(partition_key_name).eq(partition_key_value)

    if sort_key_name and sort_key_value:
        key_condition = key_condition & Key(sort_key_name).eq(sort_key_value)

    # Perform the query with the consistent read option
    response = table.query(KeyConditionExpression=key_condition,
ConsistentRead=consistent_read)

    return response
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen nach TTL-Elementen

Im folgenden Codebeispiel wird gezeigt, wie TTL-Elemente abgefragt werden.

SDK für Python (Boto3)

Abfragen eines gefilterten Ausdrucks zum Sammeln von TTL-Elementen in einer DynamoDB-Tabelle mithilfe von. AWS SDK für Python (Boto3)

```
from datetime import datetime

import boto3
```

```
def query_dynamodb_items(table_name, partition_key):
    """

    :param table_name: Name of the DynamoDB table
    :param partition_key:
    :return:
    """
    try:
        # Initialize a DynamoDB resource
        dynamodb = boto3.resource("dynamodb", region_name="us-east-1")

        # Specify your table
        table = dynamodb.Table(table_name)

        # Get the current time in epoch format
        current_time = int(datetime.now().timestamp())

        # Perform the query operation with a filter expression to exclude expired
 items
        # response = table.query(
        #
 KeyConditionExpression=boto3.dynamodb.conditions.Key('partitionKey').eq(partition_key),
        #
 FilterExpression=boto3.dynamodb.conditions.Attr('expireAt').gt(current_time)
        # )
        response = table.query(

 KeyConditionExpression=dynamodb.conditions.Key("partitionKey").eq(partition_key),
            FilterExpression=dynamodb.conditions.Attr("expireAt").gt(current_time),
        )

        # Print the items that are not expired
        for item in response["Items"]:
            print(item)

    except Exception as e:
        print(f"Error querying items: {e}")


# Call the function with your values
query_dynamodb_items("Music", "your-partition-key-value")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Abfragen von Tabellen mithilfe von Datums- und Uhrzeitmustern

Im folgenden Codebeispiel wird gezeigt, wie Datums- und Uhrzeitmuster zum Abfragen von Tabellen verwendet werden.

- Speichern und Abfragen von date/time Werten in DynamoDB.
- Implementieren von Datumsbereichsabfragen mithilfe von Sortierschlüsseln
- Formatieren von Datumszeichenfolgen für effektive Abfragen

SDK für Python (Boto3)

Abfrage unter Verwendung von Datumsbereichen in Sortierschlüsseln mit. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_date_range(
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        start_date (datetime): The start date for the query range.
        end_date (datetime): The end date for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
```

```
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date values as ISO 8601 strings
    # DynamoDB works well with ISO format for date values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key using BETWEEN operator
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response


def query_with_date_range_by_month(
    table_name, partition_key_name, partition_key_value, sort_key_name, year, month
):
    """
    Query a DynamoDB table for a specific month's data.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date
 values).
        year (int): The year to query.
        month (int): The month to query (1-12).

    Returns:
```

```
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Calculate the start and end dates for the specified month
    if month == 12:
        next_year = year + 1
        next_month = 1
    else:
        next_year = year
        next_month = month + 1

    start_date = datetime(year, month, 1)
    end_date = datetime(next_year, next_month, 1) - timedelta(microseconds=1)

    # Format the date values as ISO 8601 strings
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(KeyConditionExpression=key_condition)

    return response
```

Abfrage mit Datums-/Uhrzeitvariablen mit. AWS SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3
from boto3.dynamodb.conditions import Key


def query_with_datetime(
```

```
    table_name, partition_key_name, partition_key_value, sort_key_name, start_date,
 end_date
):
    """
    Query a DynamoDB table with a date range filter on the sort key.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        sort_key_name (str): The name of the sort key attribute (containing date/
time values).
        start_date (datetime): The start date/time for the query range.
        end_date (datetime): The end date/time for the query range.

    Returns:
        dict: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Format the date/time values as ISO 8601 strings
    # DynamoDB works well with ISO format for date/time values
    start_date_str = start_date.isoformat()
    end_date_str = end_date.isoformat()

    # Perform the query with a date range on the sort key
    key_condition = Key(partition_key_name).eq(partition_key_value) &
 Key(sort_key_name).between(
        start_date_str, end_date_str
    )

    response = table.query(
        KeyConditionExpression=key_condition,
        ExpressionAttributeValues={
            ":pk_val": partition_key_value,
            ":start_date": start_date_str,
            ":end_date": end_date_str,
        },
    )

    return response
```

```
def example_usage():
    """Example of how to use the query_with_datetime function."""
    # Example parameters
    table_name = "Events"
    partition_key_name = "EventType"
    partition_key_value = "UserLogin"
    sort_key_name = "Timestamp"

    # Create date/time variables for the query
    end_date = datetime.now()
    start_date = end_date - timedelta(days=7)  # Query events from the last 7 days

    print(f"Querying events from {start_date.isoformat()} to
 {end_date.isoformat()}")

    # Execute the query
    response = query_with_datetime(
        table_name, partition_key_name, partition_key_value, sort_key_name,
 start_date, end_date
    )

    # Process the results
    items = response.get("Items", [])
    print(f"Found {len(items)} items")

    for item in items:
        print(f"Event: {item}")
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Python (Boto3).

Verstehen der Reihenfolge der Aktualisierungsausdrücke

Im folgenden Codebeispiel wird die Reihenfolge der Aktualisierungsausdrücke erklärt.

- Erfahren, wie DynamoDB Aktualisierungsausdrücke verarbeitet
- Verstehen der Reihenfolge der Operationen in Aktualisierungsausdrücken

- Vermeiden von unerwarteten Ergebnissen durch Verständnis der Ausdrucksauswertung

SDK für Python (Boto3)

Demonstrieren Sie das Aktualisieren der Ausdrucksreihenfolge mit. AWS SDK für Python (Boto3)

```
import boto3
import json
from typing import Any, Dict, Optional


def update_with_multiple_actions(
    table_name: str,
    key: Dict[str, Any],
    update_expression: str,
    expression_attribute_names: Optional[Dict[str, str]] = None,
    expression_attribute_values: Optional[Dict[str, Any]] = None,
) -> Dict[str, Any]:
    """
    Update an item with multiple actions in a single update expression.

    This function demonstrates how to use multiple actions in a single update
 expression
    and how DynamoDB processes these actions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        update_expression (str): The update expression with multiple actions.
        expression_attribute_names (Optional[Dict[str, str]]): Expression attribute
 name placeholders.
        expression_attribute_values (Optional[Dict[str, Any]]): Expression attribute
 value placeholders.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Prepare the update parameters
```

```
    update_params = {
        "Key": key,
        "UpdateExpression": update_expression,
        "ReturnValues": "UPDATED_NEW",
    }

    # Add expression attribute names if provided
    if expression_attribute_names:
        update_params["ExpressionAttributeNames"] = expression_attribute_names

    # Add expression attribute values if provided
    if expression_attribute_values:
        update_params["ExpressionAttributeValues"] = expression_attribute_values

    # Execute the update
    response = table.update_item(**update_params)

    return response


def demonstrate_value_copying(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate that variables hold copies of existing values before modifications.

    This function creates an item with initial values, then updates it with an
 expression
    that uses the values of attributes before they are modified in the same
 expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update({"a": 1, "b": 2, "c": 3})
```

```
    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with an expression that uses values before they are
 modified
    # This expression removes 'a', then sets 'b' to the value of 'a', and 'c' to the
 value of 'b'
    update_response = table.update_item(
        Key=key, UpdateExpression="REMOVE a SET b = a, c = b",
 ReturnValues="UPDATED_NEW"
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
        "final_state": item_after,
    }


def demonstrate_action_order(table_name: str, key: Dict[str, Any]) -> Dict[str,
 Any]:
    """
    Demonstrate the order in which different action types are processed.

    This function creates an item with initial values, then updates it with an
 expression
    that includes multiple action types (SET, REMOVE, ADD, DELETE) to show the order
    in which they are processed.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to create and update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
```

```
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Step 1: Create an item with initial values
    initial_item = key.copy()
    initial_item.update(
        {
            "counter": 10,
            "set_attr": set(["A", "B", "C"]),
            "to_remove": "This will be removed",
            "to_modify": "Original value",
        }
    )

    table.put_item(Item=initial_item)

    # Step 2: Get the item to verify initial state
    response_before = table.get_item(Key=key)
    item_before = response_before.get("Item", {})

    # Step 3: Update the item with multiple action types
    # The actions will be processed in this order: REMOVE, SET, ADD, DELETE
    update_response = table.update_item(
        Key=key,
        UpdateExpression="REMOVE to_remove SET to_modify = :new_value ADD
counter :increment DELETE set_attr :elements",
        ExpressionAttributeValues={
            ":new_value": "Updated value",
            ":increment": 5,
            ":elements": set(["B"]),
        },
        ReturnValues="UPDATED_NEW",
    )

    # Step 4: Get the item to verify final state
    response_after = table.get_item(Key=key)
    item_after = response_after.get("Item", {})

    # Return the results
    return {
        "initial_state": item_before,
        "update_response": update_response,
```

```
        "final_state": item_after,
    }


def update_with_multiple_set_actions(
    table_name: str, key: Dict[str, Any], attributes: Dict[str, Any]
) -> Dict[str, Any]:
    """
    Update multiple attributes with a single SET action.

    This function demonstrates how to update multiple attributes in a single SET
 action,
    which is more efficient than using multiple separate update operations.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attributes (Dict[str, Any]): The attributes to update and their new values.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Build the update expression and expression attribute values
    update_expression = "SET "
    expression_attribute_values = {}

    # Add each attribute to the update expression
    for i, (attr_name, attr_value) in enumerate(attributes.items()):
        value_placeholder = f":val{i}"

        if i > 0:
            update_expression += ", "
        update_expression += f"{attr_name} = {value_placeholder}"

        expression_attribute_values[value_placeholder] = attr_value

    # Execute the update
    response = table.update_item(
        Key=key,
```

```
        UpdateExpression=update_expression,
        ExpressionAttributeValues=expression_attribute_values,
        ReturnValues="UPDATED_NEW",
    )

    return response


def update_with_conditional_value_copying(
    table_name: str,
    key: Dict[str, Any],
    source_attribute: str,
    target_attribute: str,
    default_value: Any,
) -> Dict[str, Any]:
    """
    Update an attribute with a value from another attribute or a default value.

    This function demonstrates how to use if_not_exists to conditionally copy a
 value
    from one attribute to another, or use a default value if the source doesn't
 exist.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        source_attribute (str): The attribute to copy the value from.
        target_attribute (str): The attribute to update.
        default_value (Any): The default value to use if the source attribute
 doesn't exist.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use if_not_exists to conditionally copy the value
    response = table.update_item(
        Key=key,
        UpdateExpression=f"SET {target_attribute} =
 if_not_exists({source_attribute}, :default)",
```

```
        ExpressionAttributeValues={":default": default_value},
        ReturnValues="UPDATED_NEW",
    )

    return response
```

Beispiel für die Verwendung von Update Expression Order with AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use update expression order of operations in DynamoDB."""
    # Example parameters
    table_name = "OrderProcessing"
    key = {"OrderId": "order123"}

    print("Example 1: Demonstrating value copying in update expressions")
    try:
        results = demonstrate_value_copying(table_name=table_name, key=key)

        print(f"Initial state: {json.dumps(results['initial_state'], default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
 default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The initial state had a=1, b=2, c=3")
        print("2. The update expression 'REMOVE a SET b = a, c = b' did the
 following:")
        print("   - Copied the value of 'a' (which was 1) to be used for 'b'")
        print("   - Copied the value of 'b' (which was 2) to be used for 'c'")
        print("   - Removed the attribute 'a'")
        print("3. The final state has b=1, c=2, and 'a' is removed")
        print(
            "4. This demonstrates that DynamoDB uses the values of attributes as
 they were BEFORE any modifications"
        )
    except Exception as e:
        print(f"Error demonstrating value copying: {e}")

    print("\nExample 2: Demonstrating the order of different action types")
    try:
```

```
        results = demonstrate_action_order(table_name=table_name, key={"OrderId":
"order456"})

        print(f"Initial state: {json.dumps(results['initial_state'], default=str)}")
        print(f"Update response: {json.dumps(results['update_response'],
default=str)}")
        print(f"Final state: {json.dumps(results['final_state'], default=str)}")

        print("\nExplanation:")
        print("1. The update expression contained multiple action types: REMOVE,
SET, ADD, DELETE")
        print("2. DynamoDB processes these actions in this order: REMOVE, SET, ADD,
DELETE")
        print("3. First, 'to_remove' was removed")
        print("4. Then, 'to_modify' was set to a new value")
        print("5. Next, 'counter' was incremented by 5")
        print("6. Finally, 'B' was removed from the set attribute")
    except Exception as e:
        print(f"Error demonstrating action order: {e}")

    print("\nExample 3: Updating multiple attributes in a single SET action")
    try:
        response = update_with_multiple_set_actions(
            table_name=table_name,
            key={"OrderId": "order789"},
            attributes={
                "Status": "Shipped",
                "ShippingDate": "2025-05-14",
                "TrackingNumber": "1Z999AA10123456784",
            },
        )

        print(
            f"Multiple attributes updated successfully:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error updating multiple attributes: {e}")

    print("\nExample 4: Conditional value copying with if_not_exists")
    try:
        response = update_with_conditional_value_copying(
            table_name=table_name,
            key={"OrderId": "order101"},
```

```
            source_attribute="PreferredShippingMethod",
            target_attribute="ShippingMethod",
            default_value="Standard",
        )

        print(
            f"Conditional value copying result:
{json.dumps(response.get('Attributes', {}), default=str)}"
        )
    except Exception as e:
        print(f"Error with conditional value copying: {e}")

    print("\nKey Points About Update Expression Order of Operations:")
    print(
        "1. Variables in expressions hold copies of attribute values as they existed
BEFORE any modifications"
    )
    print(
        "2. Multiple actions in an update expression are processed in this order:
REMOVE, SET, ADD, DELETE"
    )
    print("3. Within each action type, operations are processed from left to right")
    print("4. You can reference the same attribute multiple times in an expression")
    print("5. You can use if_not_exists() to conditionally set values based on
attribute existence")
    print(
        "6. Using a single update expression with multiple actions is more efficient
than multiple separate updates"
    )
    print("7. The update expression is atomic - either all actions succeed or none
do")
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Aktualisieren der Einstellung für den Warmdurchsatz einer Tabelle

Im folgenden Codebeispiel wird gezeigt, wie die Warmdurchsatzeinstellung einer Tabelle aktualisiert wird.

SDK für Python (Boto3)

Aktualisieren Sie mit AWS SDK für Python (Boto3) die Einstellung für den Warmdurchsatz einer DynamoDB-Tabelle.

```
from boto3 import client
from botocore.exceptions import ClientError


def update_dynamodb_table_warm_throughput(
    table_name,
    table_read_units,
    table_write_units,
    gsi_name,
    gsi_read_units,
    gsi_write_units,
    region_name="us-east-1",
):
    """
    Updates the warm throughput of a DynamoDB table and a global secondary index.

    :param table_name: The name of the table to update.
    :param table_read_units: The new read units per second for the table's warm
 throughput.
    :param table_write_units: The new write units per second for the table's warm
 throughput.
    :param gsi_name: The name of the global secondary index to update.
    :param gsi_read_units: The new read units per second for the GSI's warm
 throughput.
    :param gsi_write_units: The new write units per second for the GSI's warm
 throughput.
    :param region_name: The AWS Region name to target. defaults to us-east-1
    :return: The response from the update_table operation
    """
    try:
        ddb = client("dynamodb", region_name=region_name)

        # Update the table's warm throughput
        table_warm_throughput = {
            "ReadUnitsPerSecond": table_read_units,
            "WriteUnitsPerSecond": table_write_units,
        }

        # Update the global secondary index's warm throughput
```

```
        gsi_warm_throughput = {
            "ReadUnitsPerSecond": gsi_read_units,
            "WriteUnitsPerSecond": gsi_write_units,
        }

        # Construct the global secondary index update
        global_secondary_index_update = [
            {"Update": {"IndexName": gsi_name, "WarmThroughput":
 gsi_warm_throughput}}
        ]

        # Construct the update table request
        update_table_request = {
            "TableName": table_name,
            "GlobalSecondaryIndexUpdates": global_secondary_index_update,
            "WarmThroughput": table_warm_throughput,
        }

        # Update the table
        response = ddb.update_table(**update_table_request)
        print("Table updated successfully!")
        return response  # Make sure to return the response
    except ClientError as e:
        print(f"Error updating table: {e}")
        raise e
```

- API-Details finden Sie UpdateTable in der API-Referenz für AWS SDK für Python (Boto3).

Aktualisieren der TTL eines Elements

Das folgende Codebeispiel zeigt, wie Sie die TTL eines Elements aktualisieren.

SDK für Python (Boto3)

```
from datetime import datetime, timedelta

import boto3


def update_dynamodb_item(table_name, region, primary_key, sort_key):
    """
    Update an existing DynamoDB item with a TTL.
```

```
    :param table_name: Name of the DynamoDB table
    :param region: AWS Region of the table - example `us-east-1`
    :param primary_key: one attribute known as the partition key.
    :param sort_key: Also known as a range attribute.
    :return: Void (nothing)
    """
    try:
        # Create the DynamoDB resource.
        dynamodb = boto3.resource("dynamodb", region_name=region)
        table = dynamodb.Table(table_name)

        # Get the current time in epoch second format
        current_time = int(datetime.now().timestamp())

        # Calculate the expireAt time (90 days from now) in epoch second format
        expire_at = int((datetime.now() + timedelta(days=90)).timestamp())

        table.update_item(
            Key={"partitionKey": primary_key, "sortKey": sort_key},
            UpdateExpression="set updatedAt=:c, expireAt=:e",
            ExpressionAttributeValues={":c": current_time, ":e": expire_at},
        )

        print("Item updated successfully.")
    except Exception as e:
        print(f"Error updating item: {e}")


# Replace with your own values
update_dynamodb_item(
    "your-table-name", "us-west-2", "your-partition-key-value", "your-sort-key-
value"
)
```

- API-Details finden Sie UpdateItem in der API-Referenz für AWS SDK für Python (Boto3).

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten unter GitHub ansehen. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwenden von Operationen mit unteilbaren Zählern

Im folgenden Codebeispiel wird gezeigt, wie Sie Operationen mit unteilbaren Zählern in DynamoDB verwenden.

- Inkrementieren von unteilbaren Zählern mithilfe von ADD- und SET-Operationen
- Sicheres Erhöhen von Zählern, die möglicherweise nicht existieren
- Implementieren einer optimistischen Sperre für Zähleroperationen

SDK für Python (Boto3)

Demonstrieren Sie atomare Gegenoperationen mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Union


def increment_counter_with_add(
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int =
 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the ADD operation.

    This function demonstrates the atomic ADD operation, which is ideal for
    incrementing counters without the risk of race conditions.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the ADD operation to atomically increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="ADD #counter :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_with_set(
```

```
    table_name: str, key: Dict[str, Any], counter_name: str, increment_value: int =
 1
) -> Dict[str, Any]:
    """
    Increment a counter attribute using the SET operation with an expression.

    This function demonstrates using SET with an expression to increment a counter.
    While this works, it's generally recommended to use ADD for simple increments.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use the SET operation with an expression to increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = #counter + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def increment_counter_safely(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    increment_value: int = 1,
    initial_value: int = 0,
) -> Dict[str, Any]:
    """
```

```
    Increment a counter attribute safely, handling the case where it might not
 exist.

    This function demonstrates a best practice for incrementing counters by using
    the if_not_exists function to handle the case where the counter doesn't exist
 yet.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        increment_value (int, optional): The value to increment by. Defaults to 1.
        initial_value (int, optional): The initial value if the counter doesn't
 exist. Defaults to 0.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use SET with if_not_exists to safely increment the counter
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #counter = if_not_exists(#counter, :initial)
 + :increment",
        ExpressionAttributeNames={"#counter": counter_name},
        ExpressionAttributeValues={":increment": increment_value, ":initial":
 initial_value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def atomic_conditional_increment(
    table_name: str,
    key: Dict[str, Any],
    counter_name: str,
    condition_attribute: str,
    condition_value: Any,
    increment_value: int = 1,
```

```
) -> Union[Dict[str, Any], None]:
    """
    Atomically increment a counter only if a condition is met.

    This function demonstrates combining atomic counter operations with
    conditional expressions for more complex update scenarios.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        counter_name (str): The name of the counter attribute.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        increment_value (int, optional): The value to increment by. Defaults to 1.

    Returns:
        Optional[Dict[str, Any]]: The response from DynamoDB if successful, None if
condition failed.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Use ADD with a condition expression
        response = table.update_item(
            Key=key,
            UpdateExpression="ADD #counter :increment",
            ConditionExpression="#condition = :value",
            ExpressionAttributeNames={"#counter": counter_name, "#condition":
condition_attribute},
            ExpressionAttributeValues={":increment": increment_value, ":value":
condition_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung atomarer Zähleroperationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the atomic counter operations functions."""
    # Example parameters
    table_name = "GameScores"
    key = {"UserId": "user123", "GameId": "game456"}
    counter_name = "Score"

    print("Example 1: Incrementing a counter with ADD operation")
    try:
        response = increment_counter_with_add(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=10
        )
        print(
            f"Counter incremented successfully. New value:
 {response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with ADD: {e}")

    print("\nExample 2: Incrementing a counter with SET operation")
    try:
        response = increment_counter_with_set(
            table_name=table_name, key=key, counter_name=counter_name,
 increment_value=5
        )
        print(
            f"Counter incremented successfully. New value:
 {response.get('Attributes', {}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error incrementing counter with SET: {e}")

    print("\nExample 3: Safely incrementing a counter that might not exist")
    try:
        new_key = {"UserId": "newuser789", "GameId": "game456"}
        response = increment_counter_safely(
            table_name=table_name,
```

```
            key=new_key,
            counter_name=counter_name,
            increment_value=15,
            initial_value=100,
        )
        print(
            f"Counter safely incremented. New value: {response.get('Attributes',
{}).get(counter_name)}"
        )
    except Exception as e:
        print(f"Error safely incrementing counter: {e}")

    print("\nExample 4: Conditional counter increment")
    try:
        # Fix for mypy: Handle the case where response might be None
        result = atomic_conditional_increment(
            table_name=table_name,
            key=key,
            counter_name="Achievements",
            condition_attribute="Level",
            condition_value=5,
            increment_value=1,
        )

        if result is not None:
            print(
                f"Conditional increment succeeded. New value:
{result.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
        if response:
            print(
                f"Conditional increment succeeded. New value:
{response.get('Attributes', {}).get('Achievements')}"
            )
        else:
            print("Conditional increment failed because condition was not met.")
    except Exception as e:
        print(f"Error with conditional increment: {e}")

    print("\nComparison of ADD vs SET for counter operations:")
    print("1. ADD is specifically designed for atomic numeric increments and set
operations")
```

```
    print("2. SET with an expression can be used for more complex calculations")
    print("3. Both operations are atomic, preventing race conditions")
    print("4. ADD is more concise for simple increments")
    print("5. SET with if_not_exists() is recommended when the attribute might not
 exist")
    print("6. For counters, ADD is generally preferred for clarity and simplicity")
```

- API-Details finden Sie UpdateItem in der AWS API-Referenz für SDK für Python (Boto3).

Verwenden von bedingten Operationen

Im folgenden Codebeispiel wird gezeigt, wie bedingte Operationen in DynamoDB verwendet werden.

- Implementieren von bedingten Schreibvorgängen, um das Überschreiben von Daten zu verhindern
- Verwenden von Bedingungsausdrücken, um Geschäftsregeln durchzusetzen
- Ordnungsgemäßer Umgang mit Fehlern bei bedingten Prüfungen

SDK für Python (Boto3)

Demonstrieren Sie bedingte Operationen mit. AWS SDK für Python (Boto3)

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, Optional, Tuple, Union


def conditional_update(
    table_name: str,
    key: Dict[str, Any],
    condition_attribute: str,
    condition_value: Any,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item only if a condition is met.

    This function demonstrates how to perform a conditional update operation
```

```
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional update
        response = table.update_item(
            Key=key,
            UpdateExpression="SET #update_attr = :update_val",
            ConditionExpression="#cond_attr = :cond_val",
            ExpressionAttributeNames={
                "#update_attr": update_attribute,
                "#cond_attr": condition_attribute,
            },
            ExpressionAttributeValues={":update_val": update_value, ":cond_val":
condition_value},
            ReturnValues="UPDATED_NEW",
        )
        # Update succeeded, condition was met
        return True, response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False, None
        else:
            # Other error occurred
            raise
```

```
def conditional_delete(
    table_name: str, key: Dict[str, Any], condition_attribute: str, condition_value:
 Any
) -> bool:
    """
    Delete an item only if a condition is met.

    This function demonstrates how to perform a conditional delete operation
    and determine if the condition was met.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to delete.
        condition_attribute (str): The attribute to check in the condition.
        condition_value (Any): The value to compare against.

    Returns:
        bool: True if the delete succeeded (condition was met), False otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        # Perform the conditional delete
        table.delete_item(
            Key=key,
            ConditionExpression="#attr = :val",
            ExpressionAttributeNames={"#attr": condition_attribute},
            ExpressionAttributeValues={":val": condition_value},
        )
        # Delete succeeded, condition was met
        return True
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return False
        else:
            # Other error occurred
            raise


def optimistic_locking_update(
    table_name: str,
```

```
    key: Dict[str, Any],
    version_attribute: str,
    update_attribute: str,
    update_value: Any,
) -> Tuple[bool, Optional[Dict[str, Any]]]:
    """
    Update an item using optimistic locking with a version attribute.

    This function demonstrates how to implement optimistic locking using
    a version attribute that is incremented with each update.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        version_attribute (str): The name of the version attribute.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.

    Returns:
        Tuple[bool, Optional[Dict[str, Any]]]: A tuple containing:
            - A boolean indicating if the update succeeded
            - The response from DynamoDB if successful, None otherwise
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # First, get the current version
    try:
        response = table.get_item(
            Key=key,
            ProjectionExpression=f"#{version_attribute}",
            ExpressionAttributeNames={f"#{version_attribute}": version_attribute},
        )

        item = response.get("Item", {})
        current_version = item.get(version_attribute, 0)

        # Now, try to update with a condition on the version
        try:
            update_response = table.update_item(
                Key=key,
                UpdateExpression=f"SET #{update_attribute} = :update_val,
 #{version_attribute} = :new_version",
```

```
                ConditionExpression=f"#{version_attribute} = :current_version",
                ExpressionAttributeNames={
                    f"#{update_attribute}": update_attribute,
                    f"#{version_attribute}": version_attribute,
                },
                ExpressionAttributeValues={
                    ":update_val": update_value,
                    ":current_version": current_version,
                    ":new_version": current_version + 1,
                },
                ReturnValues="UPDATED_NEW",
            )
            # Update succeeded
            return True, update_response
        except ClientError as e:
            if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
                # Version has changed, optimistic locking failed
                return False, None
            else:
                # Other error occurred
                raise
    except ClientError:
        # Error getting the item
        raise


def conditional_check_and_update(
    table_name: str,
    key: Dict[str, Any],
    check_attribute: str,
    check_value: Any,
    update_attribute: str,
    update_value: Any,
    create_if_not_exists: bool = False,
) -> Union[Dict[str, Any], None]:
    """
    Check if an attribute has a specific value and update another attribute if it
 does.

    This function demonstrates a more complex conditional update that can also
    create the item if it doesn't exist.

    Args:
        table_name (str): The name of the DynamoDB table.
```

```
        key (Dict[str, Any]): The primary key of the item to update.
        check_attribute (str): The attribute to check in the condition.
        check_value (Any): The value to compare against.
        update_attribute (str): The attribute to update.
        update_value (Any): The new value to set.
        create_if_not_exists (bool, optional): Whether to create the item if it
doesn't exist.

    Returns:
        Union[Dict[str, Any], None]: The response from DynamoDB if successful, None
otherwise.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    try:
        if create_if_not_exists:
            # Use attribute_not_exists to create the item if it doesn't exist
            condition_expression = "attribute_not_exists(#pk) OR #check_attr
= :check_val"
            update_expression = "SET #update_attr = :update_val, #check_attr =
if_not_exists(#check_attr, :check_val)"

            # Get the partition key name from the key dictionary
            pk_name = next(iter(key))

            expression_attribute_names = {
                "#pk": pk_name,
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }
        else:
            # Only update if the check attribute has the expected value
            condition_expression = "#check_attr = :check_val"
            update_expression = "SET #update_attr = :update_val"

            expression_attribute_names = {
                "#check_attr": check_attribute,
                "#update_attr": update_attribute,
            }

        # Perform the conditional update
        response = table.update_item(
```

```
            Key=key,
            UpdateExpression=update_expression,
            ConditionExpression=condition_expression,
            ExpressionAttributeNames=expression_attribute_names,
            ExpressionAttributeValues={":check_val": check_value, ":update_val":
 update_value},
            ReturnValues="UPDATED_NEW",
        )
        return response
    except ClientError as e:
        if e.response["Error"]["Code"] == "ConditionalCheckFailedException":
            # Condition was not met
            return None
        else:
            # Other error occurred
            raise
```

Beispiel für die Verwendung bedingter Operationen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use the conditional operations functions."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Conditional Update")
    try:
        # Update the price only if the current stock is greater than 10
        success, response = conditional_update(
            table_name=table_name,
            key=key,
            condition_attribute="Stock",
            condition_value=10,
            update_attribute="Price",
            update_value=99.99,
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes", {})
```

```
            print(f"Update succeeded! New values: {attributes}")
        else:
            print("Update failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional update: {e}")

    print("\nExample 2: Conditional Delete")
    try:
        # Delete the product only if it's discontinued
        success = conditional_delete(
            table_name=table_name,
            key=key,
            condition_attribute="Status",
            condition_value="Discontinued",
        )

        if success:
            print("Delete succeeded! The item was deleted.")
        else:
            print("Delete failed because the condition was not met.")
    except Exception as e:
        print(f"Error during conditional delete: {e}")

    print("\nExample 3: Optimistic Locking")
    try:
        # Update with optimistic locking using a version attribute
        success, response = optimistic_locking_update(
            table_name=table_name,
            key=key,
            version_attribute="Version",
            update_attribute="Description",
            update_value="Updated product description",
        )

        if success:
            # Fix for mypy: Handle the case where response might be None
            attributes = {} if response is None else response.get("Attributes", {})
            print(f"Optimistic locking update succeeded! New values: {attributes}")
        else:
            print("Optimistic locking update failed because the version has
changed.")
    except Exception as e:
        print(f"Error during optimistic locking update: {e}")
```

```
    print("\nExample 4: Conditional Check and Update")
    try:
        # Update the featured status if the product is in stock
        response = conditional_check_and_update(
            table_name=table_name,
            key=key,
            check_attribute="InStock",
            check_value=True,
            update_attribute="Featured",
            update_value=True,
            create_if_not_exists=True,
        )

        if response:
            print(
                f"Conditional check and update succeeded! New values:
{response.get('Attributes', {})}"
            )
        else:
            print("Conditional check and update failed because the condition was not
met.")
    except Exception as e:
        print(f"Error during conditional check and update: {e}")

    print("\nUnderstanding Conditional Operations in DynamoDB:")
    print("1. Conditional operations help maintain data integrity")
    print("2. They prevent race conditions in concurrent environments")
    print("3. Failed conditions result in ConditionalCheckFailedException")
    print("4. No DynamoDB capacity is consumed when conditions fail")
    print("5. Optimistic locking is a common pattern using version attributes")
    print("6. Conditions can be combined with logical operators (AND, OR, NOT)")
    print("7. Conditions can use comparison operators (=, <>, <, <=, >, >=)")
    print(
        "8. attribute_exists() and attribute_not_exists() are useful for checking
attribute presence"
    )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteItem

- PutItem
- UpdateItem

Verwenden von Ausdrucksattributnamen

Im folgenden Codebeispiel wird gezeigt, wie Sie Ausdrucksattributnamen in DynamoDB verwenden.

- Arbeiten mit reservierten Wörtern in DynamoDB-Ausdrücken
- Verwenden von Platzhaltern für Ausdrucksattributnamen
- Behandeln von Sonderzeichen in Attributnamen

SDK für Python (Boto3)

Demonstrieren Sie Ausdrucksattributnamen mit AWS SDK für Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from typing import Any, Dict, List


def use_reserved_word_attribute(
    table_name: str, key: Dict[str, Any], reserved_word: str, value: Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name is a DynamoDB reserved word.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names that are DynamoDB reserved words.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        reserved_word (str): The reserved word to use as an attribute name.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
```

```
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle the reserved word
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #reserved_attr = :value",
        ExpressionAttributeNames={"#reserved_attr": reserved_word},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def use_special_character_attribute(
    table_name: str, key: Dict[str, Any], attribute_with_special_chars: str, value:
 Any
) -> Dict[str, Any]:
    """
    Update an attribute whose name contains special characters.

    This function demonstrates how to use expression attribute names to work with
    attributes that have names containing special characters like spaces, dots, or
 hyphens.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        attribute_with_special_chars (str): The attribute name with special
 characters.
        value (Any): The value to set for the attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names to handle special characters
    response = table.update_item(
        Key=key,
        UpdateExpression="SET #special_attr = :value",
```

```
        ExpressionAttributeNames={"#special_attr": attribute_with_special_chars},
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def query_with_attribute_names(
    table_name: str,
    partition_key_name: str,
    partition_key_value: str,
    filter_attribute_name: str,
    filter_value: Any,
) -> Dict[str, Any]:
    """
    Query a table using expression attribute names for both key and filter
 attributes.

    This function demonstrates how to use expression attribute names in a query
 operation
    for both the key condition expression and filter expression.

    Args:
        table_name (str): The name of the DynamoDB table.
        partition_key_name (str): The name of the partition key attribute.
        partition_key_value (str): The value of the partition key to query.
        filter_attribute_name (str): The name of the attribute to filter on.
        filter_value (Any): The value to compare against in the filter.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the query results.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Use expression attribute names for both key condition and filter
    response = table.query(
        KeyConditionExpression="#pk = :pk_val",
        FilterExpression="#filter_attr = :filter_val",
        ExpressionAttributeNames={"#pk": partition_key_name, "#filter_attr":
 filter_attribute_name},
```

```
        ExpressionAttributeValues={":pk_val": partition_key_value, ":filter_val":
 filter_value},
    )

    return response


def update_nested_attribute_with_dots(
    table_name: str, key: Dict[str, Any], path_with_dots: str, value: Any
) -> Dict[str, Any]:
    """
    Update a nested attribute using a path with dot notation.

    This function demonstrates how to use expression attribute names to work with
    nested attributes specified using dot notation.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.
        path_with_dots (str): The path to the nested attribute using dot notation
 (e.g., "a.b.c").
        value (Any): The value to set for the nested attribute.

    Returns:
        Dict[str, Any]: The response from DynamoDB containing the updated attribute
 values.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Split the path into components
    path_parts = path_with_dots.split(".")

    # Build the update expression and attribute names
    update_expression = "SET "
    expression_attribute_names = {}

    # Build the path expression
    path_expression = ""
    for i, part in enumerate(path_parts):
        name_placeholder = f"#attr{i}"
        expression_attribute_names[name_placeholder] = part
```

```
        if i == 0:
            path_expression = name_placeholder
        else:
            path_expression += f".{name_placeholder}"

    # Complete the update expression
    update_expression += f"{path_expression} = :value"

    # Execute the update
    response = table.update_item(
        Key=key,
        UpdateExpression=update_expression,
        ExpressionAttributeNames=expression_attribute_names,
        ExpressionAttributeValues={":value": value},
        ReturnValues="UPDATED_NEW",
    )

    return response


def demonstrate_attribute_name_requirements(table_name: str, key: Dict[str, Any]) ->
 Dict[str, Any]:
    """
    Demonstrate the requirements and allowed characters for attribute names.

    This function shows examples of valid and invalid attribute names and how to
    handle them using expression attribute names.

    Args:
        table_name (str): The name of the DynamoDB table.
        key (Dict[str, Any]): The primary key of the item to update.

    Returns:
        Dict[str, Any]: A dictionary containing the results of the demonstration.
    """
    # Initialize the DynamoDB resource
    dynamodb = boto3.resource("dynamodb")
    table = dynamodb.Table(table_name)

    # Examples of attribute names with different characteristics
    examples = {
        "valid_standard": "NormalAttribute",  # Standard attribute name (no
 placeholder needed)
        "valid_with_underscore": "Normal_Attribute",  # Underscore is allowed
```

```
        "valid_with_number": "Attribute123",  # Numbers are allowed
        "reserved_word": "Timestamp",  # Reserved word (requires placeholder)
        "starts_with_number": "123Attribute",  # Starts with number (valid but may
need placeholder in some contexts)
        "with_space": "Attribute Name",  # Contains space (requires placeholder)
        "with_dot": "Attribute.Name",  # Contains dot (requires placeholder)
        "with_hyphen": "Attribute-Name",  # Contains hyphen (requires placeholder)
        "with_special_chars": "Attribute#$%",  # Contains special characters
(requires placeholder)
    }

    results = {}

    # Try to update each attribute type
    for example_type, attr_name in examples.items():
        try:
            # For attributes that don't need placeholders, try direct reference
            if example_type in ["valid_standard", "valid_with_underscore",
"valid_with_number"]:
                try:
                    # Try without expression attribute names first
                    response = table.update_item(
                        Key=key,
                        UpdateExpression=f"SET {attr_name} = :value",
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": False,
                        "response": response,
                    }
                except ClientError:
                    # If direct reference fails, try with placeholder
                    response = table.update_item(
                        Key=key,
                        UpdateExpression="SET #attr = :value",
                        ExpressionAttributeNames={"#attr": attr_name},
                        ExpressionAttributeValues={":value": f"Value for
{attr_name}"},
                        ReturnValues="UPDATED_NEW",
                    )
```

```
                    results[example_type] = {
                        "attribute_name": attr_name,
                        "success": True,
                        "needed_placeholder": True,
                        "response": response,
                    }
            else:
                # For attributes that definitely need placeholders
                response = table.update_item(
                    Key=key,
                    UpdateExpression="SET #attr = :value",
                    ExpressionAttributeNames={"#attr": attr_name},
                    ExpressionAttributeValues={":value": f"Value for {attr_name}"},
                    ReturnValues="UPDATED_NEW",
                )
                results[example_type] = {
                    "attribute_name": attr_name,
                    "success": True,
                    "needed_placeholder": True,
                    "response": response,
                }
        except ClientError as e:
            results[example_type] = {"attribute_name": attr_name, "success": False,
 "error": str(e)}

    return results
```

Beispiel für die Verwendung von Ausdrucksattributnamen mit AWS SDK für Python (Boto3).

```
def example_usage():
    """Example of how to use expression attribute names in DynamoDB."""
    # Example parameters
    table_name = "Products"
    key = {"ProductId": "prod123"}

    print("Example 1: Using a reserved word as an attribute name")
    try:
        response = use_reserved_word_attribute(
            table_name=table_name, key=key, reserved_word="Timestamp",
 value="2025-05-14T12:00:00Z"
```

```
        )
        print(f"Reserved word attribute updated successfully:
{response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating reserved word attribute: {e}")

    print("\nExample 2: Using an attribute name with special characters")
    try:
        response = use_special_character_attribute(
            table_name=table_name,
            key=key,
            attribute_with_special_chars="Product Info",
            value="Special product information",
        )
        print(f"Special character attribute updated successfully:
{response.get('Attributes', {})}")
    except Exception as e:
        print(f"Error updating special character attribute: {e}")

    print("\nExample 3: Querying with expression attribute names")
    try:
        response = query_with_attribute_names(
            table_name=table_name,
            partition_key_name="Category",
            partition_key_value="Electronics",
            filter_attribute_name="Price",
            filter_value=500,
        )
        print(
            f"Query with expression attribute names returned
{len(response.get('Items', []))} items"
        )
    except Exception as e:
        print(f"Error querying with expression attribute names: {e}")

    print("\nExample 4: Updating a nested attribute with dot notation")
    try:
        response = update_nested_attribute_with_dots(
            table_name=table_name,
            key=key,
            path_with_dots="Product.Details.Specifications",
            value={"Weight": "2.5 kg", "Dimensions": "30x20x10 cm"},
        )
```

```
        print(f"Nested attribute updated successfully: {response.get('Attributes',
{})}")
    except Exception as e:
        print(f"Error updating nested attribute: {e}")

    print("\nExample 5: Demonstrating attribute name requirements")
    try:
        results = demonstrate_attribute_name_requirements(table_name=table_name,
key=key)

        print("Attribute Name Requirements Results:")
        for example_type, result in results.items():
            if result.get("success", False):
                needed_placeholder = result.get("needed_placeholder", True)
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - {'Requires'
if needed_placeholder else 'Does not require'} placeholder"
                )
            else:
                print(
                    f"  - {example_type}: '{result['attribute_name']}' - Failed:
{result.get('error', 'Unknown error')}"
                )
    except Exception as e:
        print(f"Error demonstrating attribute name requirements: {e}")

    print("\nCommon DynamoDB Reserved Words (sample):")
    reserved_words = get_common_reserved_words()
    print(", ".join(reserved_words[:20]) + "... (and many more)")

    print("\nWhen to Use Expression Attribute Names:")
    print("1. When the attribute name is a DynamoDB reserved word")
    print("2. When the attribute name contains special characters (spaces, dots,
hyphens)")
    print("3. When the attribute name begins with a number")
    print("4. When working with nested attributes using dot notation")
    print("5. When you need to reference the same attribute multiple times in an
expression")

    print("\nExpression Attribute Name Requirements:")
    print("1. Must begin with a pound sign (#)")
    print("2. After the pound sign, must contain at least one character")
    print("3. Can contain alphanumeric characters and underscore (_)")
    print("4. Are case-sensitive")
```

```
    print("5. Must be unique within a single expression")

    print("\nAttribute Name Requirements in DynamoDB:")
    print("1. Can begin with a-z, A-Z, or 0-9")
    print("2. Can contain a-z, A-Z, 0-9, underscore (_), dash (-), and dot (.)")
    print("3. Are case-sensitive")
    print("4. No length restrictions, but practical limits apply")
    print("5. Cannot be a DynamoDB reserved word if used directly in expressions")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - Query
  - UpdateItem

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel eines geplanten EventBridge Amazon-Ereignisses registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten zum späteren Abrufen in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erstellt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Beispiele für Amazon EC2 unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon EC2 beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
```

```
    :param ec2_client: A Boto3 EC2 client. This client provides low-level
                       access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(PaginationConfig={'MaxItems': 10}) #
 List only 10 security groups.
        logging.basicConfig(level=logging.INFO) # Enable logging.
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
        raise


if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- API-Details finden Sie DescribeSecurityGroups in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance, associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
        Initializes the EC2InstanceScenario with the necessary AWS service wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a remote
 execution
                            environment. Defaults to False. If True, the script
 won't prompt
                            for user interaction.
        """
        self.inst_wrapper = inst_wrapper
        self.key_wrapper = key_wrapper
        self.sg_wrapper = sg_wrapper
```

```
        self.eip_wrapper = eip_wrapper
        self.ssm_client = ssm_client
        self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
        Creates an RSA key pair for SSH access to the EC2 instance and lists
available key pairs.
        """
        console.print("**Step 1: Create a Secure Key Pair**", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4)  # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
                if time.time() - start_time > 2:
                    console.print(
                        "Taking longer than expected! Please wait...",
                        style="bold yellow",
                    )

    def create_security_group(self) -> None:
        """
```

```
        Creates a security group that controls access to the EC2 instance and adds a
rule
        to allow SSH access from the user's current public IP address.
        """
        console.print("**Step 2: Create a Security Group**", style="bold cyan")
        console.print(
            "Security groups manage access to your instance. Let's create one."
        )
        sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Security Group Name**: {sg_name}")

        # Create the security group and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Security Group") as bar:
            self.sg_wrapper.create(
                sg_name, "Security group for example: get started with instances."
            )
            time.sleep(0.5)
            bar()

        console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

        # Get the current public IP to set up SSH access.
        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        console.print(
            "Let's add a rule to allow SSH only from your current IP address."
        )
        console.print(f"- **Your Public IP Address**: {current_ip_address}")
        console.print("- Automatically adding SSH rule...")

        # Update security group rules to allow SSH and simulate with a progress bar.
        with alive_bar(1, title="Updating Security Group Rules") as bar:
            response = self.sg_wrapper.authorize_ingress(current_ip_address)
            time.sleep(0.4)
            if response and response.get("Return"):
                console.print("- **Security Group Rules Updated**.")
            else:
                console.print(
                    "- **Error**: Couldn't update security group rules.",
                    style="bold red",
                )
            bar()
```

```
        self.sg_wrapper.describe(self.sg_wrapper.security_group)

    def create_instance(self) -> None:
        """
        Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
 pair
        and security group. Displays instance details and SSH connection
 information.
        """
        # Retrieve Amazon Linux 2 AMIs from SSM.
        ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
        ami_options = []
        for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
            ami_options += page["Parameters"]
        amzn2_images = self.inst_wrapper.get_images(
            [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
        )
        console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
        console.print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
        )
        image_choice = 0
        console.print(f"- Selected AMI: {amzn2_images[image_choice]['ImageId']}\n")

        # Display instance types compatible with the selected AMI
        inst_types = self.inst_wrapper.get_instance_types(
            amzn2_images[image_choice]["Architecture"]
        )
        inst_type_choice = 0
        console.print(
            f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"
        )

        console.print("Creating your instance and waiting for it to start...")
        with alive_bar(1, title="Creating Instance") as bar:
            self.inst_wrapper.create(
                amzn2_images[image_choice]["ImageId"],
                inst_types[inst_type_choice]["InstanceType"],
                self.key_wrapper.key_pair["KeyName"],
                [self.sg_wrapper.security_group],
            )
```

```
            time.sleep(21)
            bar()

        console.print(f"**Success! Your instance is ready:**\n", style="bold green")
        self.inst_wrapper.display()

        console.print(
            "You can use SSH to connect to your instance. "
            "If the connection attempt times out, you might have to manually update
"
            "the SSH ingress rule for your IP address in the AWS Management
Console."
        )
        self._display_ssh_info()

    def _display_ssh_info(self) -> None:
        """
        Displays SSH connection information for the user to connect to the EC2
instance.
        Handles the case where the instance does or does not have an associated
public IP address.
        """
        if (
            not self.eip_wrapper.elastic_ips
            or not self.eip_wrapper.elastic_ips[0].allocation_id
        ):
            if self.inst_wrapper.instances:
                instance = self.inst_wrapper.instances[0]
                instance_id = instance["InstanceId"]

                waiter = self.inst_wrapper.ec2_client.get_waiter("instance_running")
                console.print(
                    "Waiting for the instance to be in a running state with a public
IP...",
                    style="bold cyan",
                )

                with alive_bar(1, title="Waiting for Instance to Start") as bar:
                    waiter.wait(InstanceIds=[instance_id])
                    time.sleep(20)
                    bar()

                instance = self.inst_wrapper.ec2_client.describe_instances(
                    InstanceIds=[instance_id]
```

```
                )["Reservations"][0]["Instances"][0]

                public_ip = instance.get("PublicIpAddress")
                if public_ip:
                    console.print(
                        "\nTo connect via SSH, open another command prompt and run
 the following command:",
                        style="bold cyan",
                    )
                    console.print(
                        f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
                    )
                else:
                    console.print(
                        "Instance does not have a public IP address assigned.",
                        style="bold red",
                    )
            else:
                console.print(
                    "No instance available to retrieve public IP address.",
                    style="bold red",
                )
        else:
            elastic_ip = self.eip_wrapper.elastic_ips[0]
            elastic_ip_address = elastic_ip.public_ip
            console.print(
                f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
            )

        if not self.remote_exec:
            console.print("\nOpen a new terminal tab to try the above SSH command.")
            input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
 cyan")
        console.print(
```

```
            "You can allocate an Elastic IP address and associate it with your
 instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
 IP."
        )
        self._display_ssh_info()

    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains
        changes that occur when the instance is restarted, such as the potential
 change
        in the public IP address unless an Elastic IP is associated.
        """
        console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
 cyan")
        console.print("Let's stop and start your instance to see what changes.")
        console.print("- **Stopping your instance and waiting until it's
 stopped...**")

        with alive_bar(1, title="Stopping Instance") as bar:
            self.inst_wrapper.stop()
```

```
            time.sleep(360)
            bar()

        console.print("- **Your instance is stopped. Restarting...**")

        with alive_bar(1, title="Starting Instance") as bar:
            self.inst_wrapper.start()
            time.sleep(20)
            bar()

        console.print("**Your instance is running.**", style="bold green")
        self.inst_wrapper.display()

        elastic_ip = (
            self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
        )

        if elastic_ip is None or elastic_ip.allocation_id is None:
            console.print(
                "- **Note**: Every time your instance is restarted, its public IP
address changes."
            )
        else:
            console.print(
                f"Because you have associated an Elastic IP with your instance, you
can \n"
                f"connect by using a consistent IP address after the instance
restarts: {elastic_ip.public_ip}"
            )

        self._display_ssh_info()

    def cleanup(self) -> None:
        """
        Cleans up all the resources created during the scenario, including
disassociating
        and releasing the Elastic IP, terminating the instance, deleting the
security
        group, and deleting the key pair.
        """
        console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
        console.print("Cleaning up resources:")
```

```
        for elastic_ip in self.eip_wrapper.elastic_ips:
            console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

            with alive_bar(1, title="Disassociating Elastic IP") as bar:
                self.eip_wrapper.disassociate(elastic_ip.allocation_id)
                time.sleep(2)
                bar()

            console.print("\t- **Disassociated Elastic IP from the Instance**")

            with alive_bar(1, title="Releasing Elastic IP") as bar:
                self.eip_wrapper.release(elastic_ip.allocation_id)
                time.sleep(1)
                bar()

            console.print("\t- **Released Elastic IP**")

        console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

        with alive_bar(1, title="Terminating Instance") as bar:
            self.inst_wrapper.terminate()
            time.sleep(380)
            bar()

        console.print("\t- **Terminated Instance**")

        console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

        with alive_bar(1, title="Deleting Security Group") as bar:
            self.sg_wrapper.delete(self.sg_wrapper.security_group)
            time.sleep(1)
            bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")
```

```
    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
 groups,
        launches an instance, associates an Elastic IP, and cleans up all resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
 with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = EC2InstanceScenario(
            EC2InstanceWrapper.from_client(),
            KeyPairWrapper.from_client(),
            SecurityGroupWrapper.from_client(),
            ElasticIpWrapper.from_client(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Schlüsselpaar-Aktionen umschließt.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())
```

```
    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that represents
the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
                      all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
```

```
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}' with
fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

Definieren Sie eine Klasse, die Sicherheitsgruppen-Aktionen umschließt.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of the
 current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
```

```
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose a
different name."
                )
            raise
        else:
            return self.security_group


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or None
if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
```

```
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to describe
all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security group(s),
such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
```

```
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete. Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an AWS
service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
it is still in use."
```

```
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

Definieren Sie eine Klasse, die Instance-Aktionen umschließt.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                            access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group of
the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
```

```
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Select a different instance type or launch in a different
 availability zone."
                    )
                )
            raise
        return self.instances


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
 instances in this state
                             will be displayed. Default is 'running'. Example
 states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)
```

```
        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match the
 filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise


    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
```

```
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise


    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
```

```
                f"Failed to start instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise


    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no instances
to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response = self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response
```

```
    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str, image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images


    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
 "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types. Default:
 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support the
 specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
```

```
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture}, {','.join(map(str,
 sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
 DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

Definieren Sie eine Klasse, die Elastic-IP-Aktionen umschließt.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
```

```
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the maximum
limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
```

```
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if no
Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance ID
is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
```

```
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance. When
the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:
                self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
```

```
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP address
was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise


    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address "
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise
```

```
    @staticmethod
    def get_elastic_ip_by_allocation(
        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None if
not found.
        """
        return next(
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress

- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
```

```
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP address
        constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the maximum
limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
 Support for an increase."
                )
            raise err
        return elastic_ip
```

- API-Details finden Sie AllocateAddress in der AWS API-Referenz zum SDK für Python (Boto3).

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung. `AssociateAddress`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
```

```
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param instance_id: The ID of the Amazon EC2 instance.
        :return: A response that contains the ID of the association, or None if no
Elastic IP is found.
        :raises ClientError: If the association fails, such as when the instance ID
is not found.
        """
```

```
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return None

        try:
            response = self.ec2_client.associate_address(
                AllocationId=allocation_id, InstanceId=instance_id
            )
            elastic_ip.instance_id = (
                instance_id  # Track the instance associated with this Elastic IP.
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
                logger.error(
                    f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                    "because the specified instance ID does not exist or has not
propagated fully. "
                    "Verify the instance ID and try again, or wait a few moments
before attempting to "
                    "associate the Elastic IP address."
                )
            raise
        return response
```

- API-Details finden Sie AssociateAddress in der AWS API-Referenz zum SDK für Python (Boto3).

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung. `AuthorizeSecurityGroupIngress`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.
```

```
        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                               to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                 response indicates whether the request succeeded or failed, or None
if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling for
ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already exists"
                    f"in security group '{self.security_group}'."
                )
            raise
        else:
            return response
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `CreateKeyPair`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
 Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
```

```
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that represents
the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "
                    "or delete the existing key pair before creating."
                )
            raise
        else:
            return self.key_pair
```

- API-Details finden Sie CreateKeyPair in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `CreateLaunchTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird eine Startvorlage erstellt, die ein Instance-Profil enthält, das der Instance bestimmte Berechtigungen gewährt, und ein Benutzerdaten-Bash-Skript, das nach dem Start auf der Instance ausgeführt wird.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
```

```
                                            when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
```

```
            log.error(f"Failed to create launch template
 {self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
 nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template
```

- API-Details finden Sie CreateLaunchTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `CreateSecurityGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
```

```
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of the
 current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling for
 ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose a
 different name."
                )
            raise
        else:
            return self.security_group
```

- API-Details finden Sie CreateSecurityGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateVpc`

Das folgende Codebeispiel zeigt die Verwendung. `CreateVpc`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
    def create(self, cidr_block: str) -> str:
        """
        Creates a new Amazon VPC with the specified CIDR block.

        :param cidr_block: The CIDR block for the new VPC, such as '10.0.0.0/16'.
        :return: The ID of the new VPC.
        """
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr_block)
            vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[vpc_id])
            return vpc_id
        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateVpc in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateVpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `CreateVpcEndpoint`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""
```

```
    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create_vpc_endpoint(
        self, vpc_id: str, service_name: str, route_table_ids: list[str]
    ) -> Dict[str, Any]:
        """
        Creates a new VPC endpoint for the specified service and associates it with
the specified route tables.

        :param vpc_id: The ID of the VPC to create the endpoint in.
        :param service_name: The name of the service to create the endpoint for.
        :param route_table_ids: A list of IDs of the route tables to associate with
the endpoint.
        :return: A dictionary representing the newly created VPC endpoint.
        """
        try:
            response = self.ec2_client.create_vpc_endpoint(
                VpcId=vpc_id,
                ServiceName=service_name,
                RouteTableIds=route_table_ids,
            )
            return response["VpcEndpoint"]
        except ClientError as err:
            logger.error(
                "Couldn't create VPC endpoint for service %s. Here's why: %s: %s",
                service_name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateVpcEndpoint in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteKeyPair`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
 directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
```

```
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                             if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
```

```
                    f"The key pair '{key_name}' does not exist and cannot be
 deleted. "
                    "Please verify the key pair name and try again."
                )
            raise
```

- API-Details finden Sie DeleteKeyPair in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteLaunchTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteLaunchTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
```

```
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
```

```
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DeleteLaunchTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteSecurityGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
```

```
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete. Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an AWS
 service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
 '{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
 '{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
 it does not exist."
                )
```

```
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted because
 it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise
```

- API-Details finden Sie DeleteSecurityGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteVpc`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteVpc`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
```

```
    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete(self, vpc_id: str) -> None:
        """
        Deletes the specified VPC.

        :param vpc_id: The ID of the VPC to delete.
        """
        try:
            self.ec2_client.delete_vpc(VpcId=vpc_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC %s. Here's why: %s: %s",
                vpc_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteVpc in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteVpcEndpoints`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteVpcEndpoints`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def delete_vpc_endpoints(self, vpc_endpoint_ids: list[str]) -> None:
        """
        Deletes the specified VPC endpoints.

        :param vpc_endpoint_ids: A list of IDs of the VPC endpoints to delete.
        """
        try:
            self.ec2_client.delete_vpc_endpoints(VpcEndpointIds=vpc_endpoint_ids)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC endpoints %s. Here's why: %s: %s",
                vpc_endpoint_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteVpcEndpoints in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeAvailabilityZones`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeAvailabilityZones`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
```

```
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

- API-Details finden Sie DescribeAvailabilityZones in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeIamInstanceProfileAssociations`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeIamInstanceProfileAssociations`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
```

```
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
```

```
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DescribeIamInstanceProfileAssociations in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeImages`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
```

```
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str, image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images
```

- API-Details finden Sie DescribeImages in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeInstanceTypes`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def get_instance_types(
```

```
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
 "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types. Default:
 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support the
 specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture}, {','.join(map(str,
 sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "
                    "Ensure architecture and size strings conform to
 DescribeInstanceTypes API reference."
                )
            raise
        else:
            return inst_types
```

- API-Details finden Sie DescribeInstanceTypes in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.
```

```
        :return: An instance of EC2InstanceWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
instances in this state
                             will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:
                            continue  # Skip this instance if it doesn't match the
filter

                        # Create a formatted string with instance details
                        instance_info = (
                            f"• ID: {instance['InstanceId']}\n"
                            f"• Image ID: {instance['ImageId']}\n"
                            f"• Instance type: {instance['InstanceType']}\n"
                            f"• Key name: {instance['KeyName']}\n"
                            f"• VPC ID: {instance['VpcId']}\n"
```

```
                            f"• Public IP: {instance.get('PublicIpAddress', 'N/
A')}\n"
                            f"• State: {instance_state}"
                        )
                        print(instance_info)

        except ClientError as err:
            logger.error(
                f"Failed to display instance(s). : {' '.join(map(str,
 instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
                raise
```

- API-Details finden Sie DescribeInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeKeyPairs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """
```

```
    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is stored.
                             This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                         This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2 client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())


    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
```

```
                        all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}' with
 fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise
```

- API-Details finden Sie DescribeKeyPairs in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeRouteTables`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeRouteTables`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual Private
 Cloud actions."""
```

```
    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe_route_tables(self, vpc_ids: list[str]) -> None:
        """
        Displays information about the route tables in the specified VPC.

        :param vpc_ids: A list of VPC IDs.
        """
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": vpc_ids}]
            )
            pp(response["RouteTables"])
        except ClientError as err:
            logger.error(
                "Couldn't describe route tables for VPCs %s. Here's why: %s: %s",
                vpc_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeRouteTables in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeSecurityGroups`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
 None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
 security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a high-
level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
 EC2 client.
        """
```

```
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                  If None, an open search is performed to describe
all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security group(s),
such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise
```

- API-Details finden Sie DescribeSecurityGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeSubnets`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeSubnets`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
```

```
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
```

```
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DescribeSubnets in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeVpcs`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeVpcs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

- API-Details finden Sie DescribeVpcs in der AWS API-Referenz zum SDK für Python (Boto3).

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung. `DisassociateAddress`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
 None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
```

```
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance. When
the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:
            logger.info(
                f"No association found for Elastic IP with allocation ID
{allocation_id}."
            )
            return

        try:
            # Retrieve the association ID before disassociating
            response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
            association_id = response["Addresses"][0].get("AssociationId")

            if association_id:
                self.ec2_client.disassociate_address(AssociationId=association_id)
                elastic_ip.instance_id = None  # Remove the instance association
```

```
            else:
                logger.info(
                    f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
                )

        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
                logger.error(
                    f"Failed to disassociate Elastic IP {allocation_id} "
                    "because the specified association ID for the Elastic IP address
was not found. "
                    "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                    "resource before attempting to disassociate it."
                )
            raise
```

- API-Details finden Sie DisassociateAddress in der AWS API-Referenz zum SDK für Python (Boto3).

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung. `RebootInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
```

```
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
```

```
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie RebootInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung. `ReleaseAddress`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticIpWrapper:
```

```
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str] =
None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def release(self, allocation_id: str) -> None:
```

```
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID {allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip)  # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address "
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise
```

- API-Details finden Sie ReleaseAddress in der AWS API-Referenz zum SDK für Python (Boto3).

`ReplaceIamInstanceProfileAssociation`

Das folgende Codebeispiel zeigt die Verwendung. `ReplaceIamInstanceProfileAssociation`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Dieses Beispiel ersetzt das Instance-Profil einer laufenden Instance, startet die Instance neu und sendet nach dem Start einen Befehl an die Instance.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
 is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
```

```
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie ReplaceIamInstanceProfileAssociation in der AWS API-Referenz zum SDK für Python (Boto3).

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung. `RunInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
```

```
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
                                   If not specified, the default security group of
the VPC is used.
        :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
        """
        try:
            instance_params = {
                "ImageId": image_id,
                "InstanceType": instance_type,
                "KeyName": key_pair_name,
            }
            if security_group_ids is not None:
                instance_params["SecurityGroupIds"] = security_group_ids

            response = self.ec2_client.run_instances(
                **instance_params, MinCount=1, MaxCount=1
            )
            instance = response["Instances"][0]
            self.instances.append(instance)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=[instance["InstanceId"]])
```

```
        except ClientError as err:
            params_str = "\n\t".join(
                f"{key}: {value}" for key, value in instance_params.items()
            )
            logger.error(
                f"Failed to complete instance creation request.\nRequest details:
{params_str}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InstanceLimitExceeded":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Terminate unused instances or contact AWS Support for a
 limit increase."
                    )
                )
            if error_code == "InsufficientInstanceCapacity":
                logger.error(
                    (
                        f"Insufficient capacity for instance type '{instance_type}'.
 "
                        "Select a different instance type or launch in a different
 availability zone."
                    )
                )
            raise
        return self.instances
```

- API-Details finden Sie RunInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung. `StartInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def start(self) -> Optional[Dict[str, Any]]:
        """
```

```
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_running")
            waiter.wait(InstanceIds=instance_ids)
            return start_response
        except ClientError as err:
            logger.error(
                f"Failed to start instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't start instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a stopped state before starting
them."
                )
            raise
```

- API-Details finden Sie StartInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung. `StopInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def stop(self) -> Optional[Dict[str, Any]]:
        """
```

```
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no instances
to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response = self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response
```

- API-Details finden Sie StopInstances in der AWS API-Referenz zum SDK für Python (Boto3).

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung. `TerminateInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions using
 the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
 instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
 objects. These are high-level objects that
                          wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default EC2
 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)


    def terminate(self) -> None:
        """
```

```
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                logger.error(
                    "One or more instance IDs do not exist. "
                    "Please verify the instance IDs and try again."
                )
            raise
```

- API-Details finden Sie TerminateInstances in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die Parameter aktualisieren AWS Systems Manager .

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.
```

```
        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)
```

```
        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
```

```
                    "The default security group for your VPC must allow access from
this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
```

```
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
```

```
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
 self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
 instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
```

```
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
```

```
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
 created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
 to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
```

```
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
```

```
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
 by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
```

```
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
```

```
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
```

```
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is assumed
 by the
```

```
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
```

```
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
```

```
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
```

```
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
```

```
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
```

```
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
```

```
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
```

```
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
```

```
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
```

```
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
```

```
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
```

```
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
```

```
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
```

```
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
```

```
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
```

```
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
```

```
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client
```

```
    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
and
        lower thresholds. In production, you might want to decrease the sensitivity
of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
```

```
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
```

```
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
```

```
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
```

```
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
```

```
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```

```
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
and "
                    "that you have the necessary permissions to access it.",
```

```
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
 as
        Book or Movie, and a range key named 'ItemId' that, combined with the
 MediaType,
        forms a unique identifier for the recommended item.
```

```
        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
```

```
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
 batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """
```

```
    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
```

```
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups

- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Beispiele für Amazon ECR unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon ECR

Das folgende Codebeispiel zeigt, wie Sie mit Amazon ECR beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
import argparse
from boto3 import client


def hello_ecr(ecr_client: client, repository_name: str) -> None:
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
    client and list the images in a repository.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
                           the low-level Amazon ECR service API.
    :param repository_name: The name of an Amazon ECR repository in your account.
    """
    print(
        f"Hello, Amazon ECR! Let's list some images in the repository
 '{repository_name}':\n"
    )
    paginator = ecr_client.get_paginator("list_images")
    page_iterator = paginator.paginate(
        repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
    )

    image_names: [str] = []
    for page in page_iterator:
        for schedule in page["imageIds"]:
            image_names.append(schedule["imageTag"])

    print(f"{len(image_names)} image(s) retrieved.")
    for schedule_name in image_names:
        print(f"\t{schedule_name}")
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
    parser.add_argument(
        "--repository-name",
        type=str,
        help="the name of an Amazon ECR repository in your account.",
        required=True,
    )
    args = parser.parse_args()

    hello_ecr(boto3.client("ecr"), args.repository_name)
```

- Weitere API-Informationen finden Sie unter listImages in der API-Referenz zum AWS -SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Amazon-ECR-Repository.
- Legen Sie Repository-Richtlinien fest.
- Rufen Sie Repository-URIs ab.
- Rufen Sie Amazon-ECR-Autorisierungs-Token ab.
- Legen Sie Richtlinien für den Lebenszyklus von Amazon-ECR-Repositorys fest.
- Pushen Sie ein Docker-Image in ein Amazon-ECR-Repository.
- Verifizieren Sie das Vorhandensein eines Images in einem Amazon-ECR-Repository.
- Listen Sie die Amazon-ECR-Repositorys für Ihr Konto auf und erhalten Sie detaillierte Informationen dazu.
- Löschen Sie Amazon-ECR-Repositorys.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class ECRGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to perform basic operations using
    Amazon ECR.
    """

    def __init__(
        self,
        ecr_wrapper: ECRWrapper,
        docker_client: docker.DockerClient,
    ):
        self.ecr_wrapper = ecr_wrapper
        self.docker_client = docker_client
        self.tag = "echo-text"
        self.repository_name = "ecr-basics"
        self.docker_image = None
        self.full_tag_name = None
        self.repository = None

    def run(self, role_arn: str) -> None:
        """
        Runs the scenario.
        """
        print(
            """
The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container
 registry
service provided by AWS. It allows developers and organizations to securely
store, manage, and deploy Docker container images.
ECR provides a simple and scalable way to manage container images throughout their
 lifecycle,
from building and testing to production deployment.
```

```
The `ECRWrapper' class is a wrapper for the Boto3 'ecr' client. The 'ecr' client
 provides a set of methods to
programmatically interact with the Amazon ECR service. This allows developers to
automate the storage, retrieval, and management of container images as part of their
 application
deployment pipelines. With ECR, teams can focus on building and deploying their
applications without having to worry about the underlying infrastructure required to
host and manage a container registry.

This scenario walks you through how to perform key operations for this service.
Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(
            f"""
* Create an ECR repository.

An ECR repository is a private Docker container repository provided
by Amazon Web Services (AWS). It is a managed service that makes it easy
to store, manage, and deploy Docker container images.
        """
        )
        print(f"Creating a repository named {self.repository_name}")
        self.repository = self.ecr_wrapper.create_repository(self.repository_name)
        print(f"The ARN of the ECR repository is
 {self.repository['repositoryArn']}")
        repository_uri = self.repository["repositoryUri"]
        press_enter_to_continue()
        print_dashes()

        print(
            f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
            """
        )
        print(f"Building a docker image from 'docker_files/Dockerfile'")
        self.full_tag_name = f"{repository_uri}:{self.tag}"
        self.docker_image = self.docker_client.images.build(
```

```
            path="docker_files", tag=self.full_tag_name
        )[0]
        print(f"Docker image {self.full_tag_name} successfully built.")
        press_enter_to_continue()
        print_dashes()

        if role_arn is None:
            print(
                """
* Because an IAM role ARN was not provided, a role policy will not be set for this
 repository.
            """
            )
        else:
            print(
                """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is crucial
 for maintaining
the security and integrity of your container images. The repository policy allows
 you to
define specific rules and restrictions for accessing and managing the images stored
 within your ECR
repository.
        """
            )

            self.grant_role_download_access(role_arn)
            print(f"Download access granted to the IAM role ARN {role_arn}")
            press_enter_to_continue()
            print_dashes()

            print(
                """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
            """
            )

            policy_text =
 self.ecr_wrapper.get_repository_policy(self.repository_name)
            print("Policy Text:")
```

```
            print(f"{policy_text}")
            press_enter_to_continue()
            print_dashes()

        print(
            """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon ECR
 registry.
The `get_authorization_token` method of the `ecr` client is responsible for securely
 accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any operations
 on the
ECR repository, such as pushing, pulling, or managing your Docker images.
        """
        )

        authorization_token = self.ecr_wrapper.get_authorization_token()
        print("Authorization token retrieved.")
        press_enter_to_continue()
        print_dashes()
        print(
            """
* Get the ECR Repository URI.

The URI  of an Amazon ECR repository is important. When you want to deploy a
 container image to
a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)
or Amazon Elastic Container Service (ECS), you need to specify the full image URI,
which includes the ECR repository URI. This allows the container runtime to pull the
correct container image from the ECR repository.
        """
        )
        repository_descriptions = self.ecr_wrapper.describe_repositories(
            [self.repository_name]
        )
        repository_uri = repository_descriptions[0]["repositoryUri"]
        print(f"Repository URI found: {repository_uri}")
```

```
        press_enter_to_continue()
        print_dashes()

        print(
            """
* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in
 your ECR repositories.
These policies allow you to automatically remove old or unused Docker images from
 your repositories,
freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container
 registry
by automatically removing older and potentially unused images, ensuring that the
storage is optimized and the registry remains up-to-date.
            """
        )
        press_enter_to_continue()
        self.put_expiration_policy()
        print(f"An expiration policy was added to the repository.")
        print_dashes()

        print(
            """
* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when
 pushing the image to the
ECR repository.
        """
        )
        decoded_authorization =
 base64.b64decode(authorization_token).decode("utf-8")
        username, password = decoded_authorization.split(":")

        resp = self.docker_client.api.push(
            repository=repository_uri,
            auth_config={"username": username, "password": password},
            tag=self.tag,
            stream=True,
            decode=True,
        )
```

```
        for line in resp:
            print(line)

        print_dashes()

        print("* Verify if the image is in the ECR Repository.")
        image_descriptions = self.ecr_wrapper.describe_images(
            self.repository_name, [self.tag]
        )
        if len(image_descriptions) > 0:
            print("Image found in ECR Repository.")
        else:
            print("Image not found in ECR Repository.")
        press_enter_to_continue()
        print_dashes()

        print(
            "* As an optional step, you can interact with the image in Amazon ECR by
 using the CLI."
        )
        if q.ask(
            "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
            q.is_yesno,
        ):
            print(
                f"""
1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you need
 to authenticate with the registry. You can do this using the AWS CLI:

    aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin {repository_uri.split("/")[0]}

2. Describe the image using this command:

   aws ecr describe-images --repository-name {self.repository_name} --image-ids
 imageTag={self.tag}

3. Run the Docker container and view the output using this command:

   docker run --rm {self.full_tag_name}
"""
            )
```

```
        self.cleanup(True)

    def cleanup(self, ask: bool):
        """
        Deletes the resources created in this scenario.
        :param ask: If True, prompts the user to confirm before deleting the
resources.
        """
        if self.repository is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the ECR repository
'{self.repository_name}? (y/n) "
            )
        ):
            print(f"Deleting the ECR repository '{self.repository_name}'.")
            self.ecr_wrapper.delete_repository(self.repository_name)

        if self.full_tag_name is not None and (
            not ask
            or q.ask(
                f"Would you like to delete the local Docker image
'{self.full_tag_name}? (y/n) "
            )
        ):
            print(f"Deleting the docker image '{self.full_tag_name}'.")
            self.docker_client.images.remove(self.full_tag_name)

    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
```

```
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )


    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )



if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Run Amazon ECR getting started scenario."
    )
    parser.add_argument(
        "--iam-role-arn",
        type=str,
        default=None,
        help="an optional IAM role ARN that will be granted access to download
 images from a repository.",
        required=False,
```

```
    )
    parser.add_argument(
        "--no-art",
        action="store_true",
        help="accessibility setting that suppresses art in the console output.",
    )
    args = parser.parse_args()
    no_art = args.no_art
    iam_role_arn = args.iam_role_arn
    demo = None
    a_docker_client = None
    try:
        a_docker_client = docker.from_env()
        if not a_docker_client.ping():
            raise docker.errors.DockerException("Docker is not running.")
    except docker.errors.DockerException as err:
        logging.error(
            """
        The Python Docker client could not be created.
        Do you have Docker installed and running?
        Here is the error message:
        %s
        """,
            err,
        )
        sys.exit("Error with Docker.")
    try:
        an_ecr_wrapper = ECRWrapper.from_client()
        demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
        demo.run(iam_role_arn)

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup(False)
```

ECRWrapper-Klasse, die Amazon ECR-Aktionen umschließt.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryAlreadyExistsException":
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.
```

```
        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_repository_policy(self, repository_name: str) -> str:
        """
```

```
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.
```

```
        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
```

```
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

Aktionen

`CreateRepository`

Das folgende Codebeispiel zeigt die Verwendung`CreateRepository`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def create_repository(self, repository_name: str) -> dict[str, any]:
        """
        Creates an ECR repository.

        :param repository_name: The name of the repository to create.
        :return: A dictionary of the created repository.
        """
        try:
            response =
 self.ecr_client.create_repository(repositoryName=repository_name)
            return response["repository"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryAlreadyExistsException":
```

```
                print(f"Repository {repository_name} already exists.")
                response = self.ecr_client.describe_repositories(
                    repositoryNames=[repository_name]
                )
                return self.describe_repositories([repository_name])[0]
            else:
                logger.error(
                    "Error creating repository %s. Here's why %s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie CreateRepository in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteRepository`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteRepository`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def delete_repository(self, repository_name: str):
        """
        Deletes an ECR repository.

        :param repository_name: The name of the repository to delete.
        """
        try:
            self.ecr_client.delete_repository(
                repositoryName=repository_name, force=True
            )
            print(f"Deleted repository {repository_name}.")
        except ClientError as err:
            logger.error(
                "Couldn't delete repository %s.. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteRepository in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeImages`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_images(
        self, repository_name: str, image_ids: list[str] = None
    ) -> list[dict]:
        """
        Describes ECR images.

        :param repository_name: The name of the repository to describe images for.
        :param image_ids: The optional IDs of images to describe.
        :return: The list of image descriptions.
        """
        try:
            params = {
                "repositoryName": repository_name,
            }
            if image_ids is not None:
                params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

            paginator = self.ecr_client.get_paginator("describe_images")
            image_descriptions = []
            for page in paginator.paginate(**params):
                image_descriptions.extend(page["imageDetails"])
            return image_descriptions
        except ClientError as err:
            logger.error(
                "Couldn't describe images. Here's why %s",
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- API-Details finden Sie DescribeImages in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeRepositories`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def describe_repositories(self, repository_names: list[str]) -> list[dict]:
        """
        Describes ECR repositories.

        :param repository_names: The names of the repositories to describe.
        :return: The list of repository descriptions.
```

```
        """
        try:
            response = self.ecr_client.describe_repositories(
                repositoryNames=repository_names
            )
            return response["repositories"]
        except ClientError as err:
            logger.error(
                "Couldn't describe repositories. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeRepositories in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAuthorizationToken`

Das folgende Codebeispiel zeigt die Verwendung. `GetAuthorizationToken`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
```

```
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_authorization_token(self) -> str:
        """
        Gets an authorization token for an ECR repository.

        :return: The authorization token.
        """
        try:
            response = self.ecr_client.get_authorization_token()
            return response["authorizationData"][0]["authorizationToken"]
        except ClientError as err:
            logger.error(
                "Couldn't get authorization token. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie GetAuthorizationToken in der AWS API-Referenz zum SDK für Python (Boto3).

`GetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `GetRepositoryPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client
```

```
    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def get_repository_policy(self, repository_name: str) -> str:
        """
        Gets the policy for an ECR repository.

        :param repository_name: The name of the repository to get the policy for.
        :return: The policy text.
        """
        try:
            response = self.ecr_client.get_repository_policy(
                repositoryName=repository_name
            )
            return response["policyText"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
                raise
            else:
                logger.error(
                    "Couldn't get repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie GetRepositoryPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`PutLifeCyclePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `PutLifeCyclePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
        """
        Puts a lifecycle policy for an ECR repository.

        :param repository_name: The name of the repository to put the lifecycle
 policy for.
        :param lifecycle_policy_text: The lifecycle policy text to put.
        """
        try:
            self.ecr_client.put_lifecycle_policy(
                repositoryName=repository_name,
                lifecyclePolicyText=lifecycle_policy_text,
            )
            print(f"Put lifecycle policy for repository {repository_name}.")
```

```
        except ClientError as err:
            logger.error(
                "Couldn't put lifecycle policy for repository %s. Here's why %s",
                repository_name,
                err.response["Error"]["Message"],
            )
            raise
```

Beispiel, das eine Richtlinie für ein Ablaufdatum festlegt.

```
    def put_expiration_policy(self):
        """
        Puts an expiration policy on the ECR repository.
        """
        policy_json = {
            "rules": [
                {
                    "rulePriority": 1,
                    "description": "Expire images older than 14 days",
                    "selection": {
                        "tagStatus": "any",
                        "countType": "sinceImagePushed",
                        "countUnit": "days",
                        "countNumber": 14,
                    },
                    "action": {"type": "expire"},
                }
            ]
        }

        self.ecr_wrapper.put_lifecycle_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- API-Details finden Sie PutLifeCyclePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

SetRepositoryPolicy

Das folgende Codebeispiel zeigt die Verwendung. `SetRepositoryPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ECRWrapper:
    def __init__(self, ecr_client: client):
        self.ecr_client = ecr_client

    @classmethod
    def from_client(cls) -> "ECRWrapper":
        """
        Creates a ECRWrapper instance with a default Amazon ECR client.

        :return: An instance of ECRWrapper initialized with the default Amazon ECR
 client.
        """
        ecr_client = boto3.client("ecr")
        return cls(ecr_client)


    def set_repository_policy(self, repository_name: str, policy_text: str):
        """
        Sets the policy for an ECR repository.

        :param repository_name: The name of the repository to set the policy for.
        :param policy_text: The policy text to set.
        """
        try:
            self.ecr_client.set_repository_policy(
                repositoryName=repository_name, policyText=policy_text
            )
            print(f"Set repository policy for repository {repository_name}.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "RepositoryPolicyNotFoundException":
                logger.error("Repository does not exist. %s.", repository_name)
```

```
                raise
            else:
                logger.error(
                    "Couldn't set repository policy for repository %s. Here's why
%s",
                    repository_name,
                    err.response["Error"]["Message"],
                )
                raise
```

Beispiel, das einer IAM-Rolle Download-Zugriff gewährt.

```
    def grant_role_download_access(self, role_arn: str):
        """
        Grants the specified role access to download images from the ECR repository.

        :param role_arn: The ARN of the role to grant access to.
        """
        policy_json = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowDownload",
                    "Effect": "Allow",
                    "Principal": {"AWS": role_arn},
                    "Action": ["ecr:BatchGetImage"],
                }
            ],
        }

        self.ecr_wrapper.set_repository_policy(
            self.repository_name, json.dumps(policy_json)
        )
```

- API-Details finden Sie SetRepositoryPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

Elastic Load Balancing – Beispiele für Version 2 unter Verwendung von SDK für Java 2.x

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Elastic Load Balancing — Version 2 Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Elastic Load Balancing

Das folgende Codebeispiel zeigt die ersten Schritte mit Elastic Load Balancing.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
```

```
def hello_elbv2(elbv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Elastic Load Balancing V2 client
 and list
    up to ten of the load balancers for your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param elbv2_client: A Boto3 Elastic Load Balancing V2 client object.
    """
    print("Hello, Elastic Load Balancing! Let's list some of your load balancers:")
    load_balancers = elbv2_client.describe_load_balancers(PageSize=10).get(
        "LoadBalancers", []
    )
    if load_balancers:
        for lb in load_balancers:
            print(f"\t{lb['LoadBalancerName']}: {lb['DNSName']}")
    else:
        print("Your account doesn't have any load balancers.")


if __name__ == "__main__":
    hello_elbv2(boto3.client("elbv2"))
```

- API-Details finden Sie DescribeLoadBalancers in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`CreateListener`

Das folgende Codebeispiel zeigt die Verwendung. `CreateListener`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
 the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
 listener for.
        :param target_group: An existing target group that is added as a listener to
 the
                              load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
```

```
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie CreateListener in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `CreateLoadBalancer`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
 subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
```

```
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer
```

- API-Details finden Sie CreateLoadBalancer in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTargetGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
        lower thresholds. In production, you might want to decrease the sensitivity
 of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
```

```
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie CreateTargetGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteLoadBalancer`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteLoadBalancer`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
 {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
```

```
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DeleteLoadBalancer in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTargetGroup`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTargetGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
 self.elb_client.describe_target_groups(Names=[target_group_name])
```

```
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
```

```
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )
```

- API-Details finden Sie DeleteTargetGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeLoadBalancers`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeLoadBalancers`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def get_endpoint(self, load_balancer_name) -> str:
        """
```

```
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

- API-Details finden Sie DescribeLoadBalancers in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeTargetHealth`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeTargetHealth`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""
```

```
    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

- API-Details finden Sie DescribeTargetHealth in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die Parameter aktualisieren AWS Systems Manager .

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
```

```
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")
```

```
        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
```

```
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
```

```
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
```

```
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
 self.recommendation.table_name)

        logging.info(
```

```
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
```

```
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
to avoid unexpected charges."
            )


def main() -> None:
```

```
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
```

```
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
```

```
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
```

```
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
```

```
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
```

```
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
```

```
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
```

```
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
```

```
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
```

```
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")
```

```
    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
```

```
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
```

```
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.
```

```
        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
```

```
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
```

```
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
```

```
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
```

```
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
```

```
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.
```

```
        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.
```

```
        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
```

```
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
how
        the load balancer forwards requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
and
        lower thresholds. In production, you might want to decrease the sensitivity
of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
```

```
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )
```

```
    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
```

```
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
```

```
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
```

```
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")
```

```
    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
```

```
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client
```

```
    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.
```

```
        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                            service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
```

```
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups

- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Beispiele für Amazon EMR unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EMR Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AddJobFlowSteps`

Das folgende Codebeispiel zeigt die Verwendung`AddJobFlowSteps`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie einen Spark-Schritt hinzu, der vom Cluster ausgeführt wird, sobald er hinzugefügt wurde.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):
    """
    Adds a job step to the specified cluster. This example adds a Spark
    step, which is run by the cluster as soon as it is added.

    :param cluster_id: The ID of the cluster.
    :param name: The name of the step.
    :param script_uri: The URI where the Python script is stored.
    :param script_args: Arguments to pass to the Python script.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly added step.
    """
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id,
            Steps=[
                {
                    "Name": name,
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            script_uri,
```

```
                            *script_args,
                        ],
                    },
                }
            ],
        )
        step_id = response["StepIds"][0]
        logger.info("Started step with ID %s", step_id)
    except ClientError:
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)
        raise
    else:
        return step_id
```

Führen Sie einen Befehl des Amazon-EMR-Dateisystems (EMRFS) als Auftragsschritt auf einem Cluster aus. Dies kann verwendet werden, um EMRFS-Befehle auf einem Cluster zu automatisieren, anstatt Befehle manuell über eine SSH-Verbindung auszuführen.

```
import boto3
from botocore.exceptions import ClientError


def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
    """
    Add an EMRFS command as a job flow step to an existing cluster.

    :param command: The EMRFS command to run.
    :param bucket_url: The URL of a bucket that contains tracking metadata.
    :param cluster_id: The ID of the cluster to update.
    :param emr_client: The Boto3 Amazon EMR client object.
    :return: The ID of the added job flow step. Status can be tracked by calling
             the emr_client.describe_step() function.
    """
    job_flow_step = {
        "Name": "Example EMRFS Command Step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
            "Jar": "command-runner.jar",
            "Args": ["/usr/bin/emrfs", command, bucket_url],
        },
```

```
    }

    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id, Steps=[job_flow_step]
        )
        step_id = response["StepIds"][0]
        print(f"Added step {step_id} to cluster {cluster_id}.")
    except ClientError:
        print(f"Couldn't add a step to cluster {cluster_id}.")
        raise
    else:
        return step_id


def usage_demo():
    emr_client = boto3.client("emr")
    # Assumes the first waiting cluster has EMRFS enabled and has created metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=["WAITING"])["Clusters"][0]
    add_emrfs_step(
        "sync", "s3://elasticmapreduce/samples/cloudfront", cluster["Id"],
 emr_client
    )


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie AddJobFlowSteps in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeCluster`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response["Cluster"]
        logger.info("Got data for cluster %s.", cluster["Name"])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
        raise
    else:
        return cluster
```

- API-Details finden Sie DescribeCluster in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeStep`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeStep`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current state
 of
    the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response["Step"]
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- API-Details finden Sie DescribeStep in der AWS API-Referenz zum SDK für Python (Boto3).

`ListSteps`

Das folgende Codebeispiel zeigt die Verwendung. `ListSteps`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """
    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response["Steps"]
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- API-Details finden Sie ListSteps in der AWS API-Referenz zum SDK für Python (Boto3).

`RunJobFlow`

Das folgende Codebeispiel zeigt die Verwendung. `RunJobFlow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def run_job_flow(
    name,
    log_uri,
    keep_alive,
    applications,
    job_flow_role,
    service_role,
    security_groups,
    steps,
    emr_client,
):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
components.html.

    :param name: The name of the cluster.
    :param log_uri: The URI where logs are stored. This can be an Amazon S3 bucket
 URL,
                    such as 's3://my-log-bucket'.
    :param keep_alive: When True, the cluster is put into a Waiting state after all
                       steps are run. When False, the cluster terminates itself when
                       the step queue is empty.
    :param applications: The applications to install on each instance in the
 cluster,
                         such as Hive or Spark.
    :param job_flow_role: The IAM role assumed by the cluster.
    :param service_role: The IAM role assumed by the service.
    :param security_groups: The security groups to assign to the cluster instances.
```

```
                            Amazon EMR adds all needed rules to these groups, so
                            they can be empty if you require only the default rules.
    :param steps: The job flow steps to add to the cluster. These are run in order
                  when the cluster is ready.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly created cluster.
    """
    try:
        response = emr_client.run_job_flow(
            Name=name,
            LogUri=log_uri,
            ReleaseLabel="emr-5.30.1",
            Instances={
                "MasterInstanceType": "m5.xlarge",
                "SlaveInstanceType": "m5.xlarge",
                "InstanceCount": 3,
                "KeepJobFlowAliveWhenNoSteps": keep_alive,
                "EmrManagedMasterSecurityGroup": security_groups["manager"].id,
                "EmrManagedSlaveSecurityGroup": security_groups["worker"].id,
            },
            Steps=[
                {
                    "Name": step["name"],
                    "ActionOnFailure": "CONTINUE",
                    "HadoopJarStep": {
                        "Jar": "command-runner.jar",
                        "Args": [
                            "spark-submit",
                            "--deploy-mode",
                            "cluster",
                            step["script_uri"],
                            *step["script_args"],
                        ],
                    },
                }
                for step in steps
            ],
            Applications=[{"Name": app} for app in applications],
            JobFlowRole=job_flow_role.name,
            ServiceRole=service_role.name,
            EbsRootVolumeSize=10,
            VisibleToAllUsers=True,
        )
        cluster_id = response["JobFlowId"]
```

```
        logger.info("Created cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't create cluster.")
        raise
    else:
        return cluster_id
```

- API-Details finden Sie RunJobFlow in der AWS API-Referenz zum SDK für Python (Boto3).

`TerminateJobFlows`

Das folgende Codebeispiel zeigt die Verwendung. `TerminateJobFlows`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def terminate_cluster(cluster_id, emr_client):
    """
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is
 lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- API-Details finden Sie TerminateJobFlows in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen Sie einen kurzlebigen Amazon EMR-Cluster und führen Sie einen Schritt aus

Das folgende Codebeispiel zeigt, wie Sie einen kurzlebigen Amazon EMR-Cluster erstellen, der einen Schritt ausführt und nach Abschluss des Schritts automatisch beendet wird.

SDK für Python (Boto3)

Erstellen Sie mithilfe von Apache Spark einen kurzlebigen Amazon EMR-Cluster, der den Wert von Pi schätzt, um eine große Anzahl von Berechnungen zu parallelisieren. Der Job schreibt Ausgaben in Amazon EMR-Protokolle und in einen Amazon Simple Storage Service (Amazon S3)-Bucket. Der Cluster beendet sich selbst, nachdem der Job abgeschlossen wurde.

- Erstellen Sie einen Amazon-S3-Bucket und uploaden Sie ein Job-Skript.
- Erstellen Sie AWS Identity and Access Management (IAM-) Rollen.
- Erstellen Sie Amazon Elastic Compute Cloud (Amazon EC2)-Sicherheitsgruppen.
- Erstellen Sie einen kurzlebigen Cluster und führen Sie einen einzelnen Arbeitsschritt aus.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon EMR

Ausführen eines Shell-Skripts zum Installieren von Bibliotheken

Das folgende Codebeispiel zeigt, wie ein Shell-Skript auf Amazon EMR-Instances ausgeführt wird, das zusätzliche Bibliotheken installiert. AWS Systems Manager Auf diese Weise können Sie die Instance-Verwaltung automatisieren, anstatt Befehle manuell über eine SSH-Verbindung auszuführen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import argparse
import time
import boto3


def install_libraries_on_core_nodes(cluster_id, script_path, emr_client,
 ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=["CORE"]
    )["Instances"]
    core_instance_ids = [node["Ec2InstanceId"] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh",
    ]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
            InstanceIds=core_instance_ids,
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": [command]},
            TimeoutSeconds=3600,
```

```
        )["Command"]["CommandId"]
        while True:
            # Verify the previous step succeeded before running the next step.
            cmd_result = ssm_client.list_commands(CommandId=command_id)["Commands"]
[0]
            if cmd_result["StatusDetails"] == "Success":
                print(f"Command succeeded.")
                break
            elif cmd_result["StatusDetails"] in ["Pending", "InProgress"]:
                print(f"Command status is {cmd_result['StatusDetails']},
 waiting...")
                time.sleep(10)
            else:
                print(f"Command status is {cmd_result['StatusDetails']}, quitting.")
                raise RuntimeError(
                    f"Command {command} failed to run. "
                    f"Details: {cmd_result['StatusDetails']}"
                )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("cluster_id", help="The ID of the cluster.")
    parser.add_argument("script_path", help="The path to the script in Amazon S3.")
    args = parser.parse_args()

    emr_client = boto3.client("emr")
    ssm_client = boto3.client("ssm")

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ListInstances in der AWS API-Referenz zum SDK für Python (Boto3).

EventBridge Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel eines geplanten EventBridge Amazon-Ereignisses registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten zum späteren Abrufen in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erstellt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge Scheduler-Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von EventBridge Scheduler Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo Scheduler EventBridge

Das folgende Codebeispiel zeigt, wie Sie mit EventBridge Scheduler beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_scheduler(scheduler_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EventBridge Scheduler
    client and list the schedules in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param scheduler_client: A Boto3 Amazon EventBridge Scheduler Client object.
 This object wraps
                             the low-level Amazon EventBridge Scheduler service API.
    """
    print("Hello, Amazon EventBridge Scheduler! Let's list some of your schedules:
\n")
    paginator = scheduler_client.get_paginator("list_schedules")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    schedule_names: [str] = []
    for page in page_iterator:
        for schedule in page["Schedules"]:
            schedule_names.append(schedule["Name"])

    print(f"{len(schedule_names)} schedule(s) retrieved.")
    for schedule_name in schedule_names:
        print(f"\t{schedule_name}")


if __name__ == "__main__":
    hello_scheduler(boto3.client("scheduler"))
```

- API-Details finden Sie ListSchedules in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`CreateSchedule`

Das folgende Codebeispiel zeigt die Verwendung. `CreateSchedule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
 runs.
```

```
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM role.
        :param input: The input for the target.
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input": input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
```

```
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]["Message"]
                )
            raise
```

- API-Details finden Sie CreateSchedule in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung. `CreateScheduleGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.
```

```
        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule group '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie CreateScheduleGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteSchedule`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteSchedule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
```

```
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
client.

        :return: An instance of SchedulerWrapper initialized with the default
EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource was
not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]["Message"]
                )
                raise
```

- API-Details finden Sie DeleteSchedule in der AWS API-Referenz zum SDK für Python (Boto3).

DeleteScheduleGroup

Das folgende Codebeispiel zeigt die Verwendung. DeleteScheduleGroup

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
 resource was not found: %s",
```

```
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie DeleteScheduleGroup in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Geplante Ereignisse

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Stellen Sie einen CloudFormation Stack mit den erforderlichen Ressourcen bereit.
- Erstellen Sie eine EventBridge Scheduler-Zeitplangruppe.
- Erstellen Sie einen einmaligen EventBridge Scheduler-Zeitplan mit einem flexiblen Zeitfenster.
- Erstellen Sie einen wiederkehrenden EventBridge Scheduler-Zeitplan mit einer bestimmten Rate.
- Löschen Sie EventBridge Scheduler, den Zeitplan und die Zeitplangruppe.
- Bereinigen Sie die Ressourcen und löschen Sie den Stack.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SchedulerScenario:
    """
```

```
    A scenario that demonstrates how to use Boto3 to schedule and receive events
using
    the Amazon EventBridge Scheduler.
    """

    def __init__(
        self,
        scheduler_wrapper: SchedulerWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.eventbridge_scheduler = scheduler_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack: ServiceResource = None
        self.schedule_group_name = None
        self.sns_topic_arn = None
        self.role_arn = None

    def run(self) -> None:
        """
        Runs the scenario.
        """

        print(DASHES)
        print("Welcome to the Amazon EventBridge Scheduler Workflow.")
        print(DASHES)

        print(DASHES)
        self.prepare_application()
        print(DASHES)

        print(DASHES)
        self.create_one_time_schedule()
        print(DASHES)

        print(DASHES)
        self.create_recurring_schedule()
        print(DASHES)

        print(DASHES)
        if q.ask(
            "Do you want to delete all resources created by this workflow? (y/n) ",
            q.is_yesno,
        ):
            self.cleanup()
```

```
        print(DASHES)

        print("Amazon EventBridge Scheduler workflow completed.")

    def prepare_application(self) -> None:
        """
        Prepares the application by prompting the user setup information, deploying
a CloudFormation stack and
        creating a schedule group.
        """
        print("Preparing the application...")
        print(
            "\nThis example creates resources in a CloudFormation stack, including
an SNS topic"
            + "\nthat will be subscribed to the EventBridge Scheduler events. "
            + "\n\nYou will need to confirm the subscription in order to receive
event emails. "
        )

        email_address = q.ask("Enter an email address to use for event
subscriptions: ")
        stack_name = q.ask("Enter a name for the AWS Cloud Formation Stack: ")

        template_file = SchedulerScenario.get_template_as_string()

        parameters = [{"ParameterKey": "email", "ParameterValue": email_address}]

        self.stack = self.deploy_cloudformation_stack(
            stack_name, template_file, parameters
        )
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RoleARN":
                self.role_arn = output.get("OutputValue")
            elif output.get("OutputKey") == "SNStopicARN":
                self.sns_topic_arn = output.get("OutputValue")

        if not self.sns_topic_arn or not self.role_arn:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'sns_topic_arn'={self.sns_topic_arn}, 'role_arn'={self.role_arn}
            """
            logger.error(error_string)
            raise ValueError(error_string)
```

```
        print(f"Stack output RoleARN: {self.role_arn}")
        print(f"Stack output SNStopicARN: a")
        schedule_group_name = "scenario-schedules-group"
        schedule_group_arn = self.eventbridge_scheduler.create_schedule_group(
            schedule_group_name
        )
        print(
            f"Successfully created schedule group '{self.schedule_group_name}':
{schedule_group_arn}."
        )
        self.schedule_group_name = schedule_group_name
        print("Application preparation complete.")

    def create_one_time_schedule(self) -> None:
        """
        Creates a one-time schedule to send an initial event.
        """
        schedule_name = q.ask("Enter a name for the one-time schedule:")

        scheduled_time = datetime.now(timezone.utc) + timedelta(minutes=1)
        formatted_scheduled_time = scheduled_time.strftime("%Y-%m-%dT%H:%M:%S")

        print(
            f"Creating a one-time schedule named '{schedule_name}' "
            + f"\nto send an initial event in 1 minute with a flexible time
window..."
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"at({formatted_scheduled_time})",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"One time scheduled event test from schedule {schedule_name}.",
            delete_after_completion=True,
            use_flexible_time_window=True,
        )
        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
```

```
        print(f"You must confirm your subscription to receive event emails.")
        print(f"One-time schedule '{schedule_name}' created successfully.")

    def create_recurring_schedule(self) -> None:
        """
        Create a recurring schedule to send events at a specified rate in minutes.
        """

        print("Creating a recurring schedule to send events for one hour...")
        schedule_name = q.ask("Enter a name for the recurring schedule: ")
        schedule_rate_in_minutes = q.ask(
            "Enter the desired schedule rate (in minutes): ", q.is_int
        )

        schedule_arn = self.eventbridge_scheduler.create_schedule(
            schedule_name,
            f"rate({schedule_rate_in_minutes} minutes)",
            self.schedule_group_name,
            self.sns_topic_arn,
            self.role_arn,
            f"Recurrent event test from schedule {schedule_name}.",
        )

        print(
            f"Successfully created schedule '{schedule_name}' in schedule group
'scenario-schedules-group': {schedule_arn}."
        )
        print(f"Subscription email will receive an email from this event.")
        print(f"You must confirm your subscription to receive event emails.")

        if q.ask(
            f"Are you ready to delete the '{schedule_name}' schedule? (y/n)",
q.is_yesno
        ):
            self.eventbridge_scheduler.delete_schedule(
                schedule_name, self.schedule_group_name
            )

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str, parameters: [dict[str, str]]
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
```

```
        defined in the associated `cfn_template.yaml` AWS CloudFormation script and
are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :param parameters: The parameters for the CloudFormation stack.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=parameters,
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take a
few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
```

```
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.schedule_group_name:
            schedule_group_name = self.schedule_group_name
            self.schedule_group_name = None
            self.eventbridge_scheduler.delete_schedule_group(schedule_group_name)
            print(f"Successfully deleted schedule group '{schedule_group_name}'.")

        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)
        print("Stack deleted, demo complete.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory, "cfn_template.yaml")
        file = open(template_file_path, "r")
        return file.read()


if __name__ == "__main__":
    demo: SchedulerScenario = None
    try:
        scheduler_wrapper = SchedulerWrapper.from_client()
        cloud_formation_resource = resource("cloudformation")
        demo = SchedulerScenario(scheduler_wrapper, cloud_formation_resource)
        demo.run()

    except Exception as exception:
        logging.exception("Something went wrong with the demo!")
        if demo is not None:
            demo.cleanup()
```

SchedulerWrapper Klasse, die Amazon EventBridge Scheduler-Aktionen umschließt.

```
class SchedulerWrapper:
    def __init__(self, eventbridge_scheduler_client: client):
        self.scheduler_client = eventbridge_scheduler_client

    @classmethod
    def from_client(cls) -> "SchedulerWrapper":
        """
        Creates a SchedulerWrapper instance with a default EventBridge Scheduler
 client.

        :return: An instance of SchedulerWrapper initialized with the default
 EventBridge Scheduler client.
        """
        eventbridge_scheduler_client = boto3.client("scheduler")
        return cls(eventbridge_scheduler_client)


    def create_schedule(
        self,
        name: str,
        schedule_expression: str,
        schedule_group_name: str,
        target_arn: str,
        role_arn: str,
        input: str,
        delete_after_completion: bool = False,
        use_flexible_time_window: bool = False,
    ) -> str:
        """
        Creates a new schedule with the specified parameters.

        :param name: The name of the schedule.
        :param schedule_expression: The expression that defines when the schedule
 runs.
        :param schedule_group_name: The name of the schedule group.
        :param target_arn: The Amazon Resource Name (ARN) of the target.
        :param role_arn: The Amazon Resource Name (ARN) of the execution IAM role.
        :param input: The input for the target.
```

```
        :param delete_after_completion: Whether to delete the schedule after it
completes.
        :param use_flexible_time_window: Whether to use a flexible time window.

        :return The ARN of the created schedule.
        """
        try:
            hours_to_run = 1
            flexible_time_window_minutes = 10
            parameters = {
                "Name": name,
                "ScheduleExpression": schedule_expression,
                "GroupName": schedule_group_name,
                "Target": {"Arn": target_arn, "RoleArn": role_arn, "Input": input},
                "StartDate": datetime.now(timezone.utc),
                "EndDate": datetime.now(timezone.utc) +
timedelta(hours=hours_to_run),
            }

            if delete_after_completion:
                parameters["ActionAfterCompletion"] = "DELETE"

            if use_flexible_time_window:
                parameters["FlexibleTimeWindow"] = {
                    "Mode": "FLEXIBLE",
                    "MaximumWindowInMinutes": flexible_time_window_minutes,
                }
            else:
                parameters["FlexibleTimeWindow"] = {"Mode": "OFF"}

            response = self.scheduler_client.create_schedule(**parameters)
            return response["ScheduleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule '%s' due to a conflict. %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule: %s", err.response["Error"]["Message"]
                )
            raise
```

```
    def delete_schedule(self, name: str, schedule_group_name: str) -> None:
        """
        Deletes the schedule with the specified name and schedule group.

        :param name: The name of the schedule.
        :param schedule_group_name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule(
                Name=name, GroupName=schedule_group_name
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule with ID '%s' because the resource was
not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule: %s", err.response["Error"]["Message"]
                )
                raise


    def create_schedule_group(self, name: str) -> str:
        """
        Creates a new schedule group with the specified name and description.

        :param name: The name of the schedule group.
        :param description: The description of the schedule group.

        :return: The ARN of the created schedule group.
        """
        try:
            response = self.scheduler_client.create_schedule_group(Name=name)
            return response["ScheduleGroupArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ConflictException":
                logger.error(
                    "Failed to create schedule group '%s' due to a conflict. %s",
```

```
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error creating schedule group: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def delete_schedule_group(self, name: str) -> None:
        """
        Deletes the schedule group with the specified name.

        :param name: The name of the schedule group.
        """
        try:
            self.scheduler_client.delete_schedule_group(Name=name)
            logger.info("Schedule group %s deleted successfully.", name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Failed to delete schedule group with ID '%s' because the
resource was not found: %s",
                    name,
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Error deleting schedule group: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateSchedule
  - CreateScheduleGroup

- DeleteSchedule
- DeleteScheduleGroups

Amazon Glacier-Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Glacier Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateVault`

Das folgende Codebeispiel zeigt die Verwendung`CreateVault`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

- API-Details finden Sie CreateVault in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteArchive`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteArchive`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- API-Details finden Sie DeleteArchive in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteVault`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteVault`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
```

```
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

- API-Details finden Sie DeleteVault in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteVaultNotifications`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteVaultNotifications`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
```

```
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- API-Details finden Sie DeleteVaultNotifications in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeJob`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
```

```
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
            return job.status_code
```

- API-Details finden Sie DescribeJob in der AWS API-Referenz zum SDK für Python (Boto3).

`GetJobOutput`

Das folgende Codebeispiel zeigt die Verwendung. `GetJobOutput`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
 response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

- API-Details finden Sie GetJobOutput in der AWS API-Referenz zum SDK für Python (Boto3).

`GetVaultNotifications`

Das folgende Codebeispiel zeigt die Verwendung. `GetVaultNotifications`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.", vault.name)
            raise
        else:
            return notification
```

- API-Details finden Sie GetVaultNotifications in der AWS API-Referenz zum SDK für Python (Boto3).

`InitiateJob`

Das folgende Codebeispiel zeigt die Verwendung. `InitiateJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie eine Tresorinventur ab.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
        When the job completes, you can get the inventory by calling get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
```

```
            raise
        else:
            return job
```

Rufen Sie ein Archiv aus einem Tresor ab.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically complete
        within 3-5 hours. When the job completes, you can get the archive contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job
```

- API-Details finden Sie InitiateJob in der AWS API-Referenz zum SDK für Python (Boto3).

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListJobs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
```

```
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list
```

- API-Details finden Sie ListJobs in der AWS API-Referenz zum SDK für Python (Boto3).

`ListVaults`

Das folgende Codebeispiel zeigt die Verwendung. `ListVaults`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def list_vaults(self):
        """
        Lists vaults for the current account.
```

```
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- API-Details finden Sie ListVaults in der AWS API-Referenz zum SDK für Python (Boto3).

`SetVaultNotifications`

Das folgende Codebeispiel zeigt die Verwendung. `SetVaultNotifications`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
```

```
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name
            )
            raise
        else:
            return notification
```

- API-Details finden Sie SetVaultNotifications in der AWS API-Referenz zum SDK für Python (Boto3).

`UploadArchive`

Das folgende Codebeispiel zeigt die Verwendung. `UploadArchive`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
```

```
            raise
        else:
            return archive
```

- API-Details finden Sie UploadArchive in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Archivieren einer Datei, Erhalten von Benachrichtigungen und Initiieren von Aufträgen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon Glacier-Tresor.
- Konfigurieren des Tresors, um Benachrichtigungen in einem Amazon-SNS-Thema zu veröffentlichen.
- Hochladen einer Archivdatei in den Tresor
- Initiieren eines Auftrags zum Abrufen eines Archivs

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon Glacier-Operationen abschließt.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
```

```
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically complete
        within 3–5 hours. When the job completes, you can get the archive contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
```

```
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
```

```
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name
            )
            raise
        else:
            return notification
```

Rufen Sie Funktionen in der Wrapper-Klasse auf, um einen Tresor zu erstellen und eine Datei hochzuladen. Konfigurieren Sie dann den Tresor so, dass er Benachrichtigungen veröffentlicht, und initiieren Sie einen Auftrag zum Abrufen des Archivs.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification of
                      Amazon S3 Glacier events.
```

```
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateVault

- InitiateJob
- ListJobs
- ListVaults
- SetVaultNotifications
- UploadArchive

Abrufen von Archivinhalten und Löschen des Archivs

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Listen Sie Jobs für einen Amazon Glacier-Tresor auf und rufen Sie den Auftragsstatus ab.
- Abrufen der Ausgabe eines abgeschlossenen Auftrags zum Archivabruf
- Löschen eines Archivs
- Löschen eines Tresors

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die Amazon Glacier-Operationen abschließt.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
```

```
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
```

```
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
```

```
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

Rufen Sie Funktionen der Wrapper-Klasse auf, um Archivinhalte aus einem abgeschlossenen Auftrag abzurufen, und löschen Sie dann das Archiv.

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this script
 "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again later.")
        return
```

```
    retrieval_job = None
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

AWS Glue Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für Python (Boto3)

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError


def hello_glue():
```

```
    """
    Lists the job definitions in your AWS Glue account, using the AWS SDK for Python
 (Boto3).
    """
    try:
        # Create the Glue client
        glue = boto3.client("glue")

        # List the jobs, limiting the results to 10 per page
        paginator = glue.get_paginator("get_jobs")
        response_iterator = paginator.paginate(
            PaginationConfig={"MaxItems": 10, "PageSize": 10}
        )

        # Print the job names
        print("Here are the jobs in your account:")
        for page in response_iterator:
            for job in page["Jobs"]:
                print(f"\t{job['Name']}")

    except ClientError as e:
        print(f"Error: {e}")


if __name__ == "__main__":
    hello_glue()
```

- API-Details finden Sie ListJobs in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.

- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die die im Szenario verwendeten AWS Glue Funktionen umschließt.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
```

```
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.
```

```
        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job that
can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is run
as
                                part of the job. The script defines how the data is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table, output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by the
            # Python ETL script to determine the location of input and output data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

```
    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
```

```
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job and
 use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM) role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output data
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an animated
        spinner.

        :param seconds: The number of seconds to wait.
```

```
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the AWS
Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during job
                           runs.
        :param job_name: The name to give the job definition that is created during
the
```

```
                        scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source} and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
```

```
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source}, transforms
it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only a
few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
```

```
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the first
{lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list of
runs for "
```

```
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
            f"Let's clean up. During this example we created job definition
'{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
```

```
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must grant
 access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
```

```
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Erstellen Sie ein ETL-Skript, das von AWS Glue zum Extrahieren, Transformieren und Laden von Daten während der Auftragsausführung verwendet wird.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
```

```
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases

- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the data
        in the target.

        :param name: The name of the crawler.
```

```
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateCrawler in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung. `CreateJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job that
 can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is run
 as
                                part of the job. The script defines how the data is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API-Details finden Sie CreateJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteCrawler`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteCrawler in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteDatabase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteDatabase in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteTable in der AWS API-Referenz zum SDK für Python (Boto3).

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung. `GetCrawler`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
```

```
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler
```

- API-Details finden Sie GetCrawler in der AWS API-Referenz zum SDK für Python (Boto3).

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung. `GetDatabase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- API-Details finden Sie GetDatabase in der AWS API-Referenz für SDK für Python (Boto3).

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung. `GetJobRun`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- API-Details finden Sie GetJobRun in der AWS API-Referenz zum SDK für Python (Boto3).

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung. `GetJobRuns`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- API-Details finden Sie GetJobRuns in der AWS API-Referenz zum SDK für Python (Boto3).

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung. `GetTables`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- API-Details finden Sie GetTables in der AWS API-Referenz zum SDK für Python (Boto3).

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListJobs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- API-Details finden Sie ListJobs in der AWS API-Referenz zum SDK für Python (Boto3).

StartCrawler

Das folgende Codebeispiel zeigt die Verwendung. `StartCrawler`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie StartCrawler in der AWS API-Referenz zum SDK für Python (Boto3).

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung. `StartJobRun`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table, output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
 tables
                               that describe the source data. This is typically
 created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
```

```
        try:
            # The custom Arguments that are passed to this function are used by the
            # Python ETL script to determine the location of input and output data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]
```

- API-Details finden Sie StartJobRun in der AWS API-Referenz zum SDK für Python (Boto3).

HealthImaging Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo HealthImaging

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von HealthImaging beginnen.

SDK für Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_medical_imaging(medical_imaging_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS HealthImaging
    client and list the data stores in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param medical_imaging_client: A Boto3 AWS HealthImaging Client object.
    """
    print("Hello, Amazon Health Imaging! Let's list some of your data stores:\n")
    try:
        paginator = medical_imaging_client.get_paginator("list_datastores")
        page_iterator = paginator.paginate()
        datastore_summaries = []
        for page in page_iterator:
            datastore_summaries.extend(page["datastoreSummaries"])
        print("\tData Stores:")
        for ds in datastore_summaries:
            print(f"\t\tDatastore: {ds['datastoreName']} ID {ds['datastoreId']}")
    except ClientError as err:
        logger.error(
            "Couldn't list data stores. Here's why: %s: %s",
```

```
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise


if __name__ == "__main__":
    hello_medical_imaging(boto3.client("medical-imaging"))
```

- API-Details finden Sie ListDatastores in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktionen

`CopyImageSet`

Das folgende Codebeispiel zeigt die Verwendung`CopyImageSet`.

SDK für Python (Boto3)

Hilfsfunktion zum Kopieren eines Imagesatzes.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def copy_image_set(
        self,
        datastore_id,
        image_set_id,
        version_id,
        destination_image_set_id=None,
        destination_version_id=None,
        force=False,
        subsets=[],
    ):
```

```
        """
        Copy an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param destination_image_set_id: The ID of the optional destination image
set.
        :param destination_version_id: The ID of the optional destination image set
version.
        :param force: Force the copy.
        :param subsets: The optional subsets to copy. For example:
["12345678901234567890123456789012"].
        :return: The copied image set ID.
        """
        try:
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }
            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }
            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"][
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }
            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't copy image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return copy_results["destinationImageSetProperties"]["imageSetId"]
```

Kopieren Sie einen Imagesatz ohne Ziel.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Kopieren Sie einen Imagesatz mit Ziel.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if destination_image_set_id and destination_version_id:
                copy_image_set_information["destinationImageSet"] = {
                    "imageSetId": destination_image_set_id,
                    "latestVersionId": destination_version_id,
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
```

```
            )
```

Kopieren Sie eine Teilmenge eines Imagesatzes.

```
            copy_image_set_information = {
                "sourceImageSet": {"latestVersionId": version_id}
            }

            if len(subsets) > 0:
                copySubsetsJson = {
                    "SchemaVersion": "1.1",
                    "Study": {"Series": {"imageSetId": {"Instances": {}}}},
                }

                for subset in subsets:
                    copySubsetsJson["Study"]["Series"]["imageSetId"]["Instances"][
                        subset
                    ] = {}

                copy_image_set_information["sourceImageSet"]["DICOMCopies"] = {
                    "copiableAttributes": json.dumps(copySubsetsJson)
                }

            copy_results = self.health_imaging_client.copy_image_set(
                datastoreId=datastore_id,
                sourceImageSetId=image_set_id,
                copyImageSetInformation=copy_image_set_information,
                force=force,
            )
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie CopyImageSet in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`CreateDatastore`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatastore`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def create_datastore(self, name):
        """
        Create a data store.

        :param name: The name of the data store to create.
        :return: The data store ID.
        """
        try:
            data_store =
 self.health_imaging_client.create_datastore(datastoreName=name)
        except ClientError as err:
            logger.error(
                "Couldn't create data store %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreId"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie CreateDatastore in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatastore`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_datastore(self, datastore_id):
        """
        Delete a data store.

        :param datastore_id: The ID of the data store.
        """
        try:
            self.health_imaging_client.delete_datastore(datastoreId=datastore_id)
        except ClientError as err:
            logger.error(
                "Couldn't delete data store %s. Here's why: %s: %s",
                datastore_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie DeleteDatastore in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteImageSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteImageSet`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def delete_image_set(self, datastore_id, image_set_id):
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The delete results.
        """
        try:
            delete_results = self.health_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return delete_results
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie DeleteImageSet in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`GetDICOMImportJob`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_dicom_import_job(self, datastore_id, job_id):
        """
        Get the properties of a DICOM import job.

        :param datastore_id: The ID of the data store.
        :param job_id: The ID of the job.
        :return: The job properties.
        """
        try:
```

```
            job = self.health_imaging_client.get_dicom_import_job(
                jobId=job_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobProperties"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie GetDICOMImportJob in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDatastore`

Das folgende Codebeispiel zeigt die Verwendung`GetDatastore`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client
```

```
    def get_datastore_properties(self, datastore_id):
        """
        Get the properties of a data store.

        :param datastore_id: The ID of the data store.
        :return: The data store properties.
        """
        try:
            data_store = self.health_imaging_client.get_datastore(
                datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get data store %s. Here's why: %s: %s",
                id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return data_store["datastoreProperties"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie GetDatastore in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt die Verwendung`GetImageFrame`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K encoded
 pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.health_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie GetImageFrame in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSet`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSet`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.health_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie GetImageSet in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSetMetadata`.

SDK für Python (Boto3)

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.
```

```
        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """
        try:
            if version_id:
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:

                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            print(image_set_metadata)
            with open(metadata_file, "wb") as f:
                for chunk in
image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
```

```
                )
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
                image_set_metadata =
self.health_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie GetImageSetMetadata in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDICOMImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListDICOMImportJobs`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_dicom_import_jobs(self, datastore_id):
```

```
        """
        List the DICOM import jobs.

        :param datastore_id: The ID of the data store.
        :return: The list of jobs.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_dicom_import_jobs"
            )
            page_iterator = paginator.paginate(datastoreId=datastore_id)
            job_summaries = []
            for page in page_iterator:
                job_summaries.extend(page["jobSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list DICOM import jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job_summaries
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie ListDICOMImportJobs in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListDatastores`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_datastores(self):
        """
        List the data stores.

        :return: The list of data stores.
        """
        try:
            paginator = self.health_imaging_client.get_paginator("list_datastores")
            page_iterator = paginator.paginate()
            datastore_summaries = []
            for page in page_iterator:
                datastore_summaries.extend(page["datastoreSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't list data stores. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return datastore_summaries
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie ListDatastores in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListImageSetVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListImageSetVersions`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_image_set_versions(self, datastore_id, image_set_id):
        """
        List the image set versions.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :return: The list of image set versions.
        """
        try:
            paginator = self.health_imaging_client.get_paginator(
                "list_image_set_versions"
            )
            page_iterator = paginator.paginate(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
            image_set_properties_list = []
            for page in page_iterator:
                image_set_properties_list.extend(page["imageSetPropertiesList"])
        except ClientError as err:
            logger.error(
                "Couldn't list image set versions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return image_set_properties_list
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie ListImageSetVersions in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie ListTagsForResource in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt die Verwendung`SearchImageSets`.

SDK für Python (Boto3)

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def search_image_sets(self, datastore_id, search_filter):
```

```
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.health_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries
```

Anwendungsfall 1: EQUAL-Operator.

```
        search_filter = {
            "filters": [
                {"operator": "EQUAL", "values": [{"DICOMPatientId": patient_id}]}
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(f"Image sets found with EQUAL operator\n{image_sets}")
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudyDate und DICOMStudyTime.

```
        search_filter = {
            "filters": [
                {
                    "operator": "BETWEEN",
                    "values": [
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "19900101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                        {
                            "DICOMStudyDateAndTime": {
                                "DICOMStudyDate": "20230101",
                                "DICOMStudyTime": "000000",
                            }
                        },
                    ],
                }
            ]
        }

        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with BETWEEN operator using DICOMStudyDate and
DICOMStudyTime\n{image_sets}"
        )
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "createdAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "createdAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
```

```
                        },
                    ],
                    "operator": "BETWEEN",
                }
            ]
        }

        recent_image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            f"Image sets found with with BETWEEN operator using createdAt
\n{recent_image_sets}"
        )
```

Anwendungsfall #4: EQUAL-Operator für DICOMSeriesInstanceUID und BETWEEN für updatedAt und Sortierung der Antwort in ASC-Reihenfolge für das updatedAt-Feld.

```
        search_filter = {
            "filters": [
                {
                    "values": [
                        {
                            "updatedAt": datetime.datetime(
                                2021, 8, 4, 14, 49, 54, 429000
                            )
                        },
                        {
                            "updatedAt": datetime.datetime.now()
                            + datetime.timedelta(days=1)
                        },
                    ],
                    "operator": "BETWEEN",
                },
                {
                    "values": [{"DICOMSeriesInstanceUID": series_instance_uid}],
                    "operator": "EQUAL",
                },
            ],
            "sort": {
                "sortOrder": "ASC",
                "sortField": "updatedAt",
            },
        }
```

```
        image_sets = self.search_image_sets(data_store_id, search_filter)
        print(
            "Image sets found with EQUAL operator on DICOMSeriesInstanceUID and
BETWEEN on updatedAt and"
        )
        print(f"sort response in ASC order on updatedAt field\n{image_sets}")
```

Der folgende Code instanziiert das Objekt. MedicalImagingWrapper

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie SearchImageSets in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartDICOMImportJob`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def start_dicom_import_job(
        self, job_name, datastore_id, role_arn, input_s3_uri, output_s3_uri
    ):
        """
        Start a DICOM import job.
```

```
        :param job_name: The name of the job.
        :param datastore_id: The ID of the data store.
        :param role_arn: The Amazon Resource Name (ARN) of the role to use for the
job.
        :param input_s3_uri: The S3 bucket input prefix path containing the DICOM
files.
        :param output_s3_uri: The S3 bucket output prefix path for the result.
        :return: The job ID.
        """
        try:
            job = self.health_imaging_client.start_dicom_import_job(
                jobName=job_name,
                datastoreId=datastore_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_s3_uri,
                outputS3Uri=output_s3_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie StartDICOMImportJob in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie TagResource in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- API-Details finden Sie UntagResource in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UpdateImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`UpdateImageSetMetadata`.

SDK für Python (Boto3)

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def update_image_set_metadata(
        self, datastore_id, image_set_id, version_id, metadata, force=False
    ):
        """
        Update the metadata of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The ID of the image set version.
        :param metadata: The image set metadata as a dictionary.
            For example {"DICOMUpdates": {"updatableAttributes":
            "{\"SchemaVersion\":1.1,\"Patient\":{\"DICOM\":{\"PatientName\":
\"Garcia^Gloria\"}}}"}}
        :param: force: Force the update.
        :return: The updated image set metadata.
        """
        try:
            updated_metadata = self.health_imaging_client.update_image_set_metadata(
                imageSetId=image_set_id,
                datastoreId=datastore_id,
                latestVersionId=version_id,
                updateImageSetMetadataUpdates=metadata,
```

```
                force=force,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update image set metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return updated_metadata
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

Anwendungsfall 1: Fügen Sie ein Attribut ein oder aktualisieren Sie es.

```
            attributes = """{
                    "SchemaVersion": 1.1,
                    "Study": {
                        "DICOM": {
                            "StudyDescription": "CT CHEST"
                        }
                    }
                }"""
            metadata = {"DICOMUpdates": {"updatableAttributes": attributes}}

            self.update_image_set_metadata(
                data_store_id, image_set_id, version_id, metadata, force
            )
```

Anwendungsfall 2: Entfernen Sie ein Attribut.

```
            # Attribute key and value must match the existing attribute.
            attributes = """{
                    "SchemaVersion": 1.1,
```

```
                "Study": {
                    "DICOM": {
                        "StudyDescription": "CT CHEST"
                    }
                }
            }"""
        metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

        self.update_image_set_metadata(
            data_store_id, image_set_id, version_id, metadata, force
        )
```

Anwendungsfall 3: Entfernen Sie eine Instance.

```
        attributes = """{
                "SchemaVersion": 1.1,
                "Study": {
                    "Series": {
                        "1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {
                            "Instances": {

"1.1.1.1.1.1.12345.123456789012.123.12345678901234.1": {}
                            }
                        }
                    }
                }
            }"""
        metadata = {"DICOMUpdates": {"removableAttributes": attributes}}

        self.update_image_set_metadata(
            data_store_id, image_set_id, version_id, metadata, force
        )
```

Anwendungsfall 4: Kehren Sie zu einer früheren Version zurück.

```
        metadata = {"revertToVersionId": "1"}

        self.update_image_set_metadata(
            data_store_id, image_set_id, version_id, metadata, force
        )
```

- API-Details finden Sie UpdateImageSetMetadata in der API-Referenz für AWS SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien

Erste Schritte mit Bildsätzen und Bild-Frames

Das folgende Codebeispiel zeigt, wie DICOM-Dateien importiert und Bildrahmen heruntergeladen werden. HealthImaging

Die Implementierung ist als Befehlszeilenanwendung strukturiert.

- Richten Sie Ressourcen für einen DICOM-Import ein.
- Importieren Sie DICOM-Dateien in einen Datenspeicher.
- Rufen Sie die Bildsatz-IDs für den Importauftrag ab.
- Rufen Sie die Bild-Frame-IDs für den Importauftrag ab.
- Laden Sie die Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.
- Bereinigen Sie die Ressourcen.

SDK für Python (Boto3)

Erstellen Sie einen CloudFormation Stapel mit den erforderlichen Ressourcen.

```
    def deploy(self):
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `setup.yaml` AWS CloudFormation script and are
deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.
        """
```

```
        print("\t\tLet's deploy the stack for resource creation.")
        stack_name = q.ask("\t\tEnter a name for the stack: ", q.non_empty)

        data_store_name = q.ask(
            "\t\tEnter a name for the Health Imaging Data Store: ", q.non_empty
        )

        account_id = boto3.client("sts").get_caller_identity()["Account"]

        with open(
            "../../../../scenarios/features/healthimaging_image_sets/resources/
cfn_template.yaml"
        ) as setup_file:
            setup_template = setup_file.read()
        print(f"\t\tCreating {stack_name}.")
        stack = self.cf_resource.create_stack(
            StackName=stack_name,
            TemplateBody=setup_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
            Parameters=[
                {
                    "ParameterKey": "datastoreName",
                    "ParameterValue": data_store_name,
                },
                {
                    "ParameterKey": "userAccountID",
                    "ParameterValue": account_id,
                },
            ],
        )
        print("\t\tWaiting for stack to deploy. This typically takes a minute or
 two.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_create_complete")
        waiter.wait(StackName=stack.name)
        stack.load()
        print(f"\t\tStack status: {stack.stack_status}")

        outputs_dictionary = {
            output["OutputKey"]: output["OutputValue"] for output in stack.outputs
        }
        self.input_bucket_name = outputs_dictionary["BucketName"]
        self.output_bucket_name = outputs_dictionary["BucketName"]
        self.role_arn = outputs_dictionary["RoleArn"]
```

```
        self.data_store_id = outputs_dictionary["DatastoreID"]
        return stack
```

Kopieren Sie DICOM-Dateien in den Amazon-S3-Import-Bucket.

```
    def copy_single_object(self, key, source_bucket, target_bucket,
target_directory):
        """
        Copies a single object from a source to a target bucket.

        :param key: The key of the object to copy.
        :param source_bucket: The source bucket for the copy.
        :param target_bucket: The target bucket for the copy.
        :param target_directory: The target directory for the copy.
        """
        new_key = target_directory + "/" + key
        copy_source = {"Bucket": source_bucket, "Key": key}
        self.s3_client.copy_object(
            CopySource=copy_source, Bucket=target_bucket, Key=new_key
        )
        print(f"\n\t\tCopying {key}.")

    def copy_images(
        self, source_bucket, source_directory, target_bucket, target_directory
    ):
        """
        Copies the images from the source to the target bucket using multiple
threads.

        :param source_bucket: The source bucket for the images.
        :param source_directory: Directory within the source bucket.
        :param target_bucket: The target bucket for the images.
        :param target_directory: Directory within the target bucket.
        """

        # Get list of all objects in source bucket.
        list_response = self.s3_client.list_objects_v2(
            Bucket=source_bucket, Prefix=source_directory
        )
        objs = list_response["Contents"]
        keys = [obj["Key"] for obj in objs]
```

```
        # Copy the objects in the bucket.
        for key in keys:
            self.copy_single_object(key, source_bucket, target_bucket,
 target_directory)

        print("\t\tDone copying all objects.")
```

Importieren Sie die DICOM-Dateien in den Amazon-S3-Datenspeicher.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def start_dicom_import_job(
        self,
        data_store_id,
        input_bucket_name,
        input_directory,
        output_bucket_name,
        output_directory,
        role_arn,
    ):
        """
        Routine which starts a HealthImaging import job.
```

```
        :param data_store_id: The HealthImaging data store ID.
        :param input_bucket_name: The name of the Amazon S3 bucket containing the
 DICOM files.
        :param input_directory: The directory in the S3 bucket containing the DICOM
 files.
        :param output_bucket_name: The name of the S3 bucket for the output.
        :param output_directory: The directory in the S3 bucket to store the output.
        :param role_arn: The ARN of the IAM role with permissions for the import.
        :return: The job ID of the import.
        """

        input_uri = f"s3://{input_bucket_name}/{input_directory}/"
        output_uri = f"s3://{output_bucket_name}/{output_directory}/"
        try:
            job = self.medical_imaging_client.start_dicom_import_job(
                jobName="examplejob",
                datastoreId=data_store_id,
                dataAccessRoleArn=role_arn,
                inputS3Uri=input_uri,
                outputS3Uri=output_uri,
            )
        except ClientError as err:
            logger.error(
                "Couldn't start DICOM import job. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return job["jobId"]
```

Rufen Sie Bildsätze ab, die durch den DICOM-Importauftrag erstellt wurden.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
```

```
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_sets_for_dicom_import_job(self, datastore_id, import_job_id):
        """
        Retrieves the image sets created for an import job.

        :param datastore_id: The HealthImaging data store ID
        :param import_job_id: The import job ID
        :return: List of image set IDs
        """

        import_job = self.medical_imaging_client.get_dicom_import_job(
            datastoreId=datastore_id, jobId=import_job_id
        )

        output_uri = import_job["jobProperties"]["outputS3Uri"]

        bucket = output_uri.split("/")[2]
        key = "/".join(output_uri.split("/")[3:])

        # Try to get the manifest.
        retries = 3
        while retries > 0:
            try:
                obj = self.s3_client.get_object(
                    Bucket=bucket, Key=key + "job-output-manifest.json"
                )
                body = obj["Body"]
                break
            except ClientError as error:
                retries = retries - 1
                time.sleep(3)
        try:
```

```
            data = json.load(body)
            expression =
jmespath.compile("jobSummary.imageSetsSummary[].imageSetId")
            image_sets = expression.search(data)
        except json.decoder.JSONDecodeError as error:
            image_sets = import_job["jobProperties"]

        return image_sets


    def get_image_set(self, datastore_id, image_set_id, version_id=None):
        """
        Get the properties of an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The optional version of the image set.
        :return: The image set properties.
        """
        try:
            if version_id:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set = self.medical_imaging_client.get_image_set(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
        except ClientError as err:
            logger.error(
                "Couldn't get image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return image_set
```

Rufen Sie Bild-Frame-Informationen für Bildsätze ab.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_image_frames_for_image_set(self, datastore_id, image_set_id,
 out_directory):
        """
        Get the image frames for an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param out_directory: The directory to save the file.
        :return: The image frames.
        """
        image_frames = []
        file_name = os.path.join(out_directory,
 f"{image_set_id}_metadata.json.gzip")
        file_name = file_name.replace("/", "\\\\")
        self.get_image_set_metadata(file_name, datastore_id, image_set_id)
        try:
            with gzip.open(file_name, "rb") as f_in:
                doc = json.load(f_in)
            instances = jmespath.search("Study.Series.*.Instances[].*[]", doc)
            for instance in instances:
                rescale_slope = jmespath.search("DICOM.RescaleSlope", instance)
                rescale_intercept = jmespath.search("DICOM.RescaleIntercept",
 instance)
```

```
                image_frames_json = jmespath.search("ImageFrames[][]", instance)
                for image_frame in image_frames_json:
                    checksum_json = jmespath.search(
                        "max_by(PixelDataChecksumFromBaseToFullResolution, &Width)",
                        image_frame,
                    )
                    image_frame_info = {
                        "imageSetId": image_set_id,
                        "imageFrameId": image_frame["ID"],
                        "rescaleIntercept": rescale_intercept,
                        "rescaleSlope": rescale_slope,
                        "minPixelValue": image_frame["MinPixelValue"],
                        "maxPixelValue": image_frame["MaxPixelValue"],
                        "fullResolutionChecksum": checksum_json["Checksum"],
                    }
                    image_frames.append(image_frame_info)
            return image_frames
        except TypeError:
            return {}
        except ClientError as err:
            logger.error(
                "Couldn't get image frames for image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return image_frames


    def get_image_set_metadata(
        self, metadata_file, datastore_id, image_set_id, version_id=None
    ):
        """
        Get the metadata of an image set.

        :param metadata_file: The file to store the JSON gzipped metadata.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param version_id: The version of the image set.
        """

        try:
            if version_id:
```

```
                image_set_metadata =
 self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id,
                    datastoreId=datastore_id,
                    versionId=version_id,
                )
            else:
                image_set_metadata =
 self.medical_imaging_client.get_image_set_metadata(
                    imageSetId=image_set_id, datastoreId=datastore_id
                )
            with open(metadata_file, "wb") as f:
                for chunk in
 image_set_metadata["imageSetMetadataBlob"].iter_chunks():
                    if chunk:
                        f.write(chunk)

        except ClientError as err:
            logger.error(
                "Couldn't get image metadata. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Laden Sie Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.

```
class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
```

```
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def get_pixel_data(
        self, file_path_to_write, datastore_id, image_set_id, image_frame_id
    ):
        """
        Get an image frame's pixel data.

        :param file_path_to_write: The path to write the image frame's HTJ2K encoded
pixel data.
        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        :param image_frame_id: The ID of the image frame.
        """
        try:
            image_frame = self.medical_imaging_client.get_image_frame(
                datastoreId=datastore_id,
                imageSetId=image_set_id,
                imageFrameInformation={"imageFrameId": image_frame_id},
            )
            with open(file_path_to_write, "wb") as f:
                for chunk in image_frame["imageFrameBlob"].iter_chunks():
                    f.write(chunk)
        except ClientError as err:
            logger.error(
                "Couldn't get image frame. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def download_decode_and_check_image_frames(
        self, data_store_id, image_frames, out_directory
    ):
        """
        Downloads image frames, decodes them, and uses the checksum to validate
        the decoded images.

        :param data_store_id: The HealthImaging data store ID.
```

```
        :param image_frames: A list of dicts containing image frame information.
        :param out_directory: A directory for the downloaded images.
        :return: True if the function succeeded; otherwise, False.
        """
        total_result = True
        for image_frame in image_frames:
            image_file_path = f"{out_directory}/
image_{image_frame['imageFrameId']}.jph"
            self.get_pixel_data(
                image_file_path,
                data_store_id,
                image_frame["imageSetId"],
                image_frame["imageFrameId"],
            )

            image_array = self.jph_image_to_opj_bitmap(image_file_path)
            crc32_checksum = image_frame["fullResolutionChecksum"]
            # Verify checksum.
            crc32_calculated = zlib.crc32(image_array)
            image_result = crc32_checksum == crc32_calculated
            print(
                f"\t\tImage checksum verified for {image_frame['imageFrameId']}:
 {image_result }"
            )
            total_result = total_result and image_result
        return total_result

    @staticmethod
    def jph_image_to_opj_bitmap(jph_file):
        """
        Decode the image to a bitmap using an OPENJPEG library.
        :param jph_file: The file to decode.
        :return: The decoded bitmap as an array.
        """
        # Use format 2 for the JPH file.
        params = openjpeg.utils.get_parameters(jph_file, 2)
        print(f"\n\t\tImage parameters for {jph_file}: \n\t\t{params}")

        image_array = openjpeg.utils.decode(jph_file, 2)

        return image_array
```

Bereinigen Sie die Ressourcen.

```
    def destroy(self, stack):
        """
        Destroys the resources managed by the CloudFormation stack, and the
 CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """

        print(f"\t\tCleaning up resources and {stack.name}.")
        data_store_id = None
        for oput in stack.outputs:
            if oput["OutputKey"] == "DatastoreID":
                data_store_id = oput["OutputValue"]
        if data_store_id is not None:
            print(f"\t\tDeleting image sets in data store {data_store_id}.")
            image_sets = self.medical_imaging_wrapper.search_image_sets(
                data_store_id, {}
            )
            image_set_ids = [image_set["imageSetId"] for image_set in image_sets]

            for image_set_id in image_set_ids:
                self.medical_imaging_wrapper.delete_image_set(
                    data_store_id, image_set_id
                )
                print(f"\t\tDeleted image set with id : {image_set_id}")

        print(f"\t\tDeleting {stack.name}.")
        stack.delete()
        print("\t\tWaiting for stack removal. This may take a few minutes.")
        waiter = self.cf_resource.meta.client.get_waiter("stack_delete_complete")
        waiter.wait(StackName=stack.name)
        print("\t\tStack delete complete.")




class MedicalImagingWrapper:
    """Encapsulates AWS HealthImaging functionality."""

    def __init__(self, medical_imaging_client, s3_client):
        """
```

```
        :param medical_imaging_client: A Boto3 Amazon MedicalImaging client.
        :param s3_client: A Boto3 S3 client.
        """
        self.medical_imaging_client = medical_imaging_client
        self.s3_client = s3_client

    @classmethod
    def from_client(cls):
        medical_imaging_client = boto3.client("medical-imaging")
        s3_client = boto3.client("s3")
        return cls(medical_imaging_client, s3_client)


    def search_image_sets(self, datastore_id, search_filter):
        """
        Search for image sets.

        :param datastore_id: The ID of the data store.
        :param search_filter: The search filter.
            For example: {"filters" : [{ "operator": "EQUAL", "values":
[{"DICOMPatientId": "3524578"}]}]}.
        :return: The list of image sets.
        """
        try:
            paginator =
self.medical_imaging_client.get_paginator("search_image_sets")
            page_iterator = paginator.paginate(
                datastoreId=datastore_id, searchCriteria=search_filter
            )
            metadata_summaries = []
            for page in page_iterator:
                metadata_summaries.extend(page["imageSetsMetadataSummaries"])
        except ClientError as err:
            logger.error(
                "Couldn't search image sets. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return metadata_summaries


    def delete_image_set(self, datastore_id, image_set_id):
```

```
        """
        Delete an image set.

        :param datastore_id: The ID of the data store.
        :param image_set_id: The ID of the image set.
        """
        try:
            delete_results = self.medical_imaging_client.delete_image_set(
                imageSetId=image_set_id, datastoreId=datastore_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete image set. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Taggen eines Datenspeichers

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Datenspeicher taggen.

SDK für Python (Boto3)

So taggen Sie einen Datenspeicher.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.tag_resource(data_store_arn, {"Deployment":
 "Development"})
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

So listen Sie Tags für einen Datenspeicher auf.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"
```

```
    medical_imaging_wrapper.list_tags_for_resource(data_store_arn)
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return tags["tags"]
```

So entfernen Sie Tags von einem Datenspeicher.

```
    a_data_store_arn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012"

    medical_imaging_wrapper.untag_resource(data_store_arn, ["Deployment"])
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
    client = boto3.client("medical-imaging")
    medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Taggen eines Imagesatzes

Das folgende Codebeispiel zeigt, wie Sie einen HealthImaging Bildsatz taggen.

SDK für Python (Boto3)

So taggen Sie einen Imagesatz.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.tag_resource(image_set_arn, {"Deployment":
 "Development"})
```

Die Hilfsfunktion zum Taggen einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def tag_resource(self, resource_arn, tags):
        """
        Tag a resource.

        :param resource_arn: The ARN of the resource.
        :param tags: The tags to apply.
        """
        try:
            self.health_imaging_client.tag_resource(resourceArn=resource_arn,
 tags=tags)
        except ClientError as err:
```

```
            logger.error(
                "Couldn't tag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

So listen Sie Tags für einen Imagesatz auf.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.list_tags_for_resource(image_set_arn)
```

Die Hilfsfunktion zum Auflisten von Tags einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def list_tags_for_resource(self, resource_arn):
        """
        List the tags for a resource.

        :param resource_arn: The ARN of the resource.
        :return: The list of tags.
        """
        try:
            tags = self.health_imaging_client.list_tags_for_resource(
                resourceArn=resource_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't list tags for resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return tags["tags"]
```

So entfernen Sie Tags von einem Imagesatz.

```
    an_image_set_arn = (
        "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/"
        "imageset/12345678901234567890123456789012"
    )

    medical_imaging_wrapper.untag_resource(image_set_arn, ["Deployment"])
```

Die Hilfsfunktion zum Entfernen von Tags von einer Ressource.

```
class MedicalImagingWrapper:
    def __init__(self, health_imaging_client):
        self.health_imaging_client = health_imaging_client


    def untag_resource(self, resource_arn, tag_keys):
        """
        Untag a resource.

        :param resource_arn: The ARN of the resource.
        :param tag_keys: The tag keys to remove.
        """
        try:
            self.health_imaging_client.untag_resource(
                resourceArn=resource_arn, tagKeys=tag_keys
            )
        except ClientError as err:
            logger.error(
                "Couldn't untag resource. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das MedicalImagingWrapper Objekt.

```
client = boto3.client("medical-imaging")
medical_imaging_wrapper = MedicalImagingWrapper(client)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

HealthLake Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren HealthLake.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Erstellen Sie einen Fhir-Datenspeicher`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFHIRDatastore`

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def create_fhir_datastore(
        self,
        datastore_name: str,
        sse_configuration: dict[str, any] = None,
        identity_provider_configuration: dict[str, any] = None,
    ) -> dict[str, str]:
        """
        Creates a new HealthLake data store.
        When creating a SMART on FHIR data store, the following parameters are
required:
        - sse_configuration: The server-side encryption configuration for a SMART on
FHIR-enabled data store.
        - identity_provider_configuration: The identity provider configuration for a
SMART on FHIR-enabled data store.

        :param datastore_name: The name of the data store.
        :param sse_configuration: The server-side encryption configuration for a
SMART on FHIR-enabled data store.
        :param identity_provider_configuration: The identity provider configuration
for a SMART on FHIR-enabled data store.
        :return: A dictionary containing the data store information.
        """
        try:
```

```
            parameters = {"DatastoreName": datastore_name, "DatastoreTypeVersion":
 "R4"}
            if (
                sse_configuration is not None
                and identity_provider_configuration is not None
            ):
                # Creating a SMART on FHIR-enabled data store
                parameters["SseConfiguration"] = sse_configuration
                parameters[
                    "IdentityProviderConfiguration"
                ] = identity_provider_configuration

            response = self.health_lake_client.create_fhir_datastore(**parameters)
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't create data store %s. Here's why %s",
                datastore_name,
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code zeigt ein Beispiel für Parameter für einen SMART FHIR-enabled HealthLake on-Datenspeicher.

```
            sse_configuration = {
                "KmsEncryptionConfig": {"CmkType": "AWS_OWNED_KMS_KEY"}
            }
            # TODO: Update the metadata to match your environment.
            metadata = {
                "issuer": "https://ehr.example.com",
                "jwks_uri": "https://ehr.example.com/.well-known/jwks.json",
                "authorization_endpoint": "https://ehr.example.com/auth/authorize",
                "token_endpoint": "https://ehr.token.com/auth/token",
                "token_endpoint_auth_methods_supported": [
                    "client_secret_basic",
                    "foo",
                ],
                "grant_types_supported": ["client_credential", "foo"],
                "registration_endpoint": "https://ehr.example.com/auth/register",
                "scopes_supported": ["openId", "profile", "launch"],
```

```
                "response_types_supported": ["code"],
                "management_endpoint": "https://ehr.example.com/user/manage",
                "introspection_endpoint": "https://ehr.example.com/user/introspect",
                "revocation_endpoint": "https://ehr.example.com/user/revoke",
                "code_challenge_methods_supported": ["S256"],
                "capabilities": [
                    "launch-ehr",
                    "sso-openid-connect",
                    "client-public",
                ],
            }
            # TODO: Update the IdpLambdaArn.
            identity_provider_configuration = {
                "AuthorizationStrategy": "SMART_ON_FHIR_V1",
                "FineGrainedAuthorizationEnabled": True,
                "IdpLambdaArn": "arn:aws:lambda:your-region:your-account-
id:function:your-lambda-name",
                "Metadata": json.dumps(metadata),
            }
            data_store = self.create_fhir_datastore(
                datastore_name, sse_configuration, identity_provider_configuration
            )
```

- Weitere API-Informationen finden Sie unter CreateFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Fhir-Datenspeicher

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFHIRDatastore`

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
```

```
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def delete_fhir_datastore(self, datastore_id: str) -> None:
        """
        Deletes a HealthLake data store.
        :param datastore_id: The data store ID.
        """
        try:
            self.health_lake_client.delete_fhir_datastore(DatastoreId=datastore_id)
        except ClientError as err:
            logger.exception(
                "Couldn't delete data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter DeleteFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DescribeFHIRDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DescribeFHIRDatastore`.

SDK für Python (Boto3)

```
    @classmethod
```

```
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_datastore(self, datastore_id: str) -> dict[str, any]:
        """
        Describes a HealthLake data store.
        :param datastore_id: The data store ID.
        :return: The data store description.
        """
        try:
            response = self.health_lake_client.describe_fhir_datastore(
                DatastoreId=datastore_id
            )
            return response["DatastoreProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe data store with ID %s. Here's why %s",
                datastore_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter DescribeFHIRDatastore in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DescribeFHIRExportJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeFHIRExportJob`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_export_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake export job.
        :param datastore_id: The data store ID.
        :param job_id: The export job ID.
        :return: The export job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_export_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ExportJobProperties"]
        except ClientError as err:
            logger.exception(
                "Couldn't describe export job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeFHIRExportJob in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DescribeFHIRImportJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeFHIRImportJob`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def describe_fhir_import_job(
        self, datastore_id: str, job_id: str
    ) -> dict[str, any]:
        """
        Describes a HealthLake import job.
        :param datastore_id: The data store ID.
        :param job_id: The import job ID.
        :return: The import job description.
        """
        try:
            response = self.health_lake_client.describe_fhir_import_job(
                DatastoreId=datastore_id, JobId=job_id
            )
            return response["ImportJobProperties"]
        except ClientError as err:
```

```
            logger.exception(
                "Couldn't describe import job with ID %s. Here's why %s",
                job_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeFHIRImportJob in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListFHIRDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListFHIRDatastores`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_datastores(self) -> list[dict[str, any]]:
        """
        Lists all HealthLake data stores.
        :return: A list of data store descriptions.
        """
        try:
```

```
            next_token = None
            datastores = []

            # Loop through paginated results.
            while True:
                parameters = {}
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
 self.health_lake_client.list_fhir_datastores(**parameters)
                datastores.extend(response["DatastorePropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break

            return datastores
        except ClientError as err:
            logger.exception(
                "Couldn't list data stores. Here's why %s", err.response["Error"]
["Message"]
            )
            raise
```

- Weitere API-Informationen finden Sie unter ListFHIRDatastores in der API-Referenz zum AWS -SDK für Python (Boto3).

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListFHIRExportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListFHIRExportJobs`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
```

```
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_fhir_export_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake export jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The export job name.
        :param job_status: The export job status.
        :param submitted_before: The export job submitted before the specified date.
        :param submitted_after: The export job submitted after the specified date.
        :return: A list of export jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
```

```
                response =
self.health_lake_client.list_fhir_export_jobs(**parameters)
                jobs.extend(response["ExportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
            logger.exception(
                "Couldn't list export jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListFHIRExportJobs in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListFHIRImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListFHIRImportJobs`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)
```

```
    def list_fhir_import_jobs(
        self,
        datastore_id: str,
        job_name: str = None,
        job_status: str = None,
        submitted_before: datetime = None,
        submitted_after: datetime = None,
    ) -> list[dict[str, any]]:
        """
        Lists HealthLake import jobs satisfying the conditions.
        :param datastore_id: The data store ID.
        :param job_name: The import job name.
        :param job_status: The import job status.
        :param submitted_before: The import job submitted before the specified date.
        :param submitted_after: The import job submitted after the specified date.
        :return: A list of import jobs.
        """
        try:
            parameters = {"DatastoreId": datastore_id}
            if job_name is not None:
                parameters["JobName"] = job_name
            if job_status is not None:
                parameters["JobStatus"] = job_status
            if submitted_before is not None:
                parameters["SubmittedBefore"] = submitted_before
            if submitted_after is not None:
                parameters["SubmittedAfter"] = submitted_after
            next_token = None
            jobs = []
            # Loop through paginated results.
            while True:
                if next_token is not None:
                    parameters["NextToken"] = next_token
                response =
self.health_lake_client.list_fhir_import_jobs(**parameters)
                jobs.extend(response["ImportJobPropertiesList"])
                if "NextToken" in response:
                    next_token = response["NextToken"]
                else:
                    break
            return jobs
        except ClientError as err:
```

```
            logger.exception(
                "Couldn't list import jobs. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListFHIRImportJobs in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def list_tags_for_resource(self, resource_arn: str) -> dict[str, str]:
        """
        Lists the tags for a HealthLake resource.
        :param resource_arn: The resource ARN.
        :return: The tags for the resource.
        """
        try:
```

```
            response = self.health_lake_client.list_tags_for_resource(
                ResourceARN=resource_arn
            )
            return response["Tags"]
        except ClientError as err:
            logger.exception(
                "Couldn't list tags for resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListTagsForResource in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartFHIRExportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartFHIRExportJob`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_export_job(
```

```
        self,
        job_name: str,
        datastore_id: str,
        output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake export job.
        :param job_name: The export job name.
        :param datastore_id: The data store ID.
        :param output_s3_uri: The output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The export job.
        """
        try:
            response = self.health_lake_client.start_fhir_export_job(
                OutputDataConfig={
                    "S3Configuration": {"S3Uri": output_s3_uri, "KmsKeyId":
kms_key_id}
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
                JobName=job_name,
            )

            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start export job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie StartFHIRExportJob in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

StartFHIRImportJob

Das folgende Codebeispiel zeigt die VerwendungStartFHIRImportJob.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def start_fhir_import_job(
        self,
        job_name: str,
        datastore_id: str,
        input_s3_uri: str,
        job_output_s3_uri: str,
        kms_key_id: str,
        data_access_role_arn: str,
    ) -> dict[str, str]:
        """
        Starts a HealthLake import job.
        :param job_name: The import job name.
        :param datastore_id: The data store ID.
        :param input_s3_uri: The input S3 URI.
        :param job_output_s3_uri: The job output S3 URI.
        :param kms_key_id: The KMS key ID associated with the output S3 bucket.
        :param data_access_role_arn: The data access role ARN.
        :return: The import job.
        """
```

```
        try:
            response = self.health_lake_client.start_fhir_import_job(
                JobName=job_name,
                InputDataConfig={"S3Uri": input_s3_uri},
                JobOutputDataConfig={
                    "S3Configuration": {
                        "S3Uri": job_output_s3_uri,
                        "KmsKeyId": kms_key_id,
                    }
                },
                DataAccessRoleArn=data_access_role_arn,
                DatastoreId=datastore_id,
            )
            return response
        except ClientError as err:
            logger.exception(
                "Couldn't start import job. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie StartFHIRImportJob in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.
```

```
        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def tag_resource(self, resource_arn: str, tags: list[dict[str, str]]) -> None:
        """
        Tags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tags: The tags to add to the resource.
        """
        try:
            self.health_lake_client.tag_resource(ResourceARN=resource_arn,
Tags=tags)
        except ClientError as err:
            logger.exception(
                "Couldn't tag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie TagResource in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für Python (Boto3)

```
    @classmethod
    def from_client(cls) -> "HealthLakeWrapper":
```

```
        """
        Creates a HealthLakeWrapper instance with a default AWS HealthLake client.

        :return: An instance of HealthLakeWrapper initialized with the default
HealthLake client.
        """
        health_lake_client = boto3.client("healthlake")
        return cls(health_lake_client)


    def untag_resource(self, resource_arn: str, tag_keys: list[str]) -> None:
        """
        Untags a HealthLake resource.
        :param resource_arn: The resource ARN.
        :param tag_keys: The tag keys to remove from the resource.
        """
        try:
            self.health_lake_client.untag_resource(
                ResourceARN=resource_arn, TagKeys=tag_keys
            )
        except ClientError as err:
            logger.exception(
                "Couldn't untag resource %s. Here's why %s",
                resource_arn,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UntagResource in der AWS API-Referenz für SDK für Python (Boto3).

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def main():
    """
```

```
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ListPolicies in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.

- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
```

```
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
```

```
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
```

```
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume "
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
```

```
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
```

```
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
```

```
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn, "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Fügen Sie einer Rolle mithilfe des Boto3-Richtlinienobjekts eine Richtlinie an.

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

Fügen Sie einer Rolle mithilfe des Boto3-Rollenobjekts eine Richtlinie an.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- API-Details finden Sie AttachRolePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`AttachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `AttachUserPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def attach_policy(user_name, policy_arn):
    """
```

```
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- API-Details finden Sie AttachUserPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAccessKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
```

```
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- API-Details finden Sie CreateAccessKey in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAccountAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
```

```
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- API-Details finden Sie CreateAccountAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `CreateInstanceProfile`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel werden eine Richtlinie, eine Rolle und ein Instance-Profil erstellt und alle miteinander verknüpft.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.
```

```
        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
```

```
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
```

```
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

- API-Details finden Sie CreateInstanceProfile in der AWS API-Referenz zum SDK für Python (Boto3).

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `CreatePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
```

```
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- API-Details finden Sie CreatePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`CreatePolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `CreatePolicyVersion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- API-Details finden Sie CreatePolicyVersion in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung. `CreateRole`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- API-Details finden Sie CreateRole in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung. `CreateServiceLinkedRole`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
    else:
        return role
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung. `CreateUser`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- API-Details finden Sie CreateUser in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAccessKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- API-Details finden Sie DeleteAccessKey in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAccountAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- API-Details finden Sie DeleteAccountAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteInstanceProfile`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteInstanceProfile`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird die Rolle aus dem Instance-Profil entfernt, alle mit der Rolle verknüpften Richtlinien getrennt und alle Ressourcen gelöscht.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
```

```
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
```

- API-Details finden Sie DeleteInstanceProfile in der AWS API-Referenz zum SDK für Python (Boto3).

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `DeletePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- API-Details finden Sie DeletePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteRole`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- API-Details finden Sie DeleteRole in der AWS API-Referenz für SDK für Python (Boto3).

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteUser`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
```

```
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- API-Details finden Sie DeleteUser in der AWS API-Referenz zum SDK für Python (Boto3).

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `DetachRolePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Trennen Sie eine Richtlinie von einer Rolle mithilfe des Boto3-Richtlinienobjekts.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

Trennen Sie eine Richtlinie von einer Rolle mithilfe des Boto3-Rollenobjekts.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- API-Details finden Sie DetachRolePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`DetachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `DetachUserPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
```

```
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- API-Details finden Sie DetachUserPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`GenerateCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `GenerateCredentialReport`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
```

```
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
        raise
    else:
        return response
```

- API-Details finden Sie GenerateCredentialReport in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung. `GetAccessKeyLastUsed`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
```

```
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- API-Details finden Sie GetAccessKeyLastUsed in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAccountAuthorizationDetails`

Das folgende Codebeispiel zeigt die Verwendung. `GetAccountAuthorizationDetails`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
```

```
        raise
    else:
        return account_details
```

- API-Details finden Sie GetAccountAuthorizationDetails in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `GetAccountPasswordPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
 {pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention: {pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
 {pw_policy.require_lowercase_characters}"
```

```
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
 {pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAccountSummary`

Das folgende Codebeispiel zeigt die Verwendung. `GetAccountSummary`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
```

```
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- API-Details finden Sie GetAccountSummary in der AWS API-Referenz zum SDK für Python (Boto3).

`GetCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung. `GetCredentialReport`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- API-Details finden Sie GetCredentialReport in der AWS API-Referenz zum SDK für Python (Boto3).

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `GetPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- API-Details finden Sie GetPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`GetPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung. `GetPolicyVersion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- API-Details finden Sie GetPolicyVersion in der AWS API-Referenz zum SDK für Python (Boto3).

GetRole

Das folgende Codebeispiel zeigt die Verwendung. `GetRole`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- API-Details finden Sie GetRole in der AWS API-Referenz zum SDK für Python (Boto3).

ListAccessKeys

Das folgende Codebeispiel zeigt die Verwendung. `ListAccessKeys`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- API-Details finden Sie ListAccessKeys in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung. `ListAccountAliases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- API-Details finden Sie ListAccountAliases in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung. `ListAttachedRolePolicies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.
```

```
    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS API-Referenz zum SDK für Python (Boto3).

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung. `ListGroups`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- API-Details finden Sie ListGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung. `ListPolicies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- API-Details finden Sie ListPolicies in der AWS API-Referenz zum SDK für Python (Boto3).

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung. `ListRolePolicies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- API-Details finden Sie ListRolePolicies in der AWS API-Referenz zum SDK für Python (Boto3).

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung. `ListRoles`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- API-Details finden Sie ListRoles in der AWS API-Referenz zum SDK für Python (Boto3).

`Listet SAML-Anbieter auf`

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der API-Referenz zum AWS -SDK für Python (Boto3).

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
```

```
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- API-Details finden Sie ListUsers in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateAccessKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
```

```
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate", key_id
        )
        raise
```

- API-Details finden Sie UpdateAccessKey in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateUser`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
```

```
    return user
```

- API-Details finden Sie UpdateUser in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Verwalten eines ausfallsicheren Services

Das folgende Codebeispiel zeigt, wie Sie einen Webservice mit Load Balancing erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Gruppe von Amazon EC2 Auto Scaling, um Amazon Elastic Compute Cloud (Amazon EC2)-Instances basierend auf einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Verarbeiten und verteilen Sie HTTP-Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2-Instance einen Python-Webserver aus, um HTTP-Anfragen zu verarbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Reaktion des Webservers auf Anfragen und Zustandsprüfungen, indem Sie die Parameter aktualisieren AWS Systems Manager .

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
```

```
    """
    Manages the deployment, demonstration, and destruction of resources for the
resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example, such
as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
```

```
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
```

```
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port is
 open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access from
 this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging from
 your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
```

```
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer endpoint.
Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as sending
requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer endpoint.")
                endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
```

```
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check to
 update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
 failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
 failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a non-
existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
```

```
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance replacement.")
        self.demo_choices()
```

```
        logging.info("Simulating a complete failure of the recommendation service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up manually
to avoid unexpected charges."
```

```
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM policies
 and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
```

```
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto-Scaling- und Amazon-EC2-Aktionen beinhaltet.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
```

```
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are created
by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI that
is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```
    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
```

```
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to attach
 to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role {role_name}.")
```

```
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is assumed
by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as AmazonSSMManagedInstanceCore
to grant
                                     use of Systems Manager to send commands to the
instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
```

```
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response = self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance {instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
```

```
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance {instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the profile
is
        replaced, the instance is rebooted to ensure that it uses the new profile.
When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to associate
with
                                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                            instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
```

```
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or is
not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
```

```
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
```

```
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.
The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a permissions
policy
                                     to create and attach to the instance profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
```

```
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
```

```
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones
```

```
    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with availability
zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.
```

```
        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) -> None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)
```

```
            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
```

```
                "Attached load balancer target group %s to auto scaling group %s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
```

```
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting to
delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
```

```
                    "You do not have the necessary permissions to describe VPCs. "
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress from
this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
```

```
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to either
this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist. Please
check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
```

```
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}' on
port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
```

```
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones {zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Erstellen Sie eine Klasse, die Aktionen von Elastic Load Balancing beinhaltet.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group specifies
 how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened times
 and
        lower thresholds. In production, you might want to decrease the sensitivity
 of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
```

```
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group '{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing target
group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for new
ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.", target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
```

```
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the AWS
Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with any
load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn} still
exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
```

```
            f"Target group {target_group_arn} was not deleted after {max_attempts *
delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
```

```
                    f"A load balancer with the name '{load_balancer_name}' already
exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in this
account and region. "
                    "You can delete unused load balancers or request an increase in
the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests to
the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener to
the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
```

```
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load balancer
'{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.
```

```
        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer {load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
```

```
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint, retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str, Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
```

```
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not found. "
                    "Ensure the load balancer exists, is in the correct AWS region,
 and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
 correct region, "
                    "and ensure it has not been deleted or created in a different
 account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books, movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary parameters.
```

```
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended, such
as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5, "WriteCapacityUnits":
5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
```

```
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name, data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}: {err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters to
 drive
    the demonstration of resilient architecture, such as failure of a dependency or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
```

```
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False. "
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate

- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Erstellen von schreibgeschützten und schreib-und lesebereichtigten IAM-Benutzern

Das folgende Codebeispiel veranschaulicht, wie Sie Benutzer erstellen und ihnen Richtlinien zuweisen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie zwei IAM-Benutzer.
- Fügen Sie eine Richtlinie zu einem Benutzer hinzu und bringen Sie Objekte in einem Amazon S3-Bucket unter.
- Fügen Sie eine Richtlinie für den zweiten Benutzer hinzu, mit der dieser Objekte aus dem Bucket abrufen kann.

- Erhalten Sie unterschiedliche Berechtigungen für den Bucket basierend auf Anmeldeinformationen des Benutzers.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Benutzer-Aktionen umschließen.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user



def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
```

```
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

Erstellen Sie Funktionen, die IAM-Richtlinien-Aktionen umschließen.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
```

```
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Erstellen Sie Funktionen, die IAM-Zugriffsschlüssel-Aktionen umschließen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
```

```
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Benutzer mit unterschiedlichen Richtlinien zu erstellen, und verwenden Sie deren Anmeldeinformationen, um auf einen Amazon-S3-Bucket zuzugreifen.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
```

```
    The demo then shows how the users can perform only the actions they are
permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and {user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
```

```
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN: {read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to {user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )
```

```
    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials. "
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
```

```
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachUserPolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteUser
  - DetachUserPolicy
  - ListUsers
  - UpdateUser

Verwalten von Zugriffsschlüsseln

Das folgende Codebeispiel veranschaulicht, wie Sie Zugriffsschlüssel verwalten.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen und Listen Sie Zugriffsschlüssel auf.
- Finden Sie heraus, wann und wie ein Zugriffsschlüssel zuletzt verwendet wurde.
- Zugriffsschlüssel aktualisieren und löschen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Zugriffsschlüssel-Aktionen umschließen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
```

```
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
```

```
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response



def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate", key_id
        )
        raise



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
```

```
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Zugriffsschlüssel-Aktionen für den aktuellen Benutzer auszuführen.

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
```

```
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access {last_use['ServiceName']}
"
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

Verwalten von Richtlinien

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen und Auflisten von Richtlinien.
- Erstellen und Abrufen von Richtlinienversionen.
- Zurücksetzen einer Richtlinie auf eine frühere Version.
- Löschen von Richtlinien.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Richtlinien-Aktionen umschließen.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
```

```
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies



def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
```

```
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
```

```
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
```

```
                "Default version %s found for %s, but no previous version exists, so
 "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um Richtlinien zu erstellen, Versionen zu aktualisieren und Informationen über sie zu erhalten.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
```

```
    bucket_arn = f"arn:aws:s3:::amzn-s3-demo-bucket"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy

- DeletePolicyVersion
- GetPolicyVersion
- ListPolicies
- ListPolicyVersions
- SetDefaultPolicyVersion

Verwalten Sie Rollen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle.
- Anfügen und Trennen von Richtlinien für eine Rolle.
- Löschen Sie eine Rolle.

SDK für Python (Boto3)

> **Note**
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Rollen-Aktionen umschließen.

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.
```

```
    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

Verwenden Sie die Wrapper-Funktionen, um eine Rolle zu erstellen, und fügen Sie dann eine Richtlinie hinzu und trennen Sie sie.

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
```

```
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
"batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachRolePolicy
  - CreateRole
  - DeleteRole
  - DetachRolePolicy

Verwalten Ihrer Konten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie den Konto-Alias ab und aktualisieren Sie ihn.
- Generieren Sie einen Bericht über Benutzer und Anmeldeinformationen.
- Rufen Sie eine Zusammenfassung der Kontonutzung ab.
- Rufen Sie Details zu allen Benutzern, Gruppen, Rollen und Richtlinien in Ihrem Konto ab, einschließlich deren Beziehungen untereinander.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die IAM-Kontoaktionen umschließen.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
```

```
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
```

```
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
```

```
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

Rufen Sie Wrapper-Funktionen auf, um den Konto-Alias zu ändern und Berichte über das Konto zu erhalten.

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
```

```
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
cred_report.decode("utf-8").split("\n")
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
```

```
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print("-" * 88)
    print("Thanks for watching.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

Zurücksetzen einer Richtlinienversion

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie die Liste der Richtlinienversionen nach Datum sortiert ab.
- Suchen Sie die Standard-Richtlinienversion.
- Machen Sie die vorherige Version der Richtlinie zur Standardversion.
- Löschen Sie die alte Standardversion.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
```

```
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists, so
"
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
```

```
        return rollback_version
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

AWS IoT Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für Python (Boto3)

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_iot():
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT client and list
    up to 10 things in your AWS IoT account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    try:
        iot_client = boto3.client("iot")
        response = iot_client.list_things(maxResults=10)
        things = response.get("things", [])

        print("Hello, AWS IoT! Here are your things:")
        if things:
            for i, thing in enumerate(things, 1):
                print(f"{i}. {thing['thingName']}")
        else:
            print("No things found in your AWS IoT account.")
    except ClientError as e:
        if e.response["Error"]["Code"] == "UnauthorizedException":
            print("You don't have permission to access AWS IoT.")
        else:
            print(f"Couldn't access AWS IoT. Error: {e}")
    except NoCredentialsError:
        print("No AWS credentials found. Please configure your credentials.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

- API-Details finden Sie unter ListThings in der AWS API-Referenz für SDK for Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstelle ein Ding. AWS IoT
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Shadow.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche eine AWS IoT Sache.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine IoT-Wrapper-Klasse, um den Betrieb zu verwalten.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client
```

```
    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)


    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
```

```
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise



    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response = self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response


    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
```

```
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal, thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response = self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]


    def list_certificates(self):
```

```
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise


    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
```

```
                        {"sns": {"targetArn": sns_action_arn, "roleArn": role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_topic_rules(self):
        """
        Lists AWS IoT topic rules.

        :return: The list of topic rules.
        """
        try:
            rules = []
            paginator = self.iot_client.get_paginator("list_topic_rules")
            for page in paginator.paginate():
                rules.extend(page["rules"])
            logger.info("Retrieved %s topic rules.", len(rules))
            return rules
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list topic rules. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

```
    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])


    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_thing(self, thing_name):
        """
```

```
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.

        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
```

```
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response = self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

Führen Sie ein interaktives Szenario aus, das die IoT-Grundlagen demonstriert.

```
class IoTScenario:
    """Runs an interactive scenario that shows how to use AWS IoT."""

    is_interactive = True

    def __init__(self, iot_wrapper, iot_data_client, cfn_client,
 stack_name="IoTBasicsStack", template_path=None):
        """
        :param iot_wrapper: An instance of the IoTWrapper class.
        :param iot_data_client: A Boto3 IoT Data Plane client.
        :param cfn_client: A Boto3 CloudFormation client.
        :param stack_name: Name for the CloudFormation stack.
        :param template_path: Path to the CloudFormation template file.
        """
        self.iot_wrapper = iot_wrapper
        self.iot_data_client = iot_data_client
        self.cfn_client = cfn_client
        self.thing_name = None
        self.certificate_arn = None
        self.certificate_id = None
        self.rule_name = None
        self.stack_name = stack_name
        self.template_path = template_path or "../../../scenarios/basics/iot/
iot_usecase/resources/cfn_template.yaml"

    def _deploy_stack(self):
        """Deploy CloudFormation stack and return outputs."""
        with open(self.template_path, "r") as f:
            template_body = f.read()

        try:
            self.cfn_client.create_stack(
                StackName=self.stack_name,
                TemplateBody=template_body,
                Capabilities=["CAPABILITY_NAMED_IAM"]
            )

            waiter = self.cfn_client.get_waiter("stack_create_complete")
            waiter.wait(StackName=self.stack_name)
```

```
            response = self.cfn_client.describe_stacks(StackName=self.stack_name)
            outputs = {output["OutputKey"]: output["OutputValue"]
                       for output in response["Stacks"][0]["Outputs"]}
            return outputs["SNSTopicArn"], outputs["RoleArn"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                response =
 self.cfn_client.describe_stacks(StackName=self.stack_name)
                outputs = {output["OutputKey"]: output["OutputValue"]
                           for output in response["Stacks"][0]["Outputs"]}
                return outputs["SNSTopicArn"], outputs["RoleArn"]
            raise

    def _cleanup_stack(self):
        """Delete CloudFormation stack."""
        try:
            self.cfn_client.delete_stack(StackName=self.stack_name)
            waiter = self.cfn_client.get_waiter("stack_delete_complete")
            waiter.wait(StackName=self.stack_name)
            print("CloudFormation stack deleted successfully.")
        except ClientError as err:
            logger.error(f"Failed to delete stack: {err}")

    def run_scenario(self, thing_name, rule_name):
        """
        Runs the IoT basics scenario.

        :param thing_name: The name of the thing to create.
        :param rule_name: The name of the topic rule to create.
        """
        print("-" * 88)
        print("Welcome to the AWS IoT basics scenario!")
        print("-" * 88)
        print(
            "This scenario demonstrates how to interact with AWS IoT using the AWS
 SDK for Python (Boto3).\n"
            "AWS IoT provides secure, bi-directional communication between Internet-
connected devices\n"
            "and the AWS cloud. You can manage device connections, process device
 data, and build IoT applications.\n"
        )

        self.thing_name = thing_name
```

```
        self.rule_name = rule_name

        try:
            print("\nDeploying CloudFormation stack...")
            sns_topic_arn, role_arn = self._deploy_stack()
            print(f"Stack deployed. SNS Topic: {sns_topic_arn}")

            input("\nNext, we'll create an AWS IoT thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("1. Create an AWS IoT thing")
            print("-" * 88)
            response = self.iot_wrapper.create_thing(thing_name)
            print(f"Created thing: {response['thingName']}")
            print(f"Thing ARN: {response['thingArn']}")

            input("\nNext, we'll list things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("2. List things")
            print("-" * 88)
            things = self.iot_wrapper.list_things()
            print(f"Found {len(things)} thing(s) in your account")
            for thing in things[:5]:  # Show first 5
                print(f"  Thing name: {thing['thingName']}")

            input("\nNext, we'll generate a device certificate. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("3. Generate a device certificate")
            print("-" * 88)
            cert_response = self.iot_wrapper.create_keys_and_certificate()
            self.certificate_arn = cert_response["certificateArn"]
            self.certificate_id = cert_response["certificateId"]
            print(f"Created certificate: {self.certificate_id}")

            input("\nNext, we'll attach the certificate to the thing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("4. Attach the certificate to the thing")
            print("-" * 88)
            self.iot_wrapper.attach_thing_principal(thing_name,
self.certificate_arn)
            print(f"Attached certificate to thing: {thing_name}")
```

```
            input("\nNext, we'll update the thing shadow. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("5. Update the thing shadow")
            print("-" * 88)
            shadow_state = {"state": {"reported": {"temperature": 25, "humidity":
50}}}
            self.iot_wrapper.update_thing_shadow(thing_name, shadow_state)
            print(f"Updated shadow for thing: {thing_name}")

            input("\nNext, we'll get the thing shadow. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("6. Get the thing shadow")
            print("-" * 88)
            shadow = self.iot_wrapper.get_thing_shadow(thing_name)
            print(f"Shadow state: {json.dumps(shadow['state'], indent=2)}")

            input("\nNext, we'll get the AWS IoT endpoint. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("7. Get the AWS IoT endpoint")
            print("-" * 88)
            endpoint = self.iot_wrapper.describe_endpoint()
            print(f"IoT endpoint: {endpoint}")

            input("\nNext, we'll list certificates. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("8. List certificates")
            print("-" * 88)
            certificates = self.iot_wrapper.list_certificates()
            print(f"Found {len(certificates)} certificate(s)")
            for cert in certificates:
                print(f"  Certificate ID: {cert['certificateId']}")
                print(f"  Certificate ARN: {cert['certificateArn']}")
                print(f"  Status: {cert['status']}")

            input("\nNext, we'll create a topic rule. Press Enter to continue...")
if self.is_interactive else None
            print("\n" + "-" * 88)
            print("9. Create a topic rule")
            print("-" * 88)
```

```
            self.iot_wrapper.create_topic_rule(
                rule_name, f"device/{thing_name}/data", sns_topic_arn, role_arn
            )
            print(f"Created topic rule: {rule_name}")

            input("\nNext, we'll list topic rules. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("10. List topic rules")
            print("-" * 88)
            rules = self.iot_wrapper.list_topic_rules()
            print(f"Found {len(rules)} topic rule(s)")
            for rule in rules:
                print(f"  Rule name: {rule['ruleName']}")
                print(f"  Rule ARN: {rule['ruleArn']}")

            input("\nNext, we'll configure thing indexing. Press Enter to
continue...") if self.is_interactive else None
            print("\n" + "-" * 88)
            print("11. Configure thing indexing")
            print("-" * 88)
            self.iot_wrapper.update_indexing_configuration()
            print("Enabled thing indexing")
            print("Waiting for indexing to be ready...")
            time.sleep(10)

            input("\nNext, we'll search for things. Press Enter to continue...") if
self.is_interactive else None
            print("\n" + "-" * 88)
            print("12. Search for things")
            print("-" * 88)
            try:
                things = self.iot_wrapper.search_index(f"thingName:{thing_name}")
                if things:
                    print(f"Found {len(things)} thing(s) matching the query")
                    for thing in things:
                        print(f"  Thing name: {thing.get('thingName', 'N/A')}")
                        print(f"  Thing ID: {thing.get('thingId', 'N/A')}")
                else:
                    print("No things found. Indexing may take a few minutes.")
            except ClientError as err:
                if err.response["Error"]["Code"] in [
                    "IndexNotReadyException",
                    "InvalidRequestException",
```

```
                ]:
                    print("Search index not ready yet. This is expected.")
                else:
                    raise

        except ClientError as err:
            logger.error(
                "Scenario failed: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        finally:
            self._cleanup()

    def _cleanup(self):
        """Cleans up resources created during the scenario."""
        if not self.thing_name:
            return

        print("\n" + "-" * 88)
        print("Cleanup")
        print("-" * 88)

        if q.ask("Do you want to delete the resources? (y/n) ", q.is_yesno):
            try:
                if self.certificate_arn:
                    print(f"Detaching certificate from thing: {self.thing_name}")
                    self.iot_wrapper.detach_thing_principal(
                        self.thing_name, self.certificate_arn
                    )

                if self.certificate_id:
                    print(f"Deleting certificate: {self.certificate_id}")
                    self.iot_wrapper.delete_certificate(self.certificate_id)

                if self.thing_name:
                    print(f"Deleting thing: {self.thing_name}")
                    self.iot_wrapper.delete_thing(self.thing_name)

                if self.rule_name:
                    print(f"Deleting topic rule: {self.rule_name}")
                    self.iot_wrapper.delete_topic_rule(self.rule_name)
```

```
                self._cleanup_stack()
                print("Resources deleted successfully.")
            except ClientError as err:
                logger.error(
                    "Cleanup failed: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
        else:
            print("Resources will remain in your account.")

        print("\n" + "-" * 88)
        print("Thanks for using AWS IoT!")
        print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreateThing
  - CreateTopicRule
  - DeleteCertificate
  - DeleteThing
  - DeleteTopicRule
  - DescribeEndpoint
  - DescribeThing
  - DetachThingPrincipal
  - ListCertificates
  - ListThings
  - SearchIndex
  - UpdateIndexingConfiguration
  - UpdateThing

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def attach_thing_principal(self, thing_name, principal):
        """
        Attaches a certificate to an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.attach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Attached principal %s to thing %s.", principal, thing_name)
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot attach principal. Resource not found.")
                return
            logger.error(
                "Couldn't attach principal to thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie AttachThingPrincipal in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `CreateKeysAndCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
```

```
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_keys_and_certificate(self):
        """
        Creates keys and a certificate for an AWS IoT thing.

        :return: The certificate ID, ARN, and PEM.
        """
        try:
            response = self.iot_client.create_keys_and_certificate(setAsActive=True)
            logger.info("Created certificate %s.", response["certificateId"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't create keys and certificate. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung. `CreateThing`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_thing(self, thing_name):
        """
        Creates an AWS IoT thing.

        :param thing_name: The name of the thing to create.
        :return: The name and ARN of the created thing.
        """
        try:
            response = self.iot_client.create_thing(thingName=thing_name)
            logger.info("Created thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Thing %s already exists. Skipping creation.",
 thing_name)
                return None
            logger.error(
                "Couldn't create thing %s. Here's why: %s: %s",
                thing_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie CreateThing in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTopicRule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def create_topic_rule(self, rule_name, topic, sns_action_arn, role_arn):
        """
        Creates an AWS IoT topic rule.

        :param rule_name: The name of the rule.
        :param topic: The MQTT topic to subscribe to.
        :param sns_action_arn: The ARN of the SNS topic to publish to.
        :param role_arn: The ARN of the IAM role.
        """
        try:
            self.iot_client.create_topic_rule(
```

```
                ruleName=rule_name,
                topicRulePayload={
                    "sql": f"SELECT * FROM '{topic}'",
                    "actions": [
                        {"sns": {"targetArn": sns_action_arn, "roleArn": role_arn}}
                    ],
                },
            )
            logger.info("Created topic rule %s.", rule_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.info("Topic rule %s already exists. Skipping creation.",
 rule_name)
                return
            logger.error(
                "Couldn't create topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateTopicRule in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteCertificate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""
```

```
    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_certificate(self, certificate_id):
        """
        Deletes an AWS IoT certificate.

        :param certificate_id: The ID of the certificate to delete.
        """
        try:
            self.iot_client.update_certificate(
                certificateId=certificate_id, newStatus="INACTIVE"
            )
            self.iot_client.delete_certificate(certificateId=certificate_id)
            logger.info("Deleted certificate %s.", certificate_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete certificate. Resource not found.")
                return
            logger.error(
                "Couldn't delete certificate. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteCertificate in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteThing`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_thing(self, thing_name):
        """
        Deletes an AWS IoT thing.

        :param thing_name: The name of the thing to delete.
        """
        try:
            self.iot_client.delete_thing(thingName=thing_name)
            logger.info("Deleted thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot delete thing. Resource not found.")
                return
            logger.error(
```

```
                "Couldn't delete thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteThing in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTopicRule`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTopicRule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def delete_topic_rule(self, rule_name):
        """
        Deletes an AWS IoT topic rule.
```

```
        :param rule_name: The name of the rule to delete.
        """
        try:
            self.iot_client.delete_topic_rule(ruleName=rule_name)
            logger.info("Deleted topic rule %s.", rule_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete topic rule. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteTopicRule in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeEndpoint`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client
```

```
    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def describe_endpoint(self, endpoint_type="iot:Data-ATS"):
        """
        Gets the AWS IoT endpoint.

        :param endpoint_type: The endpoint type.
        :return: The endpoint.
        """
        try:
            response = self.iot_client.describe_endpoint(endpointType=endpoint_type)
            logger.info("Retrieved endpoint %s.", response["endpointAddress"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't describe endpoint. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response["endpointAddress"]
```

- API-Details finden Sie DescribeEndpoint in der AWS API-Referenz zum SDK für Python (Boto3).

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung. `DetachThingPrincipal`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def detach_thing_principal(self, thing_name, principal):
        """
        Detaches a certificate from an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param principal: The ARN of the certificate.
        """
        try:
            self.iot_client.detach_thing_principal(
                thingName=thing_name, principal=principal
            )
            logger.info("Detached principal %s from thing %s.", principal,
 thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot detach principal. Resource not found.")
                return
```

```
            logger.error(
                "Couldn't detach principal from thing. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DetachThingPrincipal in der AWS API-Referenz zum SDK für Python (Boto3).

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung. `ListCertificates`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def list_certificates(self):
        """
        Lists AWS IoT certificates.

        :return: The list of certificates.
        """
        try:
            certificates = []
            paginator = self.iot_client.get_paginator("list_certificates")
            for page in paginator.paginate():
                certificates.extend(page["certificates"])
            logger.info("Retrieved %s certificates.", len(certificates))
            return certificates
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list certificates. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListCertificates in der AWS API-Referenz zum SDK für Python (Boto3).

`ListThings`

Das folgende Codebeispiel zeigt die Verwendung. `ListThings`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
```

```
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def list_things(self):
        """
        Lists AWS IoT things.

        :return: The list of things.
        """
        try:
            things = []
            paginator = self.iot_client.get_paginator("list_things")
            for page in paginator.paginate():
                things.extend(page["things"])
            logger.info("Retrieved %s things.", len(things))
            return things
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
                logger.error(
                    "Couldn't list things. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie ListThings in der AWS API-Referenz zum SDK für Python (Boto3).

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung. `SearchIndex`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def search_index(self, query):
        """
        Searches the AWS IoT index.

        :param query: The search query.
        :return: The list of things found.
        """
        try:
            response = self.iot_client.search_index(queryString=query)
            logger.info("Found %s things.", len(response.get("things", [])))
        except ClientError as err:
            if err.response["Error"]["Code"] == "ThrottlingException":
                logger.error("Request throttled. Please try again later.")
            else:
```

```
                logger.error(
                    "Couldn't search index. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
        else:
            return response.get("things", [])
```

- API-Details finden Sie SearchIndex in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateIndexingConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateIndexingConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)
```

```
    def update_indexing_configuration(self):
        """
        Updates the AWS IoT indexing configuration to enable thing indexing.
        """
        try:
            self.iot_client.update_indexing_configuration(
                thingIndexingConfiguration={"thingIndexingMode": "REGISTRY"}
            )
            logger.info("Updated indexing configuration.")
        except ClientError as err:
            logger.error(
                "Couldn't update indexing configuration. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UpdateIndexingConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

AWS IoT data Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client

    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def get_thing_shadow(self, thing_name):
        """
        Gets the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :return: The shadow state as a dictionary.
        """
        import json
        try:
            response = self.iot_data_client.get_thing_shadow(thingName=thing_name)
            shadow = json.loads(response["payload"].read())
            logger.info("Retrieved shadow for thing %s.", thing_name)
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot get thing shadow. Resource not found.")
                return None
            logger.error(
                "Couldn't get thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return shadow
```

- API-Details finden Sie GetThingShadow in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateThingShadow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTWrapper:
    """Encapsulates AWS IoT actions."""

    def __init__(self, iot_client, iot_data_client=None):
        """
        :param iot_client: A Boto3 AWS IoT client.
        :param iot_data_client: A Boto3 AWS IoT Data Plane client.
        """
        self.iot_client = iot_client
        self.iot_data_client = iot_data_client
```

```
    @classmethod
    def from_client(cls):
        iot_client = boto3.client("iot")
        iot_data_client = boto3.client("iot-data")
        return cls(iot_client, iot_data_client)

    def update_thing_shadow(self, thing_name, shadow_state):
        """
        Updates the shadow for an AWS IoT thing.

        :param thing_name: The name of the thing.
        :param shadow_state: The shadow state as a dictionary.
        """
        import json
        try:
            self.iot_data_client.update_thing_shadow(
                thingName=thing_name, payload=json.dumps(shadow_state)
            )
            logger.info("Updated shadow for thing %s.", thing_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Cannot update thing shadow. Resource not found.")
                return
            logger.error(
                "Couldn't update thing shadow. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UpdateThingShadow in der AWS API-Referenz zum SDK für Python (Boto3).

AWS IoT SiteWise Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS IoT SiteWise.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT SiteWise

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT SiteWise beginnen.

SDK für Python (Boto3)

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_iot_sitewise(iot_sitewise_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise
    client and list the asset models in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This object
 wraps
```

```
                                the low-level AWS IoT SiteWise service API.
    """
    print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
    paginator = iot_sitewise_client.get_paginator("list_asset_models")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    asset_model_names: [str] = []
    for page in page_iterator:
        for asset_model in page["assetModelSummaries"]:
            asset_model_names.append(asset_model["name"])

    print(f"{len(asset_model_names)} asset model(s) retrieved.")
    for asset_model_name in asset_model_names:
        print(f"\t{asset_model_name}")


if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- API-Details finden Sie ListAssetModels in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein AWS IoT SiteWise Asset-Modell.
- Erstellen Sie ein AWS IoT SiteWise Asset.
- Rufen Sie die Eigenschafts-ID-Werte ab.
- Daten an ein AWS IoT SiteWise Asset senden.
- Ruft den Wert der AWS IoT SiteWise Asset-Eigenschaft ab.
- Erstellen Sie ein AWS IoT SiteWise Portal.
- Erstellen Sie ein AWS IoT SiteWise Gateway.
- Beschreiben Sie das AWS IoT SiteWise Gateway.
- Löschen Sie die AWS IoT SiteWise Vermögenswerte.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class IoTSitewiseGettingStarted:
    """
    A scenario that demonstrates how to use Boto3 to manage IoT physical assets
 using
    the AWS IoT SiteWise.
    """

    def __init__(
        self,
        iot_sitewise_wrapper: IoTSitewiseWrapper,
        cloud_formation_resource: ServiceResource,
    ):
        self.iot_sitewise_wrapper = iot_sitewise_wrapper
        self.cloud_formation_resource = cloud_formation_resource
        self.stack = None
        self.asset_model_id = None
        self.asset_id = None
        self.portal_id = None
        self.gateway_id = None

    def run(self) -> None:
        """
        Runs the scenario.
        """
        print(
            """
AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes.
It is designed to help industrial and manufacturing organizations collect data from
 their equipment and
processes, and use that data to make informed decisions about their operations.
```

```
One of the key features of AWS IoT SiteWise is its ability to connect to a wide
 range of industrial
equipment and systems, including programmable logic controllers (PLCs), sensors, and
 other
industrial devices. It can collect data from these devices and organize it into a
 unified data model,
making it easier to analyze and gain insights from the data. AWS IoT SiteWise also
 provides tools for
visualizing the data, setting up alarms and alerts, and generating reports.

Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data.
It can collect and store data from thousands of devices and process millions of data
 points per second,
making it suitable for large-scale industrial operations. Additionally, AWS IoT
 SiteWise is designed
to be secure and compliant, with features like role-based access controls, data
 encryption,
and integration with other AWS services for additional security and compliance
 features.

Let's get started...
        """
        )
        press_enter_to_continue()
        print_dashes()
        print(f"")
        print(
            f"Use AWS CloudFormation to create an IAM role that is required for this
 scenario."
        )
        template_file = IoTSitewiseGettingStarted.get_template_as_string()

        self.stack = self.deploy_cloudformation_stack(
            "python-iot-sitewise-basics", template_file
        )
        outputs = self.stack.outputs
        iam_role = None

        for output in outputs:
            if output.get("OutputKey") == "SitewiseRoleArn":
                iam_role = output.get("OutputValue")

        if iam_role is None:
```

```
            error_string = f"Failed to retrieve iam_role from CloudFormation stack."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The ARN of the IAM role is {iam_role}")
        print_dashes()
        print_dashes()
        print(f"1. Create an AWS SiteWise Asset Model")
        print(
            """
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such as
 equipment,
processes, and systems, that exist in an industrial environment. This model provides
 a structured and
hierarchical representation of these assets, allowing users to define the
 relationships and values
of each asset.

This scenario creates two asset model values: temperature and humidity.
        """
        )
        press_enter_to_continue()
        asset_model_name = "MyAssetModel1"
        temperature_property_name = "temperature"
        humidity_property_name = "humidity"
        try:
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
            self.asset_model_id = self.iot_sitewise_wrapper.create_asset_model(
                asset_model_name, properties
```

```
            )
            print(
                f"Asset Model successfully created. Asset Model ID:
 {self.asset_model_id}. "
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                self.asset_model_id =
 self.get_model_id_for_model_name(asset_model_name)
                print(
                    f"Asset Model {asset_model_name} already exists. Asset Model ID:
 {self.asset_model_id}. "
                )
            else:
                raise

        press_enter_to_continue()
        print_dashes()
        print(f"2. Create an AWS IoT SiteWise Asset")
        print(
            """
The IoT SiteWise model that we just created defines the structure and metadata for
 your physical assets.
Now we create an asset from the asset model.

        """
        )
        press_enter_to_continue()

        self.asset_id = self.iot_sitewise_wrapper.create_asset(
            "MyAsset1", self.asset_model_id
        )

        print(f"Asset created with ID: {self.asset_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()
        print(f"3. Retrieve the property ID values")
        print(
            """
To send data to an asset, we need to get the property ID values. In this scenario,
 we access the
temperature and humidity property ID values.
        """
```

```
        )
        press_enter_to_continue()
        property_ids = self.iot_sitewise_wrapper.list_asset_model_properties(
            self.asset_model_id
        )
        humidity_property_id = None
        temperature_property_id = None
        for property_id in property_ids:
            if property_id.get("name") == humidity_property_name:
                humidity_property_id = property_id.get("id")
            elif property_id.get("name") == temperature_property_name:
                temperature_property_id = property_id.get("id")
        if humidity_property_id is None or temperature_property_id is None:
            error_string = f"Failed to retrieve property IDs from Asset Model."
            logger.error(error_string)
            raise ValueError(error_string)

        print(f"The Humidity property Id is {humidity_property_id}")
        print(f"The Temperature property Id is {temperature_property_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"4. Send data to an AWS IoT SiteWise Asset")
        print(
            """
By sending data to an IoT SiteWise Asset, you can aggregate data from
multiple sources, normalize the data into a standard format, and store it in a
centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to the
 AWS IoT SiteWise asset.

        """
        )
        press_enter_to_continue()

        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
```

```
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
        self.iot_sitewise_wrapper.batch_put_asset_property_value(self.asset_id,
 values)
        print(f"Data sent successfully.")

        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"5. Retrieve the value of the IoT SiteWise Asset property")
        print(
            """
IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data
from connected equipment and sensors. One of the key benefits of reading an IoT
 SiteWise property
is the ability to gain valuable insights from your industrial data.

        """
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, temperature_property_id
        )
        print(f"The property name is '{temperature_property_name}'.")

        print(
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()

        property_value = self.iot_sitewise_wrapper.get_asset_property_value(
            self.asset_id, humidity_property_id
        )
        print(f"The property name is '{humidity_property_name}'.")
        print(
```

```
            f"The value of this property is: {property_value['value']
['doubleValue']}"
        )
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"6. Create an IoT SiteWise Portal")
        print(
            """
An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources,
such as sensors, equipment, and control systems, into a centralized platform.
        """
        )

        press_enter_to_continue()
        contact_email = q.ask("Enter a contact email for the portal:", q.non_empty)
        print("Creating the portal. The portal may take a while to become active.")
        self.portal_id = self.iot_sitewise_wrapper.create_portal(
            "MyPortal1", iam_role, contact_email
        )
        print(f"Portal created successfully. Portal ID {self.portal_id}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"7. Describe the Portal")
        print(
            """
In this step, we get a description of the portal and display the portal URL.
        """
        )
        press_enter_to_continue()
        portal_description =
 self.iot_sitewise_wrapper.describe_portal(self.portal_id)
        print(f"Portal URL: {portal_description['portalStartUrl']}")
        press_enter_to_continue()
        print_dashes()
        print_dashes()

        print(f"8. Create an IoT SiteWise Gateway")
        press_enter_to_continue()
        self.gateway_id = self.iot_sitewise_wrapper.create_gateway(
```

```
            "MyGateway1", "MyThing1"
        )
        print(f"Gateway creation completed successfully. id is {self.gateway_id}")
        print_dashes()
        print_dashes()
        print(f"9. Describe the IoT SiteWise Gateway")
        press_enter_to_continue()

        gateway_description = self.iot_sitewise_wrapper.describe_gateway(
            self.gateway_id
        )
        print(f"Gateway Name: {gateway_description['gatewayName']}")
        print(f"Gateway ARN: {gateway_description['gatewayArn']}")
        print(f"Gateway Platform:\n{gateway_description['gatewayPlatform']}")
        print(f"Gateway Creation Date: {gateway_description['gatewayArn']}")
        print_dashes()
        print_dashes()

        print(f"10. Delete the AWS IoT SiteWise Assets")
        if q.ask("Would you like to delete the IoT SiteWise Assets? (y/n)",
q.is_yesno):
            self.cleanup()
        else:
            print(f"The resources will not be deleted.")
        print_dashes()
        print_dashes()
        print(f"This concludes the AWS IoT SiteWise Scenario")

    def cleanup(self) -> None:
        """
        Deletes the CloudFormation stack and the resources created for the demo.
        """

        if self.gateway_id is not None:
            self.iot_sitewise_wrapper.delete_gateway(self.gateway_id)
            print(f"Deleted gateway with id {self.gateway_id}.")
            self.gateway_id = None
        if self.portal_id is not None:
            self.iot_sitewise_wrapper.delete_portal(self.portal_id)
            print(f"Deleted portal with id {self.portal_id}.")
            self.portal_id = None
        if self.asset_id is not None:
            self.iot_sitewise_wrapper.delete_asset(self.asset_id)
            print(f"Deleted asset with id {self.asset_id}.")
```

```
            self.iot_sitewise_wrapper.wait_asset_deleted(self.asset_id)
            self.asset_id = None
        if self.asset_model_id is not None:
            self.iot_sitewise_wrapper.delete_asset_model(self.asset_model_id)
            print(f"Deleted asset model with id {self.asset_model_id}.")
            self.asset_model_id = None
        if self.stack is not None:
            stack = self.stack
            self.stack = None
            self.destroy_cloudformation_stack(stack)

    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `SitewiseRoles-template.yaml` AWS CloudFormation
script and are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
```

```
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may take a
few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        template_file_path = os.path.join(script_dir, "SitewiseRoles-template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def get_model_id_for_model_name(self, model_name: str) -> str:
        """
        Returns the model ID for the given model name.

        :param model_name: The name of the model.
        :return: The model ID.
        """
        model_id = None
        asset_models = self.iot_sitewise_wrapper.list_asset_models()
        for asset_model in asset_models:
            if asset_model["name"] == model_name:
                model_id = asset_model["id"]
                break
        return model_id
```

IoTSitewiseWrapper Klasse, die AWS IoT SiteWise Aktionen umschließt.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model description.",
```

```
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
```

```
            raise


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator = self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def list_asset_model_properties(self, asset_model_id: str) -> List[Dict[str,
Any]]:
        """
        Lists all AWS IoT SiteWise Asset Model Properties.

        :param asset_model_id: The ID of the asset model to list values for.
        :return: A list of dictionaries containing information about each asset
model property.
        """
        try:
            asset_model_properties = []
            paginator = self.iotsitewise_client.get_paginator(
                "list_asset_model_properties"
            )
            pages = paginator.paginate(assetModelId=asset_model_id)
            for page in pages:
                asset_model_properties.extend(page["assetModelPropertySummaries"])
            return asset_model_properties
```

```
        except ClientError as err:
            logger.error(
                "Error listing asset model values. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)
            self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
```

```
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.
```

```
        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id, property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise


    def create_portal(
        self, portal_name: str, iam_role_arn: str, portal_contact_email: str
    ) -> str:
        """
        Creates an AWS IoT SiteWise Portal.

        :param portal_name: The name of the portal to create.
        :param iam_role_arn: The ARN of an IAM role.
        :param portal_contact_email: The contact email of the portal.
        :return: The ID of the created portal.
        """
        try:
            response = self.iotsitewise_client.create_portal(
                portalName=portal_name,
                roleArn=iam_role_arn,
                portalContactEmail=portal_contact_email,
            )
            portal_id = response["portalId"]
            waiter = self.iotsitewise_client.get_waiter("portal_active")
            waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
            return portal_id
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Portal %s already exists.", portal_name)
            else:
                logger.error(
                    "Error creating portal %s. Here's why %s",
                    portal_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_portal(self, portal_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to describe.
        :return: A dictionary containing information about the portal.
        """
        try:
            response = self.iotsitewise_client.describe_portal(portalId=portal_id)
            return response
        except ClientError as err:
            logger.error(
                "Error describing portal %s. Here's why %s",
                portal_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
```

```
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
        """
        try:
            response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
```

```
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_portal(self, portal_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to delete.
        """
        try:
            self.iotsitewise_client.delete_portal(portalId=portal_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Portal %s does not exist.", portal_id)
            else:
                logger.error(
                    "Error deleting portal %s. Here's why %s",
                    portal_id,
                    err.response["Error"]["Message"],
                )
            raise


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
```

```
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:
            self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise


    def wait_asset_deleted(self, asset_id: str) -> None:
        """
        Waits for an AWS IoT SiteWise Asset to be deleted.

        :param asset_id: The ID of the asset to wait for.
        """
        try:
            waiter = self.iotsitewise_client.get_waiter("asset_not_exists")
            waiter.wait(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error waiting for asset %s to be deleted. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - BatchPutAssetPropertyValue
  - CreateAsset
  - CreateAssetModel
  - CreateGateway
  - DeleteAsset
  - DeleteAssetModel
  - DeleteGateway
  - DescribeAssetModel
  - DescribeGateway
  - GetAssetPropertyValue
  - ListAssetModelProperties
  - ListAssetModels

Aktionen

`BatchPutAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchPutAssetPropertyValue`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
```

```
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def batch_put_asset_property_value(
        self, asset_id: str, values: List[Dict[str, str]]
    ) -> None:
        """
        Sends data to an AWS IoT SiteWise Asset.

        :param asset_id: The asset ID.
        :param values: A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        """
        try:
            entries = self.properties_to_values(asset_id, values)
            self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Asset %s does not exist.", asset_id)
            else:
                logger.error(
```

```
                    "Error sending data to asset. Here's why %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Eine Hilfsfunktion zum Generieren des Parameters „entries“ aus einer Werteliste.

```
    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
```

```
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries
```

Hier ist ein Beispiel für eine Werteliste, die an die Hilfsfunktion übergeben werden kann.

```
        values = [
            {
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- API-Details finden Sie BatchPutAssetPropertyValue in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAsset`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAsset`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
```

```
        :param asset_model_id: The ID of the asset model to associate with the
 asset.
        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie CreateAsset in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateAssetModel`

Das folgende Codebeispiel zeigt die Verwendung. `CreateAssetModel`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""
```

```
    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_asset_model(
        self, asset_model_name: str, properties: List[Dict[str, Any]]
    ) -> str:
        """
        Creates an AWS IoT SiteWise Asset Model.

        :param asset_model_name: The name of the asset model to create.
        :param properties: The property definitions of the asset model.
        :return: The ID of the created asset model.
        """
        try:
            response = self.iotsitewise_client.create_asset_model(
                assetModelName=asset_model_name,
                assetModelDescription="This is a sample asset model description.",
                assetModelProperties=properties,
            )
            asset_model_id = response["assetModelId"]
            waiter = self.iotsitewise_client.get_waiter("asset_model_active")
            waiter.wait(assetModelId=asset_model_id)
```

```
            return asset_model_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Asset model %s already exists.", asset_model_name)
            else:
                logger.error(
                    "Error creating asset model %s. Here's why %s",
                    asset_model_name,
                    err.response["Error"]["Message"],
                )
            raise
```

Hier finden Sie ein Beispiel für eine Eigenschaftenliste, die an die Funktion übergeben wird.

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- API-Details finden Sie CreateAssetModel in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateGateway`

Das folgende Codebeispiel zeigt die Verwendung. `CreateGateway`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def create_gateway(self, gateway_name: str, my_thing: str) -> str:
        """
        Creates an AWS IoT SiteWise Gateway.

        :param gateway_name: The name of the gateway to create.
        :param my_thing: The core device thing name.
```

```
        :return: The ID of the created gateway.
        """
        try:
            response = self.iotsitewise_client.create_gateway(
                gatewayName=gateway_name,
                gatewayPlatform={
                    "greengrassV2": {"coreDeviceThingName": my_thing},
                },
                tags={"Environment": "Production"},
            )
            gateway_id = response["gatewayId"]
            return gateway_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Gateway %s already exists.", gateway_name)
            else:
                logger.error(
                    "Error creating gateway %s. Here's why %s",
                    gateway_name,
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie CreateGateway in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAsset`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAsset`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""
```

```
    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset(self, asset_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset.

        :param asset_id: The ID of the asset to delete.
        """
        try:
            self.iotsitewise_client.delete_asset(assetId=asset_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset %s. Here's why %s",
                asset_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteAsset in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteAssetModel`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAssetModel`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.
```

```
        :param asset_model_id: The ID of the asset model to delete.
        """
        try:
            self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteAssetModel in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteGateway`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteGateway`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
```

```
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def delete_gateway(self, gateway_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to delete.
        """
        try:
            self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error deleting gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie DeleteGateway in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeGateway`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeGateway`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
 IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
        """
        Describes an AWS IoT SiteWise Gateway.

        :param gateway_id: The ID of the gateway to describe.
        :return: A dictionary containing information about the gateway.
```

```
        """
        try:
            response =
 self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Gateway %s does not exist.", gateway_id)
            else:
                logger.error(
                    "Error describing gateway %s. Here's why %s",
                    gateway_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie DescribeGateway in der AWS API-Referenz zum SDK für Python (Boto3).

`GetAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung. `GetAssetPropertyValue`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.
```

```
        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id, property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
```

```
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie GetAssetPropertyValue in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAssetModels`

Das folgende Codebeispiel zeigt die Verwendung. `ListAssetModels`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
 provides low-level
                            access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
 batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
 client.
```

```
        :return: An instance of IoTSitewiseWrapper initialized with the default AWS
IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)


    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
model.

        """
        try:
            asset_models = []
            paginator = self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListAssetModels in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon Keyspaces unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Keyspaces Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Keyspaces

Das folgende Codebeispiel zeigt, wie Sie mit der Verwendung von Amazon Keyspaces beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
 Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
```

```
    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
 wraps
                             the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
        print(ks["keyspaceName"])
        print(f"\t{ks['resourceArn']}")


if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- API-Details finden Sie ListKeyspaces in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Keyspace und eine Tabelle. Das Tabellenschema enthält Filmdaten und die zeitpunktbezogene Wiederherstellung ist aktiviert.
- Stellen Sie über eine sichere TLS-Verbindung mit SigV4-Authentifizierung eine Verbindung zum Keyspace her.
- Fragen Sie die Tabelle ab. Fügen Sie Filmdaten hinzu, rufen Sie sie ab und aktualisieren Sie sie.
- Aktualisieren Sie die Tabelle. Fügen Sie eine Spalte hinzu, um bereits gesehene Filme zu verfolgen.
- Stellen Sie den vorherigen Zustand der Tabelle wieder her und bereinigen Sie die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper

    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspaces in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only letters,
 "
            "numbers and underscores: ",
            q.non_empty,
        )
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
            print(f"Created a new keyspace.\n\t{ks_arn}.")
        print("The first 10 keyspaces in your account are:\n")
        self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema to
 hold
           movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
```

```
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
        print(f"Your table is {table['status']}. Its schema is:")
        pp(table["schemaDefinition"])
        print("\nThe tables in your keyspace are:\n")
        self.ks_wrapper.list_tables()

    @demo_func
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the connection
        to the keyspace. This function downloads a default certificate or lets you
        specify your own.
        """
        print("To connect to your keyspace, you must have a TLS certificate.")
        print("Checking for TLS certificate...")
        cert_path = os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        )
        if not os.path.exists(cert_path):
            cert_choice = q.ask(
                f"Press enter to download a certificate from {QueryManager.CERT_URL}
"
                f"or enter the full path to the certificate you want to use: "
            )
            if cert_choice:
                cert_path = cert_choice
```

```
            else:
                cert = requests.get(QueryManager.CERT_URL).text
                with open(cert_path, "w") as cert_file:
                    cert_file.write(cert)
        else:
            q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
        print(
            f"Certificate {cert_path} will be used to secure the connection to your
keyspace."
        )
        return cert_path

    @demo_func
    def query_table(self, qm, movie_file):
        """
        1. Adds movies to the table from a sample movie data file.
        2. Gets a list of movies from the table and lets you select one.
        3. Displays more information about the selected movie.
        """
        qm.add_movies(self.ks_wrapper.table_name, movie_file)
        movies = qm.get_movies(self.ks_wrapper.table_name)
        print(f"Added {len(movies)} movies to the table:")
        sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
        movie_choice = qm.get_movie(
            self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
        )
        print(movie_choice.title)
        print(f"\tReleased: {movie_choice.release_date}")
        print(f"\tPlot: {movie_choice.plot}")

    @demo_func
    def update_and_restore_table(self, qm):
        """
        1. Updates the table by adding a column to track watched movies.
        2. Marks some of the movies as watched.
        3. Gets the list of watched movies from the table.
        4. Restores to a movies_restored table at a previous point in time.
        5. Gets the list of movies from the restored table.
        """
        print("Let's add a column to record which movies you've watched.")
        pre_update_timestamp = datetime.utcnow()
        print(
```

```
            f"Recorded the current UTC time of {pre_update_timestamp} so we can
 restore the table later."
        )
        self.ks_wrapper.update_table()
        print("Waiting for your table to update...")
        table = {"status": "UPDATING"}
        while table["status"] != "ACTIVE":
            wait(5)
            table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
        print("Column 'watched' added to table.")
        q.ask(
            "Let's mark some of the movies as watched. Press Enter when you're
 ready.\n"
        )
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print("-" * 88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print("-" * 88)
        if q.ask(
            "Do you want to restore the table to the way it was before all of these
\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
            q.is_yesno,
        ):
            starting_table_name = self.ks_wrapper.table_name
            table_name_restored =
 self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {"status": "RESTORING"}
            while table["status"] != "ACTIVE":
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(
                f"Restored {starting_table_name} to {table_name_restored} "
                f"at a point in time of {pre_update_timestamp}."
            )
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
```

```
                print(movie.title)

    def cleanup(self, cert_path):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.

        :param cert_path: The path of the TLS certificate used in the demo. If the
                          certificate was downloaded during the demo, it is removed.
        """
        if q.ask(
            f"Do you want to delete your {self.ks_wrapper.table_name} table and "
            f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
            q.is_yesno,
        ):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print(
                "Keyspace deleted. If you chose to restore your table during the "
                "demo, the original table is also deleted."
            )
            if cert_path == os.path.join(
                os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
            ) and os.path.exists(cert_path):
                os.remove(cert_path)
                print("Removed certificate that was downloaded for this demo.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
```

```
        # Use a context manager to ensure the connection to the keyspace is closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../../../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
        self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Keyspace- und Tabellenaktionen umschließt.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
```

```
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
        return exists


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems": limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
```

```
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

```
    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Definieren Sie eine Klasse, die eine TLS-Verbindung zu einem Keyspace herstellt, sich mit SigV4 authentifiziert und CQL-Abfragen an eine Tabelle im Keyspace sendet.

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"
```

```
    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for
TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

    def __exit__(self, *args):
        """
        Exits the cluster. This shuts down all existing session connections.
```

```
        """
        self.cluster.__exit__(*args)

    def add_movies(self, table_name, movie_file_path):
        """
        Gets movies from a JSON file and adds them to a table in the keyspace.

        :param table_name: The name of the table.
        :param movie_file_path: The path and file name of a JSON file that contains
 movie data.
        """
        with open(movie_file_path, "r") as movie_file:
            movies = json.loads(movie_file.read())
        stmt = self.session.prepare(
            f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
 (?, ?, ?, ?);"
        )
        for movie in movies[:20]:
            self.session.execute(
                stmt,
                parameters=[
                    movie["year"],
                    movie["title"],
                    date.fromisoformat(movie["info"]["release_date"].partition("T")
[0]),
                    movie["info"]["plot"],
                ],
            )

    def get_movies(self, table_name, watched=None):
        """
        Gets the title and year of the full list of movies from the table.

        :param table_name: The name of the movie table.
        :param watched: When specified, the returned list of movies is filtered to
                        either movies that have been watched or movies that have not
                        been watched. Otherwise, all movies are returned.
        :return: A list of movies in the table.
        """
        if watched is None:
            stmt = SimpleStatement(f"SELECT title, year from {table_name}")
            params = None
        else:
            stmt = SimpleStatement(
```

```
                f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
FILTERING"
            )
            params = [watched]
        return self.session.execute(stmt, parameters=params).all()

    def get_movie(self, table_name, title, year):
        """
        Gets a single movie from the table, by title and year.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        :return: The requested movie.
        """
        return self.session.execute(
            SimpleStatement(
                f"SELECT * from {table_name} WHERE title = %s AND year = %s"
            ),
            parameters=[title, year],
        ).one()

    def watched_movie(self, table_name, title, year):
        """
        Updates a movie as having been watched.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        """
        self.session.execute(
            SimpleStatement(
                f"UPDATE {table_name} SET watched=true WHERE title = %s AND year =
%s"
            ),
            parameters=[title, year],
        )
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).

- CreateKeyspace
- CreateTable
- DeleteKeyspace
- DeleteTable
- GetKeyspace
- GetTable
- ListKeyspaces
- ListTables
- RestoreTable
- UpdateTable

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyspace`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None
```

```
    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn
```

- API-Details finden Sie CreateKeyspace in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
```

```
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]
```

- API-Details finden Sie CreateTable in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
```

```
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteKeyspace in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API-Details finden Sie DeleteTable in der AWS API-Referenz zum SDK für Python (Boto3).

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung. `GetKeyspace`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.
```

```
        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists
```

- API-Details finden Sie GetKeyspace in der AWS API-Referenz zum SDK für Python (Boto3).

`GetTable`

Das folgende Codebeispiel zeigt die Verwendung. `GetTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""
```

```
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- API-Details finden Sie GetTable in der AWS API-Referenz zum SDK für Python (Boto3).

`ListKeyspaces`

Das folgende Codebeispiel zeigt die Verwendung. `ListKeyspaces`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems": limit}):
```

```
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListKeyspaces in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung. `ListTables`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
```

```
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListTables in der AWS API-Referenz zum SDK für Python (Boto3).

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung. `RestoreTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""
```

```
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name
```

- API-Details finden Sie RestoreTable in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateTable`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
```

```
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UpdateTable in der AWS API-Referenz zum SDK für Python (Boto3).

Kinesis-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Kinesis Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`CreateStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateStream`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def create(self, name, wait_until_exists=True):
        """
        Creates a stream.

        :param name: The name of the stream.
        :param wait_until_exists: When True, waits until the service reports that
                                  the stream exists, then queries for its metadata.
        """
        try:
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)
            self.name = name
            logger.info("Created stream %s.", name)
            if wait_until_exists:
                logger.info("Waiting until exists.")
                self.stream_exists_waiter.wait(StreamName=name)
                self.describe(name)
```

```
        except ClientError:
            logger.exception("Couldn't create stream %s.", name)
            raise
```

- API-Details finden Sie CreateStream in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteStream`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteStream`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def delete(self):
        """
        Deletes a stream.
        """
        try:
            self.kinesis_client.delete_stream(StreamName=self.name)
            self._clear()
            logger.info("Deleted stream %s.", self.name)
        except ClientError:
```

```
            logger.exception("Couldn't delete stream %s.", self.name)
            raise
```

- API-Details finden Sie DeleteStream in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeStream`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeStream`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
```

```
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- API-Details finden Sie DescribeStream in der AWS API-Referenz zum SDK für Python (Boto3).

`GetRecords`

Das folgende Codebeispiel zeigt die Verwendung. `GetRecords`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def get_records(self, max_records):
        """
        Gets records from the stream. This function is a generator that first gets
        a shard iterator for the stream, then uses the shard iterator to get records
        in batches from the stream. The shard iterator can be accessed through the
```

```
        'details' property, which is populated using the 'describe' function of this
class.
        Each batch of records is yielded back to the caller until the specified
        maximum number of records has been retrieved.

        :param max_records: The maximum number of records to retrieve.
        :return: Yields the current batch of retrieved records.
        """
        try:
            response = self.kinesis_client.get_shard_iterator(
                StreamName=self.name,
                ShardId=self.details["Shards"][0]["ShardId"],
                ShardIteratorType="LATEST",
            )
            shard_iter = response["ShardIterator"]
            record_count = 0
            while record_count < max_records:
                response = self.kinesis_client.get_records(
                    ShardIterator=shard_iter, Limit=10
                )
                shard_iter = response["NextShardIterator"]
                records = response["Records"]
                logger.info("Got %s records.", len(records))
                record_count += len(records)
                yield records
        except ClientError:
            logger.exception("Couldn't get records from stream %s.", self.name)
            raise



    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response["StreamDescription"]
            logger.info("Got stream %s.", name)
        except ClientError:
```

```
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- API-Details finden Sie GetRecords in der AWS API-Referenz zum SDK für Python (Boto3).

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung. `PutRecord`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""

    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter("stream_exists")


    def put_record(self, data, partition_key):
        """
        Puts data into the stream. The data is formatted as JSON before it is passed
        to the stream.

        :param data: The data to put in the stream.
        :param partition_key: The partition key to use for the data.
```

```
        :return: Metadata about the record, including its shard ID and sequence
number.
        """
        try:
            response = self.kinesis_client.put_record(
                StreamName=self.name, Data=json.dumps(data),
PartitionKey=partition_key
            )
            logger.info("Put record in stream %s.", self.name)
        except ClientError:
            logger.exception("Couldn't put record in stream %s.", self.name)
            raise
        else:
            return response
```

- API-Details finden Sie PutRecord in der AWS API-Referenz zum SDK für Python (Boto3).

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
def lambda_handler(event, context):
```

```
    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
```

```
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

AWS KMS Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS KMS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines KMS-Schlüssels.
- Listen Sie die KMS-Schlüssel für Ihr Konto auf und rufen Sie Details über sie ab.
- Aktivieren und Deaktivieren von KMS-Schlüsseln
- Generieren Sie einen symmetrischen Datenschlüssel, der für die clientseitige Verschlüsselung verwendet werden kann.

- Generieren Sie einen asymmetrischen Schlüssel, der zum digitalen Signieren von Daten verwendet wird.
- Taggen Sie Schlüssel.
- Löschen Sie KMS-Schlüssel.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.
```

```
This program demonstrates how to interact with AWS Key Management using the AWS SDK
 for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service that
 allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
 and applications.
KMS provides a centralized and unified approach to managing encryption keys, making
 it easier to meet your
data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...
        """
        )
        q.ask("Press Enter to continue...")

        print(DASHES)
        print(f"1. Create a symmetric KMS key\n")
        print(
            f"First, the program will creates a symmetric KMS key that you can used
 to encrypt and decrypt data."
        )
        q.ask("Press Enter to continue...")
        self.key_id = self.key_manager.create_key(key_description)["KeyId"]
        print(f"A symmetric key was successfully created {self.key_id}.")
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(
            """
2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code
 checks to
determine if the key is enabled.
        """
        )
        q.ask("Press Enter to continue...")
        is_enabled = self.is_key_enabled(self.key_id)
        print(f"Is the key enabled? {is_enabled}")
        if not is_enabled:
```

```
            self.key_manager.enable_key(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"3. Encrypt data using the symmetric KMS key")
        plain_text = "Hello, AWS KMS!"
        print(
            f"""
One of the main uses of symmetric keys is to encrypt and decrypt data.
Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT
 encryption algorithm.
        """
        )
        q.ask("Press Enter to continue...")
        encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
        print(DASHES)
        print(f"4. Create an alias")
        print(
            """
Now, the program will create an alias for the KMS key. An alias is a friendly name
 that you
can associate with a KMS key. The alias name should be prefixed with 'alias/'.
        """
        )
        alias_name = q.ask("Enter an alias name: ", q.non_empty)
        self.alias_manager.create_alias(self.key_id, alias_name)
        print(f"{alias_name} was successfully created.")
        self.alias_name = alias_name
        print(DASHES)
        print(f"5. List all of your aliases")
        q.ask("Press Enter to continue...")
        self.alias_manager.list_aliases(10)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"6. Enable automatic rotation of the KMS key")
        print(
            """

By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days) from
 the enable date and every year
thereafter.
        """
        )
        q.ask("Press Enter to continue...")
```

```
        self.key_manager.enable_key_rotation(self.key_id)
        print(DASHES)
        print(f"Key rotation has been enabled for key with id {self.key_id}")
        print(
            """
7. Create a grant

A grant is a policy instrument that allows Amazon Web Services principals to use KMS
 keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage grants.
When authorizing access to a KMS key, grants are considered along with key policies
 and IAM policies.
        """
        )
        print(
            """
To create a grant you must specify a account_id. To specify the grantee account_id,
 use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM roles,
 federated users,
and assumed role users.
        """
        )
        account_id = q.ask(
            "Enter an account_id, or press enter to skip creating a grant... "
        )
        grant = None
        if account_id != "":
            grant = self.grant_manager.create_grant(
                self.key_id,
                account_id,
                [
                    "Encrypt",
                    "Decrypt",
                    "DescribeKey",
                ],
            )
            print(f"Grant created successfully with ID: {grant['GrantId']}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(DASHES)
        print(f"8. List grants for the KMS key")
        q.ask("Press Enter to continue...")
```

```
        self.grant_manager.list_grants(self.key_id)
        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"9. Revoke the grant")
        print(
            """
The revocation of a grant immediately removes the permissions and access that the
 grant had provided.
This means that any account_id (user, role, or service) that was granted access to
 perform specific
KMS operations on a KMS key will no longer be able to perform those operations.
        """
        )
        q.ask("Press Enter to continue...")

        if grant is not None:
            self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
            print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"10. Decrypt the data\n")
        print(
            """
Lets decrypt the data that was encrypted in an early step.
The code uses the same key to decrypt the string that we encrypted earlier in the
 program.
        """
        )
        q.ask("Press Enter to continue...")
        decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
        print(f"Data decrypted successfully for key ID: {self.key_id}")
        print(f"Decrypted data: {decrypted_data}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"11. Replace a key policy\n")
        print(
            """
A key policy is a resource policy for a KMS key. Key policies are the primary way to
 control
access to KMS keys. Every KMS key must have exactly one key policy. The statements
 in the key policy
determine who has permission to use the KMS key and how they can use it.
```

```
You can also use IAM policies and grants to control access to the KMS key, but every
 KMS key
must have a key policy.

By default, when you create a key by using the SDK, a policy is created that
gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.
{
"Version":"2012-10-17",
"Statement": [{
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::0000000000:root"},
"Action": "kms:*",
"Resource": "*"
}]
}
        """
        )
        account_id = q.ask("Enter your account ID or press enter to skip: ")
        if account_id != "":
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                        "Action": "kms:*",
                        "Resource": "*",
                    }
                ],
            }

            self.key_policy.set_new_policy(self.key_id, policy)
            print("Key policy replacement succeeded.")
            q.ask("Press Enter to continue...")
        else:
            print("Skipping replacing the key policy.")

        print(DASHES)
        print(f"12. Get the key policy\n")
        print(
            f"The next bit of code that runs gets the key policy to make sure it
 exists."
```

```
        )
        q.ask("Press Enter to continue...")
        policy = self.key_policy.get_policy(self.key_id)
        print(f"The key policy is: {policy}")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"13. Create an asymmetric KMS key and sign your data\n")
        print(
            """
        Signing your data with an AWS key can provide several benefits that make it
an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
        """
        )
        q.ask("Press Enter to continue...")
        print(f"Sign and verify data operation succeeded.")
        self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
        message = "Here is the message that will be digitally signed"
        signature = self.key_encryption.sign(self.asymmetric_key_id, message)
        if self.key_encryption.verify(self.asymmetric_key_id, message, signature):
            print("Signature verification succeeded.")
        else:
            print("Signature verification failed.")

        q.ask("Press Enter to continue...")
        print(DASHES)
        print(f"14. Tag your symmetric KMS Key\n")
        print(
            """
        By using tags, you can improve the overall management, security, and
governance of your
        KMS keys, making it easier to organize, track, and control access to your
encrypted data within
        your AWS environment
        """
        )
        q.ask("Press Enter to continue...")
        self.key_manager.tag_resource(self.key_id, "Environment", "Production")
        self.clean_up()
```

```
    def is_key_enabled(self, key_id: str) -> bool:
        """
        Check if the key is enabled or not.

        :param key_id: The key to check.
        :return: True if the key is enabled, otherwise False.
        """
        response = self.key_manager.describe_key(key_id)
        return response["Enabled"] is True

    def clean_up(self):
        """
        Delete resources created by this scenario.
        """
        if self.alias_name != "":
            print(f"Deleting the alias {self.alias_name}.")
            self.alias_manager.delete_alias(self.alias_name)
        window = 7  # The window in days for a scheduled deletion.
        if self.key_id != "":
            print(
                """
Warning:
Deleting a KMS key is a destructive and potentially dangerous operation. When a KMS
 key is deleted,
all data that was encrypted under the KMS key is unrecoverable.
                """
            )
            if q.ask(
                f"Do you want to delete the key with ID {self.key_id} (y/n)?",
                q.is_yesno,
            ):
                print(
                    f"The key {self.key_id} will be deleted with a window of
 {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.key_id, window)
                self.key_id = ""

        if self.asymmetric_key_id != "":
            if q.ask(
                f"Do you want to delete the asymmetric key with ID
 {self.asymmetric_key_id} (y/n)?",
```

```
                q.is_yesno,
            ):
                print(
                    f"The key {self.asymmetric_key_id} will be deleted with a window
 of {window} days. You can cancel the deletion before"
                )
                print("the window expires.")
                self.key_manager.delete_key(self.asymmetric_key_id, window)
                self.asymmetric_key_id = ""


if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverwaltung.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
```

```
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def create_asymmetric_key(self) -> str:
        """
        Creates an asymmetric key in AWS KMS for signing messages.

        :return: The ID of the created key.
        """
        try:
            key = self.kms_client.create_key(
                KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
            )["KeyMetadata"]
            self.created_keys.append(key)
            return key["KeyId"]
        except ClientError as err:
            logger.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
```

```
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselaliase.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

```
    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
```

```
            raise
```

Wrapper-Klasse und Methoden für die KMS-Schlüsselverschlüsselung.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
 Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
 {response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
```

```
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
            return self.kms_client.decrypt(KeyId=key_id, CiphertextBlob=cipher_text)
[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselberechtigungen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def list_grants(self, key_id):
        """
        Lists grants for a key.
```

```
        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

Wrapper-Klasse und Methoden für KMS-Schlüsselrichtlinien.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the existing
        policy, so care is taken to add a statement to the existing list of
 statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
 principal to make a subsequent
                       PutKeyPolicy request on the KMS key. This reduces the risk
 that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id, Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise



    def get_policy(self, key_id: str) -> dict[str, str]:
        """
```

```
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants

- ListKeys
- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_alias(self, key_id: str, alias: str) -> None:
        """
```

```
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
 key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie CreateAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung. `CreateGrant`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
```

```
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateGrant in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung. `CreateKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateKey in der AWS API-Referenz zum SDK für Python (Boto3).

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung. `Decrypt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def decrypt(self, key_id: str, cipher_text: bytes) -> str:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
        try:
```

```
            return self.kms_client.decrypt(KeyId=key_id, CiphertextBlob=cipher_text)
[
                "Plaintext"
            ].decode()
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS -SDK für Python (Boto3).

`DeleteAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlias`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
```

```
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```
    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def describe_key(self, key_id: str) -> dict[str, any]:
        """
        Describes a key.

        :param key_id: The ARN or ID of the key to describe.
        :return: Information about the key.
        """

        try:
            key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
            return key
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeKey in der AWS API-Referenz zum SDK für Python (Boto3).

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung. `DisableKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def disable_key(self, key_id: str) -> None:
        try:
            self.kms_client.disable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DisableKey in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung. `EnableKey`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key(self, key_id: str) -> None:
        """
        Enables a key. Gets the key state after each state change.

        :param key_id: The ARN or ID of the key to enable.
        """
        try:
            self.kms_client.enable_key(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie EnableKey in der AWS API-Referenz zum SDK für Python (Boto3).

`EnableKeyRotation`

Das folgende Codebeispiel zeigt die Verwendung. `EnableKeyRotation`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def enable_key_rotation(self, key_id: str) -> None:
        """
        Enables rotation for a key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        """
        try:
            self.kms_client.enable_key_rotation(KeyId=key_id)
        except ClientError as err:
```

```
            logging.error(
                "Couldn't enable rotation for key '%s'. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie EnableKeyRotation in der AWS API-Referenz zum SDK für Python (Boto3).

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung. `Encrypt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def encrypt(self, key_id: str, text: str) -> bytes:
        """
```

```
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
                logger.error(
                    "Could not encrypt because the key %s is disabled.", key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS -SDK für Python (Boto3).

`GenerateDataKey`

Das folgende Codebeispiel zeigt die Verwendung`GenerateDataKey`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/n)?
 "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)
```

- API-Details finden Sie GenerateDataKey in der AWS API-Referenz zum SDK für Python (Boto3).

GetKeyPolicy

Das folgende Codebeispiel zeigt die Verwendung. `GetKeyPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %s. Here's why: %s",
```

```
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                pprint(policy)
                return policy
        else:
            print("Skipping get policy demo.")
```

- API-Details finden Sie GetKeyPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung. `ListAliases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListAliases in der AWS API-Referenz zum SDK für Python (Boto3).

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung. `ListGrants`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
            grants = []
            page_iterator = paginator.paginate(KeyId=key_id)
            for page in page_iterator:
                grants.extend(page["Grants"])

            print(f"Grants for key {key_id}:")
            pprint(grants)
            return grants
        except ClientError as err:
            logger.error(
                "Couldn't list grants for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListGrants in der AWS API-Referenz zum SDK für Python (Boto3).

`ListKeyPolicies`

Das folgende Codebeispiel zeigt die Verwendung. `ListKeyPolicies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
```

```
                err.response["Error"]["Message"],
            )
            raise
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- API-Details finden Sie ListKeyPolicies in der AWS API-Referenz zum SDK für Python (Boto3).

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung. `ListKeys`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
```

```
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
 10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )
```

- API-Details finden Sie ListKeys in der AWS API-Referenz zum SDK für Python (Boto3).

`PutKeyPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `PutKeyPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the existing
        policy, so care is taken to add a statement to the existing list of
statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        :return: None
        """
        principal = input(
            "Enter the ARN of an IAM role to set as the principal on the policy: "
        )
        if key_id != "" and principal != "":
            # The updated policy replaces the existing policy. Add a new statement
to
            # the list along with the original policy statements.
            policy["Statement"].append(
                {
                    "Sid": "Allow access for ExampleRole",
                    "Effect": "Allow",
                    "Principal": {"AWS": principal},
                    "Action": [
                        "kms:Encrypt",
                        "kms:GenerateDataKey*",
                        "kms:Decrypt",
                        "kms:DescribeKey",
                        "kms:ReEncrypt*",
                    ],
                    "Resource": "*",
                }
```

```
            )
            try:
                self.kms_client.put_key_policy(KeyId=key_id,
 Policy=json.dumps(policy))
            except ClientError as err:
                logger.error(
                    "Couldn't set policy for key %s. Here's why %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
                raise
            else:
                print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- API-Details finden Sie PutKeyPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`ReEncrypt`

Das folgende Codebeispiel zeigt die Verwendung. `ReEncrypt`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.
```

```
        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt the
                              ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: "
        )
        if destination_key_id != "":
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id,
                    DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text,
                )["CiphertextBlob"]
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")
```

- API-Details finden Sie ReEncrypt in der AWS API-Referenz zum SDK für Python (Boto3).

`RetireGrant`

Das folgende Codebeispiel zeigt die Verwendung. `RetireGrant`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant["GrantId"],
                err.response["Error"]["Message"],
            )
        else:
```

```
            print(f"Grant {grant['GrantId']} retired.")
```

- API-Details finden Sie RetireGrant in der AWS API-Referenz zum SDK für Python (Boto3).

`RevokeGrant`

Das folgende Codebeispiel zeigt die Verwendung. `RevokeGrant`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def revoke_grant(self, key_id: str, grant_id: str) -> None:
        """
        Revokes a grant so that it can no longer be used.

        :param key_id: The ARN or ID of the key associated with the grant.
        :param grant_id: The ID of the grant to revoke.
        """
        try:
```

```
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
        except ClientError as err:
            logger.error(
                "Couldn't revoke grant %s. Here's why: %s",
                grant_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie RevokeGrant in der AWS API-Referenz zum SDK für Python (Boto3).

`ScheduleKeyDeletion`

Das folgende Codebeispiel zeigt die Verwendung. `ScheduleKeyDeletion`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is deleted.
        """

        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key_id, PendingWindowInDays=window
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ScheduleKeyDeletion in der AWS API-Referenz zum SDK für Python (Boto3).

`Sign`

Das folgende Codebeispiel zeigt die Verwendung. `Sign`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def sign(self, key_id: str, message: str) -> str:
        """
        Signs a message with a key.

        :param key_id: The ARN or ID of the key to use for signing.
        :param message: The message to sign.
        :return: The signature of the message.
        """
        try:
            return self.kms_client.sign(
                KeyId=key_id,
                Message=message.encode(),
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )["Signature"]
        except ClientError as err:
            logger.error(
                "Couldn't sign your message. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie unter Sign in der API-Referenz zum AWS -SDK für Python (Boto3).

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
        """
        Add or edit tags on a customer managed key.

        :param key_id: The ARN or ID of the key to enable rotation for.
        :param tag_key: Key for the tag.
        :param tag_value: Value for the tag.
        """
        try:
            self.kms_client.tag_resource(
                KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
            )
        except ClientError as err:
            logging.error(
                "Couldn't add a tag for the key '%s'. Here's why: %s",
```

```
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie TagResource in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateAlias`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateAlias`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
 client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.
```

```
        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
 the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with {alias}: "
        )
        if new_key_id != "":
            try:
                self.kms_client.update_alias(AliasName=alias,
 TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias,
                    new_key_id,
                    err.response["Error"]["Message"],
                )
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- API-Details finden Sie UpdateAlias in der AWS API-Referenz zum SDK für Python (Boto3).

`Verify`

Das folgende Codebeispiel zeigt die Verwendung. `Verify`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KeyEncrypt:
    def __init__(self, kms_client):
```

```
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)


    def verify(self, key_id: str, message: str, signature: str) -> bool:
        """
        Verifies a signature against a message.

        :param key_id: The ARN or ID of the key used to sign the message.
        :param message: The message to verify.
        :param signature: The signature to verify.
        :return: True when the signature matches the message, otherwise False.
        """
        try:
            response = self.kms_client.verify(
                KeyId=key_id,
                Message=message.encode(),
                Signature=signature,
                SigningAlgorithm="RSASSA_PSS_SHA_256",
            )
            valid = response["SignatureValid"]
            print(f"The signature is {'valid' if valid else 'invalid'}.")
            return valid
        except ClientError as err:
            if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
                print("The signature is not valid.")
            else:
                logger.error(
                    "Couldn't verify your signature. Here's why: %s",
                    err.response["Error"]["Message"],
                )
            raise
```

- Weitere API-Informationen finden Sie unter Verify in der API-Referenz zum AWS -SDK für Python (Boto3).

Lambda-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def main():
    """
    List the Lambda functions in your AWS account.
    """
    # Create the Lambda client
    lambda_client = boto3.client("lambda")

    # Use the paginator to list the functions
    paginator = lambda_client.get_paginator("list_functions")
    response_iterator = paginator.paginate()

    print("Here are the Lambda functions in your account:")
    for page in response_iterator:
        for function in page["Functions"]:
            print(f"  {function['FunctionName']}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ListFunctions in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie einen Lambda-Handler, der eine Zahl inkrementiert.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)


def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
 number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
```

```
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Definieren Sie einen zweiten Lambda-Handler, der arithmetische Operationen ausführt.

```
import logging
import os


logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
 function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}


def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)
```

```
    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response
```

Erstellen Sie Funktionen, die Lambda-Aktionen umschließen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
        Creates a Lambda deployment package in .zip format in an in-memory buffer.
 This
        buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
 Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, "w") as zipped:
```

```
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
        Creates an IAM role that grants the Lambda function basic permissions. If a
        role with the specified name already exists, it is used for the demo.

        :param iam_role_name: The name of the role to create.
        :return: The role and a value that indicates whether the role is newly
created.
        """
        role = self.get_iam_role(iam_role_name)
        if role is not None:
            return role, False

        lambda_assume_role_policy = {
```

```
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "lambda.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"

        try:
            role = self.iam_resource.create_role(
                RoleName=iam_role_name,
                AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy),
            )
            logger.info("Created role %s.", role.name)
            role.attach_policy(PolicyArn=policy_arn)
            logger.info("Attached basic execution policy to role %s.", role.name)
        except ClientError as error:
            if error.response["Error"]["Code"] == "EntityAlreadyExists":
                role = self.iam_resource.Role(iam_role_name)
                logger.warning("The role %s already exists. Using it.",
 iam_role_name)
            else:
                logger.exception(
                    "Couldn't create role %s or attach policy %s.",
                    iam_role_name,
                    policy_arn,
                )
                raise

        return role, True

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
```

```
            response = self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
                             must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
```

```
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                                is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included
in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
```

```
            raise
        return response


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Erstellen Sie eine Funktion, die das Szenario ausführt.

```
class UpdateFunctionWaiter(CustomWaiter):
    """A custom waiter that waits until a function is successfully updated."""

    def __init__(self, client):
        super().__init__(
```

```
            "UpdateSuccess",
            "GetFunction",
            "Configuration.LastUpdateStatus",
            {"Successful": WaitState.SUCCESS, "Failed": WaitState.FAILURE},
            client,
        )

    def wait(self, function_name):
        self._wait(FunctionName=function_name)


def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
 lambda_name):
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda handler.
    :param calculator_file: The name of the file that contains the calculator Lambda
 handler.
    :param lambda_name: The name to give resources created for the scenario, such as
 the
                        IAM role and the Lambda function.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print("-" * 88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(
```

```
            basic_file, f"{lambda_name}.py"
        )
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f"{lambda_name}.lambda_handler", iam_role,
deployment_package
        )
    else:
        print(f"Function {lambda_name} already exists.")
    print("-" * 88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
        "action": "increment",
        "number": q.ask("Give me a number to increment: ", q.is_int),
    }
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(
        f"Incrementing {action_params['number']} resulted in "
        f"{json.load(response['Payload'])}"
    )
    print("-" * 88)

    print(f"Let's update the function to an arithmetic calculator.")
    q.ask("Press Enter when you're ready.")
    print("Creating a new deployment package...")
    deployment_package = wrapper.create_deployment_package(
        calculator_file, f"{lambda_name}.py"
    )
    print(f"...and updating the {lambda_name} Lambda function.")
    update_waiter = UpdateFunctionWaiter(lambda_client)
    wrapper.update_function_code(lambda_name, deployment_package)
    update_waiter.wait(lambda_name)
    print(f"This function uses an environment variable to control logging level.")
    print(f"Let's set it to DEBUG to get the most logging.")
    wrapper.update_function_configuration(
        lambda_name, {"LOG_LEVEL": logging.getLevelName(logging.DEBUG)}
    )

    actions = ["plus", "minus", "times", "divided-by"]
    want_invoke = True
    while want_invoke:
        print(f"Let's invoke {lambda_name}. You can invoke these actions:")
```

```
        for index, action in enumerate(actions):
            print(f"{index + 1}: {action}")
        action_params = {}
        action_index = q.ask(
            "Enter the number of the action you want to take: ",
            q.is_int,
            q.in_range(1, len(actions)),
        )
        action_params["action"] = actions[action_index - 1]
        print(f"You've chosen to invoke 'x {action_params['action']} y'.")
        action_params["x"] = q.ask("Enter a value for x: ", q.is_int)
        action_params["y"] = q.ask("Enter a value for y: ", q.is_int)
        print(f"Invoking {lambda_name}...")
        response = wrapper.invoke_function(lambda_name, action_params, True)
        print(
            f"Calculating {action_params['x']} {action_params['action']}
{action_params['y']} "
            f"resulted in {json.load(response['Payload'])}"
        )
        q.ask("Press Enter to see the logs from the call.")
        print(base64.b64decode(response["LogResult"]).decode())
        want_invoke = q.ask("That was fun. Shall we do it again? (y/n) ",
q.is_yesno)
    print("-" * 88)

    if q.ask(
        "Do you want to list all of the functions in your account? (y/n) ",
q.is_yesno
    ):
        wrapper.list_functions()
    print("-" * 88)

    if q.ask("Ready to delete the function and role? (y/n) ", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print("-" * 88)
```

```
if __name__ == "__main__":
    try:
        run_scenario(
            boto3.client("lambda"),
            boto3.resource("iam"),
            "lambda_handler_basic.py",
            "lambda_handler_calculator.py",
            "doc_example_lambda_calculator",
        )
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
```

```
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def create_function(
        self, function_name, handler_name, iam_role, deployment_package
    ):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
                             must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the function
                                   code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime="python3.9",
                Role=iam_role.arn,
                Handler=handler_name,
                Code={"ZipFile": deployment_package},
                Publish=True,
            )
            function_arn = response["FunctionArn"]
            waiter = self.lambda_client.get_waiter("function_active_v2")
            waiter.wait(FunctionName=function_name)
            logger.info(
                "Created function '%s' with ARN: '%s'.",
                function_name,
                response["FunctionArn"],
            )
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- API-Details finden Sie CreateFunction in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFunction`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

- API-Details finden Sie DeleteFunction in der AWS API-Referenz zum SDK für Python (Boto3).

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung. `GetFunction`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response = self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- API-Details finden Sie GetFunction in der AWS API-Referenz zum SDK für Python (Boto3).

Invoke

Das folgende Codebeispiel zeigt die Verwendung. `Invoke`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                                is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included
 in
                        the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType="Tail" if get_log else "None",
            )
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Python (Boto3).

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            func_paginator = self.lambda_client.get_paginator("list_functions")
            for func_page in func_paginator.paginate():
                for func in func_page["Functions"]:
                    print(func["FunctionName"])
                    desc = func.get("Description")
                    if desc:
                        print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
        except ClientError as err:
            logger.error(
                "Couldn't list functions. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- API-Details finden Sie ListFunctions in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateFunctionCode`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
 contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes in
                                   .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package
            )
        except ClientError as err:
```

```
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie UpdateFunctionCode in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateFunctionConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource


    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
```

```
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={"Variables": env_vars}
            )
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen Sie eine REST-API, um Daten zu verfolgen COVID-19

Das folgende Codebeispiel zeigt, wie eine REST-API erstellt wird, die ein System simuliert, um tägliche Fälle COVID-19 in den Vereinigten Staaten anhand fiktiver Daten zu verfolgen.

SDK für Python (Boto3)

Zeigt, wie AWS Chalice mit dem verwendet wird AWS SDK für Python (Boto3) , um eine serverlose REST-API zu erstellen, die Amazon API Gateway und Amazon DynamoDB AWS Lambda verwendet. Die REST-API simuliert ein System, das anhand fiktiver Daten tägliche Fälle COVID-19 in den Vereinigten Staaten verfolgt. Lernen Sie Folgendes:

- Verwenden Sie AWS Chalice, um Routen in Lambda-Funktionen zu definieren, die aufgerufen werden, um REST-Anfragen zu bearbeiten, die über das API Gateway eingehen.
- Verwenden Sie Lambda-Funktionen zum Abrufen und Speichern von Daten in einer DynamoDB-Tabelle, um REST-Anforderungen zu bearbeiten.
- Definieren Sie die Tabellenstruktur und die Ressourcen für Sicherheitsrollen in einer AWS CloudFormation Vorlage.

- Verwenden Sie AWS Chalice und CloudFormation , um alle erforderlichen Ressourcen zu verpacken und bereitzustellen.
- Wird verwendet CloudFormation , um alle erstellten Ressourcen zu bereinigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- CloudFormation
- DynamoDB
- Lambda

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

SDK für Python (Boto3)

Zeigt, wie Sie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) -API und AWS Chalice verwenden, um eine REST-API zu erstellen, die von einer Amazon Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldeinformationen verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Erstellen einer Messenger-Anwendung

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

SDK für Python (Boto3)

Zeigt, wie Sie with verwenden, AWS Step Functions um eine Messenger-Anwendung zu erstellen, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. AWS SDK für Python (Boto3) Die State-Machine ist in eine AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht wird.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step-Funktionen

Erstellen einer Websocket-Chat-Anwendung

Das folgende Codebeispiel zeigt, wie eine Chat-Anwendung erstellt wird, die von einer auf Amazon API Gateway basierenden Websocket-API bereitgestellt wird.

SDK für Python (Boto3)

Zeigt, wie Sie das AWS SDK für Python (Boto3) mit Amazon API Gateway V2 verwenden, um eine Websocket-API zu erstellen, die in Amazon DynamoDB integriert ist. AWS Lambda

- Erstellen Sie eine WebSocket-API, die von API Gateway bereitgestellt wird.
- Definieren Sie einen Lambda-Handler, der Verbindungen in DynamoDB speichert und Nachrichten an andere Chat-Teilnehmer sendet.
- Stellen Sie eine Verbindung zur Websocket-Chat-Anwendung her und senden Sie Nachrichten mit dem Websockets-Paket.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.

- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten unter GitHub ansehen. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel eines geplanten EventBridge Amazon-Ereignisses registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten zum späteren Abrufen in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erstellt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']
```

```
# Fetch RDS Auth Token
def get_auth_token():
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import base64
def lambda_handler(event, context):

    for record in event['Records']:
        try:
            print(f"Processed Kinesis Event - EventID: {record['eventID']}")
            record_data = base64.b64decode(record['kinesis']
['data']).decode('utf-8')
            print(f"Record Data: {record_data}")
            # TODO: Do interesting work based on the new data
        except Exception as e:
            print(f"An error occurred {e}")
            raise e
    print(f"Successfully processed {len(event['Records'])} records.")
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    print(json.dumps(event, indent=2))

    for record in event['Records']:
        log_dynamodb_record(record)

def log_dynamodb_record(record):
    print(record['eventID'])
    print(record['eventName'])
    print(f"DynamoDB Record: {json.dumps(record['dynamodb'])}")
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Python.

```
import json

def lambda_handler(event, context):
    for record in event.get('events', []):
        log_document_db_event(record)
    return 'OK'
```

```
def log_document_db_event(record):
    event_data = record.get('event', {})
    operation_type = event_data.get('operationType', 'Unknown')
    db = event_data.get('ns', {}).get('db', 'Unknown')
    collection = event_data.get('ns', {}).get('coll', 'Unknown')
    full_document = event_data.get('fullDocument', {})

    print(f"Operation type: {operation_type}")
    print(f"db: {db}")
    print(f"collection: {collection}")
    print("Full document:", json.dumps(full_document, indent=2))
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
```

```
            print('Message:', msg)
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
```

```
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your
 bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["kinesis"]["sequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def handler(event, context):
    records = event.get("Records")
    curRecordSequenceNumber = ""

    for record in records:
        try:
            # Process your record
            curRecordSequenceNumber = record["dynamodb"]["SequenceNumber"]
        except Exception as e:
            # Return failed record's sequence number
            return {"batchItemFailures":[{"itemIdentifier":
 curRecordSequenceNumber}]}

    return {"batchItemFailures":[]}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier": record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Beispiele für Managed Service für Apache Flink unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Python (Boto3) Managed Service für Apache Flink Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Datengenerator

Aktionen

`AddApplicationInput`

Das folgende Codebeispiel zeigt die Verwendung`AddApplicationInput`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_input(self, input_prefix, stream_arn, input_schema):
        """
        Adds an input stream to the application. The input stream data is mapped
        to an in-application stream that can be processed by your code running in
        Kinesis Data Analytics.

        :param input_prefix: The prefix prepended to in-application input stream
 names.
```

```
        :param stream_arn: The ARN of the input stream.
        :param input_schema: A schema that maps the data in the input stream to the
                             runtime environment. This can be automatically
generated
                             by using `discover_input_schema` or you can create it
                             yourself.
        :return: Metadata about the newly added input.
        """
        try:
            response = self.analytics_client.add_application_input(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Input={
                    "NamePrefix": input_prefix,
                    "KinesisStreamsInput": {"ResourceARN": stream_arn},
                    "InputSchema": input_schema,
                },
            )
            self.version_id = response["ApplicationVersionId"]
            logger.info("Add input stream %s to application %s.", stream_arn,
self.name)
        except ClientError:
            logger.exception(
                "Couldn't add input stream %s to application %s.", stream_arn,
self.name
            )
            raise
        else:
            return response
```

- API-Details finden Sie AddApplicationInput in der AWS API-Referenz zum SDK für Python (Boto3).

`AddApplicationOutput`

Das folgende Codebeispiel zeigt die Verwendung. `AddApplicationOutput`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps data
        from the specified in-application stream to the output stream.

        :param in_app_stream_name: The name of the in-application stream to map
                                   to the output stream.
        :param output_arn: The ARN of the output stream.
        :return: A list of metadata about the output resources currently assigned
                 to the application.
        """
        try:
            response = self.analytics_client.add_application_output(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Output={
                    "Name": in_app_stream_name,
                    "KinesisStreamsOutput": {"ResourceARN": output_arn},
                    "DestinationSchema": {"RecordFormatType": "JSON"},
                },
```

```
            )
            outputs = response["OutputDescriptions"]
            self.version_id = response["ApplicationVersionId"]
            logging.info(
                "Added output %s to %s, which now has %s outputs.",
                output_arn,
                self.name,
                len(outputs),
            )
        except ClientError:
            logger.exception("Couldn't add output %s to %s.", output_arn, self.name)
            raise
        else:
            return outputs
```

- API-Details finden Sie AddApplicationOutput in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateApplication`

Das folgende Codebeispiel zeigt die Verwendung. `CreateApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
```

```
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def create(self, app_name, role_arn, env="SQL-1_0"):
        """
        Creates a Kinesis Data Analytics application.

        :param app_name: The name of the application.
        :param role_arn: The ARN of a role that can be assumed by Kinesis Data
                         Analytics and grants needed permissions.
        :param env: The runtime environment of the application, such as SQL. Code
                    uploaded to the application runs in this environment.
        :return: Metadata about the newly created application.
        """
        try:
            response = self.analytics_client.create_application(
                ApplicationName=app_name,
                RuntimeEnvironment=env,
                ServiceExecutionRole=role_arn,
            )
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Application %s created.", app_name)
        except ClientError:
            logger.exception("Couldn't create application %s.", app_name)
            raise
        else:
            return details
```

- API-Details finden Sie CreateApplication in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteApplication`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def delete(self):
        """
        Deletes an application.
        """
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp
            )
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- API-Details finden Sie DeleteApplication in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeApplication`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe(self, name):
        """
        Gets metadata about an application.

        :param name: The name of the application to look up.
        :return: Metadata about the application.
        """
        try:
            response =
 self.analytics_client.describe_application(ApplicationName=name)
            details = response["ApplicationDetail"]
            self._update_details(details)
            logger.info("Got metadata for application %s.", name)
        except ClientError:
            logger.exception("Couldn't get metadata for application %s.", name)
            raise
        else:
```

```
            return details
```

- API-Details finden Sie DescribeApplication in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeApplicationSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeApplicationSnapshot`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def describe_snapshot(self, application_name, snapshot_name):
        """
        Gets metadata about a previously saved application snapshot.

        :param application_name: The name of the application.
        :param snapshot_name: The name of the snapshot.
        :return: Metadata about the snapshot.
        """
```

```
        try:
            response = self.analytics_client.describe_application_snapshot(
                ApplicationName=application_name, SnapshotName=snapshot_name
            )
            snapshot = response["SnapshotDetails"]
            logger.info(
                "Got metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get metadata for snapshot %s of application %s.",
                snapshot_name,
                application_name,
            )
            raise
        else:
            return snapshot
```

- API-Details finden Sie DescribeApplicationSnapshot in der AWS API-Referenz zum SDK für Python (Boto3).

`DiscoverInputSchema`

Das folgende Codebeispiel zeigt die Verwendung. `DiscoverInputSchema`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
```

```
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def discover_input_schema(self, stream_arn, role_arn):
        """
        Discovers a schema that maps data in a stream to a format that is usable by
        an application's runtime environment. The stream must be active and have
        enough data moving through it for the service to sample. The returned schema
        can be used when you add the stream as an input to the application or you
can
        write your own schema.

        :param stream_arn: The ARN of the stream to map.
        :param role_arn: A role that lets Kinesis Data Analytics read from the
stream.
        :return: The discovered schema of the data in the input stream.
        """
        try:
            response = self.analytics_client.discover_input_schema(
                ResourceARN=stream_arn,
                ServiceExecutionRole=role_arn,
                InputStartingPositionConfiguration={"InputStartingPosition": "NOW"},
            )
            schema = response["InputSchema"]
            logger.info("Discovered input schema for stream %s.", stream_arn)
        except ClientError:
            logger.exception(
                "Couldn't discover input schema for stream %s.", stream_arn
            )
            raise
        else:
            return schema
```

- API-Details finden Sie DiscoverInputSchema in der AWS API-Referenz zum SDK für Python (Boto3).

StartApplication

Das folgende Codebeispiel zeigt die Verwendung. `StartApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def start(self, input_id):
        """
        Starts an application. After the application is running, it reads from the
        specified input stream and runs the application code on the incoming data.

        :param input_id: The ID of the input to read.
        """
        try:
            self.analytics_client.start_application(
                ApplicationName=self.name,
                RunConfiguration={
                    "SqlRunConfigurations": [
                        {
```

```
                            "InputId": input_id,
                            "InputStartingPositionConfiguration": {
                                "InputStartingPosition": "NOW"
                            },
                        }
                    ]
                },
            )
            logger.info("Started application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't start application %s.", self.name)
            raise
```

- API-Details finden Sie StartApplication in der AWS API-Referenz zum SDK für Python (Boto3).

`StopApplication`

Das folgende Codebeispiel zeigt die Verwendung. `StopApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None
```

```
    def stop(self):
        """
        Stops an application. This stops the application from processing data but
        does not delete any resources.
        """
        try:
            self.analytics_client.stop_application(ApplicationName=self.name)
            logger.info("Stopping application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't stop application %s.", self.name)
            raise
```

- API-Details finden Sie StopApplication in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateApplication`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateApplication`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird der Code aktualisiert, der in einer vorhandenen Anwendung ausgeführt wird.

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""

    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
```

```
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None


    def update_code(self, code):
        """
        Updates the code that runs in the application. The code must run in the
        runtime environment of the application, such as SQL. Application code
        typically reads data from in-application streams and transforms it in some
way.

        :param code: The code to upload. This completely replaces any existing code
                     in the application.
        :return: Metadata about the application.
        """
        try:
            response = self.analytics_client.update_application(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                ApplicationConfigurationUpdate={
                    "ApplicationCodeConfigurationUpdate": {
                        "CodeContentTypeUpdate": "PLAINTEXT",
                        "CodeContentUpdate": {"TextContentUpdate": code},
                    }
                },
            )
            details = response["ApplicationDetail"]
            self.version_id = details["ApplicationVersionId"]
            logger.info("Update code for application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't update code for application %s.", self.name)
            raise
        else:
            return details
```

- API-Details finden Sie UpdateApplication in der AWS API-Referenz zum SDK für Python (Boto3).

Datengenerator

Generieren eines Streams mit einem Referrer

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit einem Referrer erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"REFERRER": "http://www.amazon.com"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Blutdruckanomalien

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Blutdruckanomalien erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


class PressureType(Enum):
    low = "LOW"
    normal = "NORMAL"
    high = "HIGH"


def get_blood_pressure(pressure_type):
    pressure = {"BloodPressureLevel": pressure_type.value}
    if pressure_type == PressureType.low:
        pressure["Systolic"] = random.randint(50, 80)
        pressure["Diastolic"] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure["Systolic"] = random.randint(90, 120)
        pressure["Diastolic"] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure["Systolic"] = random.randint(130, 200)
        pressure["Diastolic"] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure
```

```
def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low
            if rnd < 0.005
            else PressureType.high
            if rnd > 0.995
            else PressureType.normal
        )
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Daten in Spalten

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Daten in Spalten erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3
```

```
STREAM_NAME = "ExampleInputStream"


def get_data():
    return {"Col_A": "a", "Col_B": "b", "Col_C": "c", "Col_E_Unstructured": "x,y,z"}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Herzfrequenzanomalien

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Herzfrequenzanomalien erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
from enum import Enum
import json
import random
import boto3
```

```
STREAM_NAME = "ExampleInputStream"


class RateType(Enum):
    normal = "NORMAL"
    high = "HIGH"


def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {"heartRate": rate, "rateType": rate_type.value}


def generate(stream_name, kinesis_client, output=True):
    while True:
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey",
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Hotspots

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit einem Hotspot erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_hotspot(field, spot_size):
    hotspot = {
        "left": field["left"] + random.random() * (field["width"] - spot_size),
        "width": spot_size,
        "top": field["top"] + random.random() * (field["height"] - spot_size),
        "height": spot_size,
    }
    return hotspot


def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        "x": rectangle["left"] + random.random() * rectangle["width"],
        "y": rectangle["top"] + random.random() * rectangle["height"],
        "is_hot": "Y" if rectangle is hotspot else "N",
    }
    return {"Data": json.dumps(point), "PartitionKey": "partition_key"}


def generate(
    stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client
):
    """
```

```
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)
        ]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)


if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={"left": 0, "width": 10, "top": 0, "height": 10},
        hotspot_size=1,
        hotspot_weight=0.2,
        batch_size=10,
        kinesis_client=boto3.client("kinesis"),
    )
```

Generieren eines Streams mit Protokolleinträgen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Protokolleinträgen erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "LOGENTRY": "203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] "
        '"GET /index.php HTTP/1.1" 200 125 "-" '
        '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit gestaffelten Daten

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit gestaffelten Daten erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        "EVENT_TIME": event_time.isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey",
            )
            time.sleep(10)


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Börsentickerdaten

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Börsentickerdaten erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )
```

```
if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit zwei Datentypen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit zwei Datentypen erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"


def get_order(order_id, ticker):
    return {
        "RecordType": "Order",
        "Oid": order_id,
        "Oticker": ticker,
        "Oprice": random.randint(500, 10000),
        "Otype": "Sell",
    }


def get_trade(order_id, trade_id, ticker):
    return {
        "RecordType": "Trade",
        "Tid": trade_id,
```

```
        "Toid": order_id,
        "Tticker": ticker,
        "Tprice": random.randint(0, 3000),
    }


def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(["AAAA", "BBBB", "CCCC"])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
 PartitionKey=PARTITION_KEY
        )
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY,
            )
        order_id += 1


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

Generieren eines Streams mit Webprotokolldaten

Das folgende Codebeispiel veranschaulicht, wie Sie einen Kinesis-Stream mit Webprotokolldaten erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "log": "192.168.254.30 - John [24/May/2004:22:01:02 -0700] "
        '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

AWS Marketplace Katalog-API-Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Python (Boto3) with AWS Marketplace Catalog API Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- AMI-Produkte
- Angebote von Channel-Partnern
- Container-Produkte
- Entitäten
- Angebote
- Produkte
- Wiederverkaufsgenehmigung
- SaaS--Produkte
- Dienstprogramme

AMI-Produkte

Hinzufügen einer Dimension zu einem vorhandenen AMI-Produkt und Aktualisieren der Preisbedingungen für das Angebot

Das folgende Codebeispiel zeigt, wie Sie eine Dimension zu einem vorhandenen AMI-Produkt hinzufügen und die Preisbedingungen für das Angebot aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
```

```
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
```

```
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a dimension to an existing
 AMI product and update the offer pricing terms.
CAPI-23
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Add dimension for AMI product")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Hinzufügen einer Region, in der ein AMI-Produkt bereitgestellt wird

Das folgende Codebeispiel zeigt, wie Sie eine Region hinzufügen, in der Sie ein AMI-Produkt bereitstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to add a region where my
AMI product is deployed
CAPI-25A
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher jährlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher jährlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
```

```
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [],
                "AdditionalResources": [],
                "SupportDescription": "For support, contact support@example.com"
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
```

```
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
```

```
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-annual pricing and standard or custom EULA
CAPI-06
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
```

```
        "Create limited AMI product and public offer with hourly-annual pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher monatlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher monatlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
```

```
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
```

```
                        ]
                    },
                    {
                        "Type": "RecurringPaymentTerm",
                        "CurrencyCode": "USD",
                        "BillingPeriod": "Monthly",
                        "Price": "15.0"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public or limited AMI
product and public offer with hourly-monthly pricing and standard or custom EULA
CAPI-08
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "create limited AMI product and public offer with hourly-monthly pricing and
 standard EULA",
    )
```

```
if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines öffentlichen oder limitierten AMI-Produkts und eines öffentlichen Angebots mit stündlicher Preisgestaltung

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes AMI-Produkt und ein öffentliches Angebot mit stündlicher Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
 "arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create a public or limited AMI
 product
and public offer with hourly pricing and standard or custom EULA
CAPI-07
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited AMI product and public offer with hourly pricing and
 standard EULA",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines AMI-Produktentwurf mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen AMI-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an AMI draft product
with a draft public offer.
CAPI-02
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "AMI draft product with draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Einschränken einer Region, in der ein AMI-Produkt bereitgestellt wird

Das folgende Codebeispiel zeigt, wie Sie eine Region einschränken, in der Sie ein AMI-Produkt bereitstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to restrict a region where my
```

```
AMI product is deployed
CAPI-25B
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Add a region where my AMI product is deployed",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Beschränken der Produktsichtbarkeit

Das folgende Codebeispiel zeigt, wie Sie die Sichtbarkeit von Produkten einschränken.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change a product visibility to
 restricted
CAPI-17
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict existing AMI")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Angeben, ob AMI-Komponenten in neuen Regionen bereitgestellt werden

Das folgende Codebeispiel zeigt, wie angegeben wird, ob AMI-Assets in neuen Regionen bereitgestellt werden, die von AWS zur Unterstützung zukünftiger Regionen erstellt wurden.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
```

```
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to modify a product to support all
 future regions
CAPI-26
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update future region support")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Angebote von Channel-Partnern

Erstellen eines CPPO-Entwurfs für einen beliebigen Produkttyp

Das folgende Codebeispiel zeigt, wie Sie einen CPPO-Entwurf für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn für Käufer veröffentlichen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer name"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” CPPO
for any product type (AMI/SaaS/Container) that can be reviewed internally
```

```
before publishing to buyers
CAPI-60
"""
import os

import utils.start_changeset as sc  # noqa: E402
import utils.stringify_details as sd  # noqa: E402

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Create a draft CPPO offer for a product")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Ersatzangebot zur Wiederverkaufsgenehmigung mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein privates Ersatzangebot zur Genehmigung des Wiederverkaufs aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with contract
pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
```

```
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a resale authorization
 replacement private offer
from an existing agreement with contract pricing
CAPI-96
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create resale authorization replacement private offer with contract
 pricing",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Auflisten aller CPPOs, die von einem Channel-Partner erstellt wurden

Das folgende Codebeispiel zeigt, wie Sie alle CPPOs auflisten, die von einem Channel-Partner erstellt wurden.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all Channel Partner Offers
in an account

Program executed with no arguments:
ie. python3 list_all_cppo_offers.py

CAPI-93
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")


def get_offer_entities():
    """
    Returns a list of all offers in the account
    """

    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(Catalog="AWSMarketplace",
 EntityType="Offer")
```

```
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results are paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="Offer",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


def build_offer_list(response_list):
    """
    Cleans up list_entities response list with just list of offer IDs
    """
    offer_list = []

    for response in response_list:
        for entity in response["EntitySummaryList"]:
            offer_list.append(entity["EntityId"])

    return offer_list


def check_offer_resaleauth(offer_id):
    """
    Checks to see if an offer is based on a resale authorization
    """
    offer_response = describe_entity(offer_id)
```

```
    offer_details = json.loads(offer_response["Details"])
    if offer_details is None:
        offer_details = offer_response["DetailsDocument"]
    if "ResaleAuthorizationId" in offer_details and
 offer_details["ResaleAuthorizationId"] is not None:
        return offer_id
    else:
        return None


def describe_entity(entity_id):
    """
    General purpose describe entity call
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )
    except ClientError as e:
        logging.exception(f"Couldn't describe entity. {e}")
        raise

    return response


def get_resaleauth_offers():
    """
    Returns a list of all offers in the account that are
    based on a resale authorization
    """
    resale_offer_list = []

    response_list = get_offer_entities()
    offer_list = build_offer_list(response_list)
    for offer in offer_list:
        print ("offer id " + offer)
        offer_info = check_offer_resaleauth(offer)
        if offer_info is not None:
            resale_offer_list.append(offer_info)

    return resale_offer_list
```

```
if __name__ == "__main__":
    print(get_resaleauth_offers())
```

- API-Details finden Sie ListEntities in der AWS API-Referenz zum SDK für Python (Boto3).

Auflisten aller gemeinsamen Genehmigungen des Wiederverkaufs, die einem Channel-Partner zur Verfügung stehen

Das folgende Codebeispiel zeigt, wie Sie alle gemeinsamen Genehmigung des Wiederverkaufs auflisten, die einem Channel-Partner zur Verfügung stehen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to list all resale authorizations
shared to an account

Program executed with no arguments:
ie. python3 list_all_resale_authorizations.py

CAPI-94
"""

import logging

import boto3
import utils.helpers as hlp  # noqa: E402
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-catalog")
```

```
def get_shared_entities():
    next_token = ""  # nosec: B105
    response_list = []

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType="ResaleAuthorization",
            OwnershipType="SHARED",
        )
    except ClientError as e:
        logging.exception(f"Couldn't list entities. {e}")
        raise

    response_list.append(response)

    # Results can be paginated depending on number of entities returned
    while "NextToken" in response:
        next_token = response["NextToken"]

        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType="ResaleAuthorization",
                OwnershipType="SHARED",
                NextToken=next_token,
            )
        except ClientError as e:
            logging.exception(f"Couldn't list entities. {e}")
            raise

        if "NextToken" in response:
            response_list.append(response)

    return response_list


if __name__ == "__main__":
    response_list = get_shared_entities()
    hlp.pretty_print_datetime(response_list)
```

- API-Details finden Sie ListEntities in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen eines CPPO und Anhängen einer Käufer-EULA

Das folgende Codebeispiel zeigt, wie Sie ein CPPO veröffentlichen und eine Käufer-EULA anhängen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
```

```
                "Type": "CustomEula",
                "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
              }
             ]
          }
      ]
        }
       },
       {
           "ChangeType": "UpdateTargeting",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "PositiveTargeting": {
                   "BuyerAccounts": ["222222222222"]
                   }
           }
       },
       {
           "ChangeType": "UpdateAvailability",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "AvailabilityEndDate": "2023-07-31"
               }
       },
       {
           "ChangeType": "UpdateValidityTerms",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "Terms": [
                   {
                       "Type": "ValidityTerm",
                       "AgreementDuration": "P450D"
                   }
               ]
        }
```

```
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish CPPO
for any product type (AMI/SaaS/Container) and append buyer EULA
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish CPPO for any product type and append buyer EULA",
    )
```

```
    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen eines CPPO mithilfe einer einmaligen Wiederverkaufsgenehmigung und Aktualisieren des Preisaufschlags

Das folgende Codebeispiel zeigt, wie ein CPPO mithilfe einer einmaligen Wiederverkaufsgenehmigung für AMI-, SaaS- oder Container-Produkte veröffentlicht und der Preisaufschlag aktualisiert wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
                }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to Create CPPO using one-time resale
authorization on AMI, SaaS or Container products and update price markup
CAPI-63
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create CPPO using one-time resale authorization and update price markup"
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen eines CPPO-Entwurfs und Aktualisieren des Preisaufschlags

Das folgende Codebeispiel zeigt, wie Sie einen CPPO-Entwurf veröffentlichen und den Preisaufschlag aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPO",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test Offer name",
                "Description":"Test Offer description"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["111111111111"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPO.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to publish “draft” CPPO
for any product type (AMI/SaaS/Container) and update price markup
CAPI-72
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)
```

```
    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Publish draft CPPO for any product type adn update price markup"
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren des Ablaufdatums eines CPPO

Das folgende Codebeispiel zeigt, wie das Ablaufdatum eines CPPO aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update the expiry
date of a CPPO offer
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update the expiry date of a CPPO offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Container-Produkte

Erstellen eines Container-Produktentwurf mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen Container-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
```

```
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create an container draft product
with a draft public offer.
CAPI-03
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft container product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines limitierten Container-Produkts mit einem öffentlichen Angebot und Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein limitiertes Containerprodukt mit einem öffentlichen Angebot, Vertragspreisen und Standard-EULA erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
```

```
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
```

```
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create limited container
product with public offer, contract pricing and standard EULA
CAPI-15
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create limited container product with public offer contract pricing and
 standard EULA",
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Entitäten

Beschreiben aller Entitäten in einem einzigen Anruf

Das folgende Codebeispiel zeigt, wie Sie alle Entitäten in einem einzigen Aufruf beschreiben.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to describe for multiple entities
 information in the AWS Marketplace Catalog
CAPI-98
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"
OFFER_ID = "offer-1111111111111"
MARKETPLACE_CATALOG = "AWSMarketplace"
```

```
def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_entities_information(mp_client):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    entity_request_list_param = [
        {'EntityId': PRODUCT_ID, 'Catalog': MARKETPLACE_CATALOG},
        {'EntityId': OFFER_ID, 'Catalog': MARKETPLACE_CATALOG}
    ]
    try:
        response = mp_client.batch_describe_entities(
            EntityRequestList=entity_request_list_param
        )

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for entities in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = get_entities_information(mp_client)
    print("Successful entities response -")
    pretty_print(response["EntityDetails"])
    print("Failed entities response -")
    pretty_print(response["Errors"])
```

```
if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie BatchDescribeEntities in der AWS API-Referenz zum SDK für Python (Boto3).

Auflisten und Beschreiben aller Angebote, die einem Produkt zugeordnet sind

Das folgende Codebeispiel zeigt, wie Sie alle mit einem Produkt verbundenen Angebote auflisten und beschreiben.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to retrieve all offer information
related to a single product
CAPI-97
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError
from utils import helpers

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-catalog")
```

```
def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def list_entity_details(entity_type, entity_id):
    """
    Returns details about a given entity and entity type
    """

    entity_summary_list = []

    # filter will return details for given entity_id with BuyerAccounts targeting
    filter_list_param = {
        'OfferFilters':{
            'ProductId':{
                'ValueList':[entity_id]
            },
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }

    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
```

```
            EntityType=entity_type,
            EntityTypeFilters = filter_list_param,
            MaxResults=10
        )

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)

    # add results to entity_summary_list
    entity_summary_list.extend(response["EntitySummaryList"])

    # if there are more than 10 offers, paginate through the results
    while "NextToken" in response and response["NextToken"] is not None:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
                EntityTypeFilters = filter_list_param,
                NextToken=response["NextToken"],
                MaxResults=10
            )

        except ClientError as e:
            if e.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Entity ID %s not found.", entity_id)
            else:
                logger.error("Unexpected error: %s", e)

        # add results to entity_summary_list
        entity_summary_list.extend(response["EntitySummaryList"])

        return entity_summary_list

if __name__ == "__main__":
    parser = argparse.ArgumentParser()

    parser.add_argument(
        "--entity-id",
        "-eid",
        help="Provide Entity ID corresponding to a product to filter offers on",
        required=True,
```

```
    )

    args = parser.parse_args()

    # Gets a offers associated with the entity_id
    response = list_entity_details(
        "Offer",
        entity_id=args.entity_id
    )

    if response: # if response is not empty

        # list_entity_details returns a list of offers
        for offer in response:

            print("-"*128)
            print(f"Terms for Offer ID: {offer['EntityId']}")
            print("-"*128)

            #retrieve offer information for each offer
            entity_information = get_entity_information(offer["EntityId"])

            helpers.pretty_print_datetime(entity_information)

    else:
        print(f"No information found for Entity ID: {args.entity_id}")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeEntity
  - ListEntities

Angebote

Erstellen einer benutzerdefinierten Dimension für ein SaaS-Produkt und Erstellen eines privaten Angebots

Das folgende Codebeispiel zeigt, wie Sie eine benutzerdefinierte Dimension für ein SaaS-Produkt erstellen und ein privates Angebot erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
 defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a SaaS product custom
 dimension and private offer
CAPI-91
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create a SaaS product custom dimension and private offer"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebotsentwurfs für ein AMI- oder SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie einen privaten Angebotsentwurf für ein beliebiges AMI- oder SaaS-Produkt erstellen, sodass Sie es intern überprüfen können, bevor Sie es für Käufer veröffentlichen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertrag und Pay-As-You-Go Preisen für ein SaaS-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract with PAYG pricing for my SaaS product
CAPI-34
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with contract with PAYG pricing for my SaaS product",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen und einem flexiblen Zahlungsplan für ein SaaS-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
```

```
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Some kind of refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer
with contract pricing and flexible payment schedule for my SaaS product
CAPI-39
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create private offer with contract pricing and flexible payment schedule
 for my SaaS product",
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit Vertragspreisen für ein Container-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen für ein Container-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for Container product using AWS
Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
```

```
                                        "DimensionKey": "ReqPerHour",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer (target
 buyers)
for my Container product with contract pricing
CAPI-36
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish a private offer for my Container product with contract pricing",
    )

    return response
```

```
if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit Vertragspreisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Vertragspreisen für ein AMI-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 contract pricing for my AMI product
CAPI-35
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with contract pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit stündlichen jährlichen Preisen und einem flexiblen Zahlungsplan für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen und einem flexiblen Zahlungsplan für ein AMI-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
```

```
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P365D",
                        "Grants": [
                            {
                                "DimensionKey": "t2.micro",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing and flexible payment schedule for my AMI product
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with hourly annual pricing and flexible payment
 schedule for my AMI product",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit stündlichen jährlichen Preisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen jährlichen Preisen für ein AMI-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
```

```
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly annual pricing for my AMI product
CAPI-31
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly annual pricing for my AMI
 product"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit stündlichen Preisen für ein AMI-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit stündlichen Preisen für ein AMI-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-01-01"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 hourly pricing for my AMI product
CAPI-32
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "create private offer with hourly pricing for my AMI product"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit Abonnementpreisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 subscription pricing for my SaaS product
CAPI-33
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "Create private offer with subscription pricing for my SaaS
 product"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Angebots mit gestaffelten Preisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein privates Angebot mit gestaffelten Preisen für ein SaaS-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a private offer with
 tiered contract pricing for my SaaS product
```

```
CAPI-XX
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create private offer with tiered contract pricing for my SaaS product",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz für SDK für Python (Boto3).

Erstellen eines öffentlichen kostenlosen Testangebots mit Abonnementpreisen für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches kostenloses Testangebot mit Abonnementpreisen für ein SaaS-Produkt erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
 pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
```

```
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a public free trial offer
 with subscription pricing for SaaS product
CAPI-13
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create public free trial offer with subscription pricing for SaaS product",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines privaten Ersatzangebots mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein privates Ersatzangebot aus einer bestehenden Vereinbarung mit Vertragspreisen erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-111111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
 for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
```

```
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a replacement private
 offer
from an existing agreement with contract pricing
CAPI-95
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create replacement private offer with contract pricing..",
    )

    return response


if __name__ == "__main__":
```

```
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Beschreiben eines öffentlichen Angebots

Das folgende Codebeispiel veranschaulicht, wie Sie ein öffentliches Angebot beschreiben.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for offer information in the
 AWS Marketplace Catalog
CAPI-29
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

OFFER_ID = "offer-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)
```

```
def get_offer_information(mp_client, entity_id):
    """
    Returns information about a given offer
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of offer information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Offer with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an offer in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_offer_information(mp_client, OFFER_ID))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DescribeEntity in der AWS API-Referenz zum SDK für Python (Boto3).

Ablaufen des Entwurfs eines privaten Angebots

Das folgende Codebeispiel zeigt, wie Sie das Ablaufdatum eines privaten Angebots auf ein Datum in der Vergangenheit festlegen, sodass Käufer das Angebot nicht mehr sehen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create “draft” Private Offer
for any AMI or SAAS product type that can be reviewed internally
before publishing to buyers
CAPI-30
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Private offer for AMI product")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Auflisten aller privaten Angebote

Das folgende Codebeispiel zeigt, wie Sie alle privaten Angebote auflisten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) for listing offers in the AWS
 Marketplace Catalog
CAPI-40
```

```
"""
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# Constants
MAX_RESULTS = 10
CATALOG = "AWSMarketplace"
ENTITY_TYPE = "Offer"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def list_private_offers(mp_client, return_all_private_offers):
    """
    This method retrieves list of all Private Offers for this account.
    """
    entity_summary_list = []
    filter_list_param = {
        'OfferFilters': {
            'Targeting': {
                'ValueList': ["BuyerAccounts"]
            }
        }
    }
    try:
        response = mp_client.list_entities(
            Catalog=CATALOG,
            EntityType=ENTITY_TYPE,
            EntityTypeFilters=filter_list_param,
            MaxResults=MAX_RESULTS
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request: %s", e)
        raise

    entity_summary_list.extend(response["EntitySummaryList"])
```

```
    logger.info("Number of results in first iteration: %d " %
 len(entity_summary_list))

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response and return_all_private_offers:
        try:
            logger.info("Getting Next Token results: %s " % response["NextToken"])
            response = mp_client.list_entities(
                Catalog=CATALOG,
                EntityType=ENTITY_TYPE,
                EntityTypeFilters=filter_list_param,
                MaxResults=MAX_RESULTS,
                NextToken=response["NextToken"]
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request: %s", e)
            raise

        entity_summary_list.extend(response["EntitySummaryList"])
        logger.info(
            "Number of results in the current iteration: %d "
            % len(response["EntitySummaryList"])
        )

    return entity_summary_list


def get_offer_details(mp_client, offer):
    """
    Describe the details of the Offer.
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=offer["EntityId"]
        )

        return response
    except ClientError:
        logger.exception("Error: Couldn't get details of the Offer.")
        raise


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Demo  - List Private offers.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    # Get list of all Offers.
    private_offers = list_private_offers(mp_client, False)
    count = len(private_offers)

    logger.info("Number of Offers: %d " % count)
    offer_counter = 0
    # Display details of each Offer.
    for offer in private_offers:
        print("-" * 88)
        offer_counter += 1
        print("Displaying Offer details for Offer# %d" % offer_counter)
        entity = get_offer_details(mp_client, offer)
        pretty_print(entity)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren Sie ein Angebot, um einen Vertrag mit Preisangaben abzuschließen Pay-As-You-Go

Das folgende Codebeispiel zeigt, wie ein Angebot aktualisiert wird, um einen Vertrag mit Pay-As-You-Go Preisangaben abzuschließen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
```

```
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply contract
 with PAYG pricing
CAPI-21
"""

import os
```

```
import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer to apply contract with PAYG pricing")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren eines Angebots, um einen stündlichen jährlichen Preis anzuwenden

Das folgende Codebeispiel zeigt, wie Sie ein Angebot aktualisieren, um einen stündlichen jährlichen Preis anzuwenden.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply hourly
 annual pricing
CAPI-20
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer with hourly annual pricing")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren eines Angebots, um das Targeting auf bestimmte geografische Regionen anzuwenden

Das folgende Codebeispiel zeigt, wie Sie ein Angebot aktualisieren, um das Targeting auf bestimmte geografische Regionen anzuwenden.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update an offer to apply
 targeting to certain geographic regions.
CAPI-19
```

```
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer targeting")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren des Namens und der Beschreibung eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie den Namen und die Beschreibung eines öffentlichen Angebots aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update name and description of my
 offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update name and description of my offer")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren der EULA eines Angebots

Das folgende Codebeispiel zeigt, wie Sie die EULA eines Angebots aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update EULA of my offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update EULA of my offer")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren des Ablaufdatums eines privaten Angebots auf ein zukünftiges Datum

Das folgende Codebeispiel zeigt, wie das Ablaufdatum eines privaten Angebots auf ein zukünftiges Datum aktualisiert wird, um Käufern mehr Zeit zu geben, das Angebot zu bewerten und anzunehmen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to set expiry date of a private
 offer to a date in the future so that my buyers get more time to evaluate and
 accept the offer.
CAPI-37
"""

import os

import utils.start_changeset as sc
```

```
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update offer expiration date")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren der Dauer der kostenlosen Testversion eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie die Dauer der kostenlosen Testversion eines öffentlichen kostenlosen Testangebots für ein SaaS-Produkt aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to change free trial duration for my
 SaaS product by modifying my free trial public offer
CAPI-14
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "Change free trial duration for SaaS product")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren der Rückerstattungsrichtlinie eines Angebots

Das folgende Codebeispiel zeigt, wie Sie die Rückerstattungsrichtlinie eines Angebots aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
```

```
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update refund policy of my offer
CAPI-18
"""

import os

import utils.start_changeset as sc  # type: ignore
import utils.stringify_details as sd  # type: ignore

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Update refund policy of my public offer")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Produkte

Beschreiben eines AMI-, SaaS- oder Container-Produkts

Das folgende Codebeispiel zeigt, wie Sie ein AMI-, SaaS- oder Container-Produkt beschreiben und prüfen, ob es alle Informationen enthält, die Sie über das Produkt haben möchten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
CAPI-28
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

PRODUCT_ID = "prod-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
```

```
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, PRODUCT_ID))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DescribeEntity in der AWS API-Referenz zum SDK für Python (Boto3).

Auflisten aller AMI-, SaaS- oder Container-Produkte und zugehöriger öffentlicher Angebote

Das folgende Codebeispiel zeigt, wie Sie alle AMI-, SaaS- oder Container-Produkte und zugehörige öffentliche Angebote auflisten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to display information about AMI
 products and their associated offers in the AWS Marketplace Catalog
CAPI-27
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

try:
    mp_client = boto3.client("marketplace-catalog")
except ClientError as e:
    logger.error("Could not create boto3 client.")
    raise


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def describe_entity(entity_id):
    """
    Returns entity details
    Args: entity_id str: The entity ID of the product or offer
```

```
    Returns: dict: The entity details
    """
    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace", EntityId=entity_id
        )
    except ClientError as e:
        logger.error("Could not complete describe_entity request.")
        raise

    # De-stringify the details
    response["Details"] = json.loads(response["Details"])

    return response


def get_entities(entity_type, visibility=None):
    """
    Returns list of entities for provided entity_type
    Args: entity_type str: Type of entity list to return, in our case AmiProduct or
 Offer
    Returns: list: Abbreviated list of entity information
    """
    EntitySummaryList = []

    # Get the first page of results
    try:
        response = mp_client.list_entities(
            Catalog="AWSMarketplace",
            EntityType=entity_type,
            MaxResults=MAX_PAGE_RESULTS,
        )
    except ClientError as e:
        logger.error("Could not complete list_entities request.")
        raise

    EntitySummaryList.extend(response["EntitySummaryList"])

    # Get subsequent pages of results if previous response contained a NextToken
    while "NextToken" in response:
        try:
            response = mp_client.list_entities(
                Catalog="AWSMarketplace",
                EntityType=entity_type,
```

```
                MaxResults=MAX_PAGE_RESULTS,
                NextToken=response["NextToken"],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        EntitySummaryList.extend(response["EntitySummaryList"])

    # if visibility is provided, filter the list to only include entities with that
 visibility
    if visibility is not None:
        EntitySummaryList = [
            entity for entity in EntitySummaryList if entity["Visibility"] ==
 visibility
        ]

    return EntitySummaryList


def get_enhanced_product_list(entity_type):
    """
    Returns an enhanced list of products with product details and offer details
    Args: entity_type str: Type of entity list to return, in our case AmiProduct
    Returns: list: Enhanced list of dictionary objects containing product and offer
 details
    """

    product_list = get_entities(entity_type)

    # Loop through product list and append product details to each product
    for product in product_list:
        # appends product details to product dictionary
        product["ProductDetails"] = describe_entity(product["EntityId"])["Details"]
        # creating an empty list for offer details
        product["OfferDetailsList"] = []

    return product_list


def attach_offer_details(product_list):
    """
    Loops through offer information and appends offer details to product list
    Args: product_list list: List of product dictionaries
```

```
    Returns: list: Enhanced list of dictionary objects containing product and offer
 details
    """
    offer_list = get_entities("Offer", "Public")

    # Loop through offer list and append offer details to each product
    for offer in offer_list:
        offer["OfferDetails"] = describe_entity(offer["EntityId"])["Details"]

        # Extracts product-id from offer
        product_id = offer["OfferDetails"]["ProductId"]

        # Determines if product-id referenced in offer matches product-id in product
 list
        product_dict = next(
            filter(lambda product: product["EntityId"] == product_id, product_list),
            None,
        )

        # If product-id matches, appends offer details to product dictionary
        if product_dict is not None:
            # logger.info(f"Offer product Id {offer['OfferDetails']['ProductId']}
 found in product dictionary. Updating product dictionary with offer details")
            product_dict["OfferDetailsList"].append(offer["OfferDetails"])

        else:
            # logger.info("Offer product Id {offer['OfferDetails']['ProductId']} not
 found. Skipping offer details update")
            pass

    return product_list


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieving products and public offer information....")
    print("-" * 88)

    # Builds a list of products and their details
    product_list = get_enhanced_product_list("AmiProduct")

    # Queries offer information and attaches it to the product list
```

```
    product_offer_list = attach_offer_details(product_list)

    pretty_print(product_offer_list)
    return product_offer_list


if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - DescribeEntity
  - ListEntities

Wiederverkaufsgenehmigung

Erstellen eines Entwurfs einer Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie den Entwurf einer Wiederverkaufsgenehmigung für einen beliebigen Produkttyp erstellen, sodass Sie ihn intern überprüfen können, bevor Sie ihn bei einem Vertriebspartner veröffentlichen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
 can use that to create Channel Partner Private Offer (CPPO)
CAPI-41
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "draft resale auth")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Beschreiben einer Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie eine Wiederverkaufsgenehmigung beschreiben.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) search for product information in
 the AWS Marketplace Catalog
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

resaleAuthorizationId = "resaleauthz-1111111111111"


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


def get_product_information(mp_client, entity_id):
    """
    Returns information about a given product
```

```
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of product information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Product with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for a product in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    pretty_print(get_product_information(mp_client, resaleAuthorizationId))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DescribeEntity in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaliger Wiederverkaufsgenehmigung mit einem privaten Angebot

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung mit einem privaten Angebot veröffentlichten sodass ein Channel-Partner damit ein Channel Partner Private Offer (CPPO) erstellen kann.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product so my CP
can use that to create Channel Partner Private Offer (CPPO)
CAPI-42
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
```

```
        stringified_change_set, "onetime resale auth with private offer"
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum

Das folgende Codebeispiel zeigt, wie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlicht wird, sodass ein Channel-Partner damit ein CPPO erstellen kann.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
```

```
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI product so
 my CP can use that to create Channel Partner Private Offer (CPPO)
```

```
CAPI-48
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multi-use resale auth with expiry date")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum und einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/Container
 product and add custom EULA to be sent to the buyer
CAPI-56
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multiuse resale auth with expiry date and custom
 EULA")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit einem Ablaufdatum und von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleUsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with expiry date on my SaaS/AMI/Container
 product
and add reseller contract documentation between the ISV and channel partner
CAPI-57
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)
```

```
change_set = sd.stringify_changeset(change_set_file)


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "multi use resale auth with contract doc",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung mit Ablauf und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung mit einem Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization with expiry date for any product type (AMI/
SaaS/Container) and add specific buyer account for the resale
CAPI-82
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
```

```
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Publish multi-use resale authorization with expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne ein Ablaufdatum

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne ein Ablaufdatum für ein AMI-Produkt mit stündlichen jährlichen Preisen veröffentlichen, sodass ein Channel-Partner damit ein CPPO erstellen kann.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
```

```
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI product
 so my CP can use that to create Channel Partner Private Offer (CPPO)
CAPI-52
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "multi use resale auth with no expiry date")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablaufdatum und einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add custom EULA to be sent to the buyer
```

```
CAPI-58
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and custom EULA"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablaufdatum und von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
```

```
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a multi-use resale authorization with no expiry date on my SaaS/AMI/
Container product and add reseller contract documentation between the ISV and
 channel partner
CAPI-59
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set, "multi use resale auth with no expiry date and contract doc"
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer mehrfach verwendbaren Wiederverkaufsgenehmigung ohne Ablauf und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine mehrfach verwendbare Wiederverkaufsgenehmigung ohne Ablaufdatum für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
```

```
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish multi-use resale authorization without expiry date for any product type
 (AMI/SaaS/Container) and add specific buyer account for the resale
CAPI-83
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "publish multi-use resale authorization without expiry date and add specific
 buyer account",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen eines flexiblen Zahlungsplans

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und einen flexiblen Zahlungsplan hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
```

```
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ResalePaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2023-09-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2023-12-01",
                                "ChargeAmount": "250.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-06-30",
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container) and
 add Flexible payment schedule
CAPI-78
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "onetime resale auth with flexible payment schedule")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen einer EULA

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und eine benutzerdefinierte EULA hinzufügen, die an den Käufer gesendet werden soll.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
```

```
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
```

```
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 custom EULA to be sent to the buyer
```

```
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with custom EULA")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen eines bestimmten Käuferkontos

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und ein bestimmtes Käuferkonto für den Wiederverkauf hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
```

```
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container) and
 add specific buyer account for the resale
CAPI-81
"""



import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)
```

```
def main():
    sc.usage_demo(change_set, "onetime resale authorization for specific buyer
 account")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen von Vertragsunterlagen für Wiederverkäufer

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und Vertragsunterlagen für Wiederverkäufer zwischen dem ISV- und dem Channel-Partner hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
```

```
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish a one-time resale authorization on my SaaS/AMI/Container product and add
 reseller contract documentation between the ISV and channel partner
CAPI-47
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "onetime resale auth with reseller contract doc")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen einer einmaligen Wiederverkaufsgenehmigung und Hinzufügen, ob es sich um eine Verlängerung handelt

Das folgende Codebeispiel zeigt, wie Sie eine einmalige Wiederverkaufsgenehmigung für einen beliebigen Produkttyp veröffentlichen und hinzufügen, ob es sich um eine Verlängerung handelt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish one-time resale authorization for any product type (AMI/SaaS/Container)
and add whether it is renewal or not
CAPI-90
"""
```

```
import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(stringified_change_set, "onetime resale auth renewal")

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Beschränken der Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie eine Wiederverkaufsgenehmigung beschränken.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Restrict a  authorization for any product type (AMI/SaaS/Container)
CAPI-84
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "Restrict resale authorization")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren des Namens und der Beschreibung der einmaligen oder mehrfach verwendbaren Wiederverkaufsgenehmigung

Das folgende Codebeispiel zeigt, wie Sie den Namen und die Beschreibung einer einmaligen oder mehrfach verwendbaren Wiederverkaufsgenehmigung vor der Veröffentlichung für einen beliebigen Produkttyp aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Update name/description of one-time or multi-use resale authorization before
 publishing for any product type (AMI/SaaS/Container)
CAPI-77
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "update name and description of one-time or multi-use resale authorization
 before publishing",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

SaaS--Produkte

Erstellen eines SaaS-Produktentwurfs mit einem Entwurf eines öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie einen SaaS-Produktentwurf mit einem Entwurf eines öffentlichen Angebots erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
```

```
Shows how to use the AWS SDK for Python (Boto3) to create an SaaS draft product
with a draft public offer.
CAPI-04
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Create a draft saas product with a draft public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines öffentlichen oder limitierten SaaS-Produkts und eines öffentlichen Angebots mit Vertragspreisen

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit nutzungsabhängiger Preisberechnung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
```

```
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
```

```
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract pricing and standard
 EULA
CAPI-11
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create a limited saas product with a public offer with contract pricing",
    )


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen Sie ein öffentliches oder beschränktes SaaS-Produkt und ein öffentliches Angebot mit Vertrag und Preisgestaltung Pay-As-You-Go

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder beschränktes SaaS-Produkt und ein öffentliches Angebot mit einem Vertrag mit Pay-As-You-Go Preisgestaltung erstellen. In diesem Beispiel wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
```

```
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
```

```
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with contract with PAYG pricing and
 standard EULA
CAPI-10
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(
        change_set,
        "Create limited SaaS product with public offer with contract with payg
 pricing",
    )


if __name__ == "__main__":
```

```
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen eines öffentlichen oder limitierten SaaS-Produkts und eines öffentlichen Angebots mit Abonnementpreisen

Das folgende Codebeispiel zeigt, wie Sie ein öffentliches oder limitiertes SaaS-Produkt und ein öffentliches Angebot mit Abonnementpreisen erstellen. In diesen Beispielen wird entweder eine standardmäßige oder eine benutzerdefinierte EULA erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
```

```
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
```

```
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
```

```
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to create a
public or limited SaaS product and public offer with subscription(usage) pricing and
 standard EULA
CAPI-09
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd

fname = "changeset.json"
change_set_file = os.path.join(os.path.dirname(__file__), fname)

change_set = sd.stringify_changeset(change_set_file)


def main():
    sc.usage_demo(change_set, "public saas public offer with subscription pricing")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Veröffentlichen eines SaaS-Produkts und des zugehörigen öffentlichen Angebots

Das folgende Codebeispiel zeigt, wie Sie ein SaaS-Produkt und das zugehörige öffentliche Angebot veröffentlichen. Das Produkt befindet sich standardmäßig in einem eingeschränkten Zustand.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
```

```
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
```

```
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Publish my SaaS product and associated public offer (product will be in limited
 state by default)
CAPI-05A
"""

import os

import utils.start_changeset as sc
import utils.stringify_details as sd
```

```
def main(change_set=None):
    if change_set is None:
        fname = "changeset1.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "publish saas product and associated public offer",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren der Dimensionen auf einem AMI- oder SaaS-Produkt

Das folgende Codebeispiel zeigt, wie Sie die Dimensionen auf einem AMI- oder SaaS-Produkt aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

Führen Sie dieses Skript aus, um das Changeset zu starten. Hilfsfunktionen sind in Dienstprogramme zum Starten eines Changesets im Abschnitt Dienstprogramme definiert.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to update (e.g name) dimensions on
 my AMI or SaaS product
CAPI-24
"""


import os

import utils.start_changeset as sc
import utils.stringify_details as sd


def main(change_set=None):
    if change_set is None:
```

```
        fname = "changeset.json"
        change_set_file = os.path.join(os.path.dirname(__file__), fname)
        stringified_change_set = sd.stringify_changeset(change_set_file)

    else:
        stringified_change_set = change_set

    response = sc.usage_demo(
        stringified_change_set,
        "Update name dimensions on my AMI or SaaS product",
    )

    return response


if __name__ == "__main__":
    main()
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

Dienstprogramme

Dienstprogramme zum Starten eines Changesets

Das folgende Codebeispiel zeigt, wie Sie Dienstprogramme zum Starten eines Changesets definieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

Dienstprogramme zum Starten eines Changesets.

```
"""
Purpose:
```

```
Generic function to start a changeset
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def generate_changeset(mp_client, change_set, change_set_name):
    """
    Start changeset
    """
    try:
        response = mp_client.start_change_set(
            Catalog="AWSMarketplace",
            ChangeSet=change_set,
            ChangeSetName=change_set_name,
        )
        logger.info("Changeset created!")
        logger.info("ChangeSet ID: %s", response["ChangeSetId"])
        logger.info("ChangeSet ARN: %s", response["ChangeSetArn"])

        return response

    except ClientError as e:
        logger.exception("Unexpected error: %s", e)
        raise


def usage_demo(change_set, change_set_name):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Executing changeset: " + change_set_name)
    print("-" * 88)

    mp_client = boto3.client("marketplace-catalog")

    response = generate_changeset(mp_client, change_set, change_set_name)

    return response
```

```
    print("-" * 88)
```

Dienstprogramm zum Laden eines Changesets aus einer JSON-Datei.

```
"""
Purpose:

This module will stringify the details sections of a changeset file.
"""

import json


def pretty_print(response):
    json_object = json.dumps(response, indent=4)
    print(json_object)


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


def stringify_details_sections(json_object):
    """
    Loops through every change type in the changeset to look for non-empty
    details section and stringifies them
    """
    for change_type in json_object["ChangeSet"]:
        # Only stringify details section if it is not empty
        if "Details" in change_type and change_type["Details"] != "{}":
            string_details = json.dumps(change_type["Details"])
            change_type["Details"] = string_details
        else:
            pass

    return json_object["ChangeSet"]


def stringify_changeset(file_path):
```

```
    changeset_file = open_json_file(file_path)
    changeset_stringified = stringify_details_sections(changeset_file)

    return changeset_stringified
```

- API-Details finden Sie StartChangeSet in der AWS API-Referenz zum SDK für Python (Boto3).

AWS Marketplace Beispiele für Vereinbarungs-APIs unter Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der AWS SDK für Python (Boto3) with AWS Marketplace Agreement API Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Vereinbarungen

Vereinbarungen

Akzeptieren Sie eine Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung akzeptiert wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os
```

```
sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"

    @staticmethod
    def accept_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_cancellation_request(
            agreementId=AcceptAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=AcceptAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Description: " + str(response.get("description", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    AcceptAgreementCancellationRequest.accept_agreement_cancellation_request()
```

- API-Details finden Sie AcceptAgreementCancellationRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Akzeptieren Sie eine Zahlungsaufforderung für eine Vereinbarung

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung akzeptiert wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class AcceptAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    PURCHASE_ORDER_REFERENCE = "<PURCHASE ORDER REFERENCE HERE>"

    @staticmethod
    def accept_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.accept_agreement_payment_request(
            agreementId=AcceptAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=AcceptAgreementPaymentRequest.PAYMENT_REQUEST_ID,

 purchaseOrderReference=AcceptAgreementPaymentRequest.PURCHASE_ORDER_REFERENCE,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    AcceptAgreementPaymentRequest.accept_agreement_payment_request()
```

- API-Details finden Sie AcceptAgreementPaymentRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Verlängerungsdauer einer SaaS-Vertragsvereinbarung ändern

Das folgende Codebeispiel zeigt, wie Sie eine SaaS-Vertragsvereinbarung ändern, um ihre Verlängerungsdauer hinzuzufügen oder zu aktualisieren.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model and then
 turn on
the auto-renewal setting using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a public offer that supports
auto-renewal. After acceptance, the buyer decides to amend the agreement to enable
auto-renewal via the RenewalTerm configuration.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P1M for 1 month).
  - DIMENSION_1_KEY — the dimension key defined in the offer.
"""

import boto3
```

```
from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendSaaSContractRenewalTerm:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P1M" for 1 month, "P12M" for 1 year).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the RenewalTerm in your offer.
    RENEWAL_TERM_ID = "<your-renewal-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"

    @staticmethod
    def amend_saas_contract_agreement_renewal_term():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with auto-renewal
 disabled.
        2. Wait for entitlements to become active.
        3. Amend the agreement to enable auto-renewal.
        """
```

```
        client = boto3.client("marketplace-agreement")
        cls = AmendSaaSContractRenewalTerm

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        # Initial agreement: auto-renewal disabled.
        renewal_term = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": False,
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        support_term = {"id": cls.SUPPORT_TERM_ID}

        # --- Create and accept the initial SaaS agreement request with CONTRACT
pricing model ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term,
legal_term, support_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
 agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Amend: enable auto-renewal ---
        renewal_term_amended = {
            "id": cls.RENEWAL_TERM_ID,
            "configuration": {
                "renewalTermConfiguration": {
                    "enableAutoRenew": True,
                }
            },
        }

        # Use Intent.AMEND and sourceAgreementIdentifier to target the existing
 agreement.
        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[configurable_upfront_pricing_term, renewal_term_amended,
 legal_term, support_term],
            sourceAgreementIdentifier=agreement_id,
        )
        print("Amend agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print("Amendment accepted. Auto-renewal enabled. New AgreementId: " +
 aar_response["agreementId"])


if __name__ == "__main__":
    AmendSaaSContractRenewalTerm.amend_saas_contract_agreement_renewal_term()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Ändern Sie ein AMI ConfigurableUpfrontPricingTerm für das Nutzungspreismodell

Das folgende Codebeispiel zeigt, wie eine AMI-Vereinbarung mit geändert werden kann, ConfigurableUpfrontPricingTerm indem das Modell für die Dimensionsmenge für das Nutzungspreismodell aktualisiert wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create an AMI agreement with ConfigurableUpfrontPricingTerm and
 then amend
the dimension quantity using the AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm (UBPT) — accepted first to establish
 the base agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
 UBPT agreement
     entitlements are active.
Once both agreement entitlements are available, this sample shows how to amend the
 agreement
with configurableUpfrontPricingTerm (CUPT) to increase the dimension quantity.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D for one year).
  - DIMENSION_1_KEY — the dimension key defined in the offer (e.g., instance type).
```

```
  - DIMENSION_1_VALUE — initial quantity; NEW_DIMENSION_1_VALUE — amended quantity.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # The dimension key defined in your offer (e.g., an EC2 instance type like
 "c6gn.medium").
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Initial quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in your offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # New quantity to use when amending the dimension of CUPT.
    NEW_DIMENSION_1_VALUE = 5

    @staticmethod
```

```
    def amend_ami_cupt_agreement():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm.
        2. Wait for entitlements to become active, then create and accept an
agreement request
           with configurableUpfrontPricingTerm (CUPT).
        3. Wait for CUPT entitlements to become active, then amend the dimension
quantity.
        """
        client = boto3.client("marketplace-agreement")
        cls = AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " + agreement_id)

        # Wait for entitlements to become active before creating the agreement with
CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
```

```
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
cupt_agreement_id)

        # Wait for entitlements to become active before amending.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Amend Agreement with CUPT ---
        # Increase the dimension quantity using Intent.AMEND and
sourceAgreementIdentifier.
        new_config = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
```

```
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.NEW_DIMENSION_1_VALUE,  # Increase
 quantity for this dimension key
                        },
                    ],
                }
            },
        }

        car_amend_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="AMEND",
            requestedTerms=[new_config, legal_term, validity_term],
            sourceAgreementIdentifier=cupt_agreement_id,
        )
        print(
            "Amendment of CUPT agreement request created. AgreementRequestId: "
            + car_amend_response["agreementRequestId"]
        )

        aar_amend_response = client.accept_agreement_request(
            agreementRequestId=car_amend_response["agreementRequestId"]
        )
        print("Amendment accepted. New AgreementId: " +
 aar_amend_response["agreementId"])


if __name__ == "__main__":

 AmendAmiConfigurableUpfrontPricingTermForUsagePricingModel.amend_ami_cupt_agreement()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion

Das folgende Codebeispiel zeigt, wie Sie eine neue Vereinbarung für eine kostenlose AMI-Testversion erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create an AMI Free Trial agreement
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to an AMI product that offers a free trial period.
The free trial includes a FreeTrialPricingTerm alongside a UsageBasedPricingTerm.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class NewAmiFreeTrial:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the FreeTrialPricingTerm in your offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the UsageBasedPricingTerm in your offer (applies after the trial
 ends).
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the SupportTerm in your offer.
    SUPPORT_TERM_ID = "<your-support-term-id>"
```

```
    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    @staticmethod
    def create_and_accept_ami_free_trial_agreement_request():
        """
        Create an AMI Free Trial agreement.

        The FreeTrialPricingTerm grants access at no cost for the trial period.
        The UsageBasedPricingTerm defines the charges that apply once the trial
 ends.
        """
        client = boto3.client("marketplace-agreement")

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[
                {"id": NewAmiFreeTrial.FREE_TRIAL_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.USAGE_BASED_PRICING_TERM_ID},
                {"id": NewAmiFreeTrial.SUPPORT_TERM_ID},
                {"id": NewAmiFreeTrial.LEGAL_TERM_ID},
            ],

 agreementProposalIdentifier=NewAmiFreeTrial.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "Agreement request with freeTrialPricingTerm accepted. AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewAmiFreeTrial.create_and_accept_ami_free_trial_agreement_request()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung

Das folgende Codebeispiel zeigt, wie Sie eine neue SaaS-Vertragsvereinbarung mit Vorauszahlung erstellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with CONTRACT pricing model with upfront
 payment
using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product using a
 ConfigurableUpfrontPricingTerm,
selecting an agreement duration and specifying quantities for multiple dimensions.
Tax estimation is enabled at the time of agreement creation.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P12M for 12 months).
  - DIMENSION_1_KEY, DIMENSION_2_KEY — dimension keys defined in the offer.
  - DIMENSION_1_VALUE, DIMENSION_2_VALUE — quantities for each dimension.
"""

import boto3

from utils.agreement_api_utils import generate_client_token
```

```
class NewSaaSContractWithUpfrontPayment:

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P12M" for 12 months).
    SELECTOR_VALUE = "<your-selector-value>"

    # First dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-1-key>"
    DIMENSION_1_VALUE = 10

    # Second dimension key and quantity defined in your offer.
    DIMENSION_2_KEY = "<your-dimension-2-key>"
    DIMENSION_2_VALUE = 20

    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Tax estimation setting: "ENABLED" to include estimated taxes in the agreement.
    TAX_ESTIMATION = "ENABLED"

    @staticmethod
    def create_saas_contract_agreement():
        """
        Create a SaaS agreement with CONTRACT pricing model with configurable
 upfront pricing
        for multiple dimensions and tax estimation.
        """
        client = boto3.client("marketplace-agreement")

        configurable_upfront_pricing_term = {
            "id":
 NewSaaSContractWithUpfrontPayment.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue":
 NewSaaSContractWithUpfrontPayment.SELECTOR_VALUE,
                    "dimensions": [
                        {
```

```
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_1_VALUE,
                        },
                        {
                            "dimensionKey":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_KEY,
                            "dimensionValue":
 NewSaaSContractWithUpfrontPayment.DIMENSION_2_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": NewSaaSContractWithUpfrontPayment.LEGAL_TERM_ID}

        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term],
            taxConfiguration={"taxEstimation":
 NewSaaSContractWithUpfrontPayment.TAX_ESTIMATION},

 agreementProposalIdentifier=NewSaaSContractWithUpfrontPayment.AGREEMENT_PROPOSAL_IDENTIFIER
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print(
            "SaaS agreement request with CONTRACT pricing model accepted.
 AgreementId: "
            + accept_response["agreementId"]
        )


if __name__ == "__main__":
    NewSaaSContractWithUpfrontPayment.create_saas_contract_agreement()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen aller Vereinbarungs-IDs

Das folgende Codebeispiel zeigt, wie Sie alle Vereinbarungs-IDs abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement ids
AG-09
"""

import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10


def get_agreements():
    AgreementSummaryList = []
    agreement_id_list = []
```

```
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreements["agreementViewSummaries"])

    for agreement in AgreementSummaryList:
        agreement_id_list.append(agreement["agreementId"])

    return agreement_id_list


if __name__ == "__main__":
    agreement_id_list = get_agreements()
```

```
    print(agreement_id_list)
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen aller Vereinbarungen

Das folgende Codebeispiel zeigt, wie Sie alle Vereinbarungen abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreements
AG-01
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
```

```
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

party_type_list = ["Acceptor"]
agreement_type_list = ["PurchaseAgreement"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    return agreement_results_list
```

```
if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
    helper.pretty_print_datetime(agreements_list)
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der Kunden-ID aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Kunden-ID aus einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get customer AWS account id
from a given agreement
AG-08
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    print(f"Customer account: {response['acceptor']['accountId']}")
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Informieren Sie sich über die Einzelheiten einer Anfrage zur Kündigung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Details zu einer bestimmten Anfrage zur Kündigung einer Vereinbarung abrufen können.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"

    @staticmethod
    def get_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_cancellation_request(
            agreementId=GetAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=GetAgreementCancellationRequest.AGREEMENT_CANCELLATION_REQUE
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    GetAgreementCancellationRequest.get_agreement_cancellation_request()
```

- API-Details finden Sie GetAgreementCancellationRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Informieren Sie sich über die Einzelheiten einer Zahlungsaufforderung für eine Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Details einer bestimmten Zahlungsaufforderung für eine Vereinbarung abrufen können.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class GetAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"

    @staticmethod
    def get_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.get_agreement_payment_request(
            agreementId=GetAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=GetAgreementPaymentRequest.PAYMENT_REQUEST_ID,
```

```
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge ID: " + str(response.get("chargeId", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    GetAgreementPaymentRequest.get_agreement_payment_request()
```

- API-Details finden Sie GetAgreementPaymentRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen finanzieller Angaben aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie finanzielle Angaben aus einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain financial details, such as Total Contract Value of the agreementfrom a given
 agreement
AG-14
```

```
Example Usage: python3 get_agreement_financial_details.py --agreement-id <agreement-
id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_information(agreement_id):
    try:
        agreement = mp_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_information(args.agreement_id)

    if agreement is not None:
```

```
        print(f"Agreement Id: {args.agreement_id}")
        print(
            f"Agreement Value: {agreement['estimatedCharges']['currencyCode']}
 {agreement['estimatedCharges']['agreementValue']}"
        )

    else:
        print(f"Agreement with ID {args.agreement_id} is not found")
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen von Einzelheiten zur kostenlosen Testversion aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Einzelheiten zur kostenlosen Testversion aus einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the details from an agreement of a free trial I have provided to the customer
AG-20

Example Usage: python3 get_agreement_free_trial_details.py --agreement-id
 <agreement-id>
"""

import argparse
import logging
```

```
import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        freetrial_found = False
```

```
        for term in agreement["acceptedTerms"]:
            if "freeTrialPricingTerm" in term.keys():
                helper.pretty_print_datetime(term)
                freetrial_found = True

        if not freetrial_found:
            print(f"No free trial term found for agreement: {args.agreement_id}")
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen von Informationen über eine Vereinbarung

Das folgende Codebeispiel zeigt, wie Informationen über eine Vereinbarung abgerufen werden.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement information
AG-07
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))
```

```
import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
            raise e
        else:
            logger.error("Unexpected error: %s", e)
            raise e

    return response


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement_information(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(response)
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen von Produkt- und Angebotsdetails aus einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie Produkt- und Angebotsdetails aus einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get product and offer details in
 a given agreement
AG-10
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mpa_client = boto3.client("marketplace-agreement")
mpc_client = boto3.client("marketplace-catalog")

logger = logging.getLogger(__name__)


def get_agreement_information(agreement_id):
```

```
    """
    Returns information about a given agreement
    Args: agreement_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        agreement = mpa_client.describe_agreement(agreementId=agreement_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_entity_information(entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mpc_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_components(agreement_id):
    agreement_component_list = []

    agreement = get_agreement_information(agreement_id)
```

```
    if agreement is not None:
        productIds = []
        for resource in agreement["proposalSummary"]["resources"]:
            productIds.append(resource["id"])

        for product_id in productIds:
            product_document = get_entity_information(product_id)

            product_document_dict = {}
            product_document_dict["product_id"] = product_id
            product_document_dict["document"] = product_document
            agreement_component_list.append(product_document_dict)

        offerId = agreement["proposalSummary"]["offerId"]

        offer_document = get_entity_information(offerId)

        offer_document_dict = {}
        offer_document_dict["offer_id"] = offerId
        offer_document_dict["document"] = offer_document
        agreement_component_list.append(offer_document_dict)

        return agreement_component_list

    else:
        print("Agreement with ID " + args.agreement_id + " is not found")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement_id",
        "-aid",
        help="Provide agreement ID to search for product and offer detail",
        required=True,
    )
    args = parser.parse_args()

    product_offer_detail = get_agreement_components(agreement_id=args.agreement_id)

    helper.pretty_print_datetime(product_offer_detail)
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der EULA einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die EULA einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the EULA I have entered into with my customer via the agreement
AG-18
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
# agreement id
AGREEMENT_ID = "agmt-11111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "legalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "documents"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        legalEulaArray = []
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term and ATTRIBUTE_NAME in term[TERM_NAME]:
                docs = term[TERM_NAME][ATTRIBUTE_NAME]
                for doc in docs:
                    if "type" in doc:
                        legalEulaArray.append(doc)
        return legalEulaArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)
```

```
def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der Bedingungen für die automatische Verlängerung einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Bedingungen für die automatische Verlängerung einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the auto-renewal status of the agreement
AG-15
"""

import json
import logging
import os
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper


import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-1111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "renewalTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "enableAutoRenew"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """
```

```
    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        auto_renewal = "No Auto Renewal"
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    auto_renewal = term[TERM_NAME][CONFIG_ELEM][ATTRIBUTE_NAME]
                    break
        return auto_renewal

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        print(f"Auto Renewal is {agreement}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


# open json file from path
def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)
```

```
if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der im Rahmen einer Vereinbarung gekauften Dimensionen

Das folgende Codebeispiel zeigt, wie Sie die im Rahmen einer Vereinbarung gekauften Dimensionen abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the dimensions the buyer has purchased from me via the agreement
AG-28
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimensionKeys = []

        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        if "dimensions" in confParam:
                            for dimension in confParam["dimensions"]:
                                if "dimensionKey" in dimension:
                                    dimensionKey = dimension["dimensionKey"]
                                    print(f"Dimension Key: {dimensionKey}")
```

```
                                    dimensionKeys.append(dimensionKey)
        return dimensionKeys

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der im Rahmen einer Vereinbarung gekauften Instances jeder Dimension

Das folgende Codebeispiel zeigt, wie Sie die im Rahmen einer Vereinbarung gekauften Instances jeder Dimensionen abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain instances of each dimension that buyer has purchased in the agreement
AG-30
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

logger = logging.getLogger(__name__)
```

```
def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        terms = mp_client.get_agreement_terms(agreementId=entity_id)
        dimensionKeyValueMap = {}
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                if CONFIG_ELEM in term[TERM_NAME]:
                    confParam = term[TERM_NAME][CONFIG_ELEM]
                    if ATTRIBUTE_NAME in confParam:
                        selectValue = confParam["selectorValue"]
                        dimensionKeyValueMap["selectorValue"] = selectValue
                        if "dimensions" in confParam:
                            dimensionKeyValueMap["dimensions"] =
 confParam["dimensions"]
                            """
                            for dimension in confParam['dimensions']:
                                if 'dimensionKey' in dimension:

                                    dimensionValue = dimension['dimensionValue']
                                    dimensionKey = dimension['dimensionKey']
                                    print(f"Selector: {selectValue}, Dimension Key:
 {dimensionKey}, Dimension Value: {dimensionValue}")
                                    dimensionKeyValueMap[dimensionKey] =
 dimensionValue
                            """
        return dimensionKeyValueMap

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
```

```
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen des Zahlungsplans einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Zahlungsplan einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
AG-17
"""

import json
import logging
import os
```

```
import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-11111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "paymentScheduleTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        paymentScheduleArray = []
        currencyCode = ""
        for term in terms[ROOT_ELEM]:
            if TERM_NAME in term:
                paymentSchedule = term[TERM_NAME]
```

```
                if "currencyCode" in paymentSchedule:
                    currencyCode = paymentSchedule["currencyCode"]
                if "schedule" in paymentSchedule:
                    for sch in paymentSchedule["schedule"]:
                        if "chargeDate" in sch:
                            chargeDate = sch["chargeDate"]
                            chargeAmount = sch["chargeAmount"]
                            # print(f"chargeDate: {chargeDate}, chargeAmount:
 {chargeAmount}")
                            schedule = {
                                "currencyCode": currencyCode,
                                "chargeDate": chargeDate,
                                "chargeAmount": chargeAmount,
                            }
                            paymentScheduleArray.append(schedule)

        return paymentScheduleArray

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)
```

```
if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der Preise pro Dimension in einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Preise pro Dimension in einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain pricing per each dimension in the agreement
AG-29
"""

import json
import logging
import os

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"

# to use sample file or not
USE_SAMPLE_FILE = False
SAMPLE_FILE_NAME = "mockup_agreement_terms.json"

# attribute name
ROOT_ELEM = "acceptedTerms"
TERM_NAME = "configurableUpfrontPricingTerm"
CONFIG_ELEM = "configuration"
ATTRIBUTE_NAME = "selectorValue"

TERMS_TO_SEARCH = [
    "configurableUpfrontPricingTerm",
    "usageBasedPricingTerm",
    "fixedUpfrontPricingTerm",
]


def get_agreement_information(mp_client, entity_id):
    """
    Returns customer AWS Account id about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """

    try:
        if USE_SAMPLE_FILE:
            sample_file = os.path.join(os.path.dirname(__file__), SAMPLE_FILE_NAME)
            terms = open_json_file(sample_file)
        else:
            terms = mp_client.get_agreement_terms(agreementId=entity_id)

        dimentions = []
        for term in terms[ROOT_ELEM]:
            for t in TERMS_TO_SEARCH:
                rateInfo = []
                if t in term:
                    if "type" in term[t]:
                        rateInfo.append(term[t]["type"])
                    if "currencyCode" in term[t]:
```

```
                        rateInfo.append(term[t]["currencyCode"])
                    if "rateCards" in term[t]:
                        rateInfo.append(term[t]["rateCards"])
                    dimentions.append(rateInfo)
        return dimentions

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreement_information(mp_client, AGREEMENT_ID))

    # open json file from path


def open_json_file(filename):
    with open(filename, "r") as f:
        return json.load(f)


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen des Preistyps einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Preistyp einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
AG-16
"""

import json
import logging

import boto3
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "OfferId"

# replace id value as needed
idValue = "offer-1111111111111"

MAX_PAGE_RESULTS = 10

# catalog; switch to AWSMarketplace for release
AWSMPCATALOG = "AWSMarketplace"

# product types

SaaSProduct = "SaaSProduct"
AmiProduct = "AmiProduct"
MLProduct = "MachineLearningProduct"
ContainerProduct = "ContainerProduct"
DataProduct = "DataProduct"
```

```
ProServiceProduct = "ProfessionalServicesProduct"
AiqProduct = "AiqProduct"

# Define pricing types
CCP = "CCP"
Annual = "Annual"
Contract = "Contract"
SFT = "SaaS Free Trial"
HMA = "Hourly and Monthly Agreements"
Hourly = "Hourly"
Monthly = "Monthly"
AFPS = "Annual FPS"
CFPS = "Contract FPS"
CCPFPS = "CCP with FPS"
BYOL = "BYOL"
Free = "Free"
FTH = "Free Trials and Hourly"

# Define Agreement Term Types
legal = ["LegalTerm"]
config = ["ConfigurableUpfrontPricingTerm"]
usage = ["UsageBasedPricingTerm"]
config_usage = ["ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm"]
freeTrial = ["FreeTrialPricingTerm"]
recur = ["RecurringPaymentTerm"]
usage_recur = ("UsageBasedPricingTerm", "RecurringPaymentTerm")
fixed_payment = ["FixedUpfrontPricingTerm", "PaymentScheduleTerm"]
fixed_payment_usage = [
    "FixedUpfrontPricingTerm",
    "PaymentScheduleTerm",
    "UsageBasedPricingTerm",
]
byol = ["ByolPricingTerm"]
freeTrial_usage = ("FreeTrialPricingTerm", "UsageBasedPricingTerm")
all_agreement_types_combination = (
    legal,
    config,
    usage,
    config_usage,
    freeTrial,
    recur,
    usage_recur,
    fixed_payment,
    fixed_payment_usage,
```

```
    byol,
    freeTrial_usage,
)


# get pricing type method given product type, agreement temr type and offer type if
 needed
def get_pricing_type(product_type, agreement_term_type, offer_type):
    pricing_types = {
        (SaaSProduct, frozenset(config_usage), frozenset("")): CCP,
        (DataProduct, frozenset(config_usage), frozenset("")): CCP,
        (ContainerProduct, frozenset(config), frozenset(config_usage)): Annual,
        (AmiProduct, frozenset(config), frozenset(config_usage)): Annual,
        (MLProduct, frozenset(config), frozenset(config_usage)): Annual,
        (ContainerProduct, frozenset(config), frozenset(config)): Contract,
        (AmiProduct, frozenset(config), frozenset(config)): Contract,
        (SaaSProduct, frozenset(config), frozenset("")): Contract,
        (DataProduct, frozenset(config), frozenset("")): Contract,
        (AiqProduct, frozenset(config), frozenset("")): Contract,
        (ProServiceProduct, frozenset(config), frozenset("")): Contract,
        (SaaSProduct, frozenset(freeTrial), frozenset("")): SFT,
        (AmiProduct, frozenset(usage_recur), frozenset("")): HMA,
        (SaaSProduct, frozenset(usage), frozenset("")): Hourly,
        (AmiProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(usage), frozenset("")): Hourly,
        (MLProduct, frozenset(usage), frozenset("")): Hourly,
        (ContainerProduct, frozenset(recur), frozenset("")): Monthly,
        (AmiProduct, frozenset(recur), frozenset("")): Monthly,
        (
            ContainerProduct,
            frozenset(fixed_payment),
            frozenset(fixed_payment_usage),
        ): AFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment_usage)):
 AFPS,
        (MLProduct, frozenset(fixed_payment), frozenset("")): AFPS,
        (ContainerProduct, frozenset(fixed_payment), frozenset(fixed_payment)):
 CFPS,
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): CFPS,
        (SaaSProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (DataProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (AiqProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (ProServiceProduct, frozenset(fixed_payment), frozenset("")): CFPS,
        (SaaSProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
```

```
        (DataProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AiqProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (ProServiceProduct, frozenset(fixed_payment_usage), frozenset("")): CCPFPS,
        (AmiProduct, frozenset(byol), frozenset("")): BYOL,
        (SaaSProduct, frozenset(byol), frozenset("")): BYOL,
        (ProServiceProduct, frozenset(byol), frozenset("")): BYOL,
        (AiqProduct, frozenset(byol), frozenset("")): BYOL,
        (MLProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(byol), frozenset("")): BYOL,
        (DataProduct, frozenset(byol), frozenset("")): BYOL,
        (ContainerProduct, frozenset(legal), frozenset("")): Free,
        (AmiProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (ContainerProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
        (MLProduct, frozenset(freeTrial_usage), frozenset("")): FTH,
    }

    key = (product_type, agreement_term_type, offer_type)
    if key in pricing_types:
        return pricing_types[key]
    else:
        return "Unknown"


# Example usage for testing purpose
"""
product_type = SaaSProduct
agreement_term_type = frozenset(config_usage)
offer_type = frozenset('')
pricing_type = get_pricing_type(product_type, agreement_term_type, offer_type)
print("pricing type = " + pricing_type)  # Output: CCP
"""


# check if offer term types are needed; if Y, needed
def get_offer_term_type(product_type, agreement_term_type):
    offer_term_types = {
        (ContainerProduct, frozenset(config)): "Y",
        (AmiProduct, frozenset(config)): "Y",
        (ContainerProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment)): "Y",
        (AmiProduct, frozenset(fixed_payment), frozenset(fixed_payment)): "Y",
    }

    key = (product_type, agreement_term_type)
```

```
    if key in offer_term_types:
        return offer_term_types[key]
    else:
        return


logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog=AWSMPCATALOG,
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog=AWSMPCATALOG,
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
```

```
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    # find all agreements matching the specified idType and idValue
    agreements = get_agreements(mp_client)

    for item in agreements:
        pricingType = ""
        agreement_id = item["agreementId"]

        # get term types inside offer
        offer_term_types = get_offer_term_types(item)

        # even though multiple product types are allowed for one agreement, only
 need the first one
        productType = item["proposalSummary"]["resources"][0]["type"]

        # get agreement terms types
        agreementTerm = mp_client.get_agreement_terms(agreementId=agreement_id)

        agreementTermTypes = get_agreement_term_types(agreementTerm)

        # match with agreement term type group
        matchedTermType = getMatchedTermTypesCombination(agreementTermTypes)

        # check if offer term type is needed.
        offer_term_type_needed = get_offer_term_type(
            productType, frozenset(matchedTermType)
        )
```

```
        # get pricing type given product type, agreement term types and offer type
 if needed;
        # one excpetion is Container with Legal term. LegalTerm needs to be the only
 term present
        if offer_term_type_needed is not None:
            matchedOfferTermTypes = getMatchedTermTypesCombination(offer_term_types)
            print(f"matchedOfferTermType = {matchedOfferTermTypes}")
            pricingType = get_pricing_type(
                productType,
                frozenset(matchedTermType),
                frozenset(matchedOfferTermTypes),
            )
        elif set(matchedTermType) == set(legal):
            pricingType = Free
        else:
            pricingType = get_pricing_type(
                productType, frozenset(matchedTermType), frozenset("")
            )

        print(
            f"agreementId={agreement_id};productType={productType};
 agreementTermTypes={agreementTermTypes}; matchedTermType={matchedTermType};
 offerTermTypeNeeded={offer_term_type_needed}; offer_term_types={offer_term_types}"
        )
        print(f"pricing type={pricingType}")


def getMatchedTermTypesCombination(agreementTermTypes):
    matchedCombination = ()
    for element in all_agreement_types_combination:
        if check_elements(agreementTermTypes, element):
            matchedCombination = element
    return matchedCombination


def get_offer_term_types(item):
    offer_id = item["proposalSummary"]["offerId"]
    mp_catalogAPI_client = boto3.client("marketplace-catalog")
    offer_document = get_entity_information(mp_catalogAPI_client, offer_id)
    offerDetail = offer_document["Details"]
    offerDetail_json_object = json.loads(offerDetail)
    offer_term_types = [term["Type"] for term in offerDetail_json_object["Terms"]]
    return offer_term_types
```

```
# make sure all elements in array2 exist in array1
def check_elements(array1, array2):
    for element in array2:
        if element not in array1:
            return False
    return True


def get_entity_information(mp_client, entity_id):
    """
    Returns information about a given entity
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of entity information
    """

    try:
        response = mp_client.describe_entity(
            Catalog="AWSMarketplace",
            EntityId=entity_id,
        )

        return response

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Entity with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def get_agreement_term_types(agreementTerm):
    types = []
    for term in agreementTerm["acceptedTerms"]:
        for value in term.values():
            if isinstance(value, dict) and "type" in value:
                types.append(value["type"])
    return types


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen des Produkttyps einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Produkttyp einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the Product Type of the product the agreement was created on
AG-11
"""

import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

# agreement id
AGREEMENT_ID = "agmt-111111111111111111111111"


def get_agreement_information(mp_client, entity_id):
    """
    Returns information about a given agreement
    Args: entity_id str: Entity to return
    Returns: dict: Dictionary of agreement information
    """
```

```
    try:
        agreement = mp_client.describe_agreement(agreementId=entity_id)

        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", entity_id)
        else:
            logger.error("Unexpected error: %s", e)


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for offer and product details in a given agreement by agreement
 id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    agreement = get_agreement_information(mp_client, AGREEMENT_ID)

    if agreement is not None:
        for resource in agreement["proposalSummary"]["resources"]:
            print(f"Product ID: {resource['id']}  |  Product Type:
 {resource['type']}")
    else:
        print("Agreement with ID " + AGREEMENT_ID + " is not found")


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen des Status einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie den Status einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get all agreement status
AG-13

Example Usage: python3 get_agreement_status.py --agreement-id <agreement-id>
"""

import argparse
import logging

import boto3
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)


def get_agreement(agreement_id):
    try:
        response = mp_client.describe_agreement(agreementId=agreement_id)
        return response
    except ClientError as e:
        logger.error(f"Could not complete search_agreements request. {e}")

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
```

```
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreement(agreement_id=args.agreement_id)

    if response is not None:
        print(f"Agreement status: {response['status']}")
    else:
        print(f"No agreement found for {args.agreement_id}")
```

- API-Details finden Sie DescribeAgreement in der AWS API-Referenz zum SDK für Python (Boto3).

Abrufen der Support-Bedingungen einer Vereinbarung

Das folgende Codebeispiel zeigt, wie Sie die Support-Bedingungen einer Vereinbarung abrufen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Obtain the support and refund policy I have provided to the customer for an
 agreement
AG-19

Example Usage: python3 get_agreement_support_terms.py --agreement-id <agreement-id>
```

```
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

mp_client = boto3.client("marketplace-agreement")


def get_agreement_terms(agreement_id):
    try:
        agreement = mp_client.get_agreement_terms(agreementId=agreement_id)
        return agreement

    except ClientError as e:
        if e.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Agreement with ID %s not found.", agreement_id)

        else:
            logger.error("Unexpected error: %s", e)

    return None


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--agreement-id",
        "-aid",
        help="Provide agreement ID to describe agreement status",
        required=True,
    )
    args = parser.parse_args()
```

```
    agreement = get_agreement_terms(agreement_id=args.agreement_id)

    if agreement is not None:
        support_found = False

        for term in agreement["acceptedTerms"]:
            if "supportTerm" in term.keys():
                helper.pretty_print_datetime(term)
                support_found = True

        if not support_found:
            print(f"No support term found for agreement: {args.agreement_id}")
```

- API-Details finden Sie GetAgreementTerms in der AWS API-Referenz zum SDK für Python (Boto3).

Listen Sie Anfragen zur Kündigung von Verträgen auf

Das folgende Codebeispiel zeigt, wie Anfragen zur Kündigung von Vereinbarungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementCancellationRequests:
```

```
    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_cancellation_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementCancellationRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_cancellation_requests(**kwargs)

            for summary in response.get("items", []):
                print("Cancellation Request ID: " +
 summary["agreementCancellationRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Reason Code: " + str(summary.get("reasonCode", "")))
                print("Agreement Type: " + str(summary.get("agreementType", "")))
                print("Catalog: " + str(summary.get("catalog", "")))
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementCancellationRequests.list_agreement_cancellation_requests()
```

- API-Details finden Sie ListAgreementCancellationRequests in der AWS API-Referenz zum SDK für Python (Boto3).

Zahlungsanfragen für Vereinbarungen auflisten

Das folgende Codebeispiel zeigt, wie Zahlungsanforderungen für Vereinbarungen aufgelistet werden, an denen ich als Akzeptor teilnehme.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class ListAgreementPaymentRequests:

    PARTY_TYPE = "Proposer"

    @staticmethod
    def list_agreement_payment_requests():
        client = boto3.client("marketplace-agreement")

        next_token = None

        while True:
            kwargs = {"partyType": ListAgreementPaymentRequests.PARTY_TYPE}
            if next_token:
                kwargs["nextToken"] = next_token

            response = client.list_agreement_payment_requests(**kwargs)

            for summary in response.get("items", []):
                print("Payment Request ID: " + summary["paymentRequestId"])
                print("Agreement ID: " + summary["agreementId"])
                print("Status: " + str(summary.get("status", "")))
                print("Name: " + str(summary.get("name", "")))
                print("Charge ID: " + str(summary.get("chargeId", "")))
                print("Charge Amount: " + str(summary.get("chargeAmount", "")))
                print("Currency Code: " + str(summary.get("currencyCode", "")))
```

```
                print("Created At: " + str(summary.get("createdAt", "")))
                print("Updated At: " + str(summary.get("updatedAt", "")))
                print("---")

            next_token = response.get("nextToken")
            if not next_token:
                break


if __name__ == "__main__":
    ListAgreementPaymentRequests.list_agreement_payment_requests()
```

- API-Details finden Sie ListAgreementPaymentRequests in der AWS API-Referenz zum SDK für Python (Boto3).

Lehnen Sie eine Anfrage zur Kündigung einer Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Anfrage zur Kündigung einer Vereinbarung abgelehnt wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementCancellationRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
```

```
    AGREEMENT_CANCELLATION_REQUEST_ID = "<AGREEMENT CANCELLATION REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_cancellation_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_cancellation_request(
            agreementId=RejectAgreementCancellationRequest.AGREEMENT_ID,

 agreementCancellationRequestId=RejectAgreementCancellationRequest.AGREEMENT_CANCELLATION_RE
            rejectionReason=RejectAgreementCancellationRequest.REJECTION_REASON,
        )

        print("Agreement ID: " + response["agreementId"])
        print("Cancellation Request ID: " +
 response["agreementCancellationRequestId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Reason Code: " + str(response.get("reasonCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))


if __name__ == "__main__":
    RejectAgreementCancellationRequest.reject_agreement_cancellation_request()
```

- API-Details finden Sie RejectAgreementCancellationRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Lehnen Sie eine Zahlungsaufforderung für eine Vereinbarung ab

Das folgende Codebeispiel zeigt, wie eine vom Verkäufer initiierte Zahlungsaufforderung für eine Vereinbarung abgelehnt wird.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
import sys
import os

sys.path.insert(0, os.path.join(os.path.dirname(__file__), ".."))

import boto3


class RejectAgreementPaymentRequest:

    AGREEMENT_ID = "<AGREEMENT ID HERE>"
    PAYMENT_REQUEST_ID = "<PAYMENT REQUEST ID HERE>"
    REJECTION_REASON = "<REJECTION REASON HERE>"

    @staticmethod
    def reject_agreement_payment_request():
        client = boto3.client("marketplace-agreement")

        response = client.reject_agreement_payment_request(
            agreementId=RejectAgreementPaymentRequest.AGREEMENT_ID,
            paymentRequestId=RejectAgreementPaymentRequest.PAYMENT_REQUEST_ID,
            rejectionReason=RejectAgreementPaymentRequest.REJECTION_REASON,
        )

        print("Payment Request ID: " + response["paymentRequestId"])
        print("Agreement ID: " + response["agreementId"])
        print("Status: " + str(response.get("status", "")))
        print("Status Message: " + str(response.get("statusMessage", "")))
        print("Name: " + str(response.get("name", "")))
        print("Charge Amount: " + str(response.get("chargeAmount", "")))
        print("Currency Code: " + str(response.get("currencyCode", "")))
        print("Created At: " + str(response.get("createdAt", "")))
        print("Updated At: " + str(response.get("updatedAt", "")))
```

```
if __name__ == "__main__":
    RejectAgreementPaymentRequest.reject_agreement_payment_request()
```

- API-Details finden Sie RejectAgreementPaymentRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Ersetzen Sie eine SaaS-Vertragsvereinbarung durch ein vertragsbasiertes Angebot

Das folgende Codebeispiel zeigt, wie eine SaaS-Vertragsvereinbarung durch ein neues vertragsbasiertes Angebot ersetzt wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to replace an existing SaaS agreement with CONTRACT pricing model
 with a new
Agreement-Based Offer (ABO) using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer has an active SaaS agreement with CONTRACT pricing model and
 receives a private
Agreement-Based Offer from the seller. The buyer replaces the existing agreement
with the new ABO offer, which may include updated pricing terms and payment
 schedule.

Note: Unlike other Replace samples, this example starts from an already-active
EXISTING_AGREEMENT_ID rather than creating a new agreement first. Use this
pattern when you already have an agreement ID and want to replace it directly.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
```

```
  - EXISTING_AGREEMENT_ID — the agreement ID of the active agreement to replace
 (starts with agmt-).
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new ABO
 offer.
  - Term IDs (starting with term-) — found in the new offer's term list.
"""

import boto3

from utils.agreement_api_utils import generate_client_token


class ReplaceSaaSContractWithAgreementBasedOffer:

    # The agreement ID of the active SaaS agreement with CONTRACT pricing model to
 replace (starts with "agmt-").
    EXISTING_AGREEMENT_ID = "<your-existing-agreement-id>"

    # The agreementProposalId from the new Agreement-Based Offer.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"

    # Term ID for the LegalTerm in the new offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the FixedUpfrontPricingTerm in the new offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the new offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    @staticmethod
    def replace_existing_agreement():
        """
        Replace an existing SaaS agreement with CONTRACT pricing model with a new
 Agreement-Based Offer.

        Uses Intent.REPLACE with sourceAgreementIdentifier set to the existing
 agreement ID.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSContractWithAgreementBasedOffer
```

```
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}

        # Replace the agreement with the new offer
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[fixed_upfront_pricing_term, payment_schedule_term,
 legal_term, validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=cls.EXISTING_AGREEMENT_ID,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Replace agreement request created. AgreementRequestId: " +
 agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        print("Agreement replaced with ABO. New AgreementId: " +
 accept_response["agreementId"])


if __name__ == "__main__":
    ReplaceSaaSContractWithAgreementBasedOffer.replace_existing_agreement()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Ersetzen Sie eine kostenlose SaaS-Testversion durch einen Vertrag mit nutzungsabhängiger Preisgestaltung

Das folgende Codebeispiel zeigt, wie eine kostenlose SaaS-Testversion durch einen Contract with Consumption Pricing (CCP) ersetzt wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS free trial agreement and then replace it with a
paid Contract with Consumption Pricing (CCP) offer using the AWS Marketplace
 Agreement Service APIs.

Scenario: A buyer first starts a free trial on a SaaS product. Once the trial is
 active,
they decide to convert by replacing it with a paid offer that includes a
FixedUpfrontPricingTerm, PaymentScheduleTerm, and UsageBasedPricingTerm.

Flow:
  1. Create a SaaS free trial agreement with freeTrialPricingTerm.
  2. Wait for freeTrialPricingTerm agreement entitlements to become active.
  3. Replace the free trial agreement with the paid CCP offer using Intent.REPLACE.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the free trial
 offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the paid CCP
 offer.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)
```

```
class ReplaceSaaSFreeTrialWithCCP:

    # The agreementProposalId from the free trial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-free-trial-agreement-proposal-
identifier>"

    # Term ID for the FreeTrialPricingTerm in the free trial offer.
    FREE_TRIAL_PRICING_TERM_ID = "<your-free-trial-pricing-term-id>"

    # Term ID for the LegalTerm in the free trial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the paid CCP offer to convert to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-ccp-agreement-proposal-identifier>"

    # Term ID for the FixedUpfrontPricingTerm in the CCP offer.
    FIXED_UPFRONT_PRICING_TERM_ID = "<your-fixed-upfront-pricing-term-id>"

    # Term ID for the PaymentScheduleTerm in the CCP offer.
    PAYMENT_SCHEDULE_TERM_ID = "<your-payment-schedule-term-id>"

    # Term ID for the UsageBasedPricingTerm in the CCP offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the CCP offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the CCP offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def create_saas_free_trial_and_replace_with_ccp():
        """
        Full end-to-end flow:
        1. Create a SaaS free trial agreement with freeTrialPricingTerm.
        2. Wait for freeTrialPricingTerm agreement entitlements to become active.
        3. Replace the free trial agreement with the paid CCP offer.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSFreeTrialWithCCP

        legal_term = {"id": cls.LEGAL_TERM_ID}
        free_trial_pricing_term = {"id": cls.FREE_TRIAL_PRICING_TERM_ID}
```

```
        # --- Step 1: Agreement with freeTrialPricingTerm ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[free_trial_pricing_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with freeTrialPricingTerm created.
AgreementRequestId: " + agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with freeTrialPricingTerm accepted. AgreementId: "
+ agreement_id)

        # Wait for freeTrialPricingTerm agreement entitlements to become active
before replacing.
        print("Waiting for freeTrialPricingTerm agreement entitlements to become
active...")
        entitlements_response = poll_until_entitlements_available(client,
agreement_id)
        print("freeTrialPricingTerm agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace Agreement with freeTrialPricingTerm with Paid CCP
Offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the free trial
agreement.
        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        fixed_upfront_pricing_term = {"id": cls.FIXED_UPFRONT_PRICING_TERM_ID}
        payment_schedule_term = {"id": cls.PAYMENT_SCHEDULE_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[
                usage_based_pricing_term, fixed_upfront_pricing_term,
                payment_schedule_term, validity_term, new_legal_term,
            ],
```

```
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        print(
            "Agreement with freeTrialPricingTerm replaced with paid CCP offer. New
 AgreementId: "
            + aar_response["agreementId"]
        )


if __name__ == "__main__":
    ReplaceSaaSFreeTrialWithCCP.create_saas_free_trial_and_replace_with_ccp()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Ersetzen Sie eine nutzungsabhängige SaaS-Preisklausel und kündigen Sie die vorherige Vereinbarung

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige SaaS-Preisbedingung ersetzt und die vorherige Vereinbarung gekündigt wird.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to create a SaaS agreement with usageBasedPricingTerm (UBPT) and
 then replace it
```

```
with a new offer using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer subscribes to a SaaS product with UsageBasedPricingTerm.
The buyer then converts to a different offer by replacing the existing agreement.

Flow:
  1. Create and accept the initial agreement request with UBPT.
  2. Wait for entitlements to become active.
  3. Replace the agreement with a new offer using Intent.REPLACE.
  4. Cancel the new agreement using CancelAgreement.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new offer
 to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceSaaSUsageBasedPricingTermAndCancel:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_BASED_PRICING_TERM_ID = "<your-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"
```

```
    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_BASED_PRICING_TERM_ID = "<your-new-usage-based-pricing-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    @staticmethod
    def replace_saas_ubpt_and_cancel():
        """
        Full end-to-end flow:
        1. Create and accept the initial agreement request with
UsageBasedPricingTerm.
        2. Wait for entitlements to become active.
        3. Replace the agreement with a new offer.
        4. Cancel the new agreement.
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceSaaSUsageBasedPricingTermAndCancel

        usage_based_pricing_term = {"id": cls.USAGE_BASED_PRICING_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}

        # --- Step 1: Create and accept the initial UBPT agreement request ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_based_pricing_term, validity_term, legal_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)

        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " + agreement_id)
```

```
        # Wait for entitlements to become active before replacing.
        print("Waiting for entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
 agreement_id)
        print("Entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Replace the UBPT agreement with a new offer ---
        # Use Intent.REPLACE and sourceAgreementIdentifier to replace the existing
 agreement.
        new_usage_based_pricing_term = {"id": cls.NEW_USAGE_BASED_PRICING_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_based_pricing_term, new_validity_term,
 new_legal_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
        agreement_id_with_new_offer = aar_response["agreementId"]
        print("UBPT agreement replaced. New AgreementId: " +
 agreement_id_with_new_offer)

        # --- Step 3: Cancel the new agreement ---
        client.cancel_agreement(agreementId=agreement_id_with_new_offer)
        print("The new agreement has been cancelled. AgreementId: " +
 agreement_id_with_new_offer)


if __name__ == "__main__":
    ReplaceSaaSUsageBasedPricingTermAndCancel.replace_saas_ubpt_and_cancel()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Ersetzen Sie eine nutzungsabhängige AMI-Preisklausel, behalten Sie jedoch die ConfigurableUpfrontPricingTerm

Das folgende Codebeispiel zeigt, wie eine nutzungsabhängige AMI-Preisklausel ersetzt und gleichzeitig die bestehende beibehalten wird. ConfigurableUpfrontPricingTerm

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to replace an AMI agreement with usageBasedPricingTerm with a new
 offer while
the agreement with ConfigurableUpfrontPricingTerm (CUPT) is still active, using the
AWS Marketplace Agreement Service APIs.

Scenario: An AMI product with USAGE pricing requires two agreements:
  1. An agreement with usageBasedPricingTerm — accepted first to establish the base
 agreement.
  2. An agreement with configurableUpfrontPricingTerm (CUPT) — accepted after the
     usageBasedPricingTerm agreement entitlements are active.
This sample shows how to replace only the agreement with usageBasedPricingTerm with
 a
new offer without touching the existing agreement with
 configurableUpfrontPricingTerm (CUPT).

Flow:
  1. Create and accept the initial agreement request with usageBasedPricingTerm.
  2. Create and accept the agreement request with configurableUpfrontPricingTerm
 (CUPT)
     (after usageBasedPricingTerm agreement entitlements are active).
  3. Replace the agreement with usageBasedPricingTerm with a new offer using
 Intent.REPLACE.
```

```
Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offers:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the initial offer.
  - NEW_AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the new offer
 to replace to.
  - Term IDs (starting with term-) — found in each offer's term list.
  - SELECTOR_VALUE — duration for the agreement (e.g., P365D).
  - DIMENSION_1_KEY — dimension key defined in the CUPT term.
"""

import boto3

from utils.agreement_api_utils import (
    format_output,
    generate_client_token,
    poll_until_entitlements_available,
)


class ReplaceAmiUsageBasedPricingTermButNotCupt:

    # The agreementProposalId from the initial offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in the initial offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P365D" for 365 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key defined in the CUPT term.
    DIMENSION_1_KEY = "<your-dimension-key>"

    # Quantity for the dimension.
    DIMENSION_1_VALUE = 1

    # Term ID for the UsageBasedPricingTerm in the initial offer.
    USAGE_TERM_ID = "<your-usage-term-id>"

    # Term ID for the LegalTerm in the initial offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"
```

```
    # Term ID for the ValidityTerm in the initial offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    # The agreementProposalId from the new offer to replace to.
    NEW_AGREEMENT_PROPOSAL_IDENTIFIER = "<your-new-agreement-proposal-identifier>"

    # Term ID for the UsageBasedPricingTerm in the new offer.
    NEW_USAGE_TERM_ID = "<your-new-usage-term-id>"

    # Term ID for the LegalTerm in the new offer.
    NEW_LEGAL_TERM_ID = "<your-new-legal-term-id>"

    # Term ID for the ValidityTerm in the new offer.
    NEW_VALIDITY_TERM_ID = "<your-new-validity-term-id>"

    @staticmethod
    def replace_ami_usage_when_cupt_exists():
        """
        Full end-to-end flow:
        1. Create and accept an agreement request with usageBasedPricingTerm, then
wait for entitlements.
        2. Create and accept an agreement request with CUPT, then wait for
entitlements.
        3. Replace the agreement with usageBasedPricingTerm with a new offer
           (agreement with CUPT is unaffected).
        """
        client = boto3.client("marketplace-agreement")
        cls = ReplaceAmiUsageBasedPricingTermButNotCupt

        usage_term = {"id": cls.USAGE_TERM_ID}
        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Step 1: Agreement with UBPT ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[usage_term, legal_term, validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request with UBPT created. AgreementRequestId: " +
agreement_request_id)
```

```
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id
        )
        usage_agreement_id = accept_response["agreementId"]
        print("Agreement request with UBPT accepted. AgreementId: " +
usage_agreement_id)

        # Wait for UBPT agreement entitlements to become active before creating the
agreement with CUPT.
        print("Waiting for UBPT agreement entitlements to become active...")
        entitlements_response = poll_until_entitlements_available(client,
usage_agreement_id)
        print("UBPT agreement entitlements are now active.")
        format_output(entitlements_response)

        # --- Step 2: Agreement with configurableUpfrontPricingTerm (CUPT) ---
        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        car_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        print("Agreement request with CUPT created. AgreementRequestId: " +
car_response["agreementRequestId"])

        aar_response = client.accept_agreement_request(
            agreementRequestId=car_response["agreementRequestId"]
        )
```

```
        cupt_agreement_id = aar_response["agreementId"]
        print("Agreement request with CUPT accepted. AgreementId: " +
 cupt_agreement_id)

        # Wait for CUPT agreement entitlements to become active before replacing
 usage.
        print("Waiting for CUPT agreement entitlements to become active...")
        cupt_entitlements_response = poll_until_entitlements_available(client,
 cupt_agreement_id)
        print("CUPT agreement entitlements are now active.")
        format_output(cupt_entitlements_response)

        # --- Step 3: Replace Agreement with usageBasedPricingTerm with a new offer
 ---
        new_usage_term = {"id": cls.NEW_USAGE_TERM_ID}
        new_legal_term = {"id": cls.NEW_LEGAL_TERM_ID}
        new_validity_term = {"id": cls.NEW_VALIDITY_TERM_ID}

        # Use Intent.REPLACE and sourceAgreementIdentifier pointing to the
 usageBasedPricingTerm agreement only.
        # The agreement with configurableUpfrontPricingTerm (CUPT) is NOT affected
 by this replacement.
        car_replace_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="REPLACE",
            requestedTerms=[new_usage_term, new_legal_term, new_validity_term],
            agreementProposalIdentifier=cls.NEW_AGREEMENT_PROPOSAL_IDENTIFIER,
            sourceAgreementIdentifier=usage_agreement_id,
        )
        print("Replace agreement request created. AgreementRequestId: " +
 car_replace_response["agreementRequestId"])

        aar_replace_response = client.accept_agreement_request(
            agreementRequestId=car_replace_response["agreementRequestId"]
        )
        print("Agreement with UBPT replaced. New AgreementId: " +
 aar_replace_response["agreementId"])


if __name__ == "__main__":
    ReplaceAmiUsageBasedPricingTermButNotCupt.replace_ami_usage_when_cupt_exists()
```

- API-Details finden Sie CreateAgreementRequest in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand der Konto-ID

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Konto-ID suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to get agreement by customer AWS
 account ID
AG-02
"""

import argparse
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")
logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10
```

```
def get_agreements(account_id):
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": "AcceptorId", "values": [account_id]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement and agreement["nextToken"] is not None:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {"name": "AcceptorId", "values": [account_id]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--account_id",
        "-aid",
        help="Provide accepting account ID to search for agreements",
        required=True,
    )
    args = parser.parse_args()

    response = get_agreements(account_id=args.account_id)

    helper.pretty_print_datetime(response)
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand der Vereinbarungs-ID

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Vereinbarungs-ID suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements give id
 information
AG-02-A
"""
```

```
import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# To search by offer id: OfferId; by product id: ResourceIdentifier; by product
 type: ResourceType
idType = "ResourceType"

# replace id value as needed
idValue = "SaaSProduct"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": idType, "values": [idValue]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e
```

```
        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": idType, "values": [idValue]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise e

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand des Enddatums

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand des Enddatums suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement information
 before or after end date
AG-03
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

# change to 'AfterEndTime' if after endtime is desired
beforeOrAfterEndtimeFilterName = "BeforeEndTime"

# Make sure to use the same date format as below
cutoffDate = "2322-11-18T00:00:00Z"

MAX_PAGE_RESULTS = 10
```

```
logger = logging.getLogger(__name__)


def get_agreements():
    AgreementSummaryList = []

    try:
        agreement = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            # Set PartyType filter to "Proposer" to return agreements where you are
 the proposer.
            # Change to "Acceptor" to return agreements where you are the acceptor.
            filters=[
                {"name": "PartyType", "values": ["Proposer"]},
                {"name": beforeOrAfterEndtimeFilterName, "values": [cutoffDate]},
                {"name": "AgreementType", "values": ["PurchaseAgreement"]},
            ],
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise

    AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    while "nextToken" in agreement:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreement["nextToken"],
                filters=[
                    {"name": "PartyType", "values": ["Proposer"]},
                    {
                        "name": beforeOrAfterEndtimeFilterName,
                        "values": [cutoffDate],
                    },
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise
```

```
        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


if __name__ == "__main__":
    agreements = get_agreements()
    helper.pretty_print_datetime(agreements)
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand der Angebots-ID

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Angebots-ID suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreements by offer id
AG-0
"""

import logging

import boto3
import sys
import os
```

```
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# offer id to search by
offerId = "11111111111111111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "OfferId", "values": [offerId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement and agreement["nextToken"] is not None:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "OfferId", "values": [offerId]},
```

```
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement by offer id.")
    print("-" * 88)

    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand der Produkt-ID

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand der Produkt-ID suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to search for agreement by product
 id
AG-02
"""

import logging

import boto3
import sys
import os

sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

# product id to search by
resourceId = "prod-1111111111111"

MAX_PAGE_RESULTS = 10

logger = logging.getLogger(__name__)


def get_agreements(mp_client):
    AgreementSummaryList = []
    # Set PartyType to "Proposer" to return agreements where you are the proposer.
    # Change to "Acceptor" to return agreements where you are the acceptor.
    partyTypes = ["Proposer"]
```

```
    for value in partyTypes:
        try:
            agreement = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                filters=[
                    {"name": "PartyType", "values": [value]},
                    {"name": "ResourceIdentifier", "values": [resourceId]},
                    {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                ],
            )
        except ClientError as e:
            logger.error("Could not complete list_entities request.")
            raise

        AgreementSummaryList.extend(agreement["agreementViewSummaries"])

        while "nextToken" in agreement:
            try:
                agreement = mp_client.search_agreements(
                    catalog="AWSMarketplace",
                    maxResults=MAX_PAGE_RESULTS,
                    nextToken=agreement["nextToken"],
                    filters=[
                        {"name": "PartyType", "values": [value]},
                        {"name": "ResourceIdentifier", "values": [resourceId]},
                        {"name": "AgreementType", "values": ["PurchaseAgreement"]},
                    ],
                )
            except ClientError as e:
                logger.error("Could not complete search_agreements request.")
                raise

            AgreementSummaryList.extend(agreement["agreementViewSummaries"])

    return AgreementSummaryList


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Looking for an agreement in the AWS Marketplace Catalog.")
    print("-" * 88)
```

```
    mp_client = boto3.client("marketplace-agreement")

    helper.pretty_print_datetime(get_agreements(mp_client))


if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Suchen nach Vereinbarungen anhand des Status

Das folgende Codebeispiel zeigt, wie Sie nach Vereinbarungen anhand des Status suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to use the AWS SDK for Python (Boto3) to filter agreements by status
AG-04

Example Usage: python3 search_agreements_by_status.py
"""

import logging

import boto3
import sys
import os
```

```
sys.path.append(os.path.join(os.path.dirname(__file__), "..", "..", ".."))

import utils.helpers as helper
from botocore.exceptions import ClientError

mp_client = boto3.client("marketplace-agreement")

logger = logging.getLogger(__name__)

MAX_PAGE_RESULTS = 10

# Set PartyType to "Proposer" to return agreements where you are the proposer.
# Change to "Acceptor" to return agreements where you are the acceptor.
party_type_list = ["Proposer"]
agreement_type_list = ["PurchaseAgreement"]

# Accepted values: "ACTIVE", "TERMINATED", "CANCELED", "EXPIRED", "REPLACED",
 "RENEWED"
status_list = ["ACTIVE"]

filter_list = [
    {"name": "PartyType", "values": party_type_list},
    {"name": "AgreementType", "values": agreement_type_list},
    {"name": "Status", "values": status_list},
]

agreement_results_list = []


def get_agreements(filter_list=filter_list):
    try:
        agreements = mp_client.search_agreements(
            catalog="AWSMarketplace",
            maxResults=MAX_PAGE_RESULTS,
            filters=filter_list,
        )
    except ClientError as e:
        logger.error("Could not complete search_agreements request.")
        raise e

    agreement_results_list.extend(agreements["agreementViewSummaries"])

    while "nextToken" in agreements and agreements["nextToken"] is not None:
```

```
        try:
            agreements = mp_client.search_agreements(
                catalog="AWSMarketplace",
                maxResults=MAX_PAGE_RESULTS,
                nextToken=agreements["nextToken"],
                filters=filter_list,
            )
        except ClientError as e:
            logger.error("Could not complete search_agreements request.")
            raise e

        agreement_results_list.extend(agreements["agreementViewSummaries"])

    helper.pretty_print_datetime(agreement_results_list)
    return agreement_results_list


if __name__ == "__main__":
    agreements_list = get_agreements(filter_list)
```

- API-Details finden Sie SearchAgreements in der AWS API-Referenz zum SDK für Python (Boto3).

Aktualisieren Sie Bestellungen nach Annahme der Vereinbarung

Das folgende Codebeispiel zeigt, wie Bestellungen aktualisiert werden, nachdem die Vereinbarung akzeptiert wurde.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das Repository der Codebibliothek der AWS Marketplace -API-Referenz einrichten und ausführen.

```
"""
Demonstrates how to associate a purchase order reference with a SaaS agreement with
 CONTRACT
```

```
pricing model using the AWS Marketplace Agreement Service APIs.

Scenario: A buyer creates a SaaS agreement request with CONTRACT pricing model and
 provides
a purchase order reference in AcceptAgreementRequest. After acceptance, the buyer
 lists the
resulting charges via ListAgreementCharges and associates the purchase order
 reference with a
specific charge via UpdatePurchaseOrders.

Before running this sample, replace the placeholder constants below with values from
your AWS Marketplace offer:
  - AGREEMENT_PROPOSAL_IDENTIFIER — the agreementProposalId from the offer.
  - Term IDs (starting with term-) — found in the offer's term list.
  - SELECTOR_VALUE — duration for the agreement.
  - DIMENSION_1_KEY — dimension key defined in the offer.
  - PURCHASE_ORDER_REFERENCE — your internal purchase order number (e.g.,
 po-123456).
"""

import boto3

from utils.agreement_api_utils import format_output, generate_client_token


class UpdatePurchaseOrdersAfterAgreementAcceptance:

    # Your internal purchase order reference number (e.g., "po-123456").
    PURCHASE_ORDER_REFERENCE = "po-123456"

    # The agreementProposalId from the offer.
    AGREEMENT_PROPOSAL_IDENTIFIER = "<your-agreement-proposal-identifier>"

    # Term ID for the ConfigurableUpfrontPricingTerm in your offer.
    CONFIGURABLE_UPFRONT_PRICING_TERM_ID = "<your-configurable-upfront-pricing-term-
id>"

    # Duration for the agreement (e.g., "P366D" for 366 days).
    SELECTOR_VALUE = "<your-selector-value>"

    # Dimension key and quantity defined in your offer.
    DIMENSION_1_KEY = "<your-dimension-key>"
    DIMENSION_1_VALUE = 1
```

```
    # Term ID for the LegalTerm in your offer.
    LEGAL_TERM_ID = "<your-legal-term-id>"

    # Term ID for the ValidityTerm in your offer.
    VALIDITY_TERM_ID = "<your-validity-term-id>"

    @staticmethod
    def list_agreement_charges_and_update_purchase_orders():
        """
        Full end-to-end flow:
        1. Create a SaaS agreement with CONTRACT pricing model with a purchase order
reference.
        2. List charges to retrieve charge IDs and revisions.
        3. Associate the purchase order reference with a specific charge via
UpdatePurchaseOrders.
        4. List charges again to confirm the update.
        """
        client = boto3.client("marketplace-agreement")
        cls = UpdatePurchaseOrdersAfterAgreementAcceptance

        configurable_upfront_pricing_term = {
            "id": cls.CONFIGURABLE_UPFRONT_PRICING_TERM_ID,
            "configuration": {
                "configurableUpfrontPricingTermConfiguration": {
                    "selectorValue": cls.SELECTOR_VALUE,
                    "dimensions": [
                        {
                            "dimensionKey": cls.DIMENSION_1_KEY,
                            "dimensionValue": cls.DIMENSION_1_VALUE,
                        },
                    ],
                }
            },
        }

        legal_term = {"id": cls.LEGAL_TERM_ID}
        validity_term = {"id": cls.VALIDITY_TERM_ID}

        # --- Create Agreement ---
        create_response = client.create_agreement_request(
            clientToken=generate_client_token(),
            intent="NEW",
            requestedTerms=[configurable_upfront_pricing_term, legal_term,
validity_term],
```

```
            agreementProposalIdentifier=cls.AGREEMENT_PROPOSAL_IDENTIFIER,
        )
        agreement_request_id = create_response["agreementRequestId"]
        print("Agreement request created. AgreementRequestId: " +
agreement_request_id)

        # --- Accept Agreement Request with Purchase Order ---
        # The chargeId is available from the CAR response's
chargeSummary.expectedCharges.
        charge_id = create_response["chargeSummary"]["expectedCharges"][0]["id"]
        accept_response = client.accept_agreement_request(
            agreementRequestId=agreement_request_id,
            purchaseOrders=[
                {
                    "chargeId": charge_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                }
            ],
        )
        agreement_id = accept_response["agreementId"]
        print(
            "Agreement request accepted with purchase order reference '"
            + cls.PURCHASE_ORDER_REFERENCE + "'. AgreementId: " + agreement_id
        )

        # --- List Agreement Charges ---
        list_charges_response =
client.list_agreement_charges(agreementId=agreement_id)
        print("All charges for agreement " + agreement_id + ":")
        format_output(list_charges_response)

        # --- Update Purchase Order ---
        first_charge = list_charges_response["items"][0]
        client.update_purchase_orders(
            purchaseOrders=[
                {
                    "agreementId": agreement_id,
                    "purchaseOrderReference": cls.PURCHASE_ORDER_REFERENCE,
                    "chargeRevision": first_charge["revision"],
                    "chargeId": first_charge["id"],
                }
            ],
        )
        print(
```

```
            "Purchase order reference '" + cls.PURCHASE_ORDER_REFERENCE
            + "' updated for ChargeId: " + first_charge["id"]
        )

        # --- Verify Update ---
        lac_response = client.list_agreement_charges(agreementId=agreement_id)
        print("Verified updated charge:")
        format_output(lac_response["items"][0])


if __name__ == "__main__":

 UpdatePurchaseOrdersAfterAgreementAcceptance.list_agreement_charges_and_update_purchase_ord
```

- API-Details finden Sie UpdatePurchaseOrders in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon MSK unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Python.

```
import base64

def lambda_handler(event, context):
    # Iterate through keys
    for key in event['records']:
        print('Key:', key)
        # Iterate through records
        for record in event['records'][key]:
            print('Record:', record)
            # Decode base64
            msg = base64.b64decode(record['value']).decode('utf-8')
            print('Message:', msg)
```

Neptune-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Neptune Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Neptune

Das folgende Codebeispiel zeigt die ersten Schritte mit Neptune.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError


def describe_db_clusters(neptune_client):
    """
    Describes the Amazon Neptune DB clusters using a paginator to handle multiple
 pages.
    Raises ClientError with 'ResourceNotFoundException' if no clusters are found.
    """
    paginator = neptune_client.get_paginator("describe_db_clusters")
    clusters_found = False

    for page in paginator.paginate():
        for cluster in page.get("DBClusters", []):
            clusters_found = True
            print(f"Cluster Identifier: {cluster['DBClusterIdentifier']}")
            print(f"Status: {cluster['Status']}")

    if not clusters_found:
        raise ClientError(
            {
                "Error": {
```

```
                    "Code": "ResourceNotFoundException",
                    "Message": "No Neptune DB clusters found."
                }
            },
            operation_name="DescribeDBClusters"
        )

def main():
    """
    Main entry point: creates the Neptune client and calls the describe operation.
    """
    neptune_client = boto3.client("neptune")
    try:
        describe_db_clusters(neptune_client)
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "ResourceNotFoundException":
            print(f"Resource not found: {e.response['Error']['Message']}")
        else:
            print(f"Unexpected ClientError: {e.response['Error']['Message']}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- API-Details finden Sie DescribeDBClustersPaginator in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Subnetzgruppe von Amazon Neptune.
- Erstellen Sie einen Neptun-Cluster.
- Erstellen Sie eine Neptun-Instance.
- Prüfen des Status einer Neptune-Instance.

- Zeigen Sie Details zum Neptun-Cluster an.
- Stoppen Sie den Neptune-Cluster.
- Starten Sie den Neptune-Cluster.
- Löschen Sie die Neptune-Komponenten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
import time
from botocore.exceptions import ClientError

# Constants used in this scenario
POLL_INTERVAL_SECONDS = 10
TIMEOUT_SECONDS = 1200  # 20 minutes

def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)
```

```
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise

def format_elapsed_time(seconds: int) -> str:
    mins, secs = divmod(seconds, 60)
    hours, mins = divmod(mins, 60)
    return f"{hours:02}:{mins:02}:{secs:02}"


def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")
```

```
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise

def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
```

```
        raise

def wait_for_cluster_status(
        neptune_client,
        cluster_id: str,
        desired_status: str,
        timeout_seconds: int = TIMEOUT_SECONDS,
        poll_interval_seconds: int = POLL_INTERVAL_SECONDS
):
    """
    Waits for a Neptune DB cluster to reach a desired status.

    Args:
        neptune_client (boto3.client): The Amazon Neptune client.
        cluster_id (str): The identifier of the Neptune DB cluster.
        desired_status (str): The target status (e.g., "available", "stopped").
        timeout_seconds (int): Max time to wait in seconds (default: 1200).
        poll_interval_seconds (int): Polling interval in seconds (default: 10).

    Raises:
        RuntimeError: If the desired status is not reached before timeout.
    """
    print(f"Waiting for cluster '{cluster_id}' to reach status
 '{desired_status}'...")
    start_time = time.time()

    while True:
        # Prepare request object
        describe_cluster_request = {
            'DBClusterIdentifier': cluster_id
        }

        # Call the Neptune API
        response = neptune_client.describe_db_clusters(**describe_cluster_request)
        clusters = response.get('DBClusters', [])
        current_status = clusters[0].get('Status') if clusters else None
        elapsed_seconds = int(time.time() - start_time)

        status_str = current_status if current_status else "Unknown"
        print(
            f"\r Elapsed: {format_elapsed_time(elapsed_seconds):<20}  Cluster
 status: {status_str:<20}",
            end="", flush=True
        )
```

```
        if current_status and current_status.lower() == desired_status.lower():
            print(
                f"\nNeptune cluster reached desired status '{desired_status}' after
 {format_elapsed_time(elapsed_seconds)}."
            )
            return

        if elapsed_seconds > timeout_seconds:
            raise RuntimeError(f"Timeout waiting for Neptune cluster to reach
 status: {desired_status}")

        time.sleep(poll_interval_seconds)


def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')
```

```
    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
 may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
 {message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
 flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
 to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)


def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.
```

```
    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
flush=True)
```

```
        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
 to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)



def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier: {cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
```

```
                print(f"IAM Auth Enabled:
 {cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
 {cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
 {cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
 {cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
 found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
 ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise

def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()
```

```
    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
            else:
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
 {current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
 {format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
 '{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)


def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) ->
 str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
```

```
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
 '{db_instance_id}': {e}") from e


def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.
```

```
    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException", "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
 clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}': {e}")
 from e

def get_subnet_ids(vpc_id: str) -> list[str]:
    ec2_client = boto3.client('ec2')

    describe_subnets_request = {
        'Filters': [{'Name': 'vpc-id', 'Values': [vpc_id]}]
    }
```

```
    response = ec2_client.describe_subnets(**describe_subnets_request)
    subnets = response.get('Subnets', [])
    subnet_ids = [subnet['SubnetId'] for subnet in subnets if 'SubnetId' in subnet]
    return subnet_ids


def get_default_vpc_id() -> str:
    ec2_client = boto3.client('ec2')
    describe_vpcs_request = {
        'Filters': [{'Name': 'isDefault', 'Values': ['true']}]
    }

    response = ec2_client.describe_vpcs(**describe_vpcs_request)
    vpcs = response.get('Vpcs', [])
    if not vpcs:
        raise RuntimeError("No default VPC found in this region.")

    default_vpc_id = vpcs[0]['VpcId']
    print(f"Default VPC ID: {default_vpc_id}")
    return default_vpc_id


def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
```

```
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group '{group_name}':
 {e}") from e

def wait_for_input_to_continue():
    input("\nPress <ENTER> to continue...")
    print("Continuing with the program...\n")


def run_scenario(neptune_client, subnet_group_name: str, db_instance_id: str,
 cluster_name: str):
    print("-" * 88)
    print("1. Create a Neptune DB Subnet Group")
    wait_for_input_to_continue()

    try:
```

```
        name, arn = create_subnet_group(neptune_client, subnet_group_name)
        print(f"Subnet group successfully created: {name}")

        print("-" * 88)
        print("2. Create a Neptune Cluster")
        wait_for_input_to_continue()
        db_cluster_id = create_db_cluster(neptune_client, cluster_name)

        print("-" * 88)
        print("3. Create a Neptune DB Instance")
        wait_for_input_to_continue()
        create_db_instance(neptune_client, db_instance_id, cluster_name)

        print("-" * 88)
        print("4. Check the status of the Neptune DB Instance")
        print("""
        Even though you're targeting a single DB instance,
        describe_db_instances supports pagination and can return multiple pages.

        Handling paginated responses ensures your method continues to work reliably
        even if AWS returns large or paged results.
        """)
        wait_for_input_to_continue()
        check_instance_status(neptune_client, db_instance_id, "available")

        print("-" * 88)
        print("5. Show Neptune Cluster details")
        wait_for_input_to_continue()
        describe_db_clusters(neptune_client, db_cluster_id)

        print("-" * 88)
        print("6. Stop the Amazon Neptune cluster")
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for stop_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in a stopped state.
        """)
        wait_for_input_to_continue()
        stop_db_cluster(neptune_client, db_cluster_id)
        check_instance_status(neptune_client, db_instance_id, "stopped")

        print("-" * 88)
        print("7. Start the Amazon Neptune cluster")
```

```
        print("""
            Boto3 doesn't currently offer a
            built-in waiter for start_db_cluster,
            This example implements a custom polling
            strategy until the cluster is in an available state.
        """)
        wait_for_input_to_continue()
        start_db_cluster(neptune_client, db_cluster_id)
        wait_for_cluster_status(neptune_client, db_cluster_id, "available")
        check_instance_status(neptune_client, db_instance_id, "available")

        print("All Neptune resources are now available.")
        print("-" * 88)

        print("-" * 88)
        print("8. Delete the Neptune Assets")
        print("Would you like to delete the Neptune Assets? (y/n)")
        del_ans = input().strip().lower()

        if del_ans == "y":
            print("You selected to delete the Neptune assets.")

            delete_db_instance(neptune_client, db_instance_id)
            delete_db_cluster(neptune_client, db_cluster_id)
            delete_db_subnet_group(neptune_client, subnet_group_name)

            print("Neptune resources deleted successfully")

    except ClientError as ce:
        code = ce.response["Error"]["Code"]

        if code in ("DBInstanceNotFound", "DBInstanceNotFoundFault",
"ResourceNotFound"):
            print(f"Instance '{db_instance_id}' not found.")
        elif code in ("DBClusterNotFound", "DBClusterNotFoundFault",
"ResourceNotFoundFault"):
            print(f"Cluster '{cluster_name}' not found.")
        elif code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"AWS error [{code}]: {ce.response['Error']['Message']}")
```

```
            raise  # re-raise unexpected errors

    except RuntimeError as re:
        print(f"Runtime error or timeout: {re}")


def main():
    neptune_client = boto3.client('neptune')

    # Customize the following names to match your Neptune setup
    # (You must change these to unique values for your environment)
    subnet_group_name = "neptuneSubnetGroup111"
    cluster_name = "neptuneCluster111"
    db_instance_id = "neptuneDB111"

    print("""
    Amazon Neptune is a fully managed graph database service by AWS...
    Let's get started!
    """)
    wait_for_input_to_continue()
    run_scenario(neptune_client, subnet_group_name, db_instance_id, cluster_name)

    print("""
    Thank you for checking out the Amazon Neptune Service Use demo.
    For more AWS code examples, visit:
    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-library.html
    """)

if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - CreateDBCluster
  - CreateDBInstance
  - CreateDBSubnetGroup
  - CreateGraph
  - DeleteDBCluster
  - DeleteDBInstance

- DeleteDBSubnetGroup
- DescribeDBClusters
- DescribeDBInstances
- ExecuteGremlinProfileQuery
- ExecuteGremlinQuery
- ExecuteOpenCypherExplainQuery
- ExecuteQuery
- Starten Sie DBCluster
- Stoppen Sie DBCluster

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDBCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_db_cluster(neptune_client, db_name: str) -> str:
    """
    Creates a Neptune DB cluster and returns its identifier.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        db_name (str): The desired cluster identifier.

    Returns:
        str: The DB cluster identifier.

    Raises:
        RuntimeError: For any failure or AWS error, with a user-friendly message.
    """
```

```
    request = {
        'DBClusterIdentifier': db_name,
        'Engine': 'neptune',
        'DeletionProtection': False,
        'BackupRetentionPeriod': 1
    }

    try:
        response = neptune_client.create_db_cluster(**request)
        cluster = response.get('DBCluster') or {}

        cluster_id = cluster.get('DBClusterIdentifier')
        if not cluster_id:
            raise RuntimeError("Cluster created but no ID returned.")

        print(f"DB Cluster created: {cluster_id}")
        return cluster_id

    except ClientError as e:
        code = e.response["Error"]["Code"]
        message = e.response["Error"]["Message"]

        if code in ("ServiceQuotaExceededException", "DBClusterQuotaExceededFault"):
            raise RuntimeError("You have exceeded the quota for Neptune DB
clusters.") from e
        else:
            raise RuntimeError(f"AWS error [{code}]: {message}") from e

    except Exception as e:
        raise RuntimeError(f"Unexpected error creating DB cluster '{db_name}': {e}")
from e
```

- Weitere API-Informationen finden Sie unter CreateDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_db_instance(neptune_client, db_instance_id: str, db_cluster_id: str) ->
 str:
    try:
        request = {
            'DBInstanceIdentifier': db_instance_id,
            'DBInstanceClass': 'db.r5.large',
            'Engine': 'neptune',
            'DBClusterIdentifier': db_cluster_id
        }

        print(f"Creating Neptune DB Instance: {db_instance_id}")
        response = neptune_client.create_db_instance(**request)

        instance = response.get('DBInstance')
        if not instance or 'DBInstanceIdentifier' not in instance:
            raise RuntimeError("Instance creation succeeded but no ID returned.")

        print(f"Waiting for DB Instance '{db_instance_id}' to become available...")
        waiter = neptune_client.get_waiter('db_instance_available')
        waiter.wait(
            DBInstanceIdentifier=db_instance_id,
            WaiterConfig={'Delay': 30, 'MaxAttempts': 40}
        )

        print(f"DB Instance '{db_instance_id}' is now available.")
        return instance['DBInstanceIdentifier']

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
```

```
        else:
            print(f"Couldn't create DB instance. Here's why: {code}: {message}")
        raise

    except Exception as e:
        print(f"Unexpected error creating DB instance '{db_instance_id}': {e}")
        raise RuntimeError(f"Unexpected error creating DB instance
'{db_instance_id}': {e}") from e
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`CreateDBSubnetGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSubnetGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_subnet_group(neptune_client, group_name: str):
    """
    Creates a Neptune DB subnet group and returns its name and ARN.

    Args:
        neptune_client (boto3.client): The Neptune client object.
        group_name (str): The desired name of the subnet group.

    Returns:
        tuple(str, str): (subnet_group_name, subnet_group_arn)

    Raises:
        RuntimeError: For quota errors or other AWS-related failures.
    """
    vpc_id = get_default_vpc_id()
```

```
    subnet_ids = get_subnet_ids(vpc_id)

    request = {
        'DBSubnetGroupName': group_name,
        'DBSubnetGroupDescription': 'My Neptune subnet group',
        'SubnetIds': subnet_ids,
        'Tags': [{'Key': 'Environment', 'Value': 'Dev'}]
    }

    try:
        response = neptune_client.create_db_subnet_group(**request)
        sg = response.get("DBSubnetGroup", {})
        name = sg.get("DBSubnetGroupName")
        arn = sg.get("DBSubnetGroupArn")

        if not name or not arn:
            raise RuntimeError("Response missing subnet group name or ARN.")

        print(f"Subnet group created: {name}")
        print(f"ARN: {arn}")
        return name, arn

    except ClientError as e:
        code = e.response["Error"]["Code"]
        msg = e.response["Error"]["Message"]

        if code == "ServiceQuotaExceededException":
            print("Subnet group quota exceeded.")
            raise RuntimeError("Subnet group quota exceeded.") from e
        else:
            print(f"AWS error [{code}]: {msg}")
            raise RuntimeError(f"AWS error [{code}]: {msg}") from e

    except Exception as e:
        print(f"Unexpected error creating subnet group '{group_name}': {e}")
        raise RuntimeError(f"Unexpected error creating subnet group '{group_name}':
{e}") from e
```

- API-Details finden Sie CreateDBSubnetGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateGraph`

Das folgende Codebeispiel zeigt die Verwendung. `CreateGraph`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**

"""

GRAPH_NAME = "sample-analytics-graph"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)
    execute_create_graph(client, GRAPH_NAME)

def execute_create_graph(client, graph_name):
    try:
        print("Creating Neptune graph...")
        response = client.create_graph(
            graphName=graph_name,
            provisionedMemory = 16
        )
```

```
        created_graph_name = response.get("name")
        graph_arn = response.get("arn")
        graph_endpoint = response.get("endpoint")

        print("Graph created successfully!")
        print(f"Graph Name: {created_graph_name}")
        print(f"Graph ARN: {graph_arn}")
        print(f"Graph Endpoint: {graph_endpoint}")

    except ClientError as e:
        print(f"Failed to create graph: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Failed to create graph: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- API-Details finden Sie CreateGraph in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteDBCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_cluster(neptune_client, cluster_id: str):
    """
    Deletes a Neptune DB cluster and throws exceptions to the caller.

    Args:
        neptune_client (boto3.client): The Neptune client object.
```

```
        cluster_id (str): The ID of the Neptune DB cluster to be deleted.

    Raises:
        ClientError: If the delete operation fails.
    """
    request = {
        'DBClusterIdentifier': cluster_id,
        'SkipFinalSnapshot': True
    }

    try:
        print(f"Deleting DB Cluster: {cluster_id}")
        neptune_client.delete_db_cluster(**request)

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBClusterNotFoundFault":
            print(f"Cluster '{cluster_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        else:
            print(f"Couldn't delete DB cluster. {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DeleteDBCluster in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_instance(neptune_client, instance_id: str):
    """
    Deletes a Neptune DB instance and waits for its deletion to complete.
    Raises exception to be handled by calling code.
    """
    print(f"Initiating deletion of DB Instance: {instance_id}")
    try:
        neptune_client.delete_db_instance(
            DBInstanceIdentifier=instance_id,
            SkipFinalSnapshot=True
        )

        print(f"Waiting for DB Instance '{instance_id}' to be deleted...")
        waiter = neptune_client.get_waiter('db_instance_deleted')
        waiter.wait(
            DBInstanceIdentifier=instance_id,
            WaiterConfig={
                'Delay': 30,
                'MaxAttempts': 40
            }
        )

        print(f"DB Instance '{instance_id}' successfully deleted.")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBInstanceNotFoundFault":
            print(f"Instance '{instance_id}' not found or already deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete DB instance. {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBSubnetGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBSubnetGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_db_subnet_group(neptune_client, subnet_group_name):
    """
    Deletes a Neptune DB subnet group synchronously using Boto3.

    Args:
        neptune_client (boto3.client): The Neptune client.
        subnet_group_name (str): The name of the DB subnet group to delete.

    Raises:
        ClientError: If the delete operation fails.
    """
    delete_group_request = {
        'DBSubnetGroupName': subnet_group_name
    }

    try:
        neptune_client.delete_db_subnet_group(**delete_group_request)
        print(f"# Deleting Subnet Group: {subnet_group_name}")

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "DBSubnetGroupNotFoundFault":
            print(f"Subnet group '{subnet_group_name}' not found or already
 deleted.")
        elif code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't delete subnet group. {code}: {message}")
```

```
        raise
```

- API-Details finden Sie DeleteDBSubnetGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBClusters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_db_clusters(neptune_client, cluster_id: str):
    """
    Describes details of a Neptune DB cluster, paginating if needed.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_id (str): The ID of the cluster to describe.

    Raises:
        ClientError: If there's an AWS API error (e.g., cluster not found).
    """
    paginator = neptune_client.get_paginator('describe_db_clusters')

    try:
        pages = paginator.paginate(DBClusterIdentifier=cluster_id)

        found = False
        for page in pages:
            for cluster in page.get('DBClusters', []):
                found = True
                print(f"Cluster Identifier: {cluster.get('DBClusterIdentifier')}")
                print(f"Status: {cluster.get('Status')}")
                print(f"Engine: {cluster.get('Engine')}")
```

```
                print(f"Engine Version: {cluster.get('EngineVersion')}")
                print(f"Endpoint: {cluster.get('Endpoint')}")
                print(f"Reader Endpoint: {cluster.get('ReaderEndpoint')}")
                print(f"Availability Zones: {cluster.get('AvailabilityZones')}")
                print(f"Subnet Group: {cluster.get('DBSubnetGroup')}")
                print("VPC Security Groups:")
                for vpc_group in cluster.get('VpcSecurityGroups', []):
                    print(f"  - {vpc_group.get('VpcSecurityGroupId')}")
                print(f"Storage Encrypted: {cluster.get('StorageEncrypted')}")
                print(f"IAM Auth Enabled:
{cluster.get('IAMDatabaseAuthenticationEnabled')}")
                print(f"Backup Retention Period:
{cluster.get('BackupRetentionPeriod')} days")
                print(f"Preferred Backup Window:
{cluster.get('PreferredBackupWindow')}")
                print(f"Preferred Maintenance Window:
{cluster.get('PreferredMaintenanceWindow')}")
                print("------")

        if not found:
            # Treat empty response as cluster not found
            raise ClientError(
                {"Error": {"Code": "DBClusterNotFound", "Message": f"No cluster
found with ID '{cluster_id}'"}},
                "DescribeDBClusters"
            )

    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
permissions.")
        elif code == "DBClusterNotFound":
            print(f"Cluster '{cluster_id}' not found. Please verify the cluster
ID.")
        else:
            print(f"Couldn't describe DB cluster. Here's why: {code}: {message}")
        raise
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der API-Referenz zum AWS SDK für Python (Boto3).

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def check_instance_status(neptune_client, instance_id: str, desired_status: str):
    """
    Polls the status of a Neptune DB instance until it reaches desired_status.
    Uses pagination via describe_db_instances — even for a single instance.

    Raises:
      ClientError: If describe_db_instances fails (e.g., instance not found).
      RuntimeError: If timeout expires before reaching desired status.
    """
    paginator = neptune_client.get_paginator('describe_db_instances')
    start_time = time.time()

    while True:
        try:
            pages = paginator.paginate(DBInstanceIdentifier=instance_id)
            instances = []
            for page in pages:
                instances.extend(page.get('DBInstances', []))

        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBInstanceNotFound":
                print(f"Instance '{instance_id}' not found. Please verify the
 instance ID.")
            else:
```

```
                print(f"Failed to describe DB instance. {code}: {message}")
            raise

        current_status = instances[0].get('DBInstanceStatus') if instances else None
        elapsed = int(time.time() - start_time)

        print(f"\rElapsed: {format_elapsed_time(elapsed)}  Status:
{current_status}", end="", flush=True)

        if current_status and current_status.lower() == desired_status.lower():
            print(f"\nInstance '{instance_id}' reached '{desired_status}' in
{format_elapsed_time(elapsed)}.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for '{instance_id}' to reach
'{desired_status}'")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

`ExecuteGremlinProfileQuery`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteGremlinProfileQuery`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Replace this with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify Endpoint]:8182"

def main():
```

```
    """
    Entry point of the program. Initializes the Neptune client and executes the
 Gremlin query.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_gremlin_query(neptune_client)


def execute_gremlin_query(neptune_client):
    """
    Executes a Gremlin query against an Amazon Neptune database.
    """
    try:
        print("Querying Neptune...")

        response = neptune_client.execute_gremlin_explain_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )

        print("Full Response:")
        print(response['output'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Error calling Neptune: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ExecuteGremlinProfileQuery in der AWS API-Referenz zum SDK für Python (Boto3).

`ExecuteGremlinQuery`

Das folgende Codebeispiel zeigt die Verwendung. `ExecuteGremlinQuery`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**

"""

# Replace with your actual Neptune endpoint
NEPTUNE_ENDPOINT = "https://[Specify-Your-Endpoint]:8182"

def main():
    """
    Entry point of the program. Initializes the Neptune client and runs both EXPLAIN
 and PROFILE queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})
```

```
    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    try:
        run_profile_query(neptune_client)
    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")

def run_profile_query(neptune_client):
    """
    Runs a PROFILE query on the Neptune graph database.
    """
    print("Running Gremlin PROFILE query...")

    try:
        response = neptune_client.execute_gremlin_profile_query(
            gremlinQuery="g.V().has('code', 'ANC')"
        )
        print("Profile Query Result:")
        output = response.get("output")
        if output:
            print(output.read().decode('utf-8'))
        else:
            print("No explain output returned.")
    except Exception as e:
        print(f"Failed to execute PROFILE query: {str(e)}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ExecuteGremlinQuery in der AWS API-Referenz zum SDK für Python (Boto3).

`ExecuteOpenCypherExplainQuery`

Das folgende Codebeispiel zeigt die Verwendung. `ExecuteOpenCypherExplainQuery`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Replace with your actual Neptune endpoint URL
NEPTUNE_ENDPOINT = "https://<your-neptune-endpoint>:8182"

def main():
    """
    Entry point: Create Neptune client and execute different OpenCypher queries.
    """
    config = Config(connect_timeout=10, read_timeout=30, retries={'max_attempts':
 3})

    neptune_client = boto3.client(
        "neptunedata",
        endpoint_url=NEPTUNE_ENDPOINT,
        config=config
    )

    execute_open_cypher_query_without_params(neptune_client)
    execute_open_cypher_query_with_params(neptune_client)
    execute_open_cypher_explain_query(neptune_client)

def execute_open_cypher_query_without_params(client):
    """
    Executes a simple OpenCypher query without parameters.
    """
    try:
        print("\nRunning OpenCypher query without parameters...")
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n"
        )
        print("Results:")
```

```
        print(resp['results'])

    except Exception as e:
        print(f"Error in simple OpenCypher query: {str(e)}")


def execute_open_cypher_query_with_params(client):
    """
    Executes an OpenCypher query using parameters.
    """
    try:
        print("\nRunning OpenCypher query with parameters...")
        parameters = {'code': 'ANC'}
        resp = client.execute_open_cypher_query(
            openCypherQuery="MATCH (n {code: $code}) RETURN n",
            parameters=json.dumps(parameters)
        )
        print("Results:")
        print(resp['results'])

    except Exception as e:
        print(f"Error in parameterized OpenCypher query: {str(e)}")

def execute_open_cypher_explain_query(client):
    """
    Runs an OpenCypher EXPLAIN query in debug mode.
    """
    try:
        print("\nRunning OpenCypher EXPLAIN query (debug mode)...")
        resp = client.execute_open_cypher_explain_query(
            openCypherQuery="MATCH (n {code: 'ANC'}) RETURN n",
            explainMode="details"
        )
        results = resp.get('results')
        if results is None:
            print("No explain results returned.")
        else:
            try:
                print("Explain Results:")
                print(results.read().decode('UTF-8'))
            except Exception as e:
                print(f"Error in OpenCypher EXPLAIN query: {str(e)}")

    except ClientError as e:
```

```
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"BotoCore error: {str(e)}")
    except Exception as e:
        print(f"Unexpected error: {str(e)}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie ExecuteOpenCypherExplainQuery in der AWS API-Referenz zum SDK für Python (Boto3).

`ExecuteQuery`

Das folgende Codebeispiel zeigt die Verwendung. `ExecuteQuery`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Running this example.

--------------------------------------------------------------------------------
VPC Networking Requirement:
--------------------------------------------------------------------------------
Amazon Neptune must be accessed from **within the same VPC** as the Neptune cluster.
It does not expose a public endpoint, so this code must be executed from:

  - An **AWS Lambda function** configured to run inside the same VPC
  - An **EC2 instance** or **ECS task** running in the same VPC
  - A connected environment such as a **VPN**, **AWS Direct Connect**, or a **peered
 VPC**
"""
```

```
GRAPH_ID = "<your-graph-id>"

def main():
    config = Config(retries={"total_max_attempts": 1, "mode": "standard"},
 read_timeout=None)
    client = boto3.client("neptune-graph", config=config)

    try:
        print("\n--- Running OpenCypher query without parameters ---")
        run_open_cypher_query(client, GRAPH_ID)

        print("\n--- Running OpenCypher query with parameters ---")
        run_open_cypher_query_with_params(client, GRAPH_ID)

        print("\n--- Running OpenCypher explain query ---")
        run_open_cypher_explain_query(client, GRAPH_ID)

    except Exception as e:
        print(f"Unexpected error in main: {e}")

def run_open_cypher_query(client, graph_id):
    """
    Run an OpenCypher query without parameters.
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_query_with_params(client, graph_id):
    """
    Run an OpenCypher query with parameters.
    """
    try:
```

```
        parameters = {'code': 'ANC'}
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: $code}) RETURN n",
            language='OPEN_CYPHER',
            parameters=parameters
        )
        print(resp['payload'].read().decode('UTF-8'))

    except client.exceptions.InternalServerException as e:
        print(f"InternalServerException: {e.response['Error']['Message']}")
    except ClientError as e:
        print(f"ClientError: {e.response['Error']['Message']}")
    except Exception as e:  # <--- ADD THIS BLOCK
        print(f"Unexpected error: {e}")

def run_open_cypher_explain_query(client, graph_id):
    """
    Run an OpenCypher explain query (explainMode = "debug").
    """
    try:
        resp = client.execute_query(
            graphIdentifier=graph_id,
            queryString="MATCH (n {code: 'ANC'}) RETURN n",
            language='OPEN_CYPHER',
            explainMode='DETAILS'
        )
        print(resp['payload'].read().decode('UTF-8'))

    except ClientError as e:
        print(f"Neptune error: {e.response['Error']['Message']}")
    except BotoCoreError as e:
        print(f"Unexpected Boto3 error: {str(e)}")
    except Exception as e:  # <-- Add this generic catch
        print(f"Unexpected error: {str(e)}")

if __name__ == "__main__":
    main()
```

- API-Details finden Sie ExecuteQuery in der AWS API-Referenz zum SDK für Python (Boto3).

StartDBCluster

Das folgende Codebeispiel zeigt die Verwendung. `StartDBCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def start_db_cluster(neptune_client, cluster_identifier: str):
    """
    Starts an Amazon Neptune DB cluster and waits until it reaches 'available'.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: Propagates AWS API issues like resource not found.
        RuntimeError: If cluster doesn't reach 'available' within timeout.
    """
    try:
        # Initial wait in case the cluster was just stopped
        time.sleep(30)
        neptune_client.start_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't start DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
```

```
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'available':
            print(f"\n# Cluster '{cluster_identifier}' is available.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
to become available.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter StartDBCluster in der API-Referenz zum AWS -SDK für Python (Boto3).

`StopDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`StopDBCluster`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def stop_db_cluster(neptune_client, cluster_identifier: str):
    """
    Stops an Amazon Neptune DB cluster and waits until it's fully stopped.

    Args:
        neptune_client (boto3.client): The Neptune client.
        cluster_identifier (str): The DB cluster identifier.

    Raises:
        ClientError: For AWS API errors (e.g., resource not found).
        RuntimeError: If the cluster doesn't stop within the timeout.
    """
    try:
        neptune_client.stop_db_cluster(DBClusterIdentifier=cluster_identifier)
    except ClientError as err:
        code = err.response["Error"]["Code"]
        message = err.response["Error"]["Message"]

        if code == "AccessDeniedException":
            print("Access denied. Please ensure you have the necessary
 permissions.")
        else:
            print(f"Couldn't stop DB cluster. Here's why: {code}: {message}")
        raise

    start_time = time.time()
    paginator = neptune_client.get_paginator('describe_db_clusters')

    while True:
        try:
            pages = paginator.paginate(DBClusterIdentifier=cluster_identifier)
            clusters = []
            for page in pages:
                clusters.extend(page.get('DBClusters', []))
```

```
        except ClientError as err:
            code = err.response["Error"]["Code"]
            message = err.response["Error"]["Message"]

            if code == "DBClusterNotFound":
                print(f"Cluster '{cluster_identifier}' not found while polling. It
may have been deleted.")
            else:
                print(f"Couldn't describe DB cluster. Here's why: {code}:
{message}")
            raise

        status = clusters[0].get('Status') if clusters else None
        elapsed = time.time() - start_time

        print(f"\rElapsed: {int(elapsed)}s – Cluster status: {status}", end="",
flush=True)

        if status and status.lower() == 'stopped':
            print(f"\nCluster '{cluster_identifier}' is now stopped.")
            return

        if elapsed > TIMEOUT_SECONDS:
            raise RuntimeError(f"Timeout waiting for cluster '{cluster_identifier}'
to stop.")

        time.sleep(POLL_INTERVAL_SECONDS)
```

- Weitere API-Informationen finden Sie unter StopDBCluster in der API-Referenz zum AWS -SDK für Python (Boto3).

Organizations-Beispiele unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Organizations Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachPolicy`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account, or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- API-Details finden Sie AttachPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `CreatePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON before
                    it is sent to AWS. The specific format depends on the policy
 type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
    :return: The newly created policy.
    """
    try:
        response = orgs_client.create_policy(
            Name=name,
            Description=description,
            Content=json.dumps(content),
            Type=policy_type,
        )
        policy = response["Policy"]
        logger.info("Created policy %s.", name)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- API-Details finden Sie CreatePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

DeletePolicy

Das folgende Codebeispiel zeigt die Verwendung. `DeletePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- API-Details finden Sie DeletePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

DescribePolicy

Das folgende Codebeispiel zeigt die Verwendung. `DescribePolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- API-Details finden Sie DescribePolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`DetachPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `DetachPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- API-Details finden Sie DetachPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung. `ListPolicies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- API-Details finden Sie ListPolicies in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Partner Central unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Partner Central Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AssignOpportunity`

Das folgende Codebeispiel zeigt die Verwendung`AssignOpportunity`.

SDK für Python (Boto3)

Weisen Sie eine bestehende Opportunity einem anderen Benutzer neu zu.

```
#!/usr/bin/env python

"""
Purpose
PC-API-07 Assigning a new owner
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def assign_opportunity(identifier):
    assign_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "Assignee": {
            "BusinessTitle": "OpportunityOwner",
            "Email": "test@test.com",
            "FirstName": "John",
            "LastName": "Doe"
        }
    }
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.assign_opportunity(**assign_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "04236468"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Assigning a new owner to an opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(assign_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie AssignOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

`AssociateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung. `AssociateOpportunity`

SDK für Python (Boto3)

Erstellen Sie eine formelle Zuordnung zwischen einer Opportunity und verschiedenen verwandten Entitäten.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
```

```
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def associate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0059717"
    opportunityIdentifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Associate Opportunity.")
    print("-" * 88)
```

```
    helper.pretty_print_datetime(associate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie AssociateOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

`CreateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung. `CreateOpportunity`

SDK für Python (Boto3)

Erstellen Sie eine Opportunity.

```
#!/usr/bin/env python
import boto3
import logging
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
import utils.stringify_details as sd
from botocore.client import ClientError
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

def create_opportunity(partner_central_client):
    create_opportunity_request = helper.remove_nulls(sd.stringify_json("src/
create_opportunity/createOpportunity.json"))
    try:
        # Perform an API call
        response =
 partner_central_client.create_opportunity(**create_opportunity_request)

        helper.pretty_print_datetime(response)

        # Retrieve the opportunity details
```

```
        get_response = partner_central_client.get_opportunity(
            Identifier=response["Id"],
            Catalog=CATALOG_TO_USE
        )
        helper.pretty_print_datetime(get_response)
        return response
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Create Opportunity.")
    print("-" * 88)

    partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
    )

    create_opportunity(partner_central_client)

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie CreateOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

`DisassociateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung. `DisassociateOpportunity`

SDK für Python (Boto3)

Entfernen Sie eine bestehende Verknüpfung zwischen einer Opportunity und verwandten Entitäten.

```
#!/usr/bin/env python

"""
```

```
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def disassociate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0049999"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(disassociate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie DisassociateOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

`GetAwsOpportunitySummary`

Das folgende Codebeispiel zeigt die Verwendung. `GetAwsOpportunitySummary`

SDK für Python (Boto3)

Ruft eine Zusammenfassung einer AWS Opportunity ab.

```
#!/usr/bin/env python

"""
Purpose
PC-API-25 Retrieves a summary of an AWS Opportunity. LifeCycle.ReviewStatus=Approved
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)
```

```
def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "RelatedOpportunityIdentifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_aws_opportunity_summary(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get AWS Opportunity summary.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetAwsOpportunitySummary in der API-Referenz zum AWS SDK für Python (Boto3).

`GetEngagementInvitation`

Das folgende Codebeispiel zeigt die Verwendung. `GetEngagementInvitation`

SDK für Python (Boto3)

Ruft die Details einer Einladung zur Verlobung ab, die von AWS einem Partner geteilt wurde.

```
#!/usr/bin/env python
```

```
"""
Purpose
PC-API-22  GetOpportunityEngagementInvitation - Retrieves details of a specific
 engagement invitation.
This operation allows partners to view the invitation and its associated
 information,
such as the customer, project, and lifecycle details.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity_engagement_invitation(identifier):
    get_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_engagement_invitation_re
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral-selling:us-east-1:aws:catalog/Sandbox/
engagement-invitation/engi-0000000IS0Qga"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Given the ARN identifier, retrieve details of Opportunity Engagement
 Invitation.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity_engagement_invitation(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetEngagementInvitation in der API-Referenz zum AWS SDK für Python (Boto3).

`GetOpportunity`

Das folgende Codebeispiel zeigt die Verwendung. `GetOpportunity`

SDK für Python (Boto3)

Rufen Sie eine Opportunity ab.

```
#!/usr/bin/env python

"""
Purpose
PC-API -08 Get updated Opportunity given opportunity id
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)
```

```
def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response = partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie GetOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

`ListEngagementInvitations`

Das folgende Codebeispiel zeigt die Verwendung. `ListEngagementInvitations`

SDK für Python (Boto3)

Ruft eine Liste mit Einladungen zur Interaktion ab, die an den Partner gesendet wurden.

```
#!/usr/bin/env python

"""
```

```
Purpose
PC-API-21 ListEngagementInvitations - Retrieves a list of engagement invitations
 based on specified criteria.
This operation allows partners to view all invitations to engagement.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def list_engagement_invitations():
    list_engagement_invitations_request ={
        "Catalog": CATALOG_TO_USE,
        "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_engagement_invitations(**list_engagement_invitations_request)
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Retrieve list of Engagement Invitations.")
    print("-" * 88)

    helper.pretty_print_datetime(list_engagement_invitations())

if __name__ == "__main__":
```

```
    usage_demo()
```

- API-Details finden Sie ListEngagementInvitations in der AWS API-Referenz für SDK für Python (Boto3).

`ListOpportunities`

Das folgende Codebeispiel zeigt die Verwendung. `ListOpportunities`

SDK für Python (Boto3)

Listen Sie Opportunitys auf.

```
#!/usr/bin/env python

"""
Purpose
PC-API -18 Getting list of Opportunities
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_opportunities():

    opportunity_list = []

    list_opportunities_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
```

```
    try:
        # Perform an API call
        response =
 partner_central_client.list_opportunities(**list_opportunities_request)
        opportunity_list.extend(response["OpportunitySummaries"])

        while "NextToken" in response and response["NextToken"] is not None:
            list_opportunities_request["NextToken"] = response["NextToken"]
            response =
 partner_central_client.list_opportunities(**list_opportunities_request)
            opportunity_list.extend(response["OpportunitySummaries"])

        return opportunity_list

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of Opportunities.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_opportunities())

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie ListOpportunities in der AWS API-Referenz für SDK für Python (Boto3).

`ListSolutions`

Das folgende Codebeispiel zeigt die Verwendung. `ListSolutions`

SDK für Python (Boto3)

Ruft eine Liste von Partner Solutions ab, die der Partner bei Partner Central registriert hat.

```
#!/usr/bin/env python
```

```
"""
Purpose
PC-API-10 Getting list of solutions
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_solutions():
    list_solutions_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response = partner_central_client.list_solutions(**list_solutions_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Getting list of solutions.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_solutions())

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie ListSolutions in der AWS API-Referenz für SDK für Python (Boto3).

`RejectEngagementInvitation`

Das folgende Codebeispiel zeigt die Verwendung. `RejectEngagementInvitation`

SDK für Python (Boto3)

Lehnt einen AWS geteilten EngagementInvitation Inhalt ab.

```
#!/usr/bin/env python

"""
Purpose
PC-API-05 AWS Originated AO rejection - RejectOpportunityEngagementInvitation -
 Rejects a engagement invitation.
This action indicates that the partner does not wish to participate in the
 engagement and
provides a reason for the rejection.
Upon rejection, a OpportunityEngagementInvitationRejected event is triggered.
Subsequently, the invitation will no longer be available for the partner to act on.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def reject_opportunity_engagement_invitation(identifier, reject_reason):
    reject_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
```

```
        "RejectionReason": reject_reason
    }
    try:
        # Perform an API call
        response =
 partner_central_client.reject_engagement_invitation(**reject_opportunity_engagement_invitat
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isviga"
    reject_reason = "Customer problem unclear"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier and reject reason, reject the Opportunity
 Engagement Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(reject_opportunity_engagement_invitation(identifier,
 reject_reason))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie RejectEngagementInvitation in der API-Referenz zum AWS SDK für Python (Boto3).

`StartEngagementByAcceptingInvitationTask`

Das folgende Codebeispiel zeigt die Verwendung. `StartEngagementByAcceptingInvitationTask`

SDK für Python (Boto3)

Beginnt das Engagement mit der Annahme einer EngagementInvitation.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def start_engagement_by_accepting_invitation_task(identifier):

    response = get_opportunity(identifier)

    if ( response['Status'] == 'PENDING') :
```

```
        accept_opportunity_engagement_invitation_request ={
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-123456"
        }
        try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_by_accepting_invitation_task(**accept_opportunity_e
            return response

        except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None
    else:
        return None

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isusga"
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_by_accepting_invitation_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie StartEngagementByAcceptingInvitationTask in der AWS API-Referenz zum SDK für Python (Boto3).

`StartEngagementFromOpportunityTask`

Das folgende Codebeispiel zeigt die Verwendung. `StartEngagementFromOpportunityTask`

SDK für Python (Boto3)

Initiiert den Interaktionsprozess von einer bestehenden Opportunity aus, indem die Einladung zur Interaktion angenommen und eine entsprechende Opportunity im System des Partners erstellt wird.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def start_engagement_from_opportunity_task(identifier):

    start_engagement_from_opportunity_task_request ={
            "AwsSubmission": {
                "InvolvementType": "Co-Sell",
                "Visibility": "Full"
            },
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-annjqwesdsd99"
    }
    try:
            # Perform an API call
```

```
            response =
 partner_central_client.start_engagement_from_opportunity_task(**start_engagement_from_oppor
            return response

    except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Start Engagement from Opportunity Task.")
    print("-" * 88)

    helper.pretty_print_datetime(start_engagement_from_opportunity_task(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie StartEngagementFromOpportunityTask in der AWS API-Referenz für SDK für Python (Boto3).

`UpdateOpportunity`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateOpportunity`

SDK für Python (Boto3)

Aktualisieren Sie eine Opportunity.

```
#!/usr/bin/env python

"""
Purpose
PC-API-2  Updating Partner Originated Opportunity
"""
import logging
```

```
import boto3
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
from botocore.client import ClientError
import utils.stringify_details as sd
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response = partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity():
    update_opportunity_request_orig = sd.stringify_json("src/update_opportunity/
update_opportunity_technical_validation.json")
    update_opportunity_request =
 helper.remove_nulls(update_opportunity_request_orig)

    try:
        # Perform an API call
        response =
 partner_central_client.update_opportunity(**update_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
```

```
        print(err.response)

def update_opportunity_if_eligible(identifier):
    response = get_opportunity(identifier)
    if response is not None:
        return update_opportunity()
    else:
        print("Failed to retrieve opportunity details")

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Updating opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(update_opportunity_if_eligible(identifier))

if __name__ == "__main__":
    usage_demo()
```

- API-Details finden Sie UpdateOpportunity in der AWS API-Referenz für SDK für Python (Boto3).

Szenarien

Aktualisieren der zugehörigen Entität einer Opportunity

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Trennen Sie die Zuordnung einer alten Entität.
- Ordnen Sie eine neue Entität zu.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktualisieren der zugehörigen Entität einer Opportunity

```
#!/usr/bin/env python

"""
Purpose
PC-API -17 Replacing a solution
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def replace_solution(original_entity_identifier, new_entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : original_entity_identifier
    }

    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
```

```
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : new_entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    original_entity_identifier = "S-0049999"
    new_entity_identifier = "S-0050014"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Replacing a solution.")
    print("-" * 88)

    helper.pretty_print_datetime(replace_solution(original_entity_identifier,
 new_entity_identifier, opportunityIdentifier))

if __name__ == "__main__":
    usage_demo()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AssociateOpportunity
  - DisassociateOpportunity

Beispiele für Amazon Pinpoint unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Pinpoint Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung`SendMessages`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def send_email_message(
    pinpoint_client,
    app_id,
    sender,
    to_addresses,
    char_set,
    subject,
    html_message,
    text_message,
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project ID to use when you send this message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
 account
                         is in the sandbox, these addresses must be verified.
    :param char_set: The character encoding to use for the subject line and message
                     body of the email.
    :param subject: The subject line of the email.
    :param html_message: The body of the email for recipients whose email clients
 can
                         display HTML content.
    :param text_message: The body of the email for recipients whose email clients
                         don't support HTML content.
    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {
                    "EmailMessage": {
                        "FromAddress": sender,
                        "SimpleEmail": {
                            "Subject": {"Charset": char_set, "Data": subject},
                            "HtmlPart": {"Charset": char_set, "Data": html_message},
```

```
                            "TextPart": {"Charset": char_set, "Data": text_message},
                        },
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]["Result"].items()
        }


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)
    -------------------------------------
    This email was sent with Amazon Pinpoint using the AWS SDK for Python (Boto3).
    For more information, see https://aws.amazon.com/sdk-for-python/
                """
    html_message = """<html>
    <head></head>
    <body>
      <h1>Amazon Pinpoint Test (SDK for Python (Boto3)</h1>
      <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
          AWS SDK for Python (Boto3)</a>.</p>
    </body>
    </html>
                """

    print("Sending email.")
    message_ids = send_email_message(
        boto3.client("pinpoint"),
        app_id,
        sender,
```

```
        [to_address],
        char_set,
        subject,
        html_message,
        text_message,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
    main()
```

Senden Sie eine SMS-Nachricht.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_sms_message(
    pinpoint_client,
    app_id,
    origination_number,
    destination_number,
    message,
    message_type,
):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you send
                   this message. The SMS channel must be enabled for the project or
                   application.
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This phone
                               number must be associated with your Amazon Pinpoint
                               account and be in E.164 format.
    :param message: The content of the SMS message.
```

```
    :param message_type: The type of SMS message that you want to send. If you send
                         time-sensitive content, specify TRANSACTIONAL. If you send
                         marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "Body": message,
                        "MessageType": message_type,
                        "OriginationNumber": origination_number,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK for
 "
        "Python (Boto 3)."
    )
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
    message_id = send_sms_message(
        boto3.client("pinpoint"),
        app_id,
        origination_number,
        destination_number,
```

```
        message,
        message_type,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

Senden Sie eine E-Mail-Nachricht mit einer vorhandenen E-Mail-Vorlage.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_email_message(
    pinpoint_client, project_id, sender, to_addresses, template_name,
 template_version
):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: The Amazon Pinpoint project ID to use when you send this
 message.
    :param sender: The "From" address. This address must be verified in
                   Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
                         account is in the sandbox, these addresses must be
 verified.
    :param template_name: The name of the email template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
```

```
            MessageRequest={
                "Addresses": {
                    to_address: {"ChannelType": "EMAIL"} for to_address in
 to_addresses
                },
                "MessageConfiguration": {"EmailMessage": {"FromAddress": sender}},
                "TemplateConfiguration": {
                    "EmailTemplate": {
                        "Name": template_name,
                        "Version": template_version,
                    }
                },
            },
        )
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
    else:
        return {
            to_address: message["MessageId"]
            for to_address, message in response["MessageResponse"]["Result"].items()
        }


def main():
    project_id = "296b04b342374fceb661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send_templated_email_message(
        boto3.client("pinpoint"),
        project_id,
        sender,
        to_addresses,
        template_name,
        template_version,
    )
    print(f"Message sent! Message IDs: {message_ids}")


if __name__ == "__main__":
```

```
    main()
```

Senden Sie eine Textnachricht mit einer vorhandenen SMS-Vorlage.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version,
):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the
 message.
    :param template_version: The version number of the message template.

    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                "Addresses": {destination_number: {"ChannelType": "SMS"}},
                "MessageConfiguration": {
                    "SMSMessage": {
                        "MessageType": message_type,
```

```
                        "OriginationNumber": origination_number,
                    }
                },
                "TemplateConfiguration": {
                    "SMSTemplate": {"Name": template_name, "Version":
 template_version}
                },
            },
        )

    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response["MessageResponse"]["Result"][destination_number]
["MessageId"]


def main():
    region = "us-east-1"
    origination_number = "+18555550001"
    destination_number = "+14255550142"
    project_id = "7353f53e6885409fa32d07cedexample"
    message_type = "TRANSACTIONAL"
    template_name = "My_SMS_Template"
    template_version = "1"
    message_id = send_templated_sms_message(
        boto3.client("pinpoint", region_name=region),
        project_id,
        destination_number,
        message_type,
        origination_number,
        template_name,
        template_version,
    )
    print(f"Message sent! Message ID: {message_id}.")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie SendMessages in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon Pinpoint SMS und Voice API unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon Pinpoint SMS and Voice API AWS SDK für Python (Boto3) Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendVoiceMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendVoiceMessage`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def send_voice_message(
    sms_voice_client,
```

```
    origination_number,
    caller_id,
    destination_number,
    language_code,
    voice_id,
    ssml_message,
):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
                               The phone number must be associated with your Amazon
                               Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's
                      device. The phone number must be associated with your Amazon
                      Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone
                               number in E.164 format.
    :param language_code: The language to use when sending the message.
    :param voice_id: The Amazon Polly voice that you want to use to send the
 message.
    :param ssml_message: The content of the message. This example uses SSML to
 control
                         certain aspects of the message, such as the volume and the
                         speech rate. The message must not contain line breaks.
    :return: The ID of the message.
    """
    try:
        response = sms_voice_client.send_voice_message(
            DestinationPhoneNumber=destination_number,
            OriginationPhoneNumber=origination_number,
            CallerId=caller_id,
            Content={
                "SSMLMessage": {
                    "LanguageCode": language_code,
                    "VoiceId": voice_id,
                    "Text": ssml_message,
                }
            },
        )
    except ClientError:
        logger.exception(
            "Couldn't send message from %s to %s.",
```

```
            origination_number,
            destination_number,
        )
        raise
    else:
        return response["MessageId"]


def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "using the <break strength='weak'/>AWS SDK for Python (Boto3). "
        "<amazon:effect phonation='soft'>Thank you for listening."
        "</amazon:effect>"
        "</speak>"
    )
    print(f"Sending voice message from {origination_number} to
 {destination_number}.")
    message_id = send_voice_message(
        boto3.client("pinpoint-sms-voice"),
        origination_number,
        caller_id,
        destination_number,
        language_code,
        voice_id,
        ssml_message,
    )
    print(f"Message sent!\nMessage ID: {message_id}")


if __name__ == "__main__":
    main()
```

- API-Details finden Sie SendVoiceMessage in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon Polly unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
```

```
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def describe_voices(self):
        """
        Gets metadata about available voices.

        :return: The list of voice metadata.
        """
        try:
            response = self.polly_client.describe_voices()
            self.voice_metadata = response["Voices"]
            logger.info("Got metadata about %s voices.", len(self.voice_metadata))
        except ClientError:
            logger.exception("Couldn't get voice metadata.")
            raise
        else:
            return self.voice_metadata
```

- API-Details finden Sie DescribeVoices in der AWS API-Referenz zum SDK für Python (Boto3).

`GetLexicon`

Das folgende Codebeispiel zeigt die Verwendung. `GetLexicon`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
```

```
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
            raise
        else:
            return response
```

- API-Details finden Sie GetLexicon in der AWS API-Referenz zum SDK für Python (Boto3).

`GetSpeechSynthesisTask`

Das folgende Codebeispiel zeigt die Verwendung. `GetSpeechSynthesisTask`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
 status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response = self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response["SynthesisTask"]
            logger.info("Got synthesis task. Status is %s.", task["TaskStatus"])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- API-Details finden Sie GetSpeechSynthesisTask in der AWS API-Referenz zum SDK für Python (Boto3).

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung. `ListLexicons`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response["Lexicons"]
            logger.info("Got %s lexicons.", len(lexicons))
        except ClientError:
            logger.exception(
                "Couldn't get  %s.",
            )
            raise
        else:
            return lexicons
```

- API-Details finden Sie ListLexicons in der AWS API-Referenz zum SDK für Python (Boto3).

`PutLexicon`

Das folgende Codebeispiel zeigt die Verwendung. `PutLexicon`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def create_lexicon(self, name, content):
        """
        Creates a lexicon with the specified content. A lexicon contains custom
        pronunciations.

        :param name: The name of the lexicon.
        :param content: The content of the lexicon.
        """
        try:
            self.polly_client.put_lexicon(Name=name, Content=content)
            logger.info("Created lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't create lexicon %s.")
            raise
```

- API-Details finden Sie PutLexicon in der AWS API-Referenz zum SDK für Python (Boto3).

`StartSpeechSynthesisTask`

Das folgende Codebeispiel zeigt die Verwendung. `StartSpeechSynthesisTask`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def do_synthesis_task(
        self,
        text,
        engine,
        voice,
        audio_format,
        s3_bucket,
        lang_code=None,
        include_visemes=False,
        wait_callback=None,
    ):
```

```
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real
time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech. When
                             speech marks are synthesized, the output format is
JSON.
        :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                          write access to. Synthesis output is written to this
bucket.
        :param lang_code: The language code of the voice to use. This has an effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon Polly
                                to synthesize a list of visemes, using the specified
                                text and voice. A viseme represents the visual
position
                                of the face and mouth when saying part of a word.
        :param wait_callback: A callback function that is called periodically during
                              task processing, to give the caller an opportunity to
                              take action, such as to display status.
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "OutputS3BucketName": s3_bucket,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.start_speech_synthesis_task(**kwargs)
            speech_task = response["SynthesisTask"]
            logger.info("Started speech synthesis task %s.", speech_task["TaskId"])
```

```
            viseme_task = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.start_speech_synthesis_task(**kwargs)
                viseme_task = response["SynthesisTask"]
                logger.info("Started viseme synthesis task %s.",
viseme_task["TaskId"])
        except ClientError:
            logger.exception("Couldn't start synthesis task.")
            raise
        else:
            bucket = self.s3_resource.Bucket(s3_bucket)
            audio_stream = self._wait_for_task(
                10, speech_task["TaskId"], "speech", wait_callback, bucket
            )

            visemes = None
            if include_visemes:
                viseme_data = self._wait_for_task(
                    10, viseme_task["TaskId"], "viseme", wait_callback, bucket
                )
                visemes = [
                    json.loads(v) for v in viseme_data.read().decode().split() if v
                ]

            return audio_stream, visemes
```

- API-Details finden Sie StartSpeechSynthesisTask in der AWS API-Referenz zum SDK für Python (Boto3).

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung. `SynthesizeSpeech`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""

    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
 resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None


    def synthesize(
        self, text, engine, voice, audio_format, lang_code=None,
 include_visemes=False
    ):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech. When
                             speech marks are synthesized, the output format is
 JSON.
        :param lang_code: The language code of the voice to use. This has an effect
                          only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon Polly
                                to synthesize a list of visemes, using the specified
                                text and voice. A viseme represents the visual
 position
                                of the face and mouth when saying part of a word.
```

```
        :return: The audio stream that contains the synthesized speech and a list
                 of visemes that are associated with the speech audio.
        """
        try:
            kwargs = {
                "Engine": engine,
                "OutputFormat": audio_format,
                "Text": text,
                "VoiceId": voice,
            }
            if lang_code is not None:
                kwargs["LanguageCode"] = lang_code
            response = self.polly_client.synthesize_speech(**kwargs)
            audio_stream = response["AudioStream"]
            logger.info("Got audio stream spoken by %s.", voice)
            visemes = None
            if include_visemes:
                kwargs["OutputFormat"] = "json"
                kwargs["SpeechMarkTypes"] = ["viseme"]
                response = self.polly_client.synthesize_speech(**kwargs)
                visemes = [
                    json.loads(v)
                    for v in response["AudioStream"].read().decode().split()
                    if v
                ]
                logger.info("Got %s visemes.", len(visemes))
        except ClientError:
            logger.exception("Couldn't get audio stream.")
            raise
        else:
            return audio_stream, visemes
```

- API-Details finden Sie SynthesizeSpeech in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen einer Lippensynchronisationsanwendung

Das folgende Codebeispiel zeigt, wie Sie eine Lippensynchronisationsanwendung mit Amazon Polly erstellen.

SDK für Python (Boto3)

Zeigt, wie Amazon Polly und Tkinter verwendet werden, um eine Lippensynchronisierungsanwendung zu erstellen, die ein animiertes gesprochenes Gesicht zusammen mit der von Amazon Polly synthetisierten Sprache anzeigt. Lip-sync wird erreicht, indem Amazon Polly eine Liste von Bildern anfordert, die der synthetisierten Sprache entsprechen.

- Erhalten Sie Sprachmetadaten von Amazon Polly und zeigen Sie diese in einer Tkinter-Anwendung an.
- Erhalten Sie synthetisierte Sprachaufnahmen und passende Viseme-Sprachmarkierungen von Amazon Polly.
- Spielen Sie die Audioaufnahme mit synchronisierten Mundbewegungen in einem animierten Gesicht ab.
- Asynchrone Syntheseaufgaben für lange Texte übermitteln und die Ausgabe aus einem Bucket von Amazon Simple Storage Service (Amazon S3) abrufen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly

Beispiele für Amazon RDS unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"""
Purpose

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database
 Service
(Amazon RDS) to list the databases in your account.
```

```
"""

import boto3
from botocore.exceptions import ClientError

# Create an RDS client
rds_client = boto3.client("rds")

# Create a paginator for the describe_db_instances operation
paginator = rds_client.get_paginator("describe_db_instances")

try:
    # Use the paginator to get a list of DB instances
    response_iterator = paginator.paginate(
        PaginationConfig={
            "MaxItems": 123,
            "PageSize": 50,  # Adjust PageSize as needed
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response
    instances_found = False
    for page in response_iterator:
        if "DBInstances" in page and page["DBInstances"]:
            instances_found = True
            print("Your RDS instances are:")
            for db in page["DBInstances"]:
                print(db["DBInstanceIdentifier"])

    if not instances_found:
        print("No RDS instances found!")

except ClientError as e:
    print(f"Couldn't list RDS instances. Here's why: {e.response['Error']
['Message']}")
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB
 instances."""

    def __init__(self, instance_wrapper):
        """
        :param instance_wrapper: An object that wraps Amazon RDS DB instance
 actions.
        """
        self.instance_wrapper = instance_wrapper

    def create_parameter_group(self, parameter_group_name, db_engine):
        """
        Shows how to get available engine versions for a specified database engine
 and
        create a DB parameter group that is compatible with a selected engine
 family.
```

```
        :param parameter_group_name: The name given to the newly created parameter
group.
        :param db_engine: The database engine to use as a basis.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB instance parameter group named
{parameter_group_name}."
        )
        parameter_group = self.instance_wrapper.get_parameter_group(
            parameter_group_name
        )
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.instance_wrapper.get_engine_versions(db_engine)
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ", families)
            print(f"Creating a parameter group.")
            self.instance_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.instance_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def update_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
```

```
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
                )
                update_params.append(auto_inc)
        self.instance_wrapper.update_parameters(parameter_group_name, update_params)
        print(
            "You can get a list of parameters you've set by specifying a source of
'user'."
        )
        user_parameters = self.instance_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_instance(self, instance_name, db_name, db_engine, parameter_group):
        """
        Shows how to create a DB instance that contains a database of a specified
        type and is configured to use a custom DB parameter group.

        :param instance_name: The name given to the newly created DB instance.
        :param db_name: The name given to the created database.
        :param db_engine: The engine of the created database.
        :param parameter_group: The parameter group that is associated with the DB
instance.
        :return: The newly created DB instance.
        """
        print("Checking for an existing DB instance.")
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
        if db_inst is None:
            print("Let's create a DB instance.")
            admin_username = q.ask(
                "Enter an administrator user name for the database: ", q.non_empty
            )
```

```
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters): ",
                q.non_empty,
            )
            engine_versions = self.instance_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [ver["EngineVersion"] for ver in engine_versions]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            engine_selection = engine_versions[engine_index]
            print(
                "The available micro DB instance classes for your database engine
are:"
            )
            inst_opts = self.instance_wrapper.get_orderable_instances(
                engine_selection["Engine"], engine_selection["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"]
                    for opt in inst_opts
                    if "micro" in opt["DBInstanceClass"]
                }
            )
            inst_index = q.choose(
                "Which micro DB instance class do you want to use? ", inst_choices
            )
            group_name = parameter_group["DBParameterGroupName"]
            storage_type = "standard"
            allocated_storage = 5
            print(
                f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
                f"The DB instance is configured to use your custom parameter group
{group_name},\n"
                f"selected engine {engine_selection['EngineVersion']},\n"
                f"selected DB instance class {inst_choices[inst_index]},"
                f"and {allocated_storage} GiB of {storage_type} storage.\n"
                f"This typically takes several minutes."
            )
            db_inst = self.instance_wrapper.create_db_instance(
                db_name,
```

```
                instance_name,
                group_name,
                engine_selection["Engine"],
                engine_selection["EngineVersion"],
                inst_choices[inst_index],
                storage_type,
                allocated_storage,
                admin_username,
                admin_password,
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print(
            "You can now connect to your database using your favorite MySql client.
\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your DB instance. Pass the endpoint,
\n"
            "port, and administrator user name to 'mysql' and enter your password\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P {db_inst['Endpoint']
['Port']} "
            f"-u {db_inst['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Amazon RDS:\n"
```

```
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL"
        )
        print("-" * 88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB instance (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{instance_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
 instance_name)
            while snapshot.get("Status") != "available":
                wait(10)
                snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
            pp(snapshot)
            print("-" * 88)

    def cleanup(self, db_inst, parameter_group_name):
        """
        Shows how to clean up a DB instance and parameter group.
        Before the parameter group can be deleted, all associated DB instances must
 first
        be deleted.

        :param db_inst: The DB instance to delete.
        :param parameter_group_name: The DB parameter group to delete.
        """
        if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")
```

```
 self.instance_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(
                "Waiting for the DB instance to delete. This typically takes several
 minutes."
            )
            while db_inst is not None:
                wait(10)
                db_inst = self.instance_wrapper.get_db_instance(
                    db_inst["DBInstanceIdentifier"]
                )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

    def run_scenario(self, db_engine, parameter_group_name, instance_name, db_name):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
            "get started with DB instances demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(parameter_group_name,
 db_engine)
        self.update_parameters(parameter_group_name)
        db_inst = self.create_instance(
            instance_name, db_name, db_engine, parameter_group
        )
        self.display_connection(db_inst)
        self.create_snapshot(instance_name)
        self.cleanup(db_inst, parameter_group_name)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            "mysql",
            "doc-example-parameter-group",
```

```
            "doc-example-instance",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie Funktionen, die vom Szenario aufgerufen werden, um Amazon-RDS-Aktionen zu verwalten.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
```

```
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

```
    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
```

```
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def create_snapshot(self, snapshot_id, instance_id):
```

```
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return snapshot


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
```

```
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
                raise
        else:
            return db_inst


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
```

```
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_instance(
        self,
        db_name,
        instance_id,
        parameter_group_name,
        db_engine,
        db_engine_version,
        instance_class,
        storage_type,
        allocated_storage,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
instance.
        :param db_engine: The database engine of a database to create in the DB
instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
```

```
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
 new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
```

```
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie CreateDBParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDBSnapshot`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client
```

```
    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id
            )
            snapshot = response["DBSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
```

```
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Python (Boto3).

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
```

```
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBEngineVersions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
```

```
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
of
                                       engine versions to those that are compatible
with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return versions
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBInstances`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.
```

```
        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Phyton (Boto3).

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
```

```
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeDBParameters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
```

```
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator = self.rds_client.get_paginator("describe_db_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zum AWS SDK für Python (Boto3).

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zum AWS SDK für Phyton (Boto3).

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by the
DB instance.
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS API-Referenz zum SDK für Python (Boto3).

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung. `ModifyDBParameterGroup`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
 Parameters=update_parameters
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
```

```
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldeinformationen enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Python.

```
import json
import os
import boto3
import pymysql

# RDS settings
proxy_host_name = os.environ['PROXY_HOST_NAME']
port = int(os.environ['PORT'])
db_name = os.environ['DB_NAME']
db_user_name = os.environ['DB_USER_NAME']
aws_region = os.environ['AWS_REGION']


# Fetch RDS Auth Token
def get_auth_token():
```

```
    client = boto3.client('rds')
    token = client.generate_db_auth_token(
        DBHostname=proxy_host_name,
        Port=port
        DBUsername=db_user_name
        Region=aws_region
    )
    return token

def lambda_handler(event, context):
    token = get_auth_token()
    try:
        connection = pymysql.connect(
            host=proxy_host_name,
            user=db_user_name,
            password=token,
            db=db_name,
            port=port,
            ssl={'ca': 'Amazon RDS'}  # Ensure you have the CA bundle for SSL
 connection
        )

        with connection.cursor() as cursor:
            cursor.execute('SELECT %s + %s AS sum', (3, 2))
            result = cursor.fetchone()

        return result

    except Exception as e:
        return (f"Error: {str(e)}")  # Return an error message if an exception
 occurs
```

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Python (Boto3) Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldeinformationen enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon Redshift unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon Redshift

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Redshift beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_redshift(redshift_client):
```

```
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Redshift client and list
    the clusters in your account. This list might be empty if you haven't created
    any clusters.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param redshift_client: A Boto3 Redshift Client object.
    """
    print("Hello, Redshift! Let's list your clusters:")
    paginator = redshift_client.get_paginator("describe_clusters")
    clusters = []
    for page in paginator.paginate():
        clusters.extend(page["Clusters"])

    print(f"{len(clusters)} cluster(s) were found.")

    for cluster in clusters:
        print(f"  {cluster['ClusterIdentifier']}")


if __name__ == "__main__":
    hello_redshift(boto3.client("redshift"))
```

- API-Details finden Sie DescribeClusters in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines Redshift-Clusters.
- Listen Sie die Datenbanken im Cluster auf.
- Erstellen einer Tabelle mit dem Namen „Movies“.
- Befüllen der Tabelle „Movies“.
- Abfragen der Tabelle „Movies“ nach Jahr.
- Ändern des Redshift-Clusters.

- Löschen Sie den Amazon-Redshift-Cluster.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftScenario:
    """Runs an interactive scenario that shows how to get started with Redshift."""

    def __init__(self, redshift_wrapper, redshift_data_wrapper):
        self.redshift_wrapper = redshift_wrapper
        self.redshift_data_wrapper = redshift_data_wrapper

    def redhift_scenario(self, json_file_path):
        database_name = "dev"

        print(DASHES)
        print("Welcome to the Amazon Redshift SDK Getting Started example.")
        print(
            """
      This Python program demonstrates how to interact with Amazon Redshift
      using the AWS SDK for Python (Boto3).

      Amazon Redshift is a fully managed, petabyte-scale data warehouse
      service hosted in the cloud.

      The program's primary functionalities include cluster creation,
      verification of cluster readiness, listing databases, table creation,
      populating data within the table, and executing SQL statements.

      It also demonstrates querying data from the Movies table.

      Upon completion, all AWS resources are cleaned up.
    """
        )
        if not os.path.isfile(json_file_path):
            logging.error(f"The file {json_file_path} does not exist.")
            return
```

```
        print("Let's get started...")
        user_name = q.ask("Please enter your user name (default is awsuser):")
        user_name = user_name if user_name else "awsuser"

        print(DASHES)
        user_password = q.ask(
            "Please enter your user password (default is AwsUser1000):"
        )
        user_password = user_password if user_password else "AwsUser1000"

        print(DASHES)
        print(
            """A Redshift cluster refers to the collection of computing resources
and storage that work
            together to process and analyze large volumes of data."""
        )
        cluster_id = q.ask(
            "Enter a cluster identifier value (default is redshift-cluster-movies):
"
        )
        cluster_id = cluster_id if cluster_id else "redshift-cluster-movies"

        self.redshift_wrapper.create_cluster(
            cluster_id, "ra3.4xlarge", user_name, user_password, True, 2
        )

        print(DASHES)
        print(f"Wait until {cluster_id} is available. This may take a few
minutes...")
        q.ask("Press Enter to continue...")

        self.wait_cluster_available(cluster_id)

        print(DASHES)

        print(
            f"""
      When you created {cluster_id}, the dev database is created by default and
used in this scenario.

      To create a custom database, you need to have a CREATEDB privilege.
      For more information, see the documentation here:
      https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
```

```
    """
        )
        q.ask("Press Enter to continue...")
        print(DASHES)

        print(DASHES)
        print(f"List databases in {cluster_id}")
        q.ask("Press Enter to continue...")
        databases = self.redshift_data_wrapper.list_databases(
            cluster_id, database_name, user_name
        )
        print(f"The cluster contains {len(databases)} database(s).")
        for database in databases:
            print(f"    Database: {database}")
        print(DASHES)

        print(DASHES)
        print("Now you will create a table named Movies.")
        q.ask("Press Enter to continue...")

        self.create_table(cluster_id, database_name, user_name)

        print(DASHES)

        print("Populate the Movies table using the Movies.json file.")
        print(
            "Specify the number of records you would like to add to the Movies
Table."
        )
        print("Please enter a value between 50 and 200.")

        while True:
            try:
                num_records = int(q.ask("Enter a value: ", q.is_int))
                if 50 <= num_records <= 200:
                    break
                else:
                    print("Invalid input. Please enter a value between 50 and 200.")
            except ValueError:
                print("Invalid input. Please enter a value between 50 and 200.")

        self.populate_table(
            cluster_id, database_name, user_name, json_file_path, num_records
        )
```

```
        print(DASHES)
        print("Query the Movies table by year. Enter a value between 2012-2014.")

        while True:
            movie_year = int(q.ask("Enter a year: ", q.is_int))
            if 2012 <= movie_year <= 2014:
                break
            else:
                print("Invalid input. Please enter a valid year between 2012 and
2014.")

        # Function to query database
        sql_id = self.query_movies_by_year(
            database_name, user_name, movie_year, cluster_id
        )

        print(f"The identifier of the statement is {sql_id}")

        print("Checking statement status...")
        self.wait_statement_finished(sql_id)
        result = self.redshift_data_wrapper.get_statement_result(sql_id)

        self.display_movies(result)

        print(DASHES)

        print(DASHES)
        print("Now you will modify the Redshift cluster.")
        q.ask("Press Enter to continue...")

        preferred_maintenance_window = "wed:07:30-wed:08:00"
        self.redshift_wrapper.modify_cluster(cluster_id,
preferred_maintenance_window)

        print(DASHES)

        print(DASHES)
        delete = q.ask("Do you want to delete the cluster? (y/n) ", q.is_yesno)

        if delete:
            print(f"You selected to delete {cluster_id}")
            q.ask("Press Enter to continue...")
            self.redshift_wrapper.delete_cluster(cluster_id)
```

```
        else:
            print(f"Cluster {cluster_id}cluster_id was not deleted")

        print(DASHES)
        print("This concludes the Amazon Redshift SDK Getting Started scenario.")
        print(DASHES)

    def create_table(self, cluster_id, database, username):
        self.redshift_data_wrapper.execute_statement(
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql="CREATE TABLE Movies (statement_id INT PRIMARY KEY, title
VARCHAR(100), year INT)",
        )

        print("Table created: Movies")


    def populate_table(self, cluster_id, database, username, file_name, number):
        with open(file_name) as f:
            data = json.load(f)

        i = 0
        for record in data:
            if i == number:
                break

            statement_id = i
            title = record["title"]
            year = record["year"]
            i = i + 1
            parameters = [
                {"name": "statement_id", "value": str(statement_id)},
                {"name": "title", "value": title},
                {"name": "year", "value": str(year)},
            ]

            self.redshift_data_wrapper.execute_statement(
                cluster_identifier=cluster_id,
                database_name=database,
                user_name=username,
                sql="INSERT INTO Movies VALUES(:statement_id, :title, :year)",
                parameter_list=parameters,
```

```
            )

        print(f"{i} records inserted into Movies table")

    def wait_cluster_available(self, cluster_id):
        """
        Waits for a cluster to be available.

        :param cluster_id: The cluster identifier.

        Note: The cluster_available waiter can also be used.
        It is not used in this case to allow an elapsed time message.
        """
        cluster_ready = False
        start_time = time.time()

        while not cluster_ready:
            time.sleep(30)
            cluster = self.redshift_wrapper.describe_clusters(cluster_id)
            status = cluster[0]["ClusterStatus"]
            if status == "available":
                cluster_ready = True
            elif status != "creating":
                raise Exception(
                    f"Cluster {cluster_id} creation failed with status {status}."
                )

            elapsed_seconds = int(round(time.time() - start_time))
            minutes = int(elapsed_seconds // 60)
            seconds = int(elapsed_seconds % 60)

            print(f"Elapsed Time: {minutes}:{seconds:02d} - status {status}...")

            if minutes > 30:
                raise Exception(
                    f"Cluster {cluster_id} is not available after 30 minutes."
                )

    def query_movies_by_year(self, database, username, year, cluster_id):
        sql = "SELECT * FROM Movies WHERE year = :year"

        params = [{"name": "year", "value": str(year)}]

        response = self.redshift_data_wrapper.execute_statement(
```

```
            cluster_identifier=cluster_id,
            database_name=database,
            user_name=username,
            sql=sql,
            parameter_list=params,
        )

        return response["Id"]

    @staticmethod
    def display_movies(response):
        metadata = response["ColumnMetadata"]
        records = response["Records"]

        title_column_index = None
        for i in range(len(metadata)):
            if metadata[i]["name"] == "title":
                title_column_index = i
                break

        if title_column_index is None:
            print("No title column found.")
            return

        print(f"Found {len(records)} movie(s).")
        for record in records:
            print(f"   {record[title_column_index]['stringValue']}")

    def wait_statement_finished(self, sql_id):
        while True:
            time.sleep(1)
            response = self.redshift_data_wrapper.describe_statement(sql_id)
            status = response["Status"]
            print(f"Statement status is {status}.")

            if status == "FAILED":
                print(f"The query failed because {response['Error']}. Ending
program")
                raise Exception("The Query Failed. Ending program")
            elif status == "FINISHED":
                break
```

Hauptfunktion, die die Implementierung eines Szenarios zeigt.

```
def main():
    redshift_client = boto3.client("redshift")
    redshift_data_client = boto3.client("redshift-data")
    redshift_wrapper = RedshiftWrapper(redshift_client)
    redshift_data_wrapper = RedshiftDataWrapper(redshift_data_client)
    redshift_scenario = RedshiftScenario(redshift_wrapper, redshift_data_wrapper)
    redshift_scenario.redhift_scenario(
        f"{os.path.dirname(__file__)}/../../../resources/sample_files/movies.json"
    )
```

Die im Szenario verwendete Wrapper-Funktionen.

```
    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
```

```
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def execute_statement(
        self, cluster_identifier, database_name, user_name, sql, parameter_list=None
    ):
        """
        Executes a SQL statement.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param user_name: The user's name.
        :param sql: The SQL statement.
        :param parameter_list: The optional SQL statement parameters.
        :return: The SQL statement result.
        """

        try:
            kwargs = {
                "ClusterIdentifier": cluster_identifier,
                "Database": database_name,
                "DbUser": user_name,
                "Sql": sql,
            }
            if parameter_list:
                kwargs["Parameters"] = parameter_list
            response = self.client.execute_statement(**kwargs)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't execute statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
```

```
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
```

```
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def list_databases(self, cluster_identifier, database_name, database_user):
        """
        Lists databases in a cluster.

        :param cluster_identifier: The cluster identifier.
        :param database_name: The database name.
        :param database_user: The database user.
        :return: The list of databases.
        """
        try:
            paginator = self.client.get_paginator("list_databases")
            databases = []
            for page in paginator.paginate(
                ClusterIdentifier=cluster_identifier,
                Database=database_name,
                DbUser=database_user,
            ):
                databases.extend(page["Databases"])

            return databases
        except ClientError as err:
            logging.error(
                "Couldn't list databases. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.
```

```
        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier, SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def create_cluster(
        self,
        cluster_identifier,
        node_type,
        master_username,
        master_user_password,
        publicly_accessible,
        number_of_nodes,
    ):
        """
        Creates a cluster.

        :param cluster_identifier: The name of the cluster.
        :param node_type: The type of node in the cluster.
        :param master_username: The master username.
        :param master_user_password: The master user password.
        :param publicly_accessible: Whether the cluster is publicly accessible.
        :param number_of_nodes: The number of nodes in the cluster.
        :return: The cluster.
        """

        try:
```

```
            cluster = self.client.create_cluster(
                ClusterIdentifier=cluster_identifier,
                NodeType=node_type,
                MasterUsername=master_username,
                MasterUserPassword=master_user_password,
                PubliclyAccessible=publicly_accessible,
                NumberOfNodes=number_of_nodes,
            )
            return cluster
        except ClientError as err:
            logging.error(
                "Couldn't create a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- API-Details finden Sie CreateCluster in der API-Referenz für AWS SDK für Python (Boto3).

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
```

```
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def delete_cluster(self, cluster_identifier):
        """
        Deletes a cluster.

        :param cluster_identifier: The cluster identifier.
        """
        try:
            self.client.delete_cluster(
                ClusterIdentifier=cluster_identifier, SkipFinalClusterSnapshot=True
            )
        except ClientError as err:
            logging.error(
                "Couldn't delete a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- API-Details finden Sie DeleteCluster in der API-Referenz für AWS SDK für Python (Boto3).

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeClusters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def describe_clusters(self, cluster_identifier):
        """
        Describes a cluster.

        :param cluster_identifier: The cluster identifier.
        :return: A list of clusters.
        """
        try:
            kwargs = {}
            if cluster_identifier:
                kwargs["ClusterIdentifier"] = cluster_identifier

            paginator = self.client.get_paginator("describe_clusters")
            clusters = []
            for page in paginator.paginate(**kwargs):
                clusters.extend(page["Clusters"])

            return clusters

        except ClientError as err:
            logging.error(
                "Couldn't describe a cluster. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
    client = boto3.client("redshift")
    redhift_wrapper = RedshiftWrapper(client)
```

- API-Details finden Sie DescribeClusters in der API-Referenz für AWS SDK für Python (Boto3).

`DescribeStatement`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeStatement`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""

    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def describe_statement(self, statement_id):
        """
        Describes a SQL statement.
```

```
        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            response = self.client.describe_statement(Id=statement_id)
            return response
        except ClientError as err:
            logging.error(
                "Couldn't describe statement. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftDataWrapper Objekt.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- API-Details finden Sie DescribeStatement in der API-Referenz für AWS SDK für Python (Boto3).

`GetStatementResult`

Das folgende Codebeispiel zeigt die Verwendung. `GetStatementResult`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftDataWrapper:
    """Encapsulates Amazon Redshift data."""
```

```
    def __init__(self, client):
        """
        :param client: A Boto3 RedshiftDataWrapper client.
        """
        self.client = client


    def get_statement_result(self, statement_id):
        """
        Gets the result of a SQL statement.

        :param statement_id: The SQL statement identifier.
        :return: The SQL statement result.
        """
        try:
            result = {
                "Records": [],
            }
            paginator = self.client.get_paginator("get_statement_result")
            for page in paginator.paginate(Id=statement_id):
                if "ColumnMetadata" not in result:
                    result["ColumnMetadata"] = page["ColumnMetadata"]
                result["Records"].extend(page["Records"])
            return result
        except ClientError as err:
            logging.error(
                "Couldn't get statement result. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftDataWrapper Objekt.

```
    client = boto3.client("redshift-data")
    redshift_data_wrapper = RedshiftDataWrapper(client)
```

- API-Details finden Sie GetStatementResult in der API-Referenz für AWS SDK für Python (Boto3).

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung. `ModifyCluster`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RedshiftWrapper:
    """
    Encapsulates Amazon Redshift cluster operations.
    """

    def __init__(self, redshift_client):
        """
        :param redshift_client: A Boto3 Redshift client.
        """
        self.client = redshift_client


    def modify_cluster(self, cluster_identifier, preferred_maintenance_window):
        """
        Modifies a cluster.

        :param cluster_identifier: The cluster identifier.
        :param preferred_maintenance_window: The preferred maintenance window.
        """
        try:
            self.client.modify_cluster(
                ClusterIdentifier=cluster_identifier,
                PreferredMaintenanceWindow=preferred_maintenance_window,
            )
        except ClientError as err:
            logging.error(
                "Couldn't modify a cluster. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Der folgende Code instanziiert das RedshiftWrapper Objekt.

```
client = boto3.client("redshift")
redhift_wrapper = RedshiftWrapper(client)
```

- API-Details finden Sie ModifyCluster in der API-Referenz für AWS SDK für Python (Boto3).

Beispiele für Amazon Rekognition unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
```

```
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- API-Details finden Sie CompareFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCollection`

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollectionManager:
```

```
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- API-Details finden Sie CreateCollection in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteCollection`

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.", self.collection_id)
            raise
```

- API-Details finden Sie DeleteCollection in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFaces`

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
        """
        try:
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
```

```
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.", self.collection_id)
            raise
        else:
            return deleted_ids
```

- API-Details finden Sie DeleteFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeCollection`

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
```

```
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
```

```
            raise
        else:
            return self.to_dict()
```

- API-Details finden Sie DescribeCollection in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung. `DetectFaces`

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client
```

```
    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- API-Details finden Sie DetectFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung. `DetectLabels`

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
```

```
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- API-Details finden Sie DetectLabels in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung. `DetectModerationLabels`

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
```

```
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- API-Details finden Sie DetectModerationLabels in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung. `DetectText`

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
```

```
                        an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- API-Details finden Sie DetectText in der AWS API-Referenz zum SDK für Python (Boto3).

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung. `IndexFaces`

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
```

```
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.", image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- API-Details finden Sie IndexFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung. `ListCollections`

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
 self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
```

```
            return collections
```

- API-Details finden Sie ListCollections in der AWS API-Referenz zum SDK für Python (Boto3).

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung. `ListFaces`

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client
```

```
    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces), self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- API-Details finden Sie ListFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung. `RecognizeCelebrities`

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
```

```
            response =
 self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- API-Details finden Sie RecognizeCelebrities in der AWS API-Referenz zum SDK für Python (Boto3).

`SearchFaces`

Das folgende Codebeispiel zeigt die Verwendung. `SearchFaces`

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
```

```
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
```

```
        :return: The list of matching faces found in the collection. This list does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- API-Details finden Sie SearchFaces in der AWS API-Referenz zum SDK für Python (Boto3).

`SearchFacesByImage`

Das folgende Codebeispiel zeigt die Verwendung. `SearchFacesByImage`

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
```

```
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
```

```
            )
            raise
        else:
            return image_face, collection_faces
```

- API-Details finden Sie SearchFacesByImage in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen Sie eine Sammlung und finden Sie Gesichter darin

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon-Rekognition-Sammlung.
- Fügen Sie der Sammlung Bilder hinzu und erkennen Sie Gesichter darin.
- Durchsuchen Sie die Sammlung nach Gesichtern, die einem Referenzbild entsprechen.
- Löschen einer Sammlung.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung suchen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, die Amazon-Rekognition-Funktionen wrappen (verpacken).

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage
```

```
logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened and
 its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
```

```
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
```

```
            response =
 self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
```

```
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
```

```
            raise
        else:
            return self.to_dict()


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.", self.collection_id)
            raise


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
```

```
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.", image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces), self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
```

```
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
```

```
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
```

```
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
```

```
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Verwenden Sie die Wrapper-Klassen, um eine Sammlung von Gesichtern aus einer Reihe von Bildern zu erstellen und dann nach Gesichtern in der Sammlung zu suchen.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    rekognition_client = boto3.client("rekognition")
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
```

```
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
            rekognition_client,
            image_name="biking",
        ),
    ]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection("doc-example-collection-demo")
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the first face in the "
        f"list (Face ID: {faces[0].face_id}."
    )
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
    print(f"Found {len(found_faces)} matching faces.")
    for face in found_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the largest face in "
        f"{images[0].image_name}."
    )
    image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
```

```
    print(f"The largest face in {images[0].image_name} is:")
    pprint(image_face.to_dict())
    print(f"Found {len(match_faces)} matching faces.")
    for face in match_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    collection.delete_collection()
    print("Thanks for watching!")
    print("-" * 88)
```

Elemente in Bildern erkennen und anzeigen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erkennen von Elementen in Bildern mithilfe von Amazon Rekognition.
- Zeigen Sie Bilder an und zeichnen Sie Begrenzungsrahmen um die erkannten Elemente.

Weitere Informationen finden Sie unter Anzeigen von Begrenzungsrahmen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Klassen, um Amazon-Rekognition-Funktionen zu umschließen.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
```

```
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened and
 its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from the
```

```
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
```

```
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)
```

```
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

```
    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Erstellen Sie Hilfsfunktionen zum Zeichnen von Begrenzungsrahmen und Polygonen.

```
import io
import logging
from PIL import Image, ImageDraw

logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
```

```
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Erstellen Sie Klassen, um von Amazon Rekognition zurückgegebene Objekte zu analysieren.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
```

```
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
```

```
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering
```

```
class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.
```

```
        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
```

```
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.parent_name is not None:
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
```

```
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Verwenden Sie die Wrapper-Klassen, um Elemente in Bildern zu erkennen und ihre Begrenzungsrahmen anzuzeigen. Die in diesem Beispiel verwendeten Bilder finden Sie GitHub zusammen mit Anweisungen und weiterem Code weiter.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image["Bytes"],
        [[face.bounding_box for face in faces]],
        ["aqua"],
    )
    input("Press Enter to continue.")
```

```
    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
    box_sets = []
    colors = ["aqua", "red", "white", "blue", "yellow", "green"]
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst["BoundingBox"] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
    show_bounding_boxes(
        street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
    )
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client
    )
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image["Bytes"],
        [[celeb.face.bounding_box for celeb in celebs]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls", rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
```

```
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
"aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie das verwenden AWS SDK für Python (Boto3) , um eine Webanwendung zu erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingebaute Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.
- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten unter ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Beispiele für Amazon S3 unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")
```

```
    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50,  # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")


if __name__ == "__main__":
    hello_s3()
```

- API-Details finden Sie ListBuckets in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)

    bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
```

```
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )
```

```
    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)? "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
```

```
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
```

```
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- API-Details finden Sie CopyObject in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung. `CreateBucket`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Bucket mit Standardeinstellungen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this is
                                not specified, the Region configured in your shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})
```

```
            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

Erstellen Sie einen versionierten Bucket mit einer Lebenszyklus-Konfiguration.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
```

```
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- API-Details finden Sie CreateBucket in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteBucket`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- API-Details finden Sie DeleteBucket in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteBucketCors`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteBucketCors`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- API-Details finden Sie DeleteBucketCors in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteBucketLifecycle`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteBucketLifecycle`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.", self.bucket.name
            )
        except ClientError:
```

```
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- API-Details finden Sie DeleteBucketLifecycle in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteBucketPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
```

```
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- API-Details finden Sie DeleteBucketPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteObject`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie ein Objekt.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
```

```
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

Setzen Sie ein Objekt auf eine vorherige Version zurück, indem Sie spätere Versionen des Objekts löschen.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )
```

```
    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

Reaktivieren Sie ein gelöschtes Objekt, indem Sie die aktive Löschmarke des Objekts entfernen.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.
```

```
    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Erstellen Sie einen Lambda-Handler, der eine Löschmarke aus einem S3-Objekt entfernt. Dieser Handler kann verwendet werden, um irrelevante Löschmarkierungen in einem versionierten Bucket effizient zu bereinigen.

```
import logging
from urllib import parse
import boto3
```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
```

```
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
```

```
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- API-Details finden Sie DeleteObject in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteObjects`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie eine Reihe von Objekten mithilfe einer Liste von Objektschlüsseln.

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
```

```
            logger.exception("Couldn't delete any objects from bucket %s.",
 bucket.name)
            raise
        else:
            return response
```

Löschen Sie alle Objekte im Bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                              that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

Löschen Sie ein versioniertes Objekt dauerhaft, indem Sie alle seine Versionen löschen.

```
def permanently_delete_object(bucket, object_key):
    """
```

```
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- API-Details finden Sie DeleteObjects in der AWS API-Referenz zum SDK für Python (Boto3).

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung. `GetBucketAcl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
```

```
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name, acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- API-Details finden Sie GetBucketAcl in der AWS API-Referenz zum SDK für Python (Boto3).

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung. `GetBucketCors`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
```

```
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.", self.bucket.name))
            raise
        else:
            return cors
```

- API-Details finden Sie GetBucketCors in der AWS API-Referenz zum SDK für Python (Boto3).

`GetBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `GetBucketLifecycleConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
            return config.rules
```

- API-Details finden Sie GetBucketLifecycleConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `GetBucketPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- API-Details finden Sie GetBucketPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung. `GetObject`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
```

```
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        else:
            return body
```

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
 'IfModifiedSince', 'IfUnmodifiedSince'.
```

```
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of the
object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- API-Details finden Sie GetObject in der AWS API-Referenz zum SDK für Python (Boto3).

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung. `GetObjectAcl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- API-Details finden Sie GetObjectAcl in der AWS API-Referenz zum SDK für Python (Boto3).

GetObjectLegalHold

Das folgende Codebeispiel zeigt die Verwendung. `GetObjectLegalHold`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
 bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- API-Details finden Sie GetObjectLegalHold in der AWS API-Referenz zum SDK für Python (Boto3).

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `GetObjectLockConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Object-Lock-Konfiguration ab.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
 "Enabled"
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

`HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung. `HeadBucket`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
```

```
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- API-Details finden Sie HeadBucket in der AWS API-Referenz zum SDK für Python (Boto3).

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung. `ListBuckets`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.
```

```
        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
 Boto3
                            that contains collections and factory methods to create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- API-Details finden Sie ListBuckets in der AWS API-Referenz zum SDK für Python (Boto3).

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
```

```
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix are
listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.", bucket.name)
            raise
        else:
            return objects
```

- API-Details finden Sie unter ListObjects V2 in der AWS API-Referenz für das SDK für Python (Boto3).

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung. `PutBucketAcl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
 recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
```

```
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.", self.bucket.name)
            raise
```

- API-Details finden Sie PutBucketAcl in der AWS API-Referenz zum SDK für Python (Boto3).

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung. `PutBucketCors`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
```

```
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
self.bucket.name)
            raise
```

- API-Details finden Sie PutBucketCors in der AWS API-Referenz zum SDK für Python (Boto3).

`PutBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `PutBucketLifecycleConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
```

```
        Apply a lifecycle configuration to the bucket. The lifecycle configuration
can
        be used to archive or delete the objects in the bucket according to
specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- API-Details finden Sie PutBucketLifecycleConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung. `PutBucketPolicy`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BucketWrapper:
```

```
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy, self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
self.bucket.name)
            raise
```

- API-Details finden Sie PutBucketPolicy in der AWS API-Referenz zum SDK für Python (Boto3).

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung. `PutObject`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                            that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string. When
 this
                     argument is a string, it is interpreted as a file name, which
 is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
 data)
                raise

        try:
            self.object.put(Body=put_data)
```

```
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

Laden Sie ein Objekt mithilfe einer bedingten Anforderung hoch.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)




    def put_object_conditional(self, object_key: str, source_bucket: str, data:
 bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
 overwrite
        using an IfNoneMatch condition for the object key.
```

```
        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- API-Details finden Sie PutObject in der AWS API-Referenz zum SDK für Python (Boto3).

`PutObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung. `PutObjectAcl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
```

```
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                            that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- API-Details finden Sie PutObjectAcl in der AWS API-Referenz zum SDK für Python (Boto3).

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung. `PutObjectLegalHold`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key, bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key, bucket,
 e
        )
```

- API-Details finden Sie PutObjectLegalHold in der AWS API-Referenz zum SDK für Python (Boto3).

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung. `PutObjectLockConfiguration`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

PUT-Object Lock-Konfiguration.

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule": {}},
        )
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS API-Referenz zum SDK für Python (Boto3).

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung. `PutObjectRetention`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Legen Sie eine Objektaufbewahrung fest.

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
```

```
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
 far_future_date},
                BypassGovernanceRetention=True,
            )
```

- API-Details finden Sie PutObjectRetention in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine S3-Aktion ausführen kann. Verwenden Sie das Anforderungspaket, um eine Anforderung mit der URL zu stellen.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters, expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.
```

```
    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )
```

```
    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be the
 "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

Generieren Sie eine vorsignierte POST-Anforderung zum Hochladen einer Datei.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) zusammen mit Amazon Textract verwendet wird, um Text-, Formular- und Tabellenelemente in einem Dokumentbild zu erkennen. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Entitäten in Text erkennen, der aus einem Bild extrahiert wurde

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter. TextractAndComprehendNotebook.ipynb

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie das verwenden AWS SDK für Python (Boto3) , um eine Webanwendung zu erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingebaute Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.

- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten unter ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Senden von bedingten Anforderungen

Das folgende Codebeispiel zeigt, wie Sie Vorbedingungen zu Amazon-S3-Anforderungen hinzufügen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die bedingten Amazon-S3-Anforderungen zeigt.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
 requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
 setup.
sys.path.append("../../../..")
import demo_tools.question as q  # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)
```

```
class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
 actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key: str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
 {dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
            response = self.s3_client.put_object(
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
```

```
            return object_etag

        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}: {e}"
            )


    def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
        """
        Cleans up the scenario by deleting the source and destination buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

    def cleanup_bucket(self, bucket_name: str):
        """
        Cleans up the bucket by deleting all objects and then the bucket itself.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            for obj in objs:
                key = obj["Key"]
                self.s3_client.delete_object(Bucket=bucket_name, Key=key)
            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Cleaned up bucket: {bucket_name}.")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
            else:
                logger.error(f"Error deleting bucket: {error_code}")
                raise


    def display_buckets(self, source_bucket: str, dest_bucket: str):
```

```
        """
        Display a list of the objects in the test buckets.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        """
        self.list_bucket_contents(source_bucket)
        self.list_bucket_contents(dest_bucket)

    def list_bucket_contents(self, bucket_name):
        """
        Display a list of the objects in the bucket.

        :param bucket_name: The name of the bucket.
        """
        try:
            # Get list of all objects in the bucket.
            print(f"\t Items in bucket {bucket_name}")
            list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
            objs = list_response.get("Contents", [])
            if not objs:
                print("\t\tNo objects found.")
            for obj in objs:
                key = obj["Key"]
                print(f"\t\t object: {key} ETag {obj['ETag']}")
            return objs
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "NoSuchBucket":
                logger.info(f"Bucket {bucket_name} does not exist.")
            else:
                logger.error(f"Error listing bucket and objects: {error_code}")
                raise


    def display_menu(
        self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
    ):
        """
        Displays the menu of conditional request options for the user.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The key of the test object in the source bucket.
```

```
        :param etag: The etag of the test object in the source bucket.
        """

        actions = [
            "Print list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit.",
        ]

        conditions = [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ]

        condition_types = [
            "IfMatch",
            "IfNoneMatch",
            "IfModifiedSince",
            "IfUnmodifiedSince",
        ]
        copy_condition_types = [
            "CopySourceIfMatch",
            "CopySourceIfNoneMatch",
            "CopySourceIfModifiedSince",
            "CopySourceIfUnmodifiedSince",
        ]

        yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

        choice = 0
        while choice != 4:
            print("-" * 88)
            print("Choose an action to explore some example conditional requests.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                print("Listing the objects and buckets.")
                self.display_buckets(source_bucket, dest_bucket)
            elif choice == 1:
```

```
                print("Perform a conditional read.")
                condition_type = q.choose("Enter the condition type : ", conditions)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.get_object_conditional(
                        object_key, source_bucket, condition_types[condition_type],
etag
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.get_object_conditional(
                        object_key,
                        source_bucket,
                        condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 2:
                print("Perform a conditional copy.")
                condition_type = q.choose("Enter the condition type : ", conditions)
                dest_key = q.ask("Enter an object key: ", q.non_empty)
                if condition_type == 0 or condition_type == 1:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        source_bucket,
                        dest_bucket,
                        copy_condition_types[condition_type],
                        etag,
                    )
                elif condition_type == 2 or condition_type == 3:
                    self.conditional_requests.copy_object_conditional(
                        object_key,
                        dest_key,
                        copy_condition_types[condition_type],
                        yesterday_date,
                    )
            elif choice == 3:
                print(
                    "Perform a conditional write using IfNoneMatch condition on the
object key."
                )
                print("If the key is a duplicate, the write will fail.")
                object_key = q.ask("Enter an object key: ", q.non_empty)
                self.conditional_requests.put_object_conditional(
                    object_key, source_bucket, b"Conditional write example data."
                )
```

```
            elif choice == 4:
                print("Proceeding to cleanup.")


    def run_scenario(self):
        """
        Runs the interactive scenario.
        """
        print("-" * 88)
        print("Welcome to the Amazon S3 conditional requests example.")
        print("-" * 88)

        print(
            f"""\
        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.

        This example will allow you to perform conditional reads
        and writes that will succeed or fail based on your selected options.

        Sample buckets and a sample object will be created as part of the example.
        """
        )

        bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
        source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
        dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
        object_key = "test-upload-file.txt"

        try:
            etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
            self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
        finally:
            self.cleanup_scenario(source_bucket_name, dest_bucket_name)

        print("-" * 88)
        print("Thanks for watching.")
        print("-" * 88)
```

```
if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```

Eine Wrapper-Klasse, die die Operationen der bedingten Anfrage enthält.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)


class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)



    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
```

```
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of the
object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise



    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
```

```
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional write failed: Precondition failed")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise


    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
```

```
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304":  # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CopyObject
  - GetObject
  - PutObject

Versionierte Objekte in Batches mit einer Lambda-Funktion verwalten

Das folgende Codebeispiel zeigt, wie versionierte S3-Objekte in Batches mit einer Lambda-Funktion verwaltet werden.

SDK für Python (Boto3)

Zeigt, wie versionierte Objekte von Amazon Simple Storage Service (Amazon S3) stapelweise bearbeitet werden, indem Jobs erstellt werden, die AWS Lambda Funktionen aufrufen, um die Verarbeitung durchzuführen. In diesem Beispiel wird ein versionsfähiger Bucket erstellt, der die Strophen des Gedichts You Are Old, Father William von Lewis Carroll hochlädt und Amazon-S3-Batch-Aufträge verwendet, um das Gedicht auf verschiedene Arten zu verdrehen.

Lernen Sie Folgendes:

- Erstellen Sie Lambda-Funktionen, die mit versionierten Objekten arbeiten.

- Erstellen Sie ein Manifest von zu aktualisierenden Objekten.
- Erstellen Sie Batch-Aufträge, die Lambda-Funktionen zum Aktualisieren von Objekten aufrufen.
- Löschen Sie Lambda-Funktionen.
- Leeren und löschen Sie einen versionierten Bucket.

Dieses Beispiel lässt sich am besten unter ansehen. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon S3

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die Dateien mit mehreren der verfügbaren Transfer-Manager-Einstellungen übertragen. Verwenden Sie eine Callback-Klasse, um während der Dateiübertragung Callback-Fortschritte zu schreiben.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")
```

```
class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
```

```
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.
```

```
    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
```

```
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Demonstrieren Sie die Funktionen des Transfer-Managers und melden Sie Ergebnisse.

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
```

```
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
```

```
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
```

```
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
```

```
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:
                    self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.

        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )
```

```
        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
```

```
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )
```

```
    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )

    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
```

```
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Mit versionierten Objekten arbeiten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen versionierten S3 Bucket.
- Rufen Sie alle Versionen eines Objekts ab.
- Setzen Sie eine Richtlinie auf eine frühere Version zurück.
- Löschen Sie ein versioniertes Objekt und stellen Sie es wieder her.
- Löschen Sie alle Versionen eines Objekts dauerhaft.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Aktionen umschließen.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise
```

```
    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.
```

```
    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Laden Sie die Strophe eines Gedichts in ein versioniertes Objekt hoch und führen Sie eine Reihe von Aktionen dafür aus.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
```

```
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
```

```
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateBucket
  - DeleteObject
  - ListObjectVersions
  - PutBucketLifecycleConfiguration

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
```

```
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your
bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Beispiele für Amazon S3 Control mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Control Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon S3 Control

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon S3 Control.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise
```

- API-Details finden Sie ListJobs in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel zeigt, wie Sie die Kernoperationen von Amazon S3 Control erlernen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lernen Sie das S3 Batch Basics-Szenario kennen.

```
class S3BatchWrapper:
```

```
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                          access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
provides
                                 low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                           access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
            str: AWS account ID
        """
        return self.sts_client.get_caller_identity()["Account"]

    def create_bucket(self, bucket_name: str) -> None:
        """
        Create an S3 bucket.

        Args:
            bucket_name (str): Name of the bucket to create

        Raises:
            ClientError: If bucket creation fails
        """
        try:
            if self.region_name and self.region_name != 'us-east-1':
                self.s3_client.create_bucket(
                    Bucket=bucket_name,
```

```
                    CreateBucketConfiguration={
                        'LocationConstraint': self.region_name
                    }
                )
            else:
                self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
        except ClientError as e:
            print(f"Error creating bucket: {e}")
            raise

    def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
        """
        Upload files to S3 bucket including manifest file.

        Args:
            bucket_name (str): Target bucket name
            file_names (list): List of file names to upload

        Returns:
            str: ETag of the manifest file

        Raises:
            ClientError: If file upload fails
        """
        try:
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    content = f"Content for {file_name}"
                    self.s3_client.put_object(
                        Bucket=bucket_name,
                        Key=file_name,
                        Body=content.encode('utf-8')
                    )
                    print(f"Uploaded {file_name} to {bucket_name}")

            manifest_content = ""
            for file_name in file_names:
                if file_name != "job-manifest.csv":
                    manifest_content += f"{bucket_name},{file_name}\n"

            manifest_response = self.s3_client.put_object(
                Bucket=bucket_name,
```

```
                Key="job-manifest.csv",
                Body=manifest_content.encode('utf-8')
            )
            print(f"Uploaded manifest file to {bucket_name}")
            print(f"Manifest content:\n{manifest_content}")
            return manifest_response['ETag'].strip('"')

        except ClientError as e:
            print(f"Error uploading files: {e}")
            raise

    def create_s3_batch_job(self, account_id: str, role_arn: str, manifest_location:
str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
```

```
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise

    def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
        """
        Check for any failure reasons of a batch job.

        Args:
            job_id (str): ID of the batch job
```

```
            account_id (str): AWS account ID

        Returns:
            list: List of failure reasons

        Raises:
            ClientError: If checking job failure reasons fails
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            if 'FailureReasons' in response['Job']:
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
        except ClientError as e:
            print(f"Error checking job failure reasons: {e}")
            raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status: str =
'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
            desired_status (str): Target status to wait for

        Returns:
            bool: True if desired status is reached, False otherwise

        Raises:
            ClientError: If checking job status fails
        """
        print(f"Waiting for job to become {desired_status}...")
        max_attempts = 60
        attempt = 0
        while attempt < max_attempts:
            try:
                response = self.s3control_client.describe_job(
                    AccountId=account_id,
```

```
                    JobId=job_id
                )
                current_status = response['Job']['Status']
                print(f"Current job status: {current_status}")
                if current_status == desired_status:
                    return True
                if current_status == 'Suspended':
                    print("Job is in Suspended state, can proceed with activation")
                    return True
                if current_status in ['Active', 'Failed', 'Cancelled', 'Complete']:
                    print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
                    if 'FailureReasons' in response['Job']:
                        print("Failure reasons:")
                        for reason in response['Job']['FailureReasons']:
                            print(f"- {reason}")
                    return False

                time.sleep(20)
                attempt += 1
            except ClientError as e:
                print(f"Error checking job status: {e}")
                raise
        print(f"Timeout waiting for job to become {desired_status}")
        return False

    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
```

```
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")

                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return

    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
```

```
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise

    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
```

```
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
                progress = job['ProgressSummary']
                print(f"Progress Summary: Total={progress.get('TotalNumberOfTasks',
0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            tags = response.get('Tags', [])
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
                    print(f"  {tag['Key']}: {tag['Value']}")
            else:
                print(f"No tags found for job ID: {job_id}")
        except ClientError as e:
            print(f"Error getting job tags: {e}")
            raise

    def put_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Add tags to a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
```

```
        try:
            self.s3control_client.put_job_tagging(
                AccountId=account_id,
                JobId=job_id,
                Tags=[
                    {'Key': 'Environment', 'Value': 'Development'},
                    {'Key': 'Team', 'Value': 'DataProcessing'}
                ]
            )
            print(f"Additional tags were added to job {job_id}")
        except ClientError as e:
            print(f"Error adding job tags: {e}")
            raise

    def list_jobs(self, account_id: str) -> None:
        """
        List all batch jobs for the account.

        Args:
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.list_jobs(
                AccountId=account_id,
                JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
 'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
            )
            jobs = response.get('Jobs', [])
            for job in jobs:
                print(f"The job id is {job['JobId']}")
                print(f"The job priority is {job['Priority']}")
        except ClientError as e:
            print(f"Error listing jobs: {e}")
            raise

    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
```

```
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise

    def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
        """
        Clean up all resources created during the scenario.

        Args:
            bucket_name (str): Name of the bucket to clean up
            file_names (list): List of files to delete

        Raises:
            ClientError: If cleanup fails
        """
        try:
            for file_name in file_names:
                self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
                print(f"Deleted {file_name}")

            response = self.s3_client.list_objects_v2(
                Bucket=bucket_name,
                Prefix='batch-op-reports/'
            )
            if 'Contents' in response:
                for obj in response['Contents']:
                    self.s3_client.delete_object(
                        Bucket=bucket_name,
                        Key=obj['Key']
                    )
                    print(f"Deleted {obj['Key']}")

            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Deleted bucket {bucket_name}")
        except ClientError as e:
            print(f"Error in cleanup: {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def create_s3_batch_job(self, account_id: str, role_arn: str, manifest_location:
str,
                            report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:
```

```
            str: Job ID

        Raises:
            ClientError: If job creation fails
        """
        try:
            bucket_name = manifest_location.split(':::')[1].split('/')[0]
            manifest_key = 'job-manifest.csv'
            manifest_obj = self.s3_client.head_object(
                Bucket=bucket_name,
                Key=manifest_key
            )
            etag = manifest_obj['ETag'].strip('"')

            response = self.s3control_client.create_job(
                AccountId=account_id,
                Operation={
                    'S3PutObjectTagging': {
                        'TagSet': [
                            {
                                'Key': 'BatchTag',
                                'Value': 'BatchValue'
                            },
                        ]
                    }
                },
                Report={
                    'Bucket': report_bucket_name,
                    'Format': 'Report_CSV_20180820',
                    'Enabled': True,
                    'Prefix': 'batch-op-reports',
                    'ReportScope': 'AllTasks'
                },
                Manifest={
                    'Spec': {
                        'Format': 'S3BatchOperations_CSV_20180820',
                        'Fields': ['Bucket', 'Key']
                    },
                    'Location': {
                        'ObjectArn': manifest_location,
                        'ETag': etag
                    }
                },
                Priority=10,
```

```
                RoleArn=role_arn,
                Description='Batch job for tagging objects',
                ConfirmationRequired=True
            )
            job_id = response['JobId']
            print(f"The Job id is {job_id}")
            return job_id
        except ClientError as e:
            print(f"Error creating batch job: {e}")
            if 'Message' in str(e):
                print(f"Detailed error message: {e.response['Message']}")
            raise
```

- API-Details finden Sie CreateJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteJobTagging`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteJobTagging`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def delete_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Delete all tags from a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            self.s3control_client.delete_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
```

```
            print(f"You have successfully deleted {job_id} tagging.")
        except ClientError as e:
            print(f"Error deleting job tags: {e}")
            raise
```

- API-Details finden Sie DeleteJobTagging in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeJob`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def describe_job_details(self, job_id: str, account_id: str) -> None:
        """
        Describe detailed information about a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            job = response['Job']
            print(f"Job ID: {job['JobId']}")
            print(f"Description: {job.get('Description', 'N/A')}")
            print(f"Status: {job['Status']}")
            print(f"Role ARN: {job['RoleArn']}")
            print(f"Priority: {job['Priority']}")
            if 'ProgressSummary' in job:
```

```
                progress = job['ProgressSummary']
                print(f"Progress Summary: Total={progress.get('TotalNumberOfTasks', 0)}, "
                      f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                      f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise
```

- API-Details finden Sie DescribeJob in der AWS API-Referenz zum SDK für Python (Boto3).

`GetJobTagging`

Das folgende Codebeispiel zeigt die Verwendung. `GetJobTagging`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def get_job_tags(self, job_id: str, account_id: str) -> None:
        """
        Get tags associated with a batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.get_job_tagging(
                AccountId=account_id,
                JobId=job_id
            )
            tags = response.get('Tags', [])
            if tags:
                print(f"Tags for job {job_id}:")
                for tag in tags:
```

```
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise
```

- API-Details finden Sie GetJobTagging in der AWS API-Referenz zum SDK für Python (Boto3).

`PutJobTagging`

Das folgende Codebeispiel zeigt die Verwendung. `PutJobTagging`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
```

```
            print(f"Error adding job tags: {e}")
            raise
```

- API-Details finden Sie PutJobTagging in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateJobPriority`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateJobPriority`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def update_job_priority(self, job_id: str, account_id: str) -> None:
        """
        Update the priority of a batch job and start it.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended']:
                self.s3control_client.update_job_priority(
                    AccountId=account_id,
                    JobId=job_id,
                    Priority=60
                )
                print("The job priority was updated")
```

```
                try:
                    self.s3control_client.update_job_status(
                        AccountId=account_id,
                        JobId=job_id,
                        RequestedJobStatus='Ready'
                    )
                    print("Job activated successfully")
                except ClientError as activation_error:
                    print(f"Note: Could not activate job automatically:
{activation_error}")
                    print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return
```

- API-Details finden Sie UpdateJobPriority in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateJobStatus`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateJobStatus`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def cancel_job(self, job_id: str, account_id: str) -> None:
        """
        Cancel an S3 batch job.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
        """
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")

            if current_status in ['Ready', 'Suspended', 'Active']:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Cancelled'
                )
                print(f"Job {job_id} was successfully canceled.")
            elif current_status in ['Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - cannot be cancelled")
                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
            else:
                print(f"Job is in '{current_status}' state - cancel not allowed")
        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise
```

- API-Details finden Sie UpdateJobStatus in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für S3 Directory Buckets unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von S3 Directory Buckets Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.
- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.

- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
 Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
```

```
        print("")
        print_dashes()
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto 3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
 working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results in
 this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute resources
 in the same AZ as your Directory
bucket.
    """
        )
        press_enter_to_continue()

        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
 buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
 regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
 Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo (y/
n)? ",
            q.is_yesno,
        ):
```

```
            self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
 without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to allow
 S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in an
 EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
 same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
 create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in production!")
        print(
            """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
 permissions to do S3 Express One Zone actions.
            """
        )
        press_enter_to_continue()
        stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
        template_as_string = S3ExpressScenario.get_template_as_string()
        self.stack = self.deploy_cloudformation_stack(stack_name,
 template_as_string)
        regular_user_name = None
        express_user_name = None
        outputs = self.stack.outputs
        for output in outputs:
```

```
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
 'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name

    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str
    ) -> None:
        """
        Set up two S3 clients, one regular and one express, and two buckets, one
 regular and one express.
        :param express_user_name: The name of the user with S3 Express permissions.
        :param regular_user_name: The name of the user with regular S3 permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
 permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
 This client is created with the
credentials associated with the user account with the S3 Express policy attached, so
 it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3__client_with_access_key_credentials(
            regular_credentials
        )
        self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
        s3_express_client = self.create_s3__client_with_access_key_credentials(
            express_credentials
        )
        self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
```

```
        print(
            """
All the roles and policies were created and attached to the user. Then a new S3
 Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
 operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
 """
        )
        press_enter_to_continue()
        # 4. Create two buckets.
        print(
            """
3. Create two buckets.
Now we will create a Directory bucket which is the linchpin of the S3 Express One
 Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here. We'll
 also create a normal bucket, put
an object into the normal bucket, and copy it over to the Directory bucket.
"""
        )

        # Create a directory bucket. These are different from normal S3 buckets in
 subtle ways.
        bucket_prefix = q.ask(
            "Enter a bucket name prefix that will be used for both buckets: ",
            q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
        )

        # Some availability zones are not supported for Directory buckets. We'll
 choose one that is supported.
        print(
            "Now, let's choose an availability zone for the Directory bucket. We'll
 choose one that is supported."
        )
        while True:
            availability_zone = self.select_availability_zone_id(self.region)
            # Construct the parts of a directory bucket name that is made unique
 with a UUID string.
            directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
            max_uuid_length = 63 - len(bucket_prefix) - len(directory_bucket_suffix)
 - 1
            bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
            directory_bucket_name = (
```

```
                f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
            )
            regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
            configuration = {
                "Bucket": {
                    "Type": "Directory",
                    "DataRedundancy": "SingleAvailabilityZone",
                },
                "Location": {
                    "Name": availability_zone["ZoneId"],
                    "Type": "AvailabilityZone",
                },
            }
            press_enter_to_continue()
            print(
                "Now, let's create the actual Directory bucket, as well as a regular
bucket."
            )
            press_enter_to_continue()
            try:
                self.s3_express_wrapper.create_bucket(
                    directory_bucket_name, configuration
                )
                break
            except ClientError as client_error:
                if client_error.response["Error"]["Code"] == "InvalidBucketName":
                    print(
                        f"Bucket '{directory_bucket_name}' is invalid. This may be
because of selected availability zone."
                    )
                    if q.ask(
                        "Would you like to select a different availability zone? ",
                        q.is_yesno,
                    ):
                        continue
                    else:
                        raise
                else:
                    raise
        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
```

```
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
 bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
 operations are not restricted for
Directory buckets.
        """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
 Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the user
 credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls, this
 is low, but for many concurrent calls
this adds up to a lot of time saved.
"""
        )
        press_enter_to_continue()
        return bucket_object
```

```
    def demonstrate_performance(self, bucket_object: str) -> None:
        """
        Demonstrate performance differences between regular and Directory buckets.
        :param bucket_object: The name of the object to download from each bucket.
        """
        print("")
        print("6. Demonstrate performance difference.")
        print(
            """
Now, let's do a performance test. We'll download the same object from each bucket
 'downloads' times
and compare the total time needed. Note: the performance difference will be much
 more pronounced if this
example is run in an EC2 instance in the same Availability Zone as the bucket.
"""
        )
        downloads = 1000
        print(
            f"The number of downloads of the same object for this example is set at
 {downloads}."
        )
        if q.ask("Would you like to download a different number? (y/n) ",
 q.is_yesno):
            max_downloads = 1000000
            downloads = q.ask(
                f"Enter a number between 1 and {max_downloads} for the number of
 downloads: ",
                q.is_int,
                q.in_range(1, max_downloads),
            )
        # Download the object 'downloads' times from each bucket and time it to
 demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

            self.s3_express_wrapper.get_object(
                self.directory_bucket_name, bucket_object
            )

        directory_time_difference = time.time_ns() - directory_time_start
```

```
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
 bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
 while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{(difference) / 1000000000} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are not
 running on the cloud within a vpc."
            )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
 buckets.
        This is done by creating a few objects in each bucket and listing them to
 show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
 regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
 Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and therefore
 retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
 efficient it becomes. This structural
```

```
difference also causes ListObjects to behave differently, which can cause unexpected
 results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
        """
        )
        press_enter_to_continue()
        # Populate a few more files in each bucket so that we can use ListObjects
 and show the difference.
        other_object = f"other/{bucket_object}"
        alt_object = f"alt/{bucket_object}"
        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, other_object,
 "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name, other_object,
 "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name, alt_object,
 "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"   {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
 directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
 while the directory bucket actually
```

```
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
        """
        if self.directory_bucket_name is not None:
            self.s3_express_wrapper.delete_bucket_and_objects(
                self.directory_bucket_name
            )
            print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
            self.directory_bucket_name = None

        if self.regular_bucket_name is not None:

 self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
            print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
            self.regular_bucket_name = None

        if self.stack is not None:
            self.destroy_cloudformation_stack(self.stack)
            self.stack = None

        self.tear_done_vpc()

    def create_access_key(self, user_name: str) -> dict[str, any]:
        """
        Creates an access key for the user.
        :param user_name: The name of the user.
        :return: The access key for the user.
        """
        try:
            access_key = self.iam_client.create_access_key(UserName=user_name)
            return access_key["AccessKey"]
        except ClientError as client_error:
            logging.error(
                "Couldn't create the access key. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

```
    def create_s3__client_with_access_key_credentials(
        self, access_key: dict[str, any]
    ) -> client:
        """
        Creates an S3 client with access key credentials.
        :param access_key: The access key for the user.
        :return: The S3 Express One Zone client.
        """
        try:
            s3_express_client = boto3.client(
                "s3",
                aws_access_key_id=access_key["AccessKeyId"],
                aws_secret_access_key=access_key["SecretAccessKey"],
                region_name=self.region,
            )
            return s3_express_client
        except ClientError as client_error:
            logging.error(
                "Couldn't create the S3 Express One Zone client. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def select_availability_zone_id(self, region: str) -> dict[str, any]:
        """
        Selects an availability zone.
        :param region: The region to select the availability zone from.
        :return: The availability zone dictionary.
        """
        try:
            response = self.ec2_client.describe_availability_zones(
                Filters=[{"Name": "region-name", "Values": [region]}]
            )
            availability_zones = response["AvailabilityZones"]
            zone_names = [zone["ZoneName"] for zone in availability_zones]
            index = q.choose("Select an availability zone: ", zone_names)
            return availability_zones[index]
        except ClientError as client_error:
            logging.error(
                "Couldn't describe availability zones. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
```

```
    def deploy_cloudformation_stack(
        self, stack_name: str, cfn_template: str
    ) -> ServiceResource:
        """
        Deploys prerequisite resources used by the scenario. The resources are
        defined in the associated `cfn_template.yaml` AWS CloudFormation script and
are deployed
        as a CloudFormation stack, so they can be easily managed and destroyed.

        :param stack_name: The name of the CloudFormation stack.
        :param cfn_template: The CloudFormation template as a string.
        :return: The CloudFormation stack resource.
        """
        print(f"Deploying CloudFormation stack: {stack_name}.")
        stack = self.cloud_formation_resource.create_stack(
            StackName=stack_name,
            TemplateBody=cfn_template,
            Capabilities=["CAPABILITY_NAMED_IAM"],
        )
        print(f"CloudFormation stack creation started: {stack_name}")
        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

        return stack

    def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
        """
        Destroys the resources managed by the CloudFormation stack, and the
CloudFormation
        stack itself.

        :param stack: The CloudFormation stack that manages the example resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
```

```
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise
        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
            route_table_id = response["RouteTables"][0]["RouteTableId"]
```

```
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
            print(f"Created vpc endpoint {self.vpc_endpoint_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc endpoint. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def tear_done_vpc(self) -> None:
        if self.vpc_endpoint_id is not None:
            try:
                self.ec2_client.delete_vpc_endpoints(
                    VpcEndpointIds=[self.vpc_endpoint_id]
                )
                print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
                self.vpc_endpoint_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc endpoint %s. Here's why: %s",
                    self.vpc_endpoint_id,
                    client_error.response["Error"]["Message"],
                )
        if self.vpc_id is not None:
            try:
                self.ec2_client.delete_vpc(VpcId=self.vpc_id)
                print(f"Deleted vpc {self.vpc_id}")
                self.vpc_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc %s. Here's why: %s",
                    self.vpc_id,
                    client_error.response["Error"]["Message"],
                )
```

Eine Wrapper-Klasse für SDK-Funktionen von Amazon S3 Express.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:
        """
        Creates a bucket.
        :param bucket_name: The name of the bucket.
        :param bucket_configuration: The optional configuration for the bucket.
        """
        try:
            params = {"Bucket": bucket_name}
            if bucket_configuration:
                params["CreateBucketConfiguration"] = bucket_configuration
```

```
            self.s3_client.create_bucket(**params)
        except ClientError as client_error:
            # Do not log InvalidBucketName error because it is logged elsewhere.
            if client_error.response["Error"]["Code"] != "InvalidBucketName":
                logging.error(
                    "Couldn't create the bucket %s. Here's why: %s",
                    bucket_name,
                    client_error.response["Error"]["Message"],
                )
            raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
         :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a bucket
to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
                                error["Key"],
                                error["Message"],
                            )

            self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
```

```
    def put_object(self, bucket_name: str, object_key: str, content: str) -> None:
        """
        Puts an object into a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        :param content: The content of the object.
        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

    def list_objects(self, bucket: str) -> list[str]:
        """
        Lists objects in a bucket.
        :param bucket: The name of the bucket.
        :return: The list of objects in the bucket.
        """
        try:
            response = self.s3_client.list_objects_v2(Bucket=bucket)
            return response.get("Contents", [])
        except ClientError as client_error:
            logging.error(
                "Couldn't list objects in bucket %s. Here's why: %s",
                bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def copy_object(
        self,
        source_bucket: str,
        source_key: str,
        destination_bucket: str,
        destination_key: str,
    ) -> None:
```

```
        """
        Copies an object from one bucket to another.
        :param source_bucket: The source bucket.
        :param source_key: The source key.
        :param destination_bucket: The destination bucket.
        :param destination_key: The destination key.
        :return: None
        """
        try:
            self.s3_client.copy_object(
                CopySource={"Bucket": source_bucket, "Key": source_key},
                Bucket=destination_bucket,
                Key=destination_key,
            )
        except ClientError as client_error:
            logging.error(
                "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
                source_key,
                source_bucket,
                destination_bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise


    def get_object(self, bucket_name: str, object_key: str) -> None:
        """
        Gets an object from a bucket.
```

```
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        """
        try:
            self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't get the object %s from bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

Aktionen

`CreateSession`

Das folgende Codebeispiel zeigt die Verwendung`CreateSession`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
 interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)


    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
```

```
            "Couldn't create the express session for bucket %s. Here's why: %s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie CreateSession in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Secrets Manager unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Secrets Manager Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`BatchGetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetSecretValue`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class BatchGetSecretsWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def batch_get_secrets(self, filter_name):
        """
        Retrieve multiple secrets from AWS Secrets Manager using the
 batch_get_secret_value API.
        This function assumes the stack mentioned in the source code README has been
 successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param filter_name: The full or partial name of secrets to be fetched.
        :type filter_name: str
        """
        try:
            secrets = []
            response = self.client.batch_get_secret_value(
                Filters=[{"Key": "name", "Values": [f"{filter_name}"]}]
            )
            for secret in response["SecretValues"]:
                secrets.append(json.loads(secret["SecretString"]))
            if secrets:
                logger.info("Secrets retrieved successfully.")
            else:
                logger.info("Zero secrets returned without error.")
            return secrets
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"One or more requested secrets were not found with filter:
 {filter_name}"
            logger.info(msg)
            return msg
```

```
        except Exception as e:
            logger.error(f"An unknown error occurred:\n{str(e)}.")
            raise
```

- API-Details finden Sie BatchGetSecretValue in der AWS API-Referenz zum SDK für Python (Boto3).

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung. `GetSecretValue`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class GetSecretWrapper:
    def __init__(self, secretsmanager_client):
        self.client = secretsmanager_client


    def get_secret(self, secret_name):
        """
        Retrieve individual secrets from AWS Secrets Manager using the
 get_secret_value API.
        This function assumes the stack mentioned in the source code README has been
 successfully deployed.
        This stack includes 7 secrets, all of which have names beginning with
 "mySecret".

        :param secret_name: The name of the secret fetched.
        :type secret_name: str
        """
        try:
            get_secret_value_response = self.client.get_secret_value(
```

```
                SecretId=secret_name
            )
            logging.info("Secret retrieved successfully.")
            return get_secret_value_response["SecretString"]
        except self.client.exceptions.ResourceNotFoundException:
            msg = f"The requested secret {secret_name} was not found."
            logger.info(msg)
            return msg
        except Exception as e:
            logger.error(f"An unknown error occurred: {str(e)}.")
            raise
```

- API-Details finden Sie GetSecretValue in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Leihbibliothek-REST-API erstellen

Im folgenden Codebeispiel wird veranschaulicht, wie man eine Leihbibliothek erstellt, in der Kunden Bücher mithilfe einer REST-API ausleihen und zurückgeben können, die von einer Amazon-Aurora-Datenbank unterstützt wird.

SDK für Python (Boto3)

Zeigt, wie Sie die AWS SDK für Python (Boto3) mit der Amazon Relational Database Service (Amazon RDS) -API und AWS Chalice verwenden, um eine REST-API zu erstellen, die von einer Amazon Aurora-Datenbank unterstützt wird. Der Webservice ist vollständig Serverless und stellt eine einfache Leihbibliothek dar, in der die Kunden Bücher ausleihen und zurückgeben können. Lernen Sie Folgendes:

- Erstellen und verwalten Sie einen Serverless-Aurora-Datenbank-Cluster.
- Wird AWS Secrets Manager zur Verwaltung von Datenbankanmeldeinformationen verwendet.
- Implementieren Sie einen Datenspeicher-Layer, der Amazon RDS verwendet, um Daten in die und aus der Datenbank zu verschieben.
- Verwenden Sie AWS Chalice, um eine serverlose REST-API für Amazon API Gateway bereitzustellen und. AWS Lambda
- Verwenden Sie das Anforderungspaket, um Anfragen an den Webservice zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Beispiele für Amazon SES unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptFilter`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
        """
        Creates a filter that allows or blocks incoming mail from an IP address or
        range.

        :param filter_name: The name to give the filter.
        :param ip_address_or_range: The IP address or range to block or allow.
        :param allow: When True, incoming mail is allowed from the specified IP
                      address or range; otherwise, it is blocked.
        """
        try:
            policy = "Allow" if allow else "Block"
            self.ses_client.create_receipt_filter(
                Filter={
                    "Name": filter_name,
                    "IpFilter": {"Cidr": ip_address_or_range, "Policy": policy},
                }
            )
            logger.info(
                "Created receipt filter %s to %s IP of %s.",
                filter_name,
                policy,
```

```
                ip_address_or_range,
            )
        except ClientError:
            logger.exception("Couldn't create receipt filter %s.", filter_name)
            raise
```

- API-Details finden Sie CreateReceiptFilter in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung. `CreateReceiptRule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen Amazon-S3-Bucket, in dem Amazon SES Kopien eingehender E-Mails ablegen und eine Regel erstellen kann, die eingehende E-Mails für eine bestimmte Empfängerliste in den Bucket kopiert.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_bucket_for_copy(self, bucket_name):
        """
```

```
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.

        :param bucket_name: The name of the bucket to create.
        :return: The newly created bucket.
        """
        allow_ses_put_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowSESPut",
                    "Effect": "Allow",
                    "Principal": {"Service": "ses.amazonaws.com"},
                    "Action": "s3:PutObject",
                    "Resource": f"arn:aws:s3:::{bucket_name}/*",
                }
            ],
        }
        bucket = None
        try:
            bucket = self.s3_resource.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    "LocationConstraint":
self.s3_resource.meta.client.meta.region_name
                },
            )
            bucket.wait_until_exists()
            bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
            logger.info("Created bucket %s to receive copies of emails.",
bucket_name)
        except ClientError:
            logger.exception("Couldn't create bucket to receive copies of emails.")
            if bucket is not None:
                bucket.delete()
            raise
        else:
            return bucket


    def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix
    ):
```

```
        """
        Creates a rule so that all emails received by the specified recipients are
        copied to an Amazon S3 bucket.

        :param rule_set_name: The name of a previously created rule set to contain
                              this rule.
        :param rule_name: The name to give the rule.
        :param recipients: When an email is received by one of these recipients, it
                           is copied to the Amazon S3 bucket.
        :param bucket_name: The name of the bucket to receive email copies. This
                            bucket must allow Amazon SES to put objects into it.
        :param prefix: An object key prefix to give the emails copied to the bucket.
        """
        try:
            self.ses_client.create_receipt_rule(
                RuleSetName=rule_set_name,
                Rule={
                    "Name": rule_name,
                    "Enabled": True,
                    "Recipients": recipients,
                    "Actions": [
                        {
                            "S3Action": {
                                "BucketName": bucket_name,
                                "ObjectKeyPrefix": prefix,
                            }
                        }
                    ],
                },
            )
            logger.info(
                "Created rule %s to copy mail received by %s to bucket %s.",
                rule_name,
                recipients,
                bucket_name,
            )
        except ClientError:
            logger.exception("Couldn't create rule %s.", rule_name)
            raise
```

- API-Details finden Sie CreateReceiptRule in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung. `CreateReceiptRuleSet`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_rule_set(self, rule_set_name):
        """
        Creates an empty rule set. Rule sets contain individual rules and can be
        used to organize rules.

        :param rule_set_name: The name to give the rule set.
        """
        try:
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Created receipt rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't create receipt rule set %s.", rule_set_name)
            raise
```

- API-Details finden Sie CreateReceiptRuleSet in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
```

```
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise
```

- API-Details finden Sie CreateTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
```

```
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise
```

- API-Details finden Sie DeleteIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteReceiptFilter`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
```

```
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

- API-Details finden Sie DeleteReceiptFilter in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteReceiptRule`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
```

```
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.

        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name
            )
            logger.info("Removed rule %s from rule set %s.", rule_name,
rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
rule_set_name
            )
            raise
```

- API-Details finden Sie DeleteReceiptRule in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteReceiptRuleSet`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule_set(self, rule_set_name):
        """
        Deletes a rule set. When a rule set is deleted, all of the rules it contains
        are also deleted.

        :param rule_set_name: The name of the rule set to delete.
        """
        try:
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Deleted rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't delete rule set %s.", rule_set_name)
            raise
```

- API-Details finden Sie DeleteReceiptRuleSet in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

 self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
```

```
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise
```

- API-Details finden Sie DeleteTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeReceiptRuleSet`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def describe_receipt_rule_set(self, rule_set_name):
        """
        Gets data about a rule set.

        :param rule_set_name: The name of the rule set to retrieve.
        :return: Data about the rule set.
```

```
        """
        try:
            response = self.ses_client.describe_receipt_rule_set(
                RuleSetName=rule_set_name
            )
            logger.info("Got data for rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

- API-Details finden Sie DescribeReceiptRuleSet in der AWS API-Referenz zum SDK für Python (Boto3).

`GetIdentityVerificationAttributes`

Das folgende Codebeispiel zeigt die Verwendung. `GetIdentityVerificationAttributes`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def get_identity_status(self, identity):
        """
```

```
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- API-Details finden Sie GetIdentityVerificationAttributes in der AWS API-Referenz zum SDK für Python (Boto3).

`GetTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `GetTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
```

```
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

- API-Details finden Sie GetTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung. `ListIdentities`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

- API-Details finden Sie ListIdentities in der AWS API-Referenz zum SDK für Python (Boto3).

`ListReceiptFilters`

Das folgende Codebeispiel zeigt die Verwendung. `ListReceiptFilters`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.

        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response["Filters"]
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
```

```
            raise
        else:
            return filters
```

- API-Details finden Sie ListReceiptFilters in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTemplates`

Das folgende Codebeispiel zeigt die Verwendung. `ListTemplates`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
```

```
        logger.info("Extracted template tags: %s", self.template_tags)


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

- API-Details finden Sie ListTemplates in der AWS API-Referenz zum SDK für Python (Boto3).

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung. `SendEmail`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
```

```
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html, reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source, destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id
```

- API-Details finden Sie SendEmail in der AWS API-Referenz zum SDK für Python (Boto3).

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung. `SendTemplatedEmail`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
 passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
```

```
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
                              that are inserted in the template before it is sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- API-Details finden Sie SendTemplatedEmail in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
```

```
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

- API-Details finden Sie UpdateTemplate in der AWS API-Referenz für SDK für Python (Boto3).

`VerifyDomainIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `VerifyDomainIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
        """
```

```
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

- API-Details finden Sie VerifyDomainIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `VerifyEmailIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
```

```
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise
```

- API-Details finden Sie VerifyEmailIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Kopieren von E-Mail- und Domain-Identitäten von Region zu Region

Das folgende Codebeispiel zeigt, wie Amazon SES-E-Mail- und Domain-Identitäten von einer AWS Region in eine andere kopiert werden. Wenn Domänenidentitäten von Route 53 verwaltet werden, werden Überprüfungsdatensätze in die Domäne für die Zielregion kopiert.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identity_paginator = ses_client.get_paginator("list_identities")
        identity_iterator = identity_paginator.paginate(
            PaginationConfig={"PageSize": 20}
        )
        for identity_page in identity_iterator:
            for identity in identity_page["Identities"]:
                if "@" in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.",
            len(email_identities),
            len(domain_identities),
        )
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities
```

```
def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an email
    to be sent to each address. To complete verification, the recipient must follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails


def verify_domains(domain_list, ses_client):
    """
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response["VerificationToken"]
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token, domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.", domain)
    return domain_tokens


def get_hosted_zones(route53_client):
```

```
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zone_paginator = route53_client.get_paginator("list_hosted_zones")
        zone_iterator = zone_paginator.paginate(PaginationConfig={"PageSize": 20})
        zones = [
            zone for zone_page in zone_iterator for zone in zone_page["HostedZones"]
        ]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones


def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
             domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain until
 a
        # zone match is found.
        domain_split = domain.split(".")
        for index in range(0, len(domain_split) - 1):
            sub_domain = ".".join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
 '.'.
                zone_name = zone["Name"][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
```

```
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones


def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
    domain_token_record_set_name = f"_amazonses.{domain}"
    record_set_paginator = route53_client.get_paginator("list_resource_record_sets")
    record_set_iterator = record_set_paginator.paginate(
        HostedZoneId=zone["Id"], PaginationConfig={"PageSize": 20}
    )
    records = []
    for record_set_page in record_set_iterator:
        try:
            txt_record_set = next(
                record_set
                for record_set in record_set_page["ResourceRecordSets"]
                if record_set["Name"][:-1] == domain_token_record_set_name
                and record_set["Type"] == "TXT"
            )
            records = txt_record_set["ResourceRecords"]
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name,
                zone["Name"],
            )
            break
        except StopIteration:
            pass
    records.append({"Value": json.dumps(token)})
    changes = [
        {
            "Action": "UPSERT",
```

```
            "ResourceRecordSet": {
                "Name": domain_token_record_set_name,
                "Type": "TXT",
                "TTL": 1800,
                "ResourceRecords": records,
            },
        }
    ]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name,
            zone["Name"],
        )
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone %s.",
            err.response["Error"]["Code"],
            zone["Name"],
        )


def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """
    dkim_tokens = []
    try:
        dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)["DkimTokens"]
        logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
 domain)
    except ClientError:
        logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
    return dkim_tokens
```

```
def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
    """
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

    :param hosted_zone: The hosted zone where the records are added.
    :param domain: The domain to add.
    :param tokens: The DKIM tokens for the domain to add.
    :param route53_client: A Boto3 Route 53 client.
    """
    try:
        changes = [
            {
                "Action": "UPSERT",
                "ResourceRecordSet": {
                    "Name": f"{token}._domainkey.{domain}",
                    "Type": "CNAME",
                    "TTL": 1800,
                    "ResourceRecords": [{"Value": f"{token}.dkim.amazonses.com"}],
                },
            }
            for token in tokens
        ]
        route53_client.change_resource_record_sets(
            HostedZoneId=hosted_zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Added %s DKIM CNAME records to %s in zone %s.",
            len(tokens),
            domain,
            hosted_zone["Name"],
        )
    except ClientError:
        logger.warning(
            "Couldn't add DKIM CNAME records for %s to zone %s.",
            domain,
            hosted_zone["Name"],
        )


def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.
```

```
    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
 Delivery,
                   or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press "
            f"Enter to skip: "
        )
        if topic_arn != "":
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn
                )
                logger.info("Configured %s for %s notifications.", identity, topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity, topic
                )


def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print(
        f"Replicating Amazon SES identities and other configuration from "
        f"{source_client.meta.region_name} to
 {destination_client.meta.region_name}."
    )
    print("-" * 88)

    print(f"Retrieving identities from {source_client.meta.region_name}.")
    source_emails, source_domains = get_identities(source_client)
    print("Email addresses found:")
    print(*source_emails)
    print("Domains found:")
    print(*source_domains)

    print("Starting verification for email identities.")
    dest_emails = verify_emails(source_emails, destination_client)
    print("Getting domain tokens for domain identities.")
```

```
    dest_domain_tokens = verify_domains(source_domains, destination_client)

    # Get Route 53 hosted zones and match them with Amazon SES domains.
    answer = input(
        "Is the DNS configuration for your domains managed by Amazon Route 53 (y/n)?
"
    )
    use_route53 = answer.lower() == "y"
    hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
    if use_route53:
        print("Adding or updating Route 53 TXT records for your domains.")
        domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
hosted_zones)
        for domain in domain_zones:
            add_route53_verification_record(
                domain, dest_domain_tokens[domain], domain_zones[domain],
route53_client
            )
    else:
        print(
            "Use these verification tokens to create TXT records through your DNS "
            "provider:"
        )
        pprint(dest_domain_tokens)

    answer = input("Do you want to configure DKIM signing for your identities (y/n)?
")
    if answer.lower() == "y":
        # Build a set of unique domains from email and domain identities.
        domains = {email.split("@")[1] for email in dest_emails}
        domains.update(dest_domain_tokens)
        domain_zones = find_domain_zone_matches(domains, hosted_zones)
        for domain, zone in domain_zones.items():
            answer = input(
                f"Do you want to configure DKIM signing for {domain} (y/n)? "
            )
            if answer.lower() == "y":
                dkim_tokens = generate_dkim_tokens(domain, destination_client)
                if use_route53 and zone is not None:
                    add_dkim_domain_tokens(zone, domain, dkim_tokens,
route53_client)
                else:
                    print(
```

```
                        "Add the following DKIM tokens as CNAME records through your
 "
                        "DNS provider:"
                    )
                    print(*dkim_tokens, sep="\n")

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities (y/
n)? "
    )
    if answer.lower() == "y":
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/n)? "
            )
            if answer.lower() == "y":
                configure_sns_topics(
                    identity, ["Bounce", "Delivery", "Complaint"],
 destination_client
                )

    print(f"Replication complete for {destination_client.meta.region_name}.")
    print("-" * 88)


def main():
    boto3_session = boto3.Session()
    ses_regions = boto3_session.get_available_regions("ses")
    parser = argparse.ArgumentParser(
        description="Copies email address and domain identities from one AWS Region
 to "
        "another. Optionally adds records for domain verification and DKIM "
        "signing to domains that are managed by Amazon Route 53, "
        "and sets up Amazon SNS notifications for events of interest."
    )
    parser.add_argument(
        "source_region", choices=ses_regions, help="The region to copy from."
    )
    parser.add_argument(
        "destination_region", choices=ses_regions, help="The region to copy to."
    )
    args = parser.parse_args()
    source_client = boto3.client("ses", region_name=args.source_region)
    destination_client = boto3.client("ses", region_name=args.destination_region)
```

```
    route53_client = boto3.client("route53")
    replicate(source_client, destination_client, route53_client)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

Erstellen einer Webanwendung zur Verfolgung von DynamoDB-Daten

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Amazon-DynamoDB-Tabelle verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in Amazon DynamoDB verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer DynamoDB-Tabelle gespeichert sind.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel im AWS Code Examples Repository unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon SES

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für Python (Boto3)

Zeigt, wie Sie den verwenden AWS SDK für Python (Boto3) , um einen REST-Service zu erstellen, der Arbeitselemente in einer Amazon Aurora Serverless-Datenbank verfolgt und Berichte mithilfe von Amazon Simple Email Service (Amazon SES) per E-Mail versendet. In diesem Beispiel wird das Flask-Web-Framework für das HTTP-Routing verwendet und in eine React-Webseite integriert, um eine voll funktionsfähige Webanwendung zu präsentieren.

- Erstellen Sie einen Flask-REST-Service, der sich in integrieren lässt. AWS-Services
- Lesen, schreiben und aktualisieren Sie Arbeitsaufgaben, die in einer Aurora-Serverless-Datenbank gespeichert sind.
- Erstellen Sie ein AWS Secrets Manager Geheimnis, das Datenbankanmeldeinformationen enthält, und verwenden Sie es, um Aufrufe an die Datenbank zu authentifizieren.
- Verwenden Sie Amazon SES, um E-Mail-Berichte über Arbeitsaufgaben zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Erkennen von Objekten in Bildern

Die folgenden Codebeispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie das verwenden AWS SDK für Python (Boto3) , um eine Webanwendung zu erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingebaute Webseite JavaScript , die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit erstellt wurde Flask-RESTful.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS , um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.
- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten unter ansehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Generieren von Anmeldeinformationen für die Verbindung mit einem SMTP-Endpunkt

Das folgende Code-Beispiel zeigt, wie Sie Anmeldeinformationen erzeugen, um eine Verbindung mit einem Amazon-SES-SMTP-Endpunkt herzustellen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
    "us-gov-east-1",  # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04
```

```
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))


if __name__ == "__main__":
    main()
```

Verifizieren einer E-Mail-Identität und Senden von Nachrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Hinzufügen und verifizieren einer E-Mail-Adresse mit Amazon SES.

- Senden einer standardmäßigen E-Mail-Nachricht.
- Erstellen Sie eine Vorlage und senden Sie eine E-Mail-Nachricht mit Vorlage.
- Senden einer Nachricht mithilfe eines Amazon-SES-SMTP-Servers.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verifizieren einer E-Mail-Adresse mit Amazon SES und senden von Nachrichten.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    ses_client = boto3.client("ses")
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input("Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == "Success"
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): "
        )
        if answer.lower() == "y":
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete
 verification.")
            print("Waiting for verification...")
```

```
            try:
                ses_identity.wait_until_identity_exists(email)
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(
                    f"Verification timeout exceeded. You must complete the "
                    f"steps in the email sent to {email} to verify the address."
                )

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail demo!</
p>"

        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email,
            SesDestination([email]),
            "Amazon SES demo",
            test_message_text,
            test_message_html,
        )
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            "name": "doc-example-template",
            "subject": "Example of an email template.",
            "text": "This is what {{name}} will {{action}} if {{name}} can't display
 "
            "HTML.",
            "html": "<p><i>This</i> is what {{name}} will {{action}} if {{name}} "
            "<b>can</b> display HTML.</p>",
        }
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {"name": email.split("@")[0], "action": "read"}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(), template_data
            )
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
```

```
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f"email-smtp.{region}.amazonaws.com"
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == "y":
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print("-" * 88)
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-Identitätsvorgängen.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
```

```
        """
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise


    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
        identity has been successfully verified or until it exceeds its maximum
 time.
```

```
        :param identity: The identity to wait for.
        """
        try:
            waiter = self.ses_client.get_waiter("identity_exists")
            logger.info("Waiting until %s exists.", identity)
            waiter.wait(Identities=[identity])
        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.", identity)
            raise


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
```

```
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-Vorlagenvorgängen.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()
```

```
    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
```

```
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
```

```
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

Erstellen von Funktionen zum Umschließen von Amazon-SES-E-Mail-Vorgängen.

```
class SesDestination:
    """Contains data about an email destination."""

    def __init__(self, tos, ccs=None, bccs=None):
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
```

```
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {"ToAddresses": self.tos}
        if self.ccs is not None:
            svc_format["CcAddresses"] = self.ccs
        if self.bccs is not None:
            svc_format["BccAddresses"] = self.bccs
        return svc_format



class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html, reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
```

```
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source, destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
                              that are inserted in the template before it is sent.
        :return: The ID of the message, assigned by Amazon SES.
```

```
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTemplate
  - DeleteIdentity
  - DeleteTemplate
  - GetIdentityVerificationAttributes
  - GetTemplate
  - ListIdentities
  - ListTemplates
  - SendEmail

- SendTemplatedEmail
- UpdateTemplate
- VerifyDomainIdentity
- VerifyEmailIdentity

Beispiele für Amazon SES API v2 unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon SES-API v2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo Amazon SES API v2

Das folgende Codebeispiel zeigt, wie Sie mit der Amazon SES-API v2 beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def hello_sesv2(sesv2_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon SESv2 client and
    list the email identities in your account. This example uses the default
    settings specified in your shared credentials and config files.

    :param sesv2_client: A Boto3 SESv2 client object.
    """
    print("Hello, Amazon SESv2. Let's list up to 5 email identities:\n")

    try:
        response = sesv2_client.list_email_identities(PageSize=5)
        identities = response["EmailIdentities"]

        if not identities:
            print(
                "No email identities found. "
                "Use CreateEmailIdentity to add one."
            )
        else:
            for identity in identities:
                print(
                    f"  Identity: {identity['IdentityName']}"
                    f"  Type: {identity['IdentityType']}"
                    f"  Status: {identity['VerificationStatus']}"
                    f"  Sending: {'Enabled' if identity['SendingEnabled'] else
 'Disabled'}"
                )
            print(f"\nShowing {len(identities)} email identity(ies).")

    except ClientError as err:
        logger.error(
            "Couldn't list email identities. Here's why: %s: %s",
            err.response["Error"]["Code"],
```

```
            err.response["Error"]["Message"],
        )
        raise
```

- API-Details finden Sie ListEmailIdentities in der AWS API-Referenz zum SDK für Python (Boto3).

Aktionen

`CreateContact`

Das folgende Codebeispiel zeigt die Verwendung. `CreateContact`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()
```

```
class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
 Skipping...")
                else:
```

```
                    raise e
```

- API-Details finden Sie CreateContact in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateContactList`

Das folgende Codebeispiel zeigt die Verwendung. `CreateContactList`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
```

```
        self.sleep = sleep


        try:
            self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e
```

- API-Details finden Sie CreateContactList in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `CreateEmailIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
```

```
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e
```

- API-Details finden Sie CreateEmailIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `CreateEmailTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
```

```
                raise e
```

- API-Details finden Sie CreateEmailTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteContactList`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteContactList`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)
```

- API-Details finden Sie DeleteContactList in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteEmailIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:

 self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
 successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not exist.")
                else:
                    print(e)
```

- API-Details finden Sie DeleteEmailIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteEmailTemplate`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
```

```
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- API-Details finden Sie DeleteEmailTemplate in der AWS API-Referenz zum SDK für Python (Boto3).

`GetEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung. `GetEmailIdentity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie GetEmailIdentity in der AWS API-Referenz zum SDK für Python (Boto3).

`ListContacts`

Das folgende Codebeispiel zeigt die Verwendung. `ListContacts`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
```

```
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e
```

- API-Details finden Sie ListContacts in der AWS API-Referenz zum SDK für Python (Boto3).

`SendBulkEmail`

Das folgende Codebeispiel zeigt die Verwendung. `SendBulkEmail`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client

    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)
```

```
    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
```

```
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise
```

- API-Details finden Sie SendBulkEmail in der AWS API-Referenz zum SDK für Python (Boto3).

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung. `SendEmail`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht an alle Mitglieder der Kontaktliste.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
```

```
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()




class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
```

Sendet mithilfe einer Vorlage eine Nachricht an alle Mitglieder der Kontaktliste.

```
def main():
    """
```

```
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )
```

- API-Details finden Sie SendEmail in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Szenario „E-Mail-Anhänge“

Das folgende Codebeispiel zeigt, wie E-Mails mit Anhängen mithilfe der Amazon SES-API v2 gesendet werden.

- Erstellen Sie eine E-Mail-Vorlage für Massensendungen.
- Senden Sie eine einfache E-Mail mit einem Dateianhang.
- Senden Sie eine einfache E-Mail mit einem Inline-Bild.
- Senden Sie Massen-E-Mails mit Vorlagen und Anhängen.
- Bereinigen Sie die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem E-Mail-Anhänge demonstriert werden.

```
class SESv2EmailAttachmentsScenario:
    """
    Demonstrates sending emails with attachments using Amazon SESv2.

    This scenario demonstrates:
    1. Setting up an email identity and template.
    2. Sending a simple email with a file attachment.
    3. Sending a simple email with an inline image.
    4. Sending bulk templated emails with attachments.
    5. Cleaning up created resources.
    """

    TEMPLATE_NAME = "AttachmentDemoTemplate"

    def __init__(self, sesv2_wrapper: SESv2Wrapper) -> None:
        """
        :param sesv2_wrapper: An instance of the SESv2Wrapper class.
```

```
        """
        self.sesv2_wrapper = sesv2_wrapper
        self.sender_email = ""
        self.recipient_emails: list = []
        self.identity_was_created = False

    def run_scenario(self) -> None:
        """Runs the SESv2 email attachments scenario."""
        print("-" * 88)
        print("Welcome to the Amazon SESv2 Email Attachments Scenario.")
        print("-" * 88)
        print(
            "This scenario demonstrates how to send emails with attachments\n"
            "using SESv2 attachment support. SES handles MIME\n"
            "construction automatically, so you don't need to build raw\n"
            "MIME messages.\n"
        )

        try:
            self._setup()
            self._step1_send_email_with_attachment()
            self._step2_send_email_with_inline_image()
            self._step3_send_bulk_email_with_attachments()
        except Exception as e:
            logger.error("Scenario failed: %s", e)
            print(f"\nThe scenario encountered an error: {e}")
        finally:
            self._cleanup()

    # ---------- Setup ----------

    def _setup(self) -> None:
        """
        Prompts for configuration, verifies the sender identity, prepares a
        sample attachment, and creates an email template.
        """
        print("\n--- Setup ---\n")

        # Prompt for sender and recipient addresses.
        print(
            "Both sender and recipient addresses must be verified if your\n"
            "account is in the SES sandbox.\n"
        )
        self.sender_email = q.ask(
```

```
            "Enter a verified sender email address: "
        )
        recipient_input = q.ask(
            "Enter one or more recipient email addresses (comma-separated): "
        )
        self.recipient_emails = [
            addr.strip() for addr in recipient_input.split(",") if addr.strip()
        ]

        # Verify the sender identity.
        print(f"\nChecking identity for {self.sender_email}...")
        try:
            identity_info = self.sesv2_wrapper.get_email_identity(
                self.sender_email
            )
            verified = identity_info.get("VerifiedForSendingStatus", False)
            if verified:
                print(f"  {self.sender_email} is verified and ready to send.")
            else:
                print(
                    f"  {self.sender_email} exists but is not yet verified."
                )
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Identity {self.sender_email} not found. "
                    "Creating it now..."
                )
                result = self.sesv2_wrapper.create_email_identity(
                    self.sender_email
                )
                self.identity_was_created = True
                print(
                    f"  Identity created. Verification status: "
                    f"{result.get('VerifiedForSendingStatus', False)}"
                )
                print(
                    "  Check your inbox and click the verification link "
                    "before continuing."
                )
                q.ask("Press Enter when you have verified the address...")
            else:
                raise
```

```
        # Create the email template for the bulk-send step.
        print("\nCreating email template for the bulk email step...")
        try:
            self.sesv2_wrapper.create_email_template(
                template_name=self.TEMPLATE_NAME,
                subject="Bulk Email with Attachment for {{name}}",
                html_body=(
                    "<h1>Hello {{name}}</h1>"
                    "<p>Please find the attached document.</p>"
                ),
                text_body=(
                    "Hello {{name}}, Please find the attached document."
                ),
            )
            print(f"  Template '{self.TEMPLATE_NAME}' created.\n")
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' already exists. "
                    "Using it.\n"
                )
            else:
                raise

    # ---------- Step 1: Simple email with file attachment ----------

    def _step1_send_email_with_attachment(self) -> None:
        """Sends a simple email with a text file attachment."""
        print("\n--- Step 1: Send a Simple Email with a File Attachment ---\n")
        print(
            "Creating a sample text file attachment and sending it with\n"
            "the Simple email content type. SES constructs the MIME message\n"
            "automatically.\n"
        )

        # Prepare a sample text file as bytes.
        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report text file",
```

```
            "ContentTransferEncoding": "BASE64",
        }

        print(
            "Note: When using an AWS SDK, the SDK handles base64 encoding\n"
            "automatically. Direct API callers must encode content themselves.\n"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Simple Email with Attachment",
            html_body=(
                "<h1>Attachment Demo</h1>"
                "<p>Please see the attached <b>report document</b>.</p>"
            ),
            text_body="Please see the attached report document.",
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  SES automatically constructed the MIME message with the "
            "attachment.\n"
        )

    # ---------- Step 2: Simple email with inline image ----------

    def _step2_send_email_with_inline_image(self) -> None:
        """Sends a simple email with an inline image that renders in HTML."""
        print("\n--- Step 2: Send a Simple Email with an Inline Image ---\n")
        print(
            "This step demonstrates INLINE disposition. The image renders\n"
            "directly in the HTML body using a 'cid:' reference instead of\n"
            "appearing as a downloadable attachment.\n"
        )

        # Create a minimal 1x1 red PNG (valid PNG file).
        sample_image = (
            b"\x89PNG\r\n\x1a\n"  # PNG signature
            b"\x00\x00\x00\rIHDR\x00\x00\x00\x01\x00\x00\x00\x01"
            b"\x08\x02\x00\x00\x00\x90wS\xde"  # 1x1 RGB
            b"\x00\x00\x00\x0cIDATx\x9cc\xf8\x0f\x00\x00\x01\x01"
            b"\x00\x05\x18\xd8N"  # compressed data
```

```
            b"\x00\x00\x00\x00IEND\xaeB`\x82"  # IEND
        )

        attachment = {
            "RawContent": sample_image,
            "FileName": "logo.png",
            "ContentType": "image/png",
            "ContentDisposition": "INLINE",
            "ContentId": "logo123",
            "ContentDescription": "Company logo",
            "ContentTransferEncoding": "BASE64",
        }

        html_body = (
            "<html><body>"
            "<h1>Inline Image Demo</h1>"
            "<p>Here is our logo:</p>"
            '<img src="cid:logo123" alt="Company Logo">'
            "</body></html>"
        )

        message_id = self.sesv2_wrapper.send_email(
            from_address=self.sender_email,
            to_addresses=self.recipient_emails,
            subject="SESv2 Attachment Demo - Inline Image",
            html_body=html_body,
            text_body=(
                "Inline Image Demo - Please view this email in an "
                "HTML-capable client to see the embedded image."
            ),
            attachments=[attachment],
        )

        print(f"  Email sent. MessageId: {message_id}")
        print(
            "  The ContentId 'logo123' is referenced in the HTML body via\n"
            "  'cid:logo123', which lets the image render inline.\n"
        )

    # ---------- Step 3: Bulk templated email with attachments ----------

    def _step3_send_bulk_email_with_attachments(self) -> None:
        """Sends bulk templated emails with attachments to multiple recipients."""
        print("\n--- Step 3: Send Bulk Templated Emails with Attachments ---\n")
```

```
        print(
            "Using SendBulkEmail to send a templated email with an attachment\n"
            "to multiple recipients in a single API call. Each recipient gets\n"
            "personalized content via template data.\n"
        )

        sample_content = b"This is a sample report attachment."

        attachment = {
            "RawContent": sample_content,
            "FileName": "sample-report.txt",
            "ContentType": "text/plain",
            "ContentDisposition": "ATTACHMENT",
            "ContentDescription": "Sample report for bulk recipients",
            "ContentTransferEncoding": "BASE64",
        }

        # Build one entry per recipient with personalized names.
        names = ["Alice", "Bob", "Charlie", "Diana", "Eve"]
        bulk_entries = []
        for i, email in enumerate(self.recipient_emails):
            name = names[i] if i < len(names) else f"Recipient{i + 1}"
            bulk_entries.append(
                {
                    "Destination": {"ToAddresses": [email]},
                    "ReplacementEmailContent": {
                        "ReplacementTemplate": {
                            "ReplacementTemplateData": json.dumps(
                                {"name": name}
                            )
                        }
                    },
                }
            )

        results = self.sesv2_wrapper.send_bulk_email(
            from_address=self.sender_email,
            template_name=self.TEMPLATE_NAME,
            default_template_data='{"name": "Valued Customer"}',
            bulk_entries=bulk_entries,
            attachments=[attachment],
        )

        print("  Bulk email results:")
```

```
        for idx, result in enumerate(results):
            status = result.get("Status", "Unknown")
            msg_id = result.get("MessageId", "N/A")
            error = result.get("Error", "")
            recipient = (
                self.recipient_emails[idx]
                if idx < len(self.recipient_emails)
                else "Unknown"
            )
            print(f"    {recipient}: Status={status}, MessageId={msg_id}")
            if error:
                print(f"      Error: {error}")

        print(
            "\n  All recipients receive the same attachment(s) defined in\n"
            "  DefaultContent. Template data is personalized per recipient.\n"
        )

    # ---------- Cleanup ----------

    def _cleanup(self) -> None:
        """Deletes the email template and optionally the email identity."""
        print("\n--- Cleanup ---\n")

        # Delete the email template.
        try:
            self.sesv2_wrapper.delete_email_template(self.TEMPLATE_NAME)
            print(f"  Template '{self.TEMPLATE_NAME}' deleted.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                print(
                    f"  Template '{self.TEMPLATE_NAME}' was already deleted."
                )
            else:
                logger.error("Failed to delete template: %s", err)

        # Optionally delete the email identity.
        if self.identity_was_created and self.sender_email:
            delete_identity = q.ask(
                f"Delete the email identity '{self.sender_email}'? (y/n) ",
                q.is_yesno,
            )
            if delete_identity:
                try:
```

```
                    self.sesv2_wrapper.delete_email_identity(
                        self.sender_email
                    )
                    print(
                        f"  Email identity '{self.sender_email}' deleted."
                    )
                except ClientError as err:
                    if err.response["Error"]["Code"] == "NotFoundException":
                        print(
                            f"  Identity '{self.sender_email}' was "
                            "already deleted."
                        )
                    else:
                        logger.error(
                            "Failed to delete identity: %s", err
                        )
            else:
                print(
                    f"  Skipping identity deletion for {self.sender_email}."
                )
        else:
            print(
                "  Sender identity was pre-existing. Skipping deletion."
            )

        print("\nAll resources have been cleaned up.")
        print("-" * 88)
```

Erstellen Sie eine SESv2-Wrapper-Klasse zur Verwaltung von Vorgängen.

```
class SESv2Wrapper:
    """Encapsulates Amazon SESv2 email sending actions."""

    def __init__(self, sesv2_client: Any) -> None:
        """
        Initializes the SESv2Wrapper with an SESv2 client.

        :param sesv2_client: A Boto3 SESv2 client.
        """
        self.sesv2_client = sesv2_client
```

```
    @classmethod
    def from_client(cls) -> "SESv2Wrapper":
        """
        Creates an SESv2Wrapper instance with a default Boto3 SESv2 client.

        :return: A new SESv2Wrapper instance.
        """
        sesv2_client = boto3.client("sesv2")
        return cls(sesv2_client)


    def get_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
        Gets information about an email identity, including its verification status.

        :param email_address: The email address or domain to look up.
        :return: A dictionary with identity information including verification
status.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            response = self.sesv2_client.get_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Got email identity for %s.", email_address)
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found.", email_address
                )
            else:
                logger.error(
                    "Couldn't get email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_identity(self, email_address: str) -> Dict[str, Any]:
        """
```

```
        Starts the process of verifying an email identity (email address or domain).

        :param email_address: The email address or domain to verify.
        :return: A dictionary with the identity type and verification status.
        :raises ClientError: If the limit is exceeded (LimitExceededException).
        """
        try:
            response = self.sesv2_client.create_email_identity(
                EmailIdentity=email_address
            )
            logger.info(
                "Started verification for email identity %s.", email_address
            )
            return response
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email identity %s. You have exceeded "
                    "the maximum number of email identities. "
                    "Use an existing verified identity.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't create email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def create_email_template(
        self,
        template_name: str,
        subject: str,
        html_body: str,
        text_body: str,
    ) -> None:
        """
        Creates an email template for use with templated and bulk email sends.

        :param template_name: The name for the new template.
```

```
        :param subject: The subject line of the template. May include
{{placeholders}}.
        :param html_body: The HTML body of the template.
        :param text_body: The plain text body of the template.
        :raises ClientError: If the template limit is exceeded
(LimitExceededException).
        """
        try:
            self.sesv2_client.create_email_template(
                TemplateName=template_name,
                TemplateContent={
                    "Subject": subject,
                    "Html": html_body,
                    "Text": text_body,
                },
            )
            logger.info("Created email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "LimitExceededException":
                logger.error(
                    "Couldn't create email template %s. You have exceeded "
                    "the maximum number of email templates. "
                    "Delete unused templates first.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't create email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_email(
        self,
        from_address: str,
        to_addresses: List[str],
        subject: str,
        html_body: str,
        text_body: str,
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> str:
```

```
        """
        Sends a simple email message with optional attachments.

        SES handles MIME construction automatically when using attachments
        with the Simple content type, so developers don't need to build
        raw MIME messages.

        :param from_address: The verified sender email address.
        :param to_addresses: A list of recipient email addresses.
        :param subject: The subject line of the email.
        :param html_body: The HTML body content.
        :param text_body: The plain text body content.
        :param attachments: An optional list of attachment dictionaries. Each
            attachment should contain 'RawContent' (bytes), 'FileName' (str),
            and optionally 'ContentType', 'ContentDisposition', 'ContentId',
            'ContentDescription', and 'ContentTransferEncoding'.
        :return: The MessageId of the sent email.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            simple_message: Dict[str, Any] = {
                "Subject": {"Data": subject},
                "Body": {
                    "Html": {"Data": html_body},
                    "Text": {"Data": text_body},
                },
            }

            if attachments:
                simple_message["Attachments"] = attachments

            response = self.sesv2_client.send_email(
                FromEmailAddress=from_address,
                Destination={"ToAddresses": to_addresses},
                Content={"Simple": simple_message},
            )
            message_id = response["MessageId"]
            logger.info(
                "Sent email from %s to %s. MessageId: %s",
                from_address,
                to_addresses,
                message_id,
            )
            return message_id
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Message was rejected. Check that attachments use "
                    "supported file types and total message size is "
                    "under 40 MB. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def send_bulk_email(
        self,
        from_address: str,
        template_name: str,
        default_template_data: str,
        bulk_entries: List[Dict[str, Any]],
        attachments: Optional[List[Dict[str, Any]]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Sends a templated email to multiple recipients in a single API call.

        All recipients receive the same attachment(s) defined in the default
        content, while template data can be personalized per recipient.

        :param from_address: The verified sender email address.
        :param template_name: The name of an existing email template.
        :param default_template_data: Default JSON template data string.
        :param bulk_entries: A list of BulkEmailEntry dicts, each containing
            'Destination' and optionally 'ReplacementEmailContent'.
        :param attachments: An optional list of attachment dicts for all
            recipients.
        :return: A list of BulkEmailEntryResult dicts with status and MessageId.
        :raises ClientError: If the message is rejected (MessageRejected).
        """
        try:
            template_content: Dict[str, Any] = {
                "TemplateName": template_name,
```

```
                "TemplateData": default_template_data,
            }

            if attachments:
                template_content["Attachments"] = attachments

            response = self.sesv2_client.send_bulk_email(
                FromEmailAddress=from_address,
                DefaultContent={"Template": template_content},
                BulkEmailEntries=bulk_entries,
            )
            results = response.get("BulkEmailEntryResults", [])
            logger.info(
                "Sent bulk email from %s to %d recipients.",
                from_address,
                len(bulk_entries),
            )
            return results
        except ClientError as err:
            if err.response["Error"]["Code"] == "MessageRejected":
                logger.error(
                    "Bulk message was rejected. Check that the template "
                    "exists, attachment file types are supported, and "
                    "total message size is within limits. Details: %s",
                    err.response["Error"]["Message"],
                )
            else:
                logger.error(
                    "Couldn't send bulk email. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_template(self, template_name: str) -> None:
        """
        Deletes an email template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If the template is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_template(
```

```
                TemplateName=template_name
            )
            logger.info("Deleted email template %s.", template_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email template %s not found or already deleted.",
                    template_name,
                )
            else:
                logger.error(
                    "Couldn't delete email template %s. Here's why: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
            raise


    def delete_email_identity(self, email_address: str) -> None:
        """
        Deletes an email identity.

        :param email_address: The email address or domain to delete.
        :raises ClientError: If the identity is not found (NotFoundException).
        """
        try:
            self.sesv2_client.delete_email_identity(
                EmailIdentity=email_address
            )
            logger.info("Deleted email identity %s.", email_address)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NotFoundException":
                logger.info(
                    "Email identity %s not found or already deleted.",
                    email_address,
                )
            else:
                logger.error(
                    "Couldn't delete email identity %s. Here's why: %s: %s",
                    email_address,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - GetEmailIdentity
  - SendBulkEmail
  - SendEmail

Newsletter-Szenario

Das folgende Codebeispiel zeigt, wie Sie das Newsletter-Szenario von Amazon SES API v2 ausführen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
```

```
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()




class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e

            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
```

```
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e

        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
```

```
                    },
                )
                print(f"Welcome email sent to '{email}'.")

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )

        try:
            self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e

        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e

        try:
```

```
            self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)

            try:

self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not exist.")
                else:
                    print(e)

        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity

- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Beispiele für Amazon SNS unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
```

```
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise
```

- API-Details finden Sie CreateTopic in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTopic`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
```

```
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- API-Details finden Sie DeleteTopic in der AWS API-Referenz zum SDK für Python (Boto3).

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung. `ListSubscriptions`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
```

```
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- API-Details finden Sie ListSubscriptions in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung. `ListTopics`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
```

```
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- API-Details finden Sie ListTopics in der AWS API-Referenz zum SDK für Python (Boto3).

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung. `Publish`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Veröffentlichen Sie eine Nachricht mit Attributen, damit ein Abonnement basierend auf Attributen filtern kann.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
```

```
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message. Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Veröffentlichen Sie eine Nachricht, die basierend auf dem Protokoll des Abonnenten unterschiedliche Formen annimmt.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
```

```
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version is
                                sent to subscribers that have protocols that are not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
```

```
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie unter Veröffentlichen in der API-Referenz zum AWS -SDK für Python (Boto3).

`SetSubscriptionAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSubscriptionAttributes`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must have
 an
        attribute that passes the filter or it will not be sent to the subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy", AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- API-Details finden Sie SetSubscriptionAttributes in der AWS API-Referenz zum SDK für Python (Boto3).

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung. `Subscribe`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone number
                         (in E.164 format) for SMS messages, or an email address for
                         email messages.
        :return: The newly added subscription.
        """
```

```
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

Abonnieren Sie eine Warteschlange für ein Thema mit optionalen Filtern.

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def subscribe_queue_to_topic(
        self,
```

```
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie unter Abonnieren in der API-Referenz zum AWS -SDK für Python (Boto3).

`Abonnement beenden`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.", subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client
```

```
    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
                raise
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS -SDK für Python (Boto3).

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) zusammen mit Amazon Textract verwendet wird, um Text-, Formular- und Tabellenelemente in einem Dokumentbild zu erkennen. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
```

```
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)
```

```
    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII letters,
        numbers, underscores, and hyphens, and must be between 1 and 256 characters
 long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
```

```
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
```

```
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
```

```
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue

Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel wird am besten auf angesehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
```

```
        subscription.

        :param phone_number: The phone number that receives the message. This must
be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zu AWS SDK für Python (Boto3).

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe {self.queue_count}
 SQS queues to the topic.
```

```
    You can select from several options for configuring the topic and the
subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication and
message filtering.")
        print()
```

```
        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/n): 
", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ", 
q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be appended 
to the topic name.")
            print()

            print(self.DASHES)
            print("""
   Because you have chosen a FIFO topic, deduplication is supported.
   Deduplication IDs are either set in the message or automatically generated
   from content using a hash function.

   If a message is successfully published to an SNS FIFO topic, any message
   published and determined to have the same deduplication ID,
   within the five-minute deduplication interval, is accepted but not delivered.

   For more information about deduplication,
   see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a deduplication 
ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ", 
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )
```

```
        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue Service
(Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ", q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn, self.topic_arn,
queue_url)
```

```
            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription to
 the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
 {self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
```

```
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
```

```
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this message:
", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n): ",
q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")
```

```
            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message', message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
```

```
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}? (y/
n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Erstellen Sie Klassen, die Amazon SNS- und Amazon SQS-Operationen zur Verwendung in dem Szenario umfassen.

```
class SnsWrapper:
```

```
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
```

```
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
```

```
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
```

```
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
```

```
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

            logger.info(f"Found {len(topics)} topics")
```

```
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
```

```
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
```

```
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
```

```
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.
```

```
        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.
```

```
        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue

- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Verwenden von API Gateway zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von Amazon API Gateway aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel veranschaulicht, wie eine REST-API für Amazon API Gateway erstellt und verwendet wird, die auf eine AWS Lambda -Funktion verweist. Der Lambda-Handler veranschaulicht, wie basierend auf HTTP-Methoden weitergeleitet wird, wie Daten aus der Abfragezeichenfolge, dem Header und dem Text abgerufen werden und wie eine JSON-Antwort zurückgegeben wird.

- Stellen Sie eine Lambda-Funktion bereit.
- REST-API für API Gateway erstellen
- Erstellen Sie eine REST-Ressource, die auf die Lambda-Funktion verweist.
- Erteilen Sie API Gateway die Berechtigung, die Lambda-Funktion aufzurufen.
- Verwenden Sie das Anforderungspaket, um Anforderungen an die REST-API zu senden.
- Bereinigen Sie alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel lässt sich am besten unter GitHub ansehen. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon SNS

Verwendung geplanter Ereignisse zum Aufrufen einer Lambda-Funktion

Das folgende Codebeispiel zeigt, wie eine AWS Lambda Funktion erstellt wird, die von einem EventBridge geplanten Amazon-Ereignis aufgerufen wird.

SDK für Python (Boto3)

Dieses Beispiel zeigt, wie eine AWS Lambda Funktion als Ziel eines geplanten EventBridge Amazon-Ereignisses registriert wird. Der Lambda-Handler schreibt eine freundliche Nachricht und die vollständigen Ereignisdaten zum späteren Abrufen in Amazon CloudWatch Logs.

- Stellt eine Lambda-Funktion bereit.
- Erstellt ein EventBridge geplantes Ereignis und macht die Lambda-Funktion zum Ziel.
- Erteilt die Erlaubnis, die EventBridge Lambda-Funktion aufrufen zu lassen.
- Druckt die neuesten Daten aus CloudWatch Logs, um das Ergebnis der geplanten Aufrufe anzuzeigen.
- Bereinigt alle Ressourcen, die während der Demo erstellt wurden.

Dieses Beispiel eignet sich am besten für. GitHub Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Beispiele für Amazon SQS unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.
```

```
    :param name: The name of the queue. This is part of the URL assigned to the
 queue.
    :param attributes: The attributes of the queue, such as maximum message size or
                       whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can be
 used
             to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(QueueName=name, Attributes=attributes)
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)
```

```
    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise
```

- API-Details finden Sie CreateQueue in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteMessage`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained in
                    the message's metadata.
    :return: None
    """
    try:
        message.delete()
        logger.info("Deleted message: %s", message.message_id)
    except ClientError as error:
        logger.exception("Couldn't delete message: %s", message.message_id)
        raise error
```

- API-Details finden Sie DeleteMessage in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteMessageBatch`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
             message deletions.
    """
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client
```

```
    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.

        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )
```

```
                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise
```

- API-Details finden Sie DeleteMessageBatch in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteQueue`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
```

```
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.

        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise
```

- API-Details finden Sie DeleteQueue in der AWS API-Referenz zum SDK für Python (Boto3).

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung. `GetQueueAttributes`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.
```

```
        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
            raise
```

- API-Details finden Sie GetQueueAttributes in der AWS API-Referenz zum SDK für Python (Boto3).

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung. `GetQueueUrl`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
        logger.info("Got queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't get queue named %s.", name)
        raise error
    else:
        return queue
```

- API-Details finden Sie GetQueueUrl in der AWS API-Referenz zum SDK für Python (Boto3).

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung. `ListQueues`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ", ".join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- API-Details finden Sie ListQueues in der AWS API-Referenz zum SDK für Python (Boto3).

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung. `ReceiveMessage`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
```

```
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages
```

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
```

```
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise
```

- API-Details finden Sie ReceiveMessage in der AWS API-Referenz zum SDK für Python (Boto3).

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung. `SendMessage`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-value
                               pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body, MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- API-Details finden Sie SendMessage in der AWS API-Referenz zum SDK für Python (Boto3).

`SendMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung. `SendMessageBatch`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
```

```
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response
```

- API-Details finden Sie SendMessageBatch in der AWS API-Referenz zum SDK für Python (Boto3).

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung. `SetQueueAttributes`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richtlinienattribut einer Warteschlange für ein Thema festlegen

```
class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
```

```
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
```

```
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise
```

- API-Details finden Sie SetQueueAttributes in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen einer Messenger-Anwendung

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

SDK für Python (Boto3)

Zeigt, wie Sie with verwenden, AWS Step Functions um eine Messenger-Anwendung zu erstellen, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. AWS SDK für Python (Boto3) Die State-Machine ist in eine AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht wird.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step-Funktionen

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) mit Amazon Textract verwendet wird, um Text-, Formular- und Tabellenelemente in einem Dokumentbild zu erkennen. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS

- Amazon SQS
- Amazon Textract

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Amazon-SNS-FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO- und eine Standard-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu dem Thema.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
```

```
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )
```

```
    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII letters,
        numbers, underscores, and hyphens, and must be between 1 and 256 characters
 long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
```

```
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                    "FifoThroughputScope": "MessageGroup",
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version":"2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
```

```
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
```

```
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Erkennen Sie Personen und Objekte in einem Video

Das folgende Codebeispiel zeigt, wie Sie Personen und Objekte in einem Video mit Amazon Rekognition erkennen.

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel wird am besten auf angesehen GitHub. Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
        self.queue_count = 2
        self.queue_urls = []
        self.subscription_arns = []
        self.tones = ["cheerful", "funny", "serious", "sincere"]

    def run_scenario(self) -> None:
        """Run the Topics and Queues feature scenario."""
        print(self.DASHES)
        print("Welcome to messaging with topics and queues.")
        print(self.DASHES)
        print(f"""
    In this scenario, you will create an SNS topic and subscribe {self.queue_count}
 SQS queues to the topic.
```

```
    You can select from several options for configuring the topic and the
subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
        """)

        try:
            # Setup Phase
            print(self.DASHES)
            self._setup_topic()
            print(self.DASHES)

            self._setup_queues()
            print(self.DASHES)

            # Demonstration Phase
            self._publish_messages()
            print(self.DASHES)

            # Examination Phase
            self._poll_queues_for_messages()
            print(self.DASHES)

            # Cleanup Phase
            self._cleanup_resources()
            print(self.DASHES)

        except Exception as e:
            logger.error(f"Scenario failed: {e}")
            print(f"There was a problem with the scenario: {e}")
            print("\nInitiating cleanup...")
            try:
                self._cleanup_resources()
            except Exception as cleanup_error:
                logger.error(f"Error during cleanup: {cleanup_error}")

        print("Messaging with topics and queues scenario is complete.")
        print(self.DASHES)

    def _setup_topic(self) -> None:
        """Set up the SNS topic to be used with the queues."""
        print("SNS topics can be configured as FIFO (First-In-First-Out).")
        print("FIFO topics deliver messages in order and support deduplication and
message filtering.")
        print()
```

```
        self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/n): 
", q.is_yesno)

        if self.use_fifo_topic:
            print(self.DASHES)
            self.topic_name = q.ask("Enter a name for your SNS topic: ", 
q.non_empty)
            print("Because you have selected a FIFO topic, '.fifo' must be appended 
to the topic name.")
            print()

            print(self.DASHES)
            print("""
   Because you have chosen a FIFO topic, deduplication is supported.
   Deduplication IDs are either set in the message or automatically generated
   from content using a hash function.

   If a message is successfully published to an SNS FIFO topic, any message
   published and determined to have the same deduplication ID,
   within the five-minute deduplication interval, is accepted but not delivered.

   For more information about deduplication,
   see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
            """)

            self.use_content_based_deduplication = q.ask(
                "Use content-based deduplication instead of entering a deduplication 
ID? (y/n): ",
                q.is_yesno
            )
        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ", 
q.non_empty)

        print(self.DASHES)

        # Create the topic
        self.topic_arn = self.sns_wrapper.create_topic(
            self.topic_name,
            self.use_fifo_topic,
            self.use_content_based_deduplication
        )
```

```
        print(f"Your new topic with the name {self.topic_name}")
        print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
        print(f"  has been created.")
        print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue Service
(Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ", q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")
                print(self.DASHES)

            # Get queue ARN
            queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

            if i == 0:
                print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
                print("enabling it to receive messages from an SNS topic.")

            # Set queue policy to allow SNS to send messages
            self.sqs_wrapper.set_queue_policy_for_topic(queue_arn, self.topic_arn,
queue_url)
```

```
            # Set up message filtering if using FIFO
            subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
 queue_name)
            self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
 queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
 filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
 attribute.")

            use_filter = q.ask(f"Filter messages for {queue_name}'s subscription to
 the topic? (y/n): ", q.is_yesno)

            if use_filter:
                filter_policy = self._create_filter_policy()

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
 {self.topic_name}")
        print(f"  with the subscription ARN {subscription_arn}")

        return subscription_arn

    def _create_filter_policy(self) -> str:
        """Create a message filter policy based on user selections."""
        print(self.DASHES)
```

```
        print("You can filter messages by one or more of the following TONE
attributes.")

        filter_selections = []
        selection_number = 0

        while True:
            print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
            for i, tone in enumerate(self.tones, 1):
                print(f"  {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

            if selection == 0:
                break
            elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
                filter_selections.append(self.tones[selection - 1])
                print(f"Added '{self.tones[selection - 1]}' to filter list.")

        if filter_selections:
            filters = {"tone": filter_selections}
            return json.dumps(filters)
        return None

    def _publish_messages(self) -> None:
        """Publish messages to the topic with various options."""
        print("Now we can publish messages.")

        keep_sending = True
        while keep_sending:
            print()
            message = q.ask("Enter a message to publish: ", q.non_empty)

            message_group_id = None
            deduplication_id = None
            tone_attribute = None

            if self.use_fifo_topic:
                print("Because you are using a FIFO topic, you must set a message
group ID.")
```

```
                print("All messages within the same group will be received in the
order they were published.")
                print()
                message_group_id = q.ask("Enter a message group ID for this message:
", q.non_empty)

                if not self.use_content_based_deduplication:
                    print("Because you are not using content-based deduplication,")
                    print("you must enter a deduplication ID.")
                    deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

                # Ask about tone attribute
                add_attribute = q.ask("Add an attribute to this message? (y/n): ",
q.is_yesno)
                if add_attribute:
                    print("Enter a number for an attribute:")
                    for i, tone in enumerate(self.tones, 1):
                        print(f"  {i}. {tone}")

                    selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
                    if 1 <= selection <= len(self.tones):
                        tone_attribute = self.tones[selection - 1]

            # Publish the message
            message_id = self.sns_wrapper.publish_message(
                self.topic_arn,
                message,
                tone_attribute,
                deduplication_id,
                message_group_id
            )

            print(f"Message published with ID: {message_id}")

            keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

    def _poll_queues_for_messages(self) -> None:
        """Poll all queues for messages and display results."""
        for i, queue_url in enumerate(self.queue_urls):
            print(f"Polling queue #{i+1} at {queue_url} for messages...")

            q.ask("Press Enter to continue...")
```

```
            messages = self._poll_queue_for_messages(queue_url)

            if messages:
                print(f"{len(messages)} message(s) were received by queue #{i+1}")
                for j, message in enumerate(messages, 1):
                    print(f"  Message {j}:")
                    # Parse the SNS message body to get the actual message
                    try:
                        sns_message = json.loads(message['Body'])
                        actual_message = sns_message.get('Message', message['Body'])
                        print(f"    {actual_message}")
                    except (json.JSONDecodeError, KeyError):
                        print(f"    {message['Body']}")

                # Delete the messages
                self.sqs_wrapper.delete_messages(queue_url, messages)
                print(f"Messages deleted from queue #{i+1}")
            else:
                print(f"No messages received by queue #{i+1}")

            print(self.DASHES)

    def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
        """Poll a single queue for messages."""
        all_messages = []
        max_polls = 3  # Limit polling to avoid infinite loops

        for poll_count in range(max_polls):
            messages = self.sqs_wrapper.receive_messages(queue_url, 10)

            if messages:
                all_messages.extend(messages)
                print(f"  Received {len(messages)} messages in poll {poll_count +
1}")
                # Small delay between polls
                time.sleep(1)
            else:
                print(f"  No messages in poll {poll_count + 1}")
                break

        return all_messages

    def _cleanup_resources(self) -> None:
```

```
        """Clean up all resources created during the scenario."""
        print("Cleaning up resources...")

        # Delete queues
        for i, queue_url in enumerate(self.queue_urls):
            if queue_url:
                delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}? (y/
n): ", q.is_yesno)
                if delete_queue:
                    try:
                        self.sqs_wrapper.delete_queue(queue_url)
                        print(f"Deleted queue #{i+1}")
                    except Exception as e:
                        print(f"Error deleting queue #{i+1}: {e}")

        # Unsubscribe from topic
        for i, subscription_arn in enumerate(self.subscription_arns):
            if subscription_arn:
                try:
                    self.sns_wrapper.unsubscribe(subscription_arn)
                    print(f"Unsubscribed subscription #{i+1}")
                except Exception as e:
                    print(f"Error unsubscribing #{i+1}: {e}")

        # Delete topic
        if self.topic_arn:
            delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
 q.is_yesno)
            if delete_topic:
                try:
                    self.sns_wrapper.delete_topic(self.topic_arn)
                    print(f"Deleted topic {self.topic_name}")
                except Exception as e:
                    print(f"Error deleting topic: {e}")

        print("Resource cleanup complete.")
```

Erstellen Sie Klassen, die Amazon SNS- und Amazon SQS-Operationen zur Verwendung in dem Szenario umfassen.

```
class SnsWrapper:
```

```
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)


    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
```

```
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
            raise


    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param queue_arn: The ARN of the SQS queue.
        :param filter_policy: Optional JSON filter policy for message filtering.
        :return: The ARN of the subscription.
        :raises ClientError: If the subscription fails.
        """
        try:
            attributes = {}
            if filter_policy:
                attributes['FilterPolicy'] = filter_policy

            response = self.sns_client.subscribe(
                TopicArn=topic_arn,
                Protocol='sqs',
                Endpoint=queue_arn,
                Attributes=attributes
```

```
            )

            subscription_arn = response['SubscriptionArn']
            logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
            return subscription_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
            raise


    def publish_message(
        self,
        topic_arn: str,
        message: str,
        tone_attribute: Optional[str] = None,
        deduplication_id: Optional[str] = None,
        message_group_id: Optional[str] = None
    ) -> str:
        """
        Publish a message to an SNS topic.

        :param topic_arn: The ARN of the SNS topic.
        :param message: The message content to publish.
        :param tone_attribute: Optional tone attribute for message filtering.
        :param deduplication_id: Optional deduplication ID for FIFO topics.
        :param message_group_id: Optional message group ID for FIFO topics.
        :return: The message ID of the published message.
        :raises ClientError: If the message publication fails.
        """
        try:
            publish_args = {
                'TopicArn': topic_arn,
                'Message': message
            }

            # Add message attributes if tone is specified
            if tone_attribute:
                publish_args['MessageAttributes'] = {
                    'tone': {
                        'DataType': 'String',
                        'StringValue': tone_attribute
                    }
```

```
                }

            # Add FIFO-specific parameters
            if message_group_id:
                publish_args['MessageGroupId'] = message_group_id

            if deduplication_id:
                publish_args['MessageDeduplicationId'] = deduplication_id

            response = self.sns_client.publish(**publish_args)

            message_id = response['MessageId']
            logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error publishing message to topic: {error_code} - {e}")
            raise


    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
                logger.error(f"Error unsubscribing: {error_code} - {e}")
```

```
                raise


    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise


    def list_topics(self) -> list:
        """
        List all SNS topics in the account using pagination.

        :return: List of topic ARNs.
        :raises ClientError: If listing topics fails.
        """
        try:
            topics = []
            paginator = self.sns_client.get_paginator('list_topics')

            for page in paginator.paginate():
                topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

            logger.info(f"Found {len(topics)} topics")
```

```
            return topics

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AuthorizationError':
                logger.error("Authorization error listing topics - check IAM
 permissions")
            else:
                logger.error(f"Error listing topics: {error_code} - {e}")
            raise


class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """
        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)


    def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
        """
        Create an SQS queue.

        :param queue_name: The name of the queue to create.
        :param is_fifo: Whether to create a FIFO queue.
        :return: The URL of the created queue.
        :raises ClientError: If the queue creation fails.
        """
        try:
```

```
            # Add .fifo suffix for FIFO queues
            if is_fifo and not queue_name.endswith('.fifo'):
                queue_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoQueue'] = 'true'

            response = self.sqs_client.create_queue(
                QueueName=queue_name,
                Attributes=attributes
            )

            queue_url = response['QueueUrl']
            logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
            return queue_url

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
            raise


    def get_queue_arn(self, queue_url: str) -> str:
        """
        Get the ARN of an SQS queue.

        :param queue_url: The URL of the queue.
        :return: The ARN of the queue.
        :raises ClientError: If getting queue attributes fails.
        """
        try:
            response = self.sqs_client.get_queue_attributes(
                QueueUrl=queue_url,
                AttributeNames=['QueueArn']
            )

            queue_arn = response['Attributes']['QueueArn']
            logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
            return queue_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error getting queue ARN: {error_code} - {e}")
```

```
            raise


    def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str, queue_url:
str) -> bool:
        """
        Set the queue policy to allow SNS to send messages to the queue.

        :param queue_arn: The ARN of the SQS queue.
        :param topic_arn: The ARN of the SNS topic.
        :param queue_url: The URL of the SQS queue.
        :return: True if successful.
        :raises ClientError: If setting the queue policy fails.
        """
        try:
            # Create policy that allows SNS to send messages to the queue
            policy = {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "sns.amazonaws.com"
                        },
                        "Action": "sqs:SendMessage",
                        "Resource": queue_arn,
                        "Condition": {
                            "ArnEquals": {
                                "aws:SourceArn": topic_arn
                            }
                        }
                    }
                ]
            }

            self.sqs_client.set_queue_attributes(
                QueueUrl=queue_url,
                Attributes={
                    'Policy': json.dumps(policy)
                }
            )

            logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
```

```
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error setting queue policy: {error_code} - {e}")
            raise


    def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
        """
        Receive messages from an SQS queue.

        :param queue_url: The URL of the queue to receive messages from.
        :param max_messages: Maximum number of messages to receive (1-10).
        :return: List of received messages.
        :raises ClientError: If receiving messages fails.
        """
        try:
            # Ensure max_messages is within valid range
            max_messages = max(1, min(10, max_messages))

            response = self.sqs_client.receive_message(
                QueueUrl=queue_url,
                MaxNumberOfMessages=max_messages,
                WaitTimeSeconds=2,  # Short polling
                MessageAttributeNames=['All']
            )

            messages = response.get('Messages', [])
            logger.info(f"Received {len(messages)} messages from {queue_url}")
            return messages

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error receiving messages: {error_code} - {e}")
            raise


    def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
        """
        Delete messages from an SQS queue in batches.
```

```
        :param queue_url: The URL of the queue.
        :param messages: List of messages to delete.
        :return: True if successful.
        :raises ClientError: If deleting messages fails.
        """
        try:
            if not messages:
                return True

            # Build delete entries for batch delete
            delete_entries = []
            for i, message in enumerate(messages):
                delete_entries.append({
                    'Id': str(i),
                    'ReceiptHandle': message['ReceiptHandle']
                })

            # Delete messages in batches of 10 (SQS limit)
            batch_size = 10
            for i in range(0, len(delete_entries), batch_size):
                batch = delete_entries[i:i + batch_size]

                response = self.sqs_client.delete_message_batch(
                    QueueUrl=queue_url,
                    Entries=batch
                )

                # Check for failures
                if 'Failed' in response and response['Failed']:
                    for failed in response['Failed']:
                        logger.warning(f"Failed to delete message: {failed}")

            logger.info(f"Deleted {len(messages)} messages from {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error deleting messages: {error_code} - {e}")
            raise


    def delete_queue(self, queue_url: str) -> bool:
        """
        Delete an SQS queue.
```

```
        :param queue_url: The URL of the queue to delete.
        :return: True if successful.
        :raises ClientError: If the queue deletion fails.
        """
        try:
            self.sqs_client.delete_queue(QueueUrl=queue_url)

            logger.info(f"Deleted queue: {queue_url}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
                logger.warning(f"Queue not found: {queue_url}")
                return True  # Already deleted
            else:
                logger.error(f"Error deleting queue: {error_code} - {e}")
                raise


    def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
        """
        List all SQS queues in the account using pagination.

        :param queue_name_prefix: Optional prefix to filter queue names.
        :return: List of queue URLs.
        :raises ClientError: If listing queues fails.
        """
        try:
            queue_urls = []
            paginator = self.sqs_client.get_paginator('list_queues')

            page_params = {}
            if queue_name_prefix:
                page_params['QueueNamePrefix'] = queue_name_prefix

            for page in paginator.paginate(**page_params):
                queue_urls.extend(page.get('QueueUrls', []))

            logger.info(f"Found {len(queue_urls)} queues")
            return queue_urls
```

```
        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            if error_code == 'AccessDenied':
                logger.error("Access denied listing queues - check IAM permissions")
            else:
                logger.error(f"Error listing queues: {error_code} - {e}")
            raise

    def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
        """
        Send a message to an SQS queue.

        :param queue_url: The URL of the queue.
        :param message_body: The message content.
        :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
        :return: The message ID.
        :raises ClientError: If sending the message fails.
        """
        try:
            send_params = {
                'QueueUrl': queue_url,
                'MessageBody': message_body,
                **kwargs
            }

            response = self.sqs_client.send_message(**send_params)

            message_id = response['MessageId']
            logger.info(f"Sent message to {queue_url} with ID: {message_id}")
            return message_id

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error sending message: {error_code} - {e}")
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue

- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Senden und Empfangen von Stapeln von Nachrichten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Amazon SQS-Warteschlange
- Senden Sie Stapel von Nachrichten an die Warteschlange.
- Empfangen Sie Stapel von Nachrichten aus der Warteschlange.
- Löschen Sie Stapel von Nachrichten aus der Warteschlange.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Funktionen, um Amazon-SQS-Nachrichtenfunktionen zu umschließen.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError
```

```
import queue_wrapper

logger = logging.getLogger(__name__)
sqs = boto3.resource("sqs")

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
             messages.
    """
    try:
        entries = [
            {
                "Id": str(ind),
                "MessageBody": msg["body"],
                "MessageAttributes": msg["attributes"],
            }
            for ind, msg in enumerate(messages)
        ]
        response = queue.send_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta["MessageId"],
                    messages[int(msg_meta["Id"])]["body"],
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
```

```
        raise error
    else:
        return response



def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=["All"],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time,
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages



def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
             message deletions.
    """
```

```
    try:
        entries = [
            {"Id": str(ind), "ReceiptHandle": msg.receipt_handle}
            for ind, msg in enumerate(messages)
        ]
        response = queue.delete_messages(Entries=entries)
        if "Successful" in response:
            for msg_meta in response["Successful"]:
                logger.info("Deleted %s",
 messages[int(msg_meta["Id"])].receipt_handle)
        if "Failed" in response:
            for msg_meta in response["Failed"]:
                logger.warning(
                    "Could not delete %s",
 messages[int(msg_meta["Id"])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

Verwenden Sie die Wrapper-Funktionen, um Nachrichten stapelweise zu senden und zu empfangen.

```
def usage_demo():
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """

    def pack_message(msg_path, msg_body, msg_line):
        return {
            "body": msg_body,
            "attributes": {
                "path": {"StringValue": msg_path, "DataType": "String"},
                "line": {"StringValue": str(msg_line), "DataType": "String"},
            },
```

```
        }

    def unpack_message(msg):
        return (
            msg.message_attributes["path"]["StringValue"],
            msg.body,
            int(msg.message_attributes["line"]["StringValue"]),
        )

    print("-" * 88)
    print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
    print("-" * 88)

    queue = queue_wrapper.create_queue("sqs-usage-demo-message-wrapper")

    with open(__file__) as file:
        lines = file.readlines()

    line = 0
    batch_size = 10
    received_lines = [None] * len(lines)
    print(f"Sending file lines in batches of {batch_size} as messages.")
    while line < len(lines):
        messages = [
            pack_message(__file__, lines[index], index)
            for index in range(line, min(line + batch_size, len(lines)))
        ]
        line = line + batch_size
        send_messages(queue, messages)
        print(".", end="")
        sys.stdout.flush()
    print(f"Done. Sent {len(lines) - 1} messages.")

    print(f"Receiving, handling, and deleting messages in batches of {batch_size}.")
    more_messages = True
    while more_messages:
        received_messages = receive_messages(queue, batch_size, 2)
        print(".", end="")
        sys.stdout.flush()
        for message in received_messages:
            path, body, line = unpack_message(message)
            received_lines[line] = body
        if received_messages:
            delete_messages(queue, received_messages)
```

```
        else:
            more_messages = False
    print("Done.")

    if all([lines[index] == received_lines[index] for index in range(len(lines))]):
        print(f"Successfully reassembled all file lines!")
    else:
        print(f"Uh oh, some lines were missed!")

    queue.delete()

    print("Thanks for watching!")
    print("-" * 88)
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for message in event['Records']:
        process_message(message)
    print("done")

def process_message(message):
    try:
        print(f"Processed message {message['body']}")
        # TODO: Do interesting work based on the new message
    except Exception as err:
        print("An error occurred")
        raise err
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

def lambda_handler(event, context):
    if event:
        batch_item_failures = []
        sqs_batch_response = {}

        for record in event["Records"]:
            try:
                print(f"Processed message: {record['body']}")
            except Exception as e:
                batch_item_failures.append({"itemIdentifier": record['messageId']})

        sqs_batch_response["batchItemFailures"] = batch_item_failures
        return sqs_batch_response
```

Beispiele für Step Functions unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen
- Szenarien

Erste Schritte

Hello Step Functions

Das folgende Codebeispiel zeigt, wie Sie mit Step Functions beginnen.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3


def hello_stepfunctions(stepfunctions_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Step Functions client and
 list
    the state machines in your account. This list might be empty if you haven't
 created
    any state machines.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param stepfunctions_client: A Boto3 Step Functions Client object.
    """
    print("Hello, Step Functions! Let's list up to 10 of your state machines:")
    state_machines = stepfunctions_client.list_state_machines(maxResults=10)
    for sm in state_machines["stateMachines"]:
        print(f"\t{sm['name']}: {sm['stateMachineArn']}")


if __name__ == "__main__":
    hello_stepfunctions(boto3.client("stepfunctions"))
```

- API-Details finden Sie ListStateMachines in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Aktivität.
- Erstellen Sie einen Zustandsautomaten aus einer Definition von Amazon States Language, die die zuvor erstellte Aktivität als Schritt enthält.
- Führen Sie den Zustandsautomaten aus und reagieren Sie auf die Aktivität mit Benutzereingaben.
- Ermitteln Sie den endgültigen Status und die Ausgabe, nachdem der Lauf abgeschlossen ist, und bereinigen Sie dann die Ressourcen.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class StateMachineScenario:
    """Runs an interactive scenario that shows how to get started using Step
 Functions."""

    def __init__(self, activity, state_machine, iam_client):
        """
        :param activity: An object that wraps activity actions.
        :param state_machine: An object that wraps state machine actions.
        :param iam_client: A Boto3 AWS Identity and Access Management (IAM) client.
        """
        self.activity = activity
        self.state_machine = state_machine
        self.iam_client = iam_client
```

```
        self.state_machine_role = None

    def prerequisites(self, state_machine_role_name):
        """
        Finds or creates an IAM role that can be assumed by Step Functions.
        A role of this kind is required to create a state machine.
        The state machine used in this example does not call any additional
services,
        so it needs no additional permissions.

        :param state_machine_role_name: The name of the role.
        :return: Data about the role.
        """
        trust_policy = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {"Service": "states.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        try:
            role = self.iam_client.get_role(RoleName=state_machine_role_name)
            print(f"Prerequisite IAM role {state_machine_role_name} already
exists.")
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                role = None
            else:
                logger.error(
                    "Couldn't get prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        if role is None:
            try:
                role = self.iam_client.create_role(
                    RoleName=state_machine_role_name,
                    AssumeRolePolicyDocument=json.dumps(trust_policy),
```

```
                )
            except ClientError as err:
                logger.error(
                    "Couldn't create prerequisite IAM role %s. Here's why: %s: %s",
                    state_machine_role_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        self.state_machine_role = role["Role"]

    def find_or_create_activity(self, activity_name):
        """
        Finds or creates a Step Functions activity.

        :param activity_name: The name of the activity.
        :return: The Amazon Resource Name (ARN) of the activity.
        """
        print("First, let's set up an activity and state machine.")
        activity_arn = self.activity.find(activity_name)
        if activity_arn is None:
            activity_arn = self.activity.create(activity_name)
            print(
                f"Activity {activity_name} created. Its Amazon Resource Name (ARN)
is "
                f"{activity_arn}."
            )
        else:
            print(f"Activity {activity_name} already exists.")
        return activity_arn

    def find_or_create_state_machine(
        self, state_machine_name, activity_arn, state_machine_file
    ):
        """
        Finds or creates a Step Functions state machine.

        :param state_machine_name: The name of the state machine.
        :param activity_arn: The ARN of an activity that is used as a step in the
state
                             machine. This ARN is injected into the state machine
                             definition that's used to create the state machine.
        :param state_machine_file: The path to a file containing the state machine
                                   definition.
```

```
        :return: The ARN of the state machine.
        """
        state_machine_arn = self.state_machine.find(state_machine_name)
        if state_machine_arn is None:
            with open(state_machine_file) as state_machine_file:
                state_machine_def = state_machine_file.read().replace(
                    "{{DOC_EXAMPLE_ACTIVITY_ARN}}", activity_arn
                )
                state_machine_arn = self.state_machine.create(
                    state_machine_name,
                    state_machine_def,
                    self.state_machine_role["Arn"],
                )
            print(f"State machine {state_machine_name} created.")
        else:
            print(f"State machine {state_machine_name} already exists.")
        print("-" * 88)
        print(f"Here's some information about state machine {state_machine_name}:")
        state_machine_info = self.state_machine.describe(state_machine_arn)
        for field in ["name", "status", "stateMachineArn", "roleArn"]:
            print(f"\t{field}: {state_machine_info[field]}")
        return state_machine_arn

    def run_state_machine(self, state_machine_arn, activity_arn):
        """
        Run the state machine. The state machine used in this example is a simple
        chat simulation. It contains an activity step in a loop that is used for
user
        interaction. When the state machine gets to the activity step, it waits for
        an external application to get task data and submit a response. This
function
        acts as the activity application by getting task input and responding with
        user input.

        :param state_machine_arn: The ARN of the state machine.
        :param activity_arn: The ARN of the activity used as a step in the state
machine.
        :return: The ARN of the run.
        """
        print(
            f"Let's run the state machine. It's a simplistic, non-AI chat simulator
"
            f"we'll call ChatSFN."
        )
```

```
        user_name = q.ask("What should ChatSFN call you? ", q.non_empty)
        run_input = {"name": user_name}
        print("Starting state machine...")
        run_arn = self.state_machine.start(state_machine_arn, json.dumps(run_input))
        action = None
        while action != "done":
            activity_task = self.activity.get_task(activity_arn)
            task_input = json.loads(activity_task["input"])
            print(f"ChatSFN: {task_input['message']}")
            action = task_input["actions"][
                q.choose("What now? ", task_input["actions"])
            ]
            task_response = {"action": action}
            self.activity.send_task_success(
                activity_task["taskToken"], json.dumps(task_response)
            )
        return run_arn

    def finish_state_machine_run(self, run_arn):
        """
        Wait for the state machine run to finish, then print final status and
output.

        :param run_arn: The ARN of the run to retrieve.
        """
        print(f"Let's get the final output from the state machine:")
        status = "RUNNING"
        while status == "RUNNING":
            run_output = self.state_machine.describe_run(run_arn)
            status = run_output["status"]
            if status == "RUNNING":
                print(
                    "The state machine is still running, let's wait for it to
finish."
                )
                wait(1)
            elif status == "SUCCEEDED":
                print(f"ChatSFN: {json.loads(run_output['output'])['message']}")
            else:
                print(f"Run status: {status}.")

    def cleanup(
        self,
        state_machine_name,
```

```
        state_machine_arn,
        activity_name,
        activity_arn,
        state_machine_role_name,
    ):
        """
        Clean up resources created by this example.

        :param state_machine_name: The name of the state machine.
        :param state_machine_arn: The ARN of the state machine.
        :param activity_name: The name of the activity.
        :param activity_arn: The ARN of the activity.
        :param state_machine_role_name: The name of the role used by the state
machine.
        """
        if q.ask(
            "Do you want to delete the state machine, activity, and role created for
this "
            "example? (y/n) ",
            q.is_yesno,
        ):
            self.state_machine.delete(state_machine_arn)
            print(f"Deleted state machine {state_machine_name}.")
            self.activity.delete(activity_arn)
            print(f"Deleted activity {activity_name}.")
            self.iam_client.delete_role(RoleName=state_machine_role_name)
            print(f"Deleted role {state_machine_role_name}.")

    def run_scenario(self, activity_name, state_machine_name):
        print("-" * 88)
        print("Welcome to the AWS Step Functions state machines demo.")
        print("-" * 88)

        activity_arn = self.find_or_create_activity(activity_name)
        state_machine_arn = self.find_or_create_state_machine(
            state_machine_name,
            activity_arn,
            "../../../resources/sample_files/chat_sfn_state_machine.json",
        )
        print("-" * 88)
        run_arn = self.run_state_machine(state_machine_arn, activity_arn)
        print("-" * 88)
        self.finish_state_machine_run(run_arn)
        print("-" * 88)
```

```
        self.cleanup(
            state_machine_name,
            state_machine_arn,
            activity_name,
            activity_arn,
            self.state_machine_role["RoleName"],
        )

        print("-" * 88)
        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        stepfunctions_client = boto3.client("stepfunctions")
        iam_client = boto3.client("iam")
        scenario = StateMachineScenario(
            Activity(stepfunctions_client),
            StateMachine(stepfunctions_client),
            iam_client,
        )
        scenario.prerequisites("doc-example-state-machine-chat")
        scenario.run_scenario("doc-example-activity", "doc-example-state-machine")
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Aktionen von Zustandsautomaten umschließt.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
```

```
        Creates a state machine with the specific definition. The state machine
assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is assumed
by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]


    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about the
run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
```

```
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["executionArn"]


    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
```

```
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

Definieren Sie eine Klasse, die Aktivitätsaktionen umschließt.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator = self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
```

```
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with an
        activity step waits for the activity to get task data and then respond with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included in
the
                           response to the get_activity_task action and must be sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self, activity_arn):
        """
        Delete an activity.
```

```
        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt die Verwendung`CreateActivity`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name):
        """
        Create an activity.

        :param name: The name of the activity to create.
        :return: The Amazon Resource Name (ARN) of the newly created activity.
        """
        try:
            response = self.stepfunctions_client.create_activity(name=name)
        except ClientError as err:
            logger.error(
                "Couldn't create activity %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["activityArn"]
```

- API-Details finden Sie CreateActivity in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung. `CreateStateMachine`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def create(self, name, definition, role_arn):
        """
        Creates a state machine with the specific definition. The state machine
 assumes
        the provided role before it starts a run.

        :param name: The name to give the state machine.
        :param definition: The Amazon States Language definition of the steps in the
                           the state machine.
        :param role_arn: The Amazon Resource Name (ARN) of the role that is assumed
 by
                         Step Functions when the state machine is run.
        :return: The ARN of the newly created state machine.
        """
        try:
```

```
            response = self.stepfunctions_client.create_state_machine(
                name=name, definition=definition, roleArn=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't create state machine %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["stateMachineArn"]
```

- API-Details finden Sie CreateStateMachine in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteActivity`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
```

```
    def delete(self, activity_arn):
        """
        Delete an activity.

        :param activity_arn: The ARN of the activity to delete.
        """
        try:
            response = self.stepfunctions_client.delete_activity(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie DeleteActivity in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteStateMachine`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
```

```
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def delete(self, state_machine_arn):
        """
        Delete a state machine and all of its run data.

        :param state_machine_arn: The ARN of the state machine to delete.
        """
        try:
            response = self.stepfunctions_client.delete_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie DeleteStateMachine in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeExecution`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeExecution`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    def describe_run(self, run_arn):
        """
        Get data about a state machine run, such as its current status or final
output.

        :param run_arn: The ARN of the run to look up.
        :return: The retrieved run data.
        """
        try:
            response = self.stepfunctions_client.describe_execution(
                executionArn=run_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe run %s. Here's why: %s: %s",
                run_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie DescribeExecution in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeStateMachine`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
```

```
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def describe(self, state_machine_arn):
        """
        Get data about a state machine.

        :param state_machine_arn: The ARN of the state machine to look up.
        :return: The retrieved state machine data.
        """
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=state_machine_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't describe state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie DescribeStateMachine in der AWS API-Referenz zum SDK für Python (Boto3).

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung. `GetActivityTask`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def get_task(self, activity_arn):
        """
        Gets task data for an activity. When a state machine is waiting for the
        specified activity, a response is returned with data from the state machine.
        When a state machine is not waiting, this call blocks for 60 seconds.

        :param activity_arn: The ARN of the activity to get task data for.
        :return: The task data for the activity.
        """
        try:
            response = self.stepfunctions_client.get_activity_task(
                activityArn=activity_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't get a task for activity %s. Here's why: %s: %s",
                activity_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- API-Details finden Sie GetActivityTask in der AWS API-Referenz zum SDK für Python (Boto3).

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung. `ListActivities`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find an activity by name. This requires listing activities until one is
 found
        with a matching name.

        :param name: The name of the activity to search for.
        :return: If found, the ARN of the activity; otherwise, None.
        """
        try:
            paginator = self.stepfunctions_client.get_paginator("list_activities")
            for page in paginator.paginate():
                for activity in page.get("activities", []):
                    if activity["name"] == name:
                        return activity["activityArn"]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list activities. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListActivities in der AWS API-Referenz zum SDK für Python (Boto3).

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung. `ListStateMachines`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Suchen Sie eine Zustandsmaschine anhand ihres Namens, indem Sie die Liste der Zustandsmaschinen für das Konto durchsuchen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def find(self, name):
        """
        Find a state machine by name. This requires listing the state machines until
        one is found with a matching name.

        :param name: The name of the state machine to search for.
```

```
        :return: The ARN of the state machine if found; otherwise, None.
        """
        try:
            paginator =
 self.stepfunctions_client.get_paginator("list_state_machines")
            for page in paginator.paginate():
                for state_machine in page.get("stateMachines", []):
                    if state_machine["name"] == name:
                        return state_machine["stateMachineArn"]
        except ClientError as err:
            logger.error(
                "Couldn't list state machines. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListStateMachines in der AWS API-Referenz zum SDK für Python (Boto3).

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung. `SendTaskSuccess`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class Activity:
    """Encapsulates Step Function activity actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
```

```
        self.stepfunctions_client = stepfunctions_client


    def send_task_success(self, task_token, task_response):
        """
        Sends a success response to a waiting activity step. A state machine with an
        activity step waits for the activity to get task data and then respond with
        either success or failure before it resumes processing.

        :param task_token: The token associated with the task. This is included in
the
                           response to the get_activity_task action and must be sent
                           without modification.
        :param task_response: The response data from the activity. This data is
                              received and processed by the state machine.
        """
        try:
            self.stepfunctions_client.send_task_success(
                taskToken=task_token, output=task_response
            )
        except ClientError as err:
            logger.error(
                "Couldn't send task success. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie SendTaskSuccess in der AWS API-Referenz zum SDK für Python (Boto3).

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung. `StartExecution`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class StateMachine:
    """Encapsulates Step Functions state machine actions."""

    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client


    def start(self, state_machine_arn, run_input):
        """
        Start a run of a state machine with a specified input. A run is also known
        as an "execution" in Step Functions.

        :param state_machine_arn: The ARN of the state machine to run.
        :param run_input: The input to the state machine, in JSON format.
        :return: The ARN of the run. This can be used to get information about the
 run,
                 including its current status and final output.
        """
        try:
            response = self.stepfunctions_client.start_execution(
                stateMachineArn=state_machine_arn, input=run_input
            )
        except ClientError as err:
            logger.error(
                "Couldn't start state machine %s. Here's why: %s: %s",
                state_machine_arn,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return response["executionArn"]
```

- API-Details finden Sie StartExecution in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen einer Messenger-Anwendung

Das folgende Codebeispiel zeigt, wie eine AWS Step Functions Messenger-Anwendung erstellt wird, die Nachrichtendatensätze aus einer Datenbanktabelle abruft.

SDK für Python (Boto3)

Zeigt, wie Sie with verwenden, AWS Step Functions um eine Messenger-Anwendung zu erstellen, die Nachrichtendatensätze aus einer Amazon DynamoDB-Tabelle abruft und sie mit Amazon Simple Queue Service (Amazon SQS) sendet. AWS SDK für Python (Boto3) Die State-Machine ist in eine AWS Lambda Funktion integriert, mit der die Datenbank nach nicht gesendeten Nachrichten durchsucht wird.

- Erstellen Sie einen Zustandsautomaten, der Nachrichtendatensätze aus einer Amazon-DynamoDB-Tabelle abruft und aktualisiert.
- Aktualisieren Sie die Definition des Zustandsautomaten, um auch Nachrichten an Amazon Simple Queue Service (Amazon SQS) zu senden.
- Starten und stoppen Sie Ausführungen des Zustandsautomaten.
- Stellen Sie vom Zustandsautomaten aus über Serviceintegrationen eine Verbindung zu Lambda, DynamoDB und Amazon SQS her.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Lambda
- Amazon SQS
- Step-Funktionen

Orchestrieren von Anwendungen mit generativer KI mit Step Functions

Im folgenden Codebeispiel wird gezeigt, wie Anwendungen mit generativer KI mit Amazon Bedrock und Step Functions erstellt und orchestriert werden.

SDK für Python (Boto3)

Das Szenario „Amazon Bedrock Serverless Prompt Chaining" zeigt, wie AWS Step Functions, Amazon Bedrock und https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html zum Erstellen und Orchestrieren komplexer und hoch skalierbarer Serverless-Anwendungen mit generativer KI verwendet werden können. Es enthält die folgenden praktischen Beispiele:

- Verfassen einer Analyse eines bestimmten Romans für einen Literaturblog. Dieses Beispiel veranschaulicht eine einfache, sequentielle Kette von Prompts.
- Generieren einer Kurzgeschichte zu einem bestimmten Thema. Dieses Beispiel veranschaulicht, wie die KI eine Liste von zuvor generierten Elementen iterativ verarbeiten kann.
- Erstellen eines Reiseplans für einen Wochenendurlaub an einem bestimmten Zielort. Dieses Beispiel veranschaulicht, wie mehrere unterschiedliche Prompts parallelisiert werden können.
- Präsentieren von Filmideen für einen menschlichen Benutzer, der als Filmproduzent fungiert. Dieses Beispiel zeigt, wie derselbe Prompt mit unterschiedlichen Inferenzparametern parallelisiert wird, wie man zu einem vorherigen Schritt in der Kette zurückkehrt und wie menschliche Eingaben in den Workflow einbezogen werden können.
- Planen einer Mahlzeit auf Grundlage der Zutaten, die der Benutzer zur Hand hat. Dieses Beispiel zeigt, wie Prompt-Chains zwei unterschiedliche KI-Konversationen beinhalten können, bei denen zwei KI-Personas miteinander debattieren, um das Endergebnis zu verbessern.
- Finde das aktuelle Repositorium mit den höchsten Trends GitHub und fasse es zusammen. Dieses Beispiel zeigt die Verkettung mehrerer KI-Agenten, die mit externen APIs interagieren.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Projekt unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Bedrock
- Amazon Bedrock Runtime
- Agenten für Amazon Bedrock

- Runtime der Agenten für Amazon Bedrock
- Step-Funktionen

AWS STS Beispiele mit SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Übernehmen Sie eine IAM-Rolle, die ein MFA-Token erfordert, und verwenden Sie temporäre Anmeldeinformationen, um Amazon-S3-Buckets für das Konto aufzulisten.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API-Details finden Sie AssumeRole in der AWS API-Referenz zum SDK für Python (Boto3).

`GetSessionToken`

Das folgende Codebeispiel zeigt die Verwendung. `GetSessionToken`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Sitzungs-Token ab, indem Sie ein MFA-Token übergeben, und verwenden Sie es, um Amazon-S3-Buckets für das Konto aufzulisten.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
```

```
    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API-Details finden Sie GetSessionToken in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Übernehmen Sie eine IAM-Rolle, die ein MFA-Token erfordert

Das folgende Code-Beispiel veranschaulicht, wie Sie eine Rolle anfügen, die ein MFA-Token erfordert.

> **Warning**
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie eine IAM-Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt.
- Erstellen Sie einen IAM-Benutzer, der nur dann berechtigt ist, die Rolle zu übernehmen, wenn MFA-Anmeldeinformationen bereitgestellt werden.
- Registrieren Sie ein MFA-Gerät für den Benutzer.
- Übernehmen der Rolle und Verwendung von temporären Anmeldeinformationen zum Auflisten von Amazon-S3-Buckets.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer, registrieren Sie ein MFA-Gerät, und erstellen Sie eine Rolle, die die Berechtigung zum Auflisten von S3-Buckets gewährt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
 MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
```

```
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": True}},
                    }
                ],
            }
        ),
```

```
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)
```

```
    return user, user_key, virtual_mfa_device, role
```

Zeigen Sie, dass die Annahme der Rolle ohne MFA-Token nicht zulässig ist.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

Übernehmen Sie die Rolle, die die Berechtigung zum Auflisten von S3-Buckets erteilt, indem Sie das erforderliche MFA-Token übergeben, und zeigen Sie, dass Buckets aufgelistet werden können.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.
```

```
    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
```

```
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
```

```
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- API-Details finden Sie AssumeRole in der AWS API-Referenz zum SDK für Python (Boto3).

Erstellen einer URL für Verbundbenutzer

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle, die nur Lesezugriff auf die Amazon-S3-Ressourcen des aktuellen Kontos gewährt.
- Rufen Sie ein Sicherheitstoken vom AWS Verbundendpunkt ab.
- Erstellen Sie eine URL, die für den Zugriff auf die Konsole mit Verbund-Anmeldeinformationen verwendet werden kann.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Rolle, die nur Lesezugriff auf die S3-Ressourcen des aktuellen Kontos gewährt.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) instance
```

```
                        that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return role
```

Rufen Sie ein Sicherheitstoken vom AWS Verbundendpunkt ab und erstellen Sie eine URL, die für den Zugriff auf die Konsole mit Verbundanmeldeinformationen verwendet werden kann.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS) that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS federation
       endpoint, includes the sign-in token for authentication, and redirects to
```

```
        the AWS Management Console with permissions defined by the role that was
        specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
granted.
                            The current user must have permission to assume the
role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
```

```
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
```

```
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- API-Details finden Sie AssumeRole in der API-Referenz für AWS SDK für Python (Boto3).

Abrufen eines Sitzungs-Tokens, das ein MFA-Token erfordert

Das folgende Code-Beispiel veranschaulicht, wie Sie ein Sitzungstoken erhalten, das ein MFA-Token erfordert.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie eigens entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie eine IAM-Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt.
- Erstellen Sie einen IAM-Benutzer, der nur dann berechtigt ist, die Rolle zu übernehmen, wenn MFA-Anmeldeinformationen bereitgestellt werden.
- Registrieren Sie ein MFA-Gerät für den Benutzer.
- Stellen Sie MFA-Anmeldeinformationen bereit, um ein Sitzungstoken abzurufen, und verwenden Sie temporäre Anmeldeinformationen, um S3-Buckets aufzulisten.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer, registrieren Sie ein MFA-Gerät und erstellen Sie eine Rolle, die dem Benutzer die Berechtigung erteilt, S3-Buckets nur aufzulisten, wenn MFA-Anmeldeinformationen verwendet werden.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
```

```
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent": True}},
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user list buckets,
"
        f"but only when MFA credentials are present."
    )
```

```
    print("Give AWS time to propagate these new resources and connections.", end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

Rufen Sie temporäre Sitzungs-Anmeldeinformationen ab, indem Sie ein MFA-Token übergeben, und verwenden Sie die Anmeldeinformationen, um S3-Buckets für das Konto aufzulisten.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Zerstören Sie die für das Demo erstellten Ressourcen.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Führen Sie dieses Szenario aus, indem Sie die zuvor definierten Funktionen verwenden.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
```

```
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- API-Details finden Sie GetSessionToken in der AWS API-Referenz zum SDK für Python (Boto3).

Support Beispiele zur Verwendung des SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Python (Boto3) with Aktionen ausführen und gängige Szenarien implementieren Support.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen

- Aktionen

Erste Schritte

Hallo Support

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von Support beginnen.

SDK für Python (Boto3)

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def hello_support(support_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Support client and count
    the available services in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param support_client: A Boto3 Support Client object.
    """
    try:
        print("Hello, AWS Support! Let's count the available Support services:")
        response = support_client.describe_services()
        print(f"There are {len(response['services'])} services available.")
    except ClientError as err:
        if err.response["Error"]["Code"] == "SubscriptionRequiredException":
            logger.info(
                "You must have a Business, Enterprise On-Ramp, or Enterprise Support
 "
```

```
                "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                "examples."
            )
        else:
            logger.error(
                "Couldn't count services. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


if __name__ == "__main__":
    hello_support(boto3.client("support"))
```

- API-Details finden Sie DescribeServices in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie verfügbare Services und Schweregrade für Fälle ab und zeigen Sie sie an.
- Erstellen Sie einen Supportfall mit einem ausgewählten Service, einer ausgewählten Kategorie und einem ausgewählten Schweregrad.
- Rufen Sie eine Liste der offenen Fälle für den aktuellen Tag ab und zeigen Sie sie an.
- Fügen Sie dem neuen Fall einen Anhangssatz und eine Mitteilung hinzu.
- Beschreiben Sie den neuen Anhang und die Mitteilung für den Fall.
- Lösen Sie den Fall.
- Rufen Sie eine Liste der gelösten Fälle für den aktuellen Tag ab und zeigen Sie sie an.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SupportCasesScenario:
    """Runs an interactive scenario that shows how to get started using AWS
 Support."""

    def __init__(self, support_wrapper):
        """
        :param support_wrapper: An object that wraps AWS Support actions.
        """
        self.support_wrapper = support_wrapper

    def display_and_select_service(self):
        """
        Lists support services and prompts the user to select one.

        :return: The support service selected by the user.
        """
        print("-" * 88)
        services_list = self.support_wrapper.describe_services("en")
        print(f"AWS Support client returned {len(services_list)} services.")
        print("Displaying first 10 services:")

        service_choices = [svc["name"] for svc in services_list[:10]]
        selected_index = q.choose(
            "Select an example support service by entering a number from the
 preceding list:",
            service_choices,
        )
        selected_service = services_list[selected_index]
        print("-" * 88)
        return selected_service

    def display_and_select_category(self, service):
        """
```

```
        Lists categories for a support service and prompts the user to select one.

        :param service: The service of the categories.
        :return: The selected category.
        """
        print("-" * 88)
        print(
            f"Available support categories for Service {service['name']}
{len(service['categories'])}:"
        )
        categories_choices = [category["name"] for category in
service["categories"]]
        selected_index = q.choose(
            "Select an example support category by entering a number from the
preceding list:",
            categories_choices,
        )
        selected_category = service["categories"][selected_index]
        print("-" * 88)
        return selected_category

    def display_and_select_severity(self):
        """
        Lists available severity levels and prompts the user to select one.

        :return: The selected severity level.
        """
        print("-" * 88)
        severity_levels_list = self.support_wrapper.describe_severity_levels("en")
        print(f"Available severity levels:")
        severity_choices = [level["name"] for level in severity_levels_list]
        selected_index = q.choose(
            "Select an example severity level by entering a number from the
preceding list:",
            severity_choices,
        )
        selected_severity = severity_levels_list[selected_index]
        print("-" * 88)
        return selected_severity

    def create_example_case(self, service, category, severity_level):
        """
        Creates an example support case with the user's selections.
```

```
        :param service: The service for the new case.
        :param category: The category for the new case.
        :param severity_level: The severity level for the new case.
        :return: The caseId of the new support case.
        """
        print("-" * 88)
        print(f"Creating new case for service {service['name']}.")
        case_id = self.support_wrapper.create_case(service, category,
severity_level)
        print(f"\tNew case created with ID {case_id}.")
        print("-" * 88)
        return case_id

    def list_open_cases(self):
        """
        List the open cases for the current day.
        """
        print("-" * 88)
        print("Let's list the open cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        open_cases = self.support_wrapper.describe_cases(start_time, end_time,
False)
        for case in open_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def create_attachment_set(self):
        """
        Create an attachment set with a sample file.

        :return: The attachment set ID of the new attachment set.
        """
        print("-" * 88)
        print("Creating attachment set with a sample file.")
        attachment_set_id = self.support_wrapper.add_attachment_to_set()
        print(f"\tNew attachment set created with ID {attachment_set_id}.")
        print("-" * 88)
        return attachment_set_id

    def add_communication(self, case_id, attachment_set_id):
        """
        Add a communication with an attachment set to the case.
```

```
        :param case_id: The ID of the case for the communication.
        :param attachment_set_id: The ID of the attachment set to
        add to the communication.
        """
        print("-" * 88)
        print(f"Adding a communication and attachment set to the case.")
        self.support_wrapper.add_communication_to_case(attachment_set_id, case_id)
        print(
            f"Added a communication and attachment set {attachment_set_id} to the
case {case_id}."
        )
        print("-" * 88)

    def list_communications(self, case_id):
        """
        List the communications associated with a case.

        :param case_id: The ID of the case.
        :return: The attachment ID of an attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attachment_id = ""
        communications =
self.support_wrapper.describe_all_case_communications(case_id)
        for communication in communications:
            print(
                f"\tCommunication created on {communication['timeCreated']} "
                f"has {len(communication['attachmentSet'])} attachments."
            )
            if len(communication["attachmentSet"]) > 0:
                attachment_id = communication["attachmentSet"][0]["attachmentId"]
        print("-" * 88)
        return attachment_id

    def describe_case_attachment(self, attachment_id):
        """
        Describe an attachment associated with a case.

        :param attachment_id: The ID of the attachment.
        """
        print("-" * 88)
        print("Let's list the communications for our case.")
        attached_file = self.support_wrapper.describe_attachment(attachment_id)
```

```
        print(f"\tAttachment includes file {attached_file}.")
        print("-" * 88)

    def resolve_case(self, case_id):
        """
        Shows how to resolve an AWS Support case by its ID.

        :param case_id: The ID of the case to resolve.
        """
        print("-" * 88)
        print(f"Resolving case with ID {case_id}.")
        case_status = self.support_wrapper.resolve_case(case_id)
        print(f"\tFinal case status is {case_status}.")
        print("-" * 88)

    def list_resolved_cases(self):
        """
        List the resolved cases for the current day.
        """
        print("-" * 88)
        print("Let's list the resolved cases for the current day.")
        start_time = str(datetime.utcnow().date())
        end_time = str(datetime.utcnow().date() + timedelta(days=1))
        resolved_cases = self.support_wrapper.describe_cases(start_time, end_time,
True)
        for case in resolved_cases:
            print(f"\tCase: {case['caseId']}: status {case['status']}.")
        print("-" * 88)

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

        print("-" * 88)
        print("Welcome to the AWS Support get started with support cases demo.")
        print("-" * 88)

        selected_service = self.display_and_select_service()
        selected_category = self.display_and_select_category(selected_service)
        selected_severity = self.display_and_select_severity()
        new_case_id = self.create_example_case(
            selected_service, selected_category, selected_severity
        )
        wait(10)
        self.list_open_cases()
```

```
        new_attachment_set_id = self.create_attachment_set()
        self.add_communication(new_case_id, new_attachment_set_id)
        new_attachment_id = self.list_communications(new_case_id)
        self.describe_case_attachment(new_attachment_id)
        self.resolve_case(new_case_id)
        wait(10)
        self.list_resolved_cases()

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = SupportCasesScenario(SupportWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die unterstützende Client-Aktionen umschließt.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a language.
```

```
        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
language.

        :param language: The language for support severity levels.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
```

```
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does not
exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a support
case.",
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
```

```
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id


    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise


    def describe_all_case_communications(self, case_id):
```

```
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator = self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return attached_file


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        else:
            return final_status


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.

        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
```

```
            cases = filter(lambda case: case["status"] == "resolved", cases)
        return cases
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

Aktionen

`AddAttachmentsToSet`

Das folgende Codebeispiel zeigt die Verwendung`AddAttachmentsToSet`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
```

```
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def add_attachment_to_set(self):
        """
        Add an attachment to a set, or create a new attachment set if one does not
exist.

        :return: The attachment set ID.
        """
        try:
            response = self.support_client.add_attachments_to_set(
                attachments=[
                    {
                        "fileName": "attachment_file.txt",
                        "data": b"This is a sample file for attachment to a support
case.",
                    }
                ]
            )
            new_set_id = response["attachmentSetId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add attachment. Here's why: %s: %s",
```

```
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return new_set_id
```

- API-Details finden Sie AddAttachmentsToSet in der AWS API-Referenz zum SDK für Python (Boto3).

`AddCommunicationToCase`

Das folgende Codebeispiel zeigt die Verwendung. `AddCommunicationToCase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)
```

```
    def add_communication_to_case(self, attachment_set_id, case_id):
        """
        Add a communication and an attachment set to a case.

        :param attachment_set_id: The ID of an existing attachment set.
        :param case_id: The ID of the case.
        """
        try:
            self.support_client.add_communication_to_case(
                caseId=case_id,
                communicationBody="This is an example communication added to a
support case.",
                attachmentSetId=attachment_set_id,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't add communication. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
```

- API-Details finden Sie AddCommunicationToCase in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateCase`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def create_case(self, service, category, severity):
        """
        Create a new support case.

        :param service: The service to use for the new case.
        :param category: The category to use for the new case.
        :param severity: The severity to use for the new case.
        :return: The caseId of the new case.
        """
        try:
            response = self.support_client.create_case(
                subject="Example case for testing, ignore.",
                serviceCode=service["code"],
                severityCode=severity["code"],
                categoryCode=category["code"],
                communicationBody="Example support case body.",
```

```
                language="en",
                issueType="customer-service",
            )
            case_id = response["caseId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't create case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return case_id
```

- API-Details finden Sie CreateCase in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeAttachment`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeAttachment`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""
```

```
    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_attachment(self, attachment_id):
        """
        Get information about an attachment by its attachmentID.

        :param attachment_id: The ID of the attachment.
        :return: The name of the attached file.
        """
        try:
            response = self.support_client.describe_attachment(
                attachmentId=attachment_id
            )
            attached_file = response["attachment"]["fileName"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get attachment description. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
```

```
            return attached_file
```

- API-Details finden Sie DescribeAttachment in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeCases`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeCases`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_cases(self, after_time, before_time, resolved):
        """
        Describe support cases over a period of time, optionally filtering
        by status.
```

```
        :param after_time: The start time to include for cases.
        :param before_time: The end time to include for cases.
        :param resolved: True to include resolved cases in the results,
            otherwise results are open cases.
        :return: The final status of the case.
        """
        try:
            cases = []
            paginator = self.support_client.get_paginator("describe_cases")
            for page in paginator.paginate(
                afterTime=after_time,
                beforeTime=before_time,
                includeResolvedCases=resolved,
                language="en",
            ):
                cases += page["cases"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe cases. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            if resolved:
                cases = filter(lambda case: case["status"] == "resolved", cases)
            return cases
```

- API-Details finden Sie DescribeCases in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeCommunications`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeCommunications`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_all_case_communications(self, case_id):
        """
        Describe all the communications for a case using a paginator.

        :param case_id: The ID of the case.
        :return: The communications for the case.
        """
        try:
            communications = []
            paginator = self.support_client.get_paginator("describe_communications")
            for page in paginator.paginate(caseId=case_id):
                communications += page["communications"]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
 subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't describe communications. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return communications
```

- API-Details finden Sie DescribeCommunications in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeServices`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeServices`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
```

```
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_services(self, language):
        """
        Get the descriptions of AWS services available for support for a language.

        :param language: The language for support services.
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of AWS service descriptions.
        """
        try:
            response = self.support_client.describe_services(language=language)
            services = response["services"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get Support services for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return services
```

- API-Details finden Sie DescribeServices in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeSeverityLevels`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeSeverityLevels`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def describe_severity_levels(self, language):
        """
        Get the descriptions of available severity levels for support cases for a
 language.

        :param language: The language for support severity levels.
```

```
        Currently, only "en" (English) and "ja" (Japanese) are supported.
        :return: The list of severity levels.
        """
        try:
            response =
self.support_client.describe_severity_levels(language=language)
            severity_levels = response["severityLevels"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
Support "
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't get severity levels for language %s. Here's why: %s:
%s",
                    language,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return severity_levels
```

- API-Details finden Sie DescribeSeverityLevels in der AWS API-Referenz zum SDK für Python (Boto3).

`ResolveCase`

Das folgende Codebeispiel zeigt die Verwendung. `ResolveCase`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SupportWrapper:
    """Encapsulates Support actions."""

    def __init__(self, support_client):
        """
        :param support_client: A Boto3 Support client.
        """
        self.support_client = support_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        support_client = boto3.client("support")
        return cls(support_client)


    def resolve_case(self, case_id):
        """
        Resolve a support case by its caseId.

        :param case_id: The ID of the case to resolve.
        :return: The final status of the case.
        """
        try:
            response = self.support_client.resolve_case(caseId=case_id)
            final_status = response["finalCaseStatus"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "SubscriptionRequiredException":
                logger.info(
                    "You must have a Business, Enterprise On-Ramp, or Enterprise
 Support "
```

```
                    "plan to use the AWS Support API. \n\tPlease upgrade your
subscription to run these "
                    "examples."
                )
            else:
                logger.error(
                    "Couldn't resolve case. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return final_status
```

- API-Details finden Sie ResolveCase in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Systems Manager unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Python (Boto3)

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo Systems Manager

Das folgende Codebeispiel zeigt die ersten Schritte mit Systems Manager.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import boto3
from botocore.exceptions import ClientError


def hello_systems_manager(ssm_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS Systems Manager
    client and list the first 5 documents in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ssm_client: A Boto3 AWS Systems Manager Client object. This object wraps
                           the low-level AWS Systems Manager service API.
    """
    print("Hello, AWS Systems Manager! Let's list some of your documents:\n")

    paginator = ssm_client.get_paginator("list_documents")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 5})
    for page in page_iterator:
        for document in page["DocumentIdentifiers"]:
            print(f"  {document['Name']}")


if __name__ == "__main__":
    try:
        hello_systems_manager(boto3.client("ssm"))
    except ClientError as err:
        print("Hello systems manager had an error.")
        print(err.response["Error"]["Code"])
```

```
        print(err.response["Error"]["Message"])
```

- API-Details finden Sie ListDocuments in der AWS API-Referenz zum SDK für Python (Boto3).

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Wartungsfenster.
- Ändern Sie den Zeitplan des Wartungsfensters.
- Erstellen Sie ein Dokument.
- Senden Sie einen Befehl an eine bestimmte EC2-Instance.
- Erstellen Sie eine. OpsItem
- Aktualisieren und lösen Sie das OpsItem.
- Löschen Sie das Wartungsfenster OpsItem und das Dokument.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario an einem Prompt aus.

```
class SystemsManagerScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Systems Manager."""

    def __init__(self, document_wrapper, maintenance_window_wrapper,
 ops_item_wrapper):
        """
        :param document_wrapper: An object that wraps Systems Manager document
 functions.
```

```
        :param maintenance_window_wrapper: An object that wraps Systems Manager
 maintenance window functions.
        :param ops_item_wrapper: An object that wraps Systems Manager OpsItem
 functions.
        """
        self.document_wrapper = document_wrapper
        self.maintenance_window_wrapper = maintenance_window_wrapper
        self.ops_item_wrapper = ops_item_wrapper

    def run(self):
        """Demonstrates how to use the AWS SDK for Python (Boto3) to get started
 with Systems Manager."""
        try:
            print("-" * 88)
            print(
                """
Welcome to the AWS Systems Manager SDK Getting Started scenario.
This program demonstrates how to interact with Systems Manager using the AWS SDK for
 Python (Boto3).
Systems Manager is the operations hub for your AWS applications and resources and a
 secure end-to-end management
solution. The program's primary functions include creating a maintenance window,
 creating a document, sending a
command to a document, listing documents, listing commands, creating an OpsItem,
 modifying an OpsItem, and deleting
Systems Manager resources. Upon completion of the program, all AWS resources are
 cleaned up.
Let's get started..."""
            )
            q.ask("Please hit Enter")

            print("-" * 88)
            print("Create a Systems Manager maintenance window.")
            maintenance_window_name = q.ask(
                "Please enter the maintenance window name (default is ssm-
maintenance-window):",
            )
            if not maintenance_window_name:
                maintenance_window_name = "ssm-maintenance-window"

            self.maintenance_window_wrapper.create(
                name=maintenance_window_name,
                schedule="cron(0 10 ? * MON-FRI *)",
                duration=2,
```

```
                cutoff=1,
                allow_unassociated_targets=True,
            )

            print("-" * 88)
            print("Modify the maintenance window by changing the schedule")
            q.ask("Please hit Enter")

            self.maintenance_window_wrapper.update(
                name=maintenance_window_name,
                schedule="cron(0 0 ? * MON *)",
                duration=24,
                cutoff=1,
                allow_unassociated_targets=True,
                enabled=True,
            )

            print("-" * 88)
            print(
                "Create a document that defines the actions that Systems Manager
 performs on your EC2 instance."
            )
            document_name = q.ask(
                "Please enter the document name (default is ssmdocument):"
            )

            if not document_name:
                document_name = "ssmdocument"

            self.document_wrapper.create(
                name=document_name,
                content="""
{
    "schemaVersion": "2.2",
    "description": "Run a simple shell command",
    "mainSteps": [
        {
            "action": "aws:runShellScript",
            "name": "runEchoCommand",
            "inputs": {
              "runCommand": [
                "echo 'Hello, world!'"
              ]
            }
```

```
        }
    ]
}
            """,
            )

            self.document_wrapper.wait_until_active()

            print(
                """
Now you have the option of running a command on an EC2 instance that echoes 'Hello,
 world!'.
In order to run this command, you must provide the instance ID of a Linux EC2
 instance. If you do
not already have a running Linux EC2 instance in your account, you can create one
 using the AWS console.
For information about creating an EC2 instance, see
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-instance-wizard.html.
            """
            )

            if q.ask(
                "Would you like to run a command on an EC2 instance? (y/n)",
                q.is_yesno,
            ):
                instance_id = q.ask(
                    "Please enter the instance ID of the EC2 instance:", q.non_empty
                )
                command_id = self.document_wrapper.send_command(
                    instance_ids=[instance_id]
                )

                self.document_wrapper.wait_command_executed(
                    command_id=command_id, instance_id=instance_id
                )

                print("-" * 88)
                print(
                    "Lets get the time when the specific command was sent to the
 specific managed node"
                )
                q.ask("Please hit Enter")
```

```
 self.document_wrapper.list_command_invocations(instance_id=instance_id)

            print("-" * 88)
            print("-" * 88)
            print(
                """
Now we will create a  Systems Manager OpsItem.
An OpsItem is a feature provided by the Systems Manager service.
It is a type of operational data item that allows you to manage and track various
 operational issues,
events, or tasks within your AWS environment.

You can create OpsItems to track and manage operational issues as they arise.
For example, you could create an OpsItem whenever your application detects a
 critical error
or an anomaly in your infrastructure.
            """
            )
            q.ask("Please hit Enter")

            self.ops_item_wrapper.create(
                title="Disk Space Alert",
                description="Created by the Systems Manager Python (Boto3) API",
                source="EC2",
                category="Performance",
                severity="2",
            )

            print("-" * 88)
            print("-" * 88)
            print(f"Now we will update  the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(
                title="Disk Space Alert",
                description=f"An update to {self.ops_item_wrapper.id}",
            )

            print(
                f"Now we will get the status of the OpsItem
 {self.ops_item_wrapper.id}"
            )
            q.ask("Please hit Enter")
```

```
            # It may take a second for the ops item to be available
            counter = 0
            while not self.ops_item_wrapper.describe() and counter < 5:
                counter += 1
                time.sleep(1)

            print(f"Now we will resolve the OpsItem {self.ops_item_wrapper.id}")
            q.ask("Please hit Enter")

            self.ops_item_wrapper.update(status="Resolved")

            print("-" * 88)
            print("-" * 88)
            if q.ask(
                "Would you like to delete the Systems Manager resources? (y/n)",
                q.is_yesno,
            ):
                print("You selected to delete the resources.")
                self.cleanup()
            else:
                print("The Systems Manager resources will not be deleted")

            print("-" * 88)
            print("This concludes the Systems Manager SDK Getting Started
 scenario.")
            print("-" * 88)

        except Exception:
            self.cleanup()
            raise

    def cleanup(self):
        self.maintenance_window_wrapper.delete()
        self.ops_item_wrapper.delete()
        self.document_wrapper.delete()


if __name__ == "__main__":
    try:
        scenario = SystemsManagerScenario(
            DocumentWrapper.from_client(),
            MaintenanceWindowWrapper.from_client(),
            OpsItemWrapper.from_client(),
```

```
        )
        scenario.run()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Definieren Sie eine Klasse, die Dokument- und Befehlsaktionen umschließt.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
TimeoutSeconds=3600
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def describe(self):
        """
        Describes the document.

        :return: Document status.
        """
        try:
            response = self.ssm_client.describe_document(Name=self.name)
            return response["Document"]["Status"]
        except ClientError as err:
            logger.error(
                "Couldn't get %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def wait_until_active(self, max_attempts=20, delay=5):
        """
        Waits until the document is active.

        :param max_attempts: The maximum number of attempts for checking the status.
        :param delay: The delay in seconds between each check.
        """
        attempt = 0
        status = ""
        while attempt <= max_attempts:
            status = self.describe()
            if status == "Active":
                break
            attempt += 1
            time.sleep(delay)

        if status != "Active":
            logger.error("Document is not active.")
        else:
```

```
            logger.info("Document is active.")

    def wait_command_executed(self, command_id, instance_id):
        """
        Waits until the command is executed on the instance.

        :param command_id: The ID of the command.
        :param instance_id: The ID of the instance.
        """

        waiter = self.ssm_client.get_waiter("command_executed")
        waiter.wait(CommandId=command_id, InstanceId=instance_id)

    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Definieren Sie eine Klasse, die Ops-Objekt-Aktionen umschließt.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
```

```
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator": "Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
                print(
                    f"The item title is {item['Title']} and the status is
{item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(self, title=None, description=None, status=None):
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
```

```
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Definieren Sie eine Klasse, die Wartungsfenster-Aktionen umschließt.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff, allow_unassociated_targets):
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
```

```
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
            return

        try:
```

```
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update(
        self, name, enabled, schedule, duration, cutoff, allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on managed
nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
```

```
            logger.error(
                "Parameter validation error when trying to update maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

Aktionen

`CreateDocument`

Das folgende Codebeispiel zeigt die Verwendung`CreateDocument`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, content, name):
        """
        Creates a document.

        :param content: The content of the document.
        :param name: The name of the document.
        """
        try:
            self.ssm_client.create_document(
                Name=name, Content=content, DocumentType="Command"
            )
            self.name = name
        except self.ssm_client.exceptions.DocumentAlreadyExists:
            print(f"Document {name} already exists.")
            self.name = name
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateDocument in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `CreateMaintenanceWindow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, name, schedule, duration, cutoff, allow_unassociated_targets):
```

```
        """
        Create an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            response = self.ssm_client.create_maintenance_window(
                Name=name,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.window_id = response["WindowId"]
            self.name = name
            logger.info("Created maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to create maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create maintenance window %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateMaintenanceWindow in der AWS API-Referenz zum SDK für Python (Boto3).

`CreateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung. `CreateOpsItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def create(self, title, source, category, severity, description):
        """
        Create an OpsItem

        :param title: The OpsItem title.
        :param source: The OpsItem source.
        :param category: The OpsItem category.
        :param severity: The OpsItem severity.
```

```
        :param description: The OpsItem description.

        """
        try:
            response = self.ssm_client.create_ops_item(
                Title=title,
                Source=source,
                Category=category,
                Severity=severity,
                Description=description,
            )
            self.id = response["OpsItemId"]
        except self.ssm_client.exceptions.OpsItemLimitExceededException as err:
            logger.error(
                "Couldn't create ops item because you have exceeded your open
OpsItem limit. "
                "Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't create ops item %s. Here's why: %s: %s",
                title,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie CreateOpsItem in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteDocument`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteDocument`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Deletes an AWS Systems Manager document.
        """
        if self.name is None:
            return

        try:
            self.ssm_client.delete_document(Name=self.name)
            print(f"Deleted document {self.name}.")
            self.name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
```

- API-Details finden Sie DeleteDocument in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteMaintenanceWindow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the associated AWS Systems Manager maintenance window.
        """
        if self.window_id is None:
```

```
            return

        try:
            self.ssm_client.delete_maintenance_window(WindowId=self.window_id)
            logger.info("Deleted maintenance window %s.", self.window_id)
            print(f"Deleted maintenance window {self.name}")
            self.window_id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete maintenance window %s. Here's why: %s: %s",
                self.window_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteMaintenanceWindow in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteOpsItem`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteOpsItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
```

```
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def delete(self):
        """
        Delete the OpsItem.
        """
        if self.id is None:
            return
        try:
            self.ssm_client.delete_ops_item(OpsItemId=self.id)
            print(f"Deleted ops item with id {self.id}")
            self.id = None
        except ClientError as err:
            logger.error(
                "Couldn't delete ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DeleteOpsItem in der AWS API-Referenz zum SDK für Python (Boto3).

`DescribeOpsItems`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeOpsItems`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def describe(self):
        """
        Describe an OpsItem.
        """
        try:
            paginator = self.ssm_client.get_paginator("describe_ops_items")
            ops_items = []
            for page in paginator.paginate(
                OpsItemFilters=[
                    {"Key": "OpsItemId", "Values": [self.id], "Operator": "Equal"}
                ]
            ):
                ops_items.extend(page["OpsItemSummaries"])

            for item in ops_items:
```

```
                print(
                    f"The item title is {item['Title']} and the status is
 {item['Status']}"
                )
            return len(ops_items) > 0
        except ClientError as err:
            logger.error(
                "Couldn't describe ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie DescribeOpsItems in der AWS API-Referenz zum SDK für Python (Boto3).

`ListCommandInvocations`

Das folgende Codebeispiel zeigt die Verwendung. `ListCommandInvocations`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None
```

```
    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def list_command_invocations(self, instance_id):
        """
        Lists the commands for an instance.

        :param instance_id: The ID of the instance.
        :return: The list of commands.
        """
        try:
            paginator = self.ssm_client.get_paginator("list_command_invocations")
            command_invocations = []
            for page in paginator.paginate(InstanceId=instance_id):
                command_invocations.extend(page["CommandInvocations"])
            num_of_commands = len(command_invocations)
            print(
                f"{num_of_commands} command invocation(s) found for instance
{instance_id}."
            )

            if num_of_commands > 10:
                print("Displaying the first 10 commands:")
                num_of_commands = 10
            date_format = "%A, %d %B %Y %I:%M%p"
            for command in command_invocations[:num_of_commands]:
                print(
                    f"   The time of command invocation is
{command['RequestedDateTime'].strftime(date_format)}"
                )
        except ClientError as err:
            logger.error(
                "Couldn't list commands for %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie ListCommandInvocations in der AWS API-Referenz für SDK für Python (Boto3).

SendCommand

Das folgende Codebeispiel zeigt die Verwendung. SendCommand

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DocumentWrapper:
    """Encapsulates AWS Systems Manager Document actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def send_command(self, instance_ids):
        """
        Sends a command to one or more instances.

        :param instance_ids: The IDs of the instances to send the command to.
        :return: The ID of the command.
        """
        try:
            response = self.ssm_client.send_command(
                InstanceIds=instance_ids, DocumentName=self.name,
 TimeoutSeconds=3600
```

```
            )
            return response["Command"]["CommandId"]
        except ClientError as err:
            logger.error(
                "Couldn't send command to %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie SendCommand in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateMaintenanceWindow`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class MaintenanceWindowWrapper:
    """Encapsulates AWS Systems Manager maintenance window actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.window_id = None
        self.name = None

    @classmethod
    def from_client(cls):
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)
```

```
    def update(
        self, name, enabled, schedule, duration, cutoff, allow_unassociated_targets
    ):
        """
        Update an AWS Systems Manager maintenance window.

        :param name: The name of the maintenance window.
        :param enabled: Whether the maintenance window is enabled to run on managed
nodes.
        :param schedule: The schedule of the maintenance window.
        :param duration: The duration of the maintenance window.
        :param cutoff: The cutoff time of the maintenance window.
        :param allow_unassociated_targets: Allow the maintenance window to run on
managed nodes, even
                                           if you haven't registered those nodes as
targets.
        """
        try:
            self.ssm_client.update_maintenance_window(
                WindowId=self.window_id,
                Name=name,
                Enabled=enabled,
                Schedule=schedule,
                Duration=duration,
                Cutoff=cutoff,
                AllowUnassociatedTargets=allow_unassociated_targets,
            )
            self.name = name
            logger.info("Updated maintenance window %s.", self.window_id)
        except ParamValidationError as error:
            logger.error(
                "Parameter validation error when trying to update maintenance window
%s. Here's why: %s",
                self.window_id,
                error,
            )
            raise
        except ClientError as err:
            logger.error(
                "Couldn't update maintenance window %s. Here's why: %s: %s",
                self.name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UpdateMaintenanceWindow in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateOpsItem`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class OpsItemWrapper:
    """Encapsulates AWS Systems Manager OpsItem actions."""

    def __init__(self, ssm_client):
        """
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.id = None

    @classmethod
    def from_client(cls):
        """
        :return: A OpsItemWrapper instance.
        """
        ssm_client = boto3.client("ssm")
        return cls(ssm_client)


    def update(self, title=None, description=None, status=None):
```

```
        """
        Update an OpsItem.

        :param title: The new OpsItem title.
        :param description: The new OpsItem description.
        :param status: The new OpsItem status.
        :return:
        """
        args = dict(OpsItemId=self.id)
        if title is not None:
            args["Title"] = title
        if description is not None:
            args["Description"] = description
        if status is not None:
            args["Status"] = status
        try:
            self.ssm_client.update_ops_item(**args)
        except ClientError as err:
            logger.error(
                "Couldn't update ops item %s. Here's why: %s: %s",
                self.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- API-Details finden Sie UpdateOpsItem in der AWS API-Referenz zum SDK für Python (Boto3).

Beispiele für Amazon Textract unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Textract Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AnalyzeDocument`

Das folgende Codebeispiel zeigt die Verwendung`AnalyzeDocument`.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def analyze_file(
        self, feature_types, *, document_file_name=None, document_bytes=None
    ):
```

```
        """
        Detects text and additional elements, such as forms or tables, in a local
image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={"Bytes": document_bytes}, FeatureTypes=feature_types
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- API-Details finden Sie AnalyzeDocument in der AWS API-Referenz zum SDK für Python (Boto3).

`DetectDocumentText`

Das folgende Codebeispiel zeigt die Verwendung. `DetectDocumentText`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
                 that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, "rb") as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.detect_document_text(
                Document={"Bytes": document_bytes}
            )
            logger.info("Detected %s blocks.", len(response["Blocks"]))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- API-Details finden Sie DetectDocumentText in der AWS API-Referenz zum SDK für Python (Boto3).

GetDocumentAnalysis

Das folgende Codebeispiel zeigt die Verwendung. `GetDocumentAnalysis`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                 detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(JobId=job_id)
            job_status = response["JobStatus"]
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
```

```
            return response
```

- API-Details finden Sie GetDocumentAnalysis in der AWS API-Referenz zum SDK für Python (Boto3).

`StartDocumentAnalysis`

Das folgende Codebeispiel zeigt die Verwendung. `StartDocumentAnalysis`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie einen asynchronen Auftrag zum Analysieren eines Dokuments.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_analysis_job(
        self,
        bucket_name,
        document_file_name,
        feature_types,
        sns_topic_arn,
```

```
        sns_role_arn,
    ):
        """
        Starts an asynchronous job to detect text and additional elements, such as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in Amazon
S3.
        :param feature_types: The types of additional document features to detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
                              where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name": document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
                FeatureTypes=feature_types,
            )
            job_id = response["JobId"]
            logger.info(
                "Started text analysis job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- API-Details finden Sie StartDocumentAnalysis in der AWS API-Referenz zum SDK für Python (Boto3).

`StartDocumentTextDetection`

Das folgende Codebeispiel zeigt die Verwendung. `StartDocumentTextDetection`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie einen asynchronen Auftrag zum Erkennen von Text in Dokumenten.

```
class TextractWrapper:
    """Encapsulates Textract functions."""

    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource


    def start_detection_job(
        self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn
    ):
        """
        Starts an asynchronous job to detect text elements in an image stored in an
        Amazon S3 bucket. Textract publishes a notification to the specified Amazon
 SNS
        topic when the job completes.
```

```
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
image.
        :param document_file_name: The name of the document image stored in Amazon
S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
                              where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
                             role that can be assumed by Textract and grants
permission
                             to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    "S3Object": {"Bucket": bucket_name, "Name": document_file_name}
                },
                NotificationChannel={
                    "SNSTopicArn": sns_topic_arn,
                    "RoleArn": sns_role_arn,
                },
            )
            job_id = response["JobId"]
            logger.info(
                "Started text detection job %s on %s.", job_id, document_file_name
            )
        except ClientError:
            logger.exception("Couldn't detect text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- API-Details finden Sie StartDocumentTextDetection in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Codebeispiele zeigen, wie die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersucht werden kann.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) zusammen mit Amazon Textract verwendet wird, um Text-, Formular- und Tabellenelemente in einem Dokumentbild zu erkennen. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.
- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Entitäten in Text erkennen, der aus einem Bild extrahiert wurde

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter. TextractAndComprehendNotebook.ipynb

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Beispiele für Amazon Transcribe unter Verwendung von SDK für Python (Boto3)

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Python (Boto3)

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

- Szenarien

Aktionen

`CreateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `CreateVocabulary`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
```

```
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- API-Details finden Sie CreateVocabulary in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteTranscriptionJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- API-Details finden Sie DeleteTranscriptionJob in der AWS API-Referenz zum SDK für Python (Boto3).

`DeleteVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteVocabulary`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- API-Details finden Sie DeleteVocabulary in der AWS API-Referenz zum SDK für Python (Boto3).

`GetTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung. `GetTranscriptionJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- API-Details finden Sie GetTranscriptionJob in der AWS API-Referenz zum SDK für Python (Boto3).

`GetVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `GetVocabulary`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- API-Details finden Sie GetVocabulary in der AWS API-Referenz zum SDK für Python (Boto3).

`ListTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListTranscriptionJobs`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in their
                       names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """
    try:
        response =
 transcribe_client.list_transcription_jobs(JobNameContains=job_filter)
        jobs = response["TranscriptionJobSummaries"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token
            )
            jobs += response["TranscriptionJobSummaries"]
            next_token = response.get("NextToken")
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- API-Details finden Sie ListTranscriptionJobs in der AWS API-Referenz zum SDK für Python (Boto3).

`ListVocabularies`

Das folgende Codebeispiel zeigt die Verwendung. `ListVocabularies`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs
```

- API-Details finden Sie ListVocabularies in der AWS API-Referenz zum SDK für Python (Boto3).

StartTranscriptionJob

Das folgende Codebeispiel zeigt die Verwendung. `StartTranscriptionJob`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
```

```
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job
```

- API-Details finden Sie StartTranscriptionJob in der AWS API-Referenz zum SDK für Python (Boto3).

`UpdateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `UpdateVocabulary`

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.
```

```
    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
        raise
```

- API-Details finden Sie UpdateVocabulary in der AWS API-Referenz zum SDK für Python (Boto3).

Szenarien

Erstellen und Verfeinern eines benutzerdefinierten Vokabulars

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Laden Sie eine Audiodatei auf Amazon S3 hoch.
- Führen Sie einen Amazon Transcribe-Auftrag aus, um die Datei zu transkribieren und die Ergebnisse zu erhalten.
- Erstellen und verfeinern Sie ein benutzerdefiniertes Vokabular, um die Transkriptionsgenauigkeit zu verbessern.
- Führen Sie Aufträge mit benutzerdefinierten Vokabularen aus und erhalten Sie die Ergebnisse.

SDK für Python (Boto3)

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Transkribieren Sie eine Audiodatei, die eine Lesung von Jabberwocky von Lewis Carroll enthält. Beginnen Sie damit, Funktionen zu erstellen, die Amazon Transcribe-Aktionen wrappen.

```
def start_job(
    job_name,
    media_uri,
    media_format,
    language_code,
    transcribe_client,
    vocabulary_name=None,
):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when transcribing
                            the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            "TranscriptionJobName": job_name,
            "Media": {"MediaFileUri": media_uri},
            "MediaFormat": media_format,
            "LanguageCode": language_code,
```

```
        }
        if vocabulary_name is not None:
            job_args["Settings"] = {"VocabularyName": vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response["TranscriptionJob"]
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job



def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name
        )
        job = response["TranscriptionJob"]
        logger.info("Got job %s.", job["TranscriptionJobName"])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job



def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
```

```
    try:
        transcribe_client.delete_transcription_job(TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise



def create_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
                          For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response



def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None, table_uri=None
):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    """
    try:
        vocab_args = {"VocabularyName": vocabulary_name, "LanguageCode":
 language_code}
        if phrases is not None:
            vocab_args["Phrases"] = phrases
        elif table_uri is not None:
            vocab_args["VocabularyFileUri"] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info("Updated custom vocabulary %s.", response["VocabularyName"])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
```

```
        raise



def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string in
                              their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response =
 transcribe_client.list_vocabularies(NameContains=vocabulary_filter)
        vocabs = response["Vocabularies"]
        next_token = response.get("NextToken")
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token
            )
            vocabs += response["Vocabularies"]
            next_token = response.get("NextToken")
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter
        )
    except ClientError:
        logger.exception(
            "Couldn't list vocabularies with filter %s.", vocabulary_filter
        )
        raise
    else:
        return vocabs



def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
```

```
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Rufen Sie die Wrapper-Funktionen auf, um Audio ohne ein benutzerdefiniertes Vokabular und anschließend mit verschiedenen Versionen eines benutzerdefinierten Vokabulars zu transkribieren, um bessere Ergebnisse zu erzielen.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    s3_resource = boto3.resource("s3")
    transcribe_client = boto3.client("transcribe")

    print("-" * 88)
    print("Welcome to the Amazon Transcribe demo!")
    print("-" * 88)

    bucket_name = f"jabber-bucket-{time.time_ns()}"
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": transcribe_client.meta.region_name
        },
    )
    media_file_name = ".media/Jabberwocky.mp3"
    media_object_key = "Jabberwocky.mp3"
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f"s3://{bucket.name}/{media_object_key}"

    job_name_simple = f"Jabber-{time.time_ns()}"
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple,
```

```
        f"s3://{bucket_name}/{media_object_key}",
        "mp3",
        "en-US",
        transcribe_client,
    )
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Creating a custom vocabulary that lists the nonsense words to try to "
        "improve the transcription."
    )
    vocabulary_name = f"Jabber-vocabulary-{time.time_ns()}"
    create_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        phrases=[
            "brillig",
            "slithy",
            "borogoves",
            "mome",
            "raths",
            "Jub-Jub",
            "frumious",
            "manxome",
            "Tumtum",
            "uffish",
            "whiffling",
            "tulgey",
            "thou",
            "frabjous",
            "callooh",
            "callay",
            "chortled",
        ],
    )
```

```
    vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocabulary_list = f"Jabber-vocabulary-list-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocabulary_list}.")
    start_job(
        job_name_vocabulary_list,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
        vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocabulary_list)
    job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
    transcript_vocabulary_list = requests.get(
        job_vocabulary_list["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
    print(transcript_vocabulary_list["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print(
        "Updating the custom vocabulary with table data that provides additional "
        "pronunciation hints."
    )
    table_vocab_file = "jabber-vocabulary-table.txt"
    bucket.upload_file(table_vocab_file, table_vocab_file)
    update_vocabulary(
        vocabulary_name,
        "en-US",
        transcribe_client,
        table_uri=f"s3://{bucket.name}/{table_vocab_file}",
    )
    vocabulary_ready_waiter.wait(vocabulary_name)

    job_name_vocab_table = f"Jabber-vocab-table-{time.time_ns()}"
    print(f"Starting transcription job {job_name_vocab_table}.")
    start_job(
        job_name_vocab_table,
        media_uri,
        "mp3",
        "en-US",
        transcribe_client,
```

```
        vocabulary_name=vocabulary_name,
    )
    transcribe_waiter.wait(job_name_vocab_table)
    job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
    transcript_vocab_table = requests.get(
        job_vocab_table["Transcript"]["TranscriptFileUri"]
    ).json()
    print(f"Transcript for job {transcript_vocab_table['jobName']}:")
    print(transcript_vocab_table["results"]["transcripts"][0]["transcript"])

    print("-" * 88)
    print("Getting data for jobs and vocabularies.")
    jabber_jobs = list_jobs("Jabber", transcribe_client)
    print(f"Found {len(jabber_jobs)} jobs:")
    for job_sum in jabber_jobs:
        job = get_job(job_sum["TranscriptionJobName"], transcribe_client)
        print(
            f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
            f"{job['Settings'].get('VocabularyName')}"
        )

    jabber_vocabs = list_vocabularies("Jabber", transcribe_client)
    print(f"Found {len(jabber_vocabs)} vocabularies:")
    for vocab_sum in jabber_vocabs:
        vocab = get_vocabulary(vocab_sum["VocabularyName"], transcribe_client)
        vocab_content = requests.get(vocab["DownloadUri"]).text
        print(f"\t{vocab['VocabularyName']} contents:")
        print(vocab_content)

    print("-" * 88)
    print("Deleting demo jobs.")
    for job_name in [job_name_simple, job_name_vocabulary_list,
job_name_vocab_table]:
        delete_job(job_name, transcribe_client)
    print("Deleting demo vocabulary.")
    delete_vocabulary(vocabulary_name, transcribe_client)
    print("Deleting demo bucket.")
    bucket.objects.delete()
    bucket.delete()
    print("Thanks for watching!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
  - CreateVocabulary
  - DeleteTranscriptionJob
  - DeleteVocabulary
  - GetTranscriptionJob
  - GetVocabulary
  - ListVocabularies
  - StartTranscriptionJob
  - UpdateVocabulary

Audio transkribieren und Auftragsdaten abrufen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen Transkriptionsauftrag mit Amazon Transcribe.
- Warten Sie, bis der -Auftrag abgeschlossen wurde.
- Ermitteln Sie die URI, unter der das Transkript gespeichert ist.

Weitere Informationen finden Sie unter Erste Schritte mit Amazon Transcribe.

SDK für Python (Boto3)

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import time
import boto3


def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
```

```
        Media={"MediaFileUri": file_uri},
        MediaFormat="wav",
        LanguageCode="en-US",
    )

    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job = transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job["TranscriptionJob"]["TranscriptionJobStatus"]
        if job_status in ["COMPLETED", "FAILED"]:
            print(f"Job {job_name} is {job_status}.")
            if job_status == "COMPLETED":
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']
['TranscriptFileUri']}."
                )
            break
        else:
            print(f"Waiting for {job_name}. Current status is {job_status}.")
        time.sleep(10)


def main():
    transcribe_client = boto3.client("transcribe")
    file_uri = "s3://test-transcribe/answer2.wav"
    transcribe_file("Example-job", file_uri, transcribe_client)


if __name__ == "__main__":
    main()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Python (Boto3).
  - GetTranscriptionJob
  - StartTranscriptionJob

Codebeispiele für SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für Ruby with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK for Ruby Developer Guide — Mehr über die Verwendung von Ruby mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- Aurora-Beispiele unter Verwendung von SDK für Ruby
- Auto-Scaling-Beispiele unter Verwendung von SDK für Ruby
- CloudTrail Beispiele zur Verwendung des SDK für Ruby
- CloudWatch Beispiele zur Verwendung des SDK für Ruby
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Ruby
- Beispiele für Amazon Comprehend unter Verwendung von SDK für Ruby
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Ruby
- DynamoDB-Beispiele unter Verwendung von SDK für Ruby
- Beispiele für Amazon EC2 unter Verwendung von SDK für Ruby
- Beispiele für Elastic Beanstalk unter Verwendung von SDK für Ruby

- EventBridge Beispiele zur Verwendung des SDK für Ruby
- AWS Glue Beispiele zur Verwendung des SDK für Ruby
- IAM-Beispiele unter Verwendung von SDK für Ruby
- Kinesis-Beispiele unter Verwendung von SDK für Ruby
- AWS KMS Beispiele mit SDK für Ruby
- Lambda-Beispiele unter Verwendung von SDK für Ruby
- Beispiele für Amazon MSK unter Verwendung von SDK für Ruby
- Beispiele für Amazon Polly unter Verwendung von SDK für Ruby
- Beispiele für Amazon RDS unter Verwendung von SDK für Ruby
- Beispiele für Amazon S3 unter Verwendung von SDK für Ruby
- Beispiele für Amazon SES unter Verwendung von SDK für Ruby
- Beispiele für Amazon SES API v2 unter Verwendung von SDK für Ruby
- Beispiele für Amazon SNS unter Verwendung von SDK für Ruby
- Beispiele für Amazon SQS unter Verwendung von SDK für Ruby
- AWS STS Beispiele mit SDK für Ruby
- Beispiele für Amazon Textract unter Verwendung von SDKs für Ruby
- Codebeispiele für Amazon Translate unter Verwendung von SDKs für Ruby

Aurora-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Aurora Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)

# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Weitere API-Informationen finden Sie unter DescribeDBClusters in der AWS SDK für Ruby - API-Referenz.

Auto-Scaling-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Auto Scaling Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
```

```
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
        @logger.info("  Min/max/desired:       #{group.min_size}/#{group.max_size}/
#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK für Ruby API-Referenz.

CloudTrail Beispiele zur Verwendung des SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren CloudTrail.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateTrail`

Das folgende Codebeispiel zeigt die Verwendung`CreateTrail`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
require 'aws-sdk-sts'

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
 bucket_name)
  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      'Version' => '2012-10-17',
      'Statement' => [
        {
          'Sid' => 'AWSCloudTrailAclCheck20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:GetBucketAcl',
          'Resource' => "arn:aws:s3:::#{bucket_name}"
        },
        {
          'Sid' => 'AWSCloudTrailWrite20150319',
          'Effect' => 'Allow',
          'Principal' => {
            'Service' => 'cloudtrail.amazonaws.com'
          },
          'Action' => 's3:PutObject',
          'Resource' => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
          'Condition' => {
            'StringEquals' => {
```

```
              's3:x-amz-acl' => 'bucket-owner-full-control'
            }
          }
        }
      ]
    }.to_json

    s3_client.put_bucket_policy(
      bucket: bucket_name,
      policy: policy
    )
    puts "Successfully added policy to bucket #{bucket_name}"
  end

  begin
    cloudtrail_client.create_trail({
                                     name: trail_name, # required
                                     s3_bucket_name: bucket_name # required
                                   })

    puts "Successfully created trail: #{trail_name}."
  rescue StandardError => e
    puts "Got error trying to create trail #{trail_name}:\n #{e}"
    puts e
    exit 1
  end
```

- API-Details finden Sie CreateTrail in der AWS SDK für Ruby API-Referenz.

`DeleteTrail`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTrail`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  client.delete_trail({
                        name: trail_name # required
                      })
  puts "Successfully deleted trail: #{trail_name}"
rescue StandardError => e
  puts "Got error trying to delete trail: #{trail_name}:"
  puts e
  exit 1
end
```

- API-Details finden Sie DeleteTrail in der AWS SDK für Ruby API-Referenz.

`ListTrails`

Das folgende Codebeispiel zeigt die Verwendung`ListTrails`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."

  resp.trail_list.each do |trail|
    puts "Name:           #{trail.name}"
    puts "S3 bucket name: #{trail.s3_bucket_name}"
    puts
  end
```

- API-Details finden Sie ListTrails in der AWS SDK für Ruby API-Referenz.

`LookupEvents`

Das folgende Codebeispiel zeigt die Verwendung`LookupEvents`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'

# @param [Object] client
def lookup_events_example(client)
  resp = client.lookup_events
  puts "Found #{resp.events.count} events:"
  resp.events.each do |e|
    puts "Event name:   #{e.event_name}"
    puts "Event ID:     #{e.event_id}"
    puts "Event time:   #{e.event_time}"
    puts 'Resources:'

    e.resources.each do |r|
      puts "  Name:       #{r.resource_name}"
      puts "  Type:       #{r.resource_type}"
      puts ''
    end
  end
end
```

- API-Details finden Sie LookupEvents in der AWS SDK für Ruby API-Referenz.

CloudWatch Beispiele zur Verwendung des SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudwatch'

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts 'No alarms found.'
  end
```

```
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- API-Details finden Sie DescribeAlarms in der AWS SDK für Ruby API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts '-' * 16
      puts "Name:           #{alarm.alarm_name}"
      puts "State value:    #{alarm.state_value}"
      puts "State reason:   #{alarm.state_reason}"
      puts "Metric:         #{alarm.metric_name}"
      puts "Namespace:      #{alarm.namespace}"
      puts "Statistic:      #{alarm.statistic}"
      puts "Period:         #{alarm.period}"
      puts "Unit:           #{alarm.unit}"
      puts "Eval. periods:  #{alarm.evaluation_periods}"
      puts "Threshold:      #{alarm.threshold}"
```

```
      puts "Comp. operator: #{alarm.comparison_operator}"

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts 'OK actions:'
        alarm.ok_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
        puts 'Alarm actions:'
        alarm.alarm_actions.each do |a|
          puts "  #{a}"
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
         alarm.insufficient_data_actions.count.positive?
        puts 'Insufficient data actions:'
        alarm.insufficient_data_actions.each do |a|
          puts "  #{a}"
        end
      end

      puts 'Dimensions:'
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: #{d.name}, Value: #{d.value}"
        end
      else
        puts '  None for this alarm.'
      end
    end
  else
    puts 'No alarms found.'
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ''
```

```
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby cw-ruby-example-show-alarms.rb REGION'
    puts 'Example: ruby cw-ruby-example-show-alarms.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts 'Available alarms:'
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK für Ruby API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
```

```
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

# Example usage:
def run_me
  alarm_name = 'ObjectsInBucket'
  alarm_description = 'Objects exist in this bucket for more than 1 day.'
  metric_name = 'NumberOfObjects'
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ['arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic']
  namespace = 'AWS/S3'
  statistic = 'Average'
  dimensions = [
    {
      name: "BucketName",
      value: "amzn-s3-demo-bucket"
    },
    {
      name: 'StorageType',
      value: 'AllStorageTypes'
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = 'Count'
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = 'GreaterThanThreshold' # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
```

```
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK für Ruby API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '    Dimensions:'
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      'Note that it could take up to 15 minutes for recently-added metrics ' \
      'to become available.'
  end
end

# Example usage:
def run_me
```

```
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisitors',
    'SiteName',
    'example.com',
    5_885.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'UniqueVisits',
    'SiteName',
    'example.com',
    8_628.0,
    'Count'
  )

  puts 'Continuing...' unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    'PageViews',
    'PageURL',
    'example.html',
    18_057.0,
    'Count'
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie ListMetrics in der AWS SDK für Ruby API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
```

```
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'amzn-s3-demo-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
```

```
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  false
end
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK für Ruby API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cloudwatch'

# Adds a datapoint to a metric in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
```

```
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  false
end
```

- API-Details finden Sie PutMetricData in der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
```

```
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- API-Details finden Sie ListUserPools in der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Comprehend unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end
```

```
def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

DynamoDB-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-dynamodb'
require 'logger'

# DynamoDBManager is a class responsible for managing DynamoDB operations
# such as listing all tables in the current AWS account.
class DynamoDBManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all DynamoDB tables in the current AWS account.
  def list_tables
    @logger.info('Here are the DynamoDB tables in your account:')

    paginator = @client.list_tables(limit: 10)
    table_names = []

    paginator.each_page do |page|
      page.table_names.each do |table_name|
        @logger.info("- #{table_name}")
        table_names << table_name
      end
    end

    if table_names.empty?
      @logger.info("You don't have any DynamoDB tables in your account.")
```

```
    else
      @logger.info("\nFound #{table_names.length} tables.")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  dynamodb_client = Aws::DynamoDB::Client.new
  manager = DynamoDBManager.new(dynamodb_client)
  manager.list_tables
end
```

- API-Details finden Sie ListTables in der AWS SDK für Ruby API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die eine DynamoDB-Tabelle enthält.

```
  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

Erstellen Sie eine Helper-Funktion zum Herunterladen und Extrahieren der JSON-Beispieldatei.

```
  # Gets sample movie data, either from a local file or by first downloading it from
  # the Amazon DynamoDB Developer Guide.
  #
  # @param movie_file_name [String] The local file name where the movie data is
 stored in JSON format.
  # @return [Hash] The movie data as a Hash.
  def fetch_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
      @logger.debug("Downloading #{movie_file_name}...")
      movie_content = URI.open(
        'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip'
      )
      movie_json = ''
```

```
      Zip::File.open_buffer(movie_content) do |zip|
        zip.each do |entry|
          movie_json = entry.get_input_stream.read
        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
  rescue StandardError => e
    puts("Failure downloading movie data:\n#{e}")
    raise
  end
```

Führen Sie ein interaktives Szenario aus, um die Tabelle zu erstellen und Aktionen darauf auszuführen.

```
  table_name = "doc-example-table-movies-#{rand(10**4)}"
  scaffold = Scaffold.new(table_name)
  dynamodb_wrapper = DynamoDBBasics.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end

  new_step(2, 'Add a new record to the DynamoDB table.')
  my_movie = {}
  my_movie[:title] = CLI::UI::Prompt.ask('Enter the title of a movie to add to the
table. E.g. The Matrix')
  my_movie[:year] = CLI::UI::Prompt.ask('What year was it released? E.g. 1989').to_i
  my_movie[:rating] = CLI::UI::Prompt.ask('On a scale of 1 - 10, how do you rate it?
E.g. 7').to_i
  my_movie[:plot] = CLI::UI::Prompt.ask('Enter a brief summary of the plot. E.g. A
man awakens to a new reality.')
  dynamodb_wrapper.add_item(my_movie)
  puts("\nNew record added:")
  puts JSON.pretty_generate(my_movie).green
```

```
 print "Done!\n".green

 new_step(3, 'Update a record in the DynamoDB table.')
 my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
 response = dynamodb_wrapper.update_item(my_movie)
 puts("Updated '#{my_movie[:title]}' with new attributes:")
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(4, 'Get a record from the DynamoDB table.')
 puts("Searching for #{my_movie[:title]} (#{my_movie[:year]})...")
 response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
 puts JSON.pretty_generate(response).green
 print "Done!\n".green

 new_step(5, 'Write a batch of items into the DynamoDB table.')
 download_file = 'moviedata.json'
 puts("Downloading movie database to #{download_file}...")
 movie_data = scaffold.fetch_movie_data(download_file)
 puts("Writing movie data from #{download_file} into your table...")
 scaffold.write_batch(movie_data)
 puts("Records added: #{movie_data.length}.")
 print "Done!\n".green

 new_step(5, 'Query for a batch of items by key.')
 loop do
   release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
   results = dynamodb_wrapper.query_items(release_year)
   if results.any?
     puts("There were #{results.length} movies released in #{release_year}:")
     results.each do |movie|
       print "\t #{movie['title']}".green
     end
     break
   else
     continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
     break unless continue.eql?('y')
   end
 end
 print "\nDone!\n".green
```

```
  new_step(6, 'Scan for a batch of items using a filter expression.')
  years = {}
  years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
 2018:')
  years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
  releases = dynamodb_wrapper.scan_items(years)
  if !releases.empty?
    puts("Found #{releases.length} movies.")
    count = Question.ask(
      'How many do you want to see? ', method(:is_int), in_range(1, releases.length)
    )
    puts("Here are your #{count} movies:")
    releases.take(count).each do |release|
      puts("\t#{release['title']}")
    end
  else
    puts("I don't know about any movies released between #{years[:start]} "\
         "and #{years[:end]}.")
  end
  print "\nDone!\n".green

  new_step(7, 'Delete an item from the DynamoDB table.')
  answer = CLI::UI::Prompt.ask("Do you want to remove '#{my_movie[:title]}'? (y/n)
 ")
  if answer.eql?('y')
    dynamodb_wrapper.delete_item(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    print "\nDone!\n".green
  end

  new_step(8, 'Delete the DynamoDB table.')
  answer = CLI::UI::Prompt.ask('Delete the table? (y/n)')
  if answer.eql?('y')
    scaffold.delete_table
    puts("Deleted #{table_name}.")
  else
    puts("Don't forget to delete the table when you're done!")
  end
  print "\nThanks for watching!\n".green
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo.')
rescue Errno::ENOENT
  true
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`BatchExecuteStatement`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Lesen Sie einen Stapel von Elementen mithilfe von PartiQL.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
```

```
    @table = @dynamodb.table(table_name)
  end

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

Löschen Sie mithilfe von PartiQL einen Stapel von Elementen.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
  def batch_execute_write(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für Ruby API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Fills an Amazon DynamoDB table with the specified data. Items are sent in
  # batches of 25 until all items are written.
  #
  # @param movies [Enumerable] The data to put in the table. Each item must contain
 at least
  #                            the keys required by the schema that was specified
 when the
  #                            table was created.
  def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
      movie_items = []
      movies[index, slice_size].each do |movie|
        movie_items.append({ put_request: { item: movie } })
      end
      @dynamo_resource.client.batch_write_item({ request_items: { @table.name =>
 movie_items } })
      index += slice_size
```

```
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:"
    )
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie BatchWriteItem in der AWS SDK für Ruby API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
```

```
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        { attribute_name: 'year', key_type: 'HASH' }, # Partition key
        { attribute_name: 'title', key_type: 'RANGE' } # Sort key
      ],
      attribute_definitions: [
        { attribute_name: 'year', attribute_type: 'N' },
        { attribute_name: 'title', attribute_type: 'S' }
      ],
      billing_mode: 'PAY_PER_REQUEST'
    )
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
    @table
  rescue Aws::DynamoDB::Errors::ServiceError => e
    @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie CreateTable in der AWS SDK für Ruby API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
```

```
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
  # @param year [Integer] The release year of the movie to delete.
  def delete_item(title, year)
    @table.delete_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie DeleteItem in der AWS SDK für Ruby API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
```

```
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie DeleteTable in der AWS SDK für Ruby API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end
```

```
  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie DescribeTable in der AWS SDK für Ruby API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wählen Sie ein einzelnes Element mithilfe von PartiQL aus.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
```

```
  end

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Aktualisieren Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Updates a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def update_rating_by_title(title, year, rating)
    request = {
      statement: "UPDATE \"#{@table.name}\" SET info.rating=? WHERE title=? and
 year=?",
      parameters: [{ "N": rating }, title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Fügen Sie ein einzelnes Element mithilfe von PartiQL hinzu.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Adds a single record to a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @param plot [String] The plot of the movie.
  # @param rating [Float] The new rating to assign the title.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def insert_item(title, year, plot, rating)
    request = {
      statement: "INSERT INTO \"#{@table.name}\" VALUE {'title': ?, 'year': ?,
 'info': ?}",
      parameters: [title, year, { 'plot': plot, 'rating': rating }]
    }
    @dynamodb.client.execute_statement(request)
  end
```

Löschen Sie ein einzelnes Element mithilfe von PartiQL.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamodb = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamodb.table(table_name)
  end

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
```

```
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }
    @dynamodb.client.execute_statement(request)
  end
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für Ruby API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: { 'year' => year, 'title' => title })
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
```

```
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie GetItem in der AWS SDK für Ruby API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Finden Sie heraus, ob eine Tabelle vorhanden ist.

```
# Encapsulates an Amazon DynamoDB table of movie data.
class Scaffold
  attr_reader :dynamo_resource, :table_name, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table_name = table_name
    @table = nil
    @logger = Logger.new($stdout)
    @logger.level = Logger::DEBUG
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    @dynamo_resource.client.describe_table(table_name: table_name)
    @logger.debug("Table #{table_name} exists")
```

```
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
    @logger.debug("Table #{table_name} doesn't exist")
    false
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie ListTables in der AWS SDK für Ruby API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        'year' => movie[:year],
        'title' => movie[:title],
```

```
        'info' => { 'plot' => movie[:plot], 'rating' => movie[:rating] }
      }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie PutItem in der AWS SDK für Ruby API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: '#yr = :year',
      expression_attribute_names: { '#yr' => 'year' },
```

```
      expression_attribute_values: { ':year' => year }
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't query for movies released in #{year}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.items
  end
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für Ruby -API-Referenz.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
```

```
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
    done = false
    start_key = nil
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für Ruby -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class DynamoDBBasics
```

```
  attr_reader :dynamo_resource, :table

  def initialize(table_name)
    client = Aws::DynamoDB::Client.new(region: 'us-east-1')
    @dynamo_resource = Aws::DynamoDB::Resource.new(client: client)
    @table = @dynamo_resource.table(table_name)
  end

  # Updates rating and plot data for a movie in the table.
  #
  # @param movie [Hash] The title, year, plot, rating of the movie.
  def update_item(movie)
    response = @table.update_item(
      key: { 'year' => movie[:year], 'title' => movie[:title] },
      update_expression: 'set info.rating=:r',
      expression_attribute_values: { ':r' => movie[:rating] },
      return_values: 'UPDATED_NEW'
    )
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't update movie #{movie[:title]} (#{movie[:year]}) in table
#{@table.name}\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    response.attributes
  end
```

- API-Details finden Sie UpdateItem in der AWS SDK für Ruby API-Referenz.

Szenarien

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, in dem eine Tabelle erstellt wird und PartiQL-Stapelabfragen ausgeführt werden.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLBatch.new(table_name)

new_step(1, 'Create a new DynamoDB table if none already exists.')
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, 'Populate DynamoDB table with movie data.')
download_file = 'moviedata.json'
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, 'Select a batch of items from the movies table.')
puts "Let's select some popular movies for side-by-side comparison."
response = sdk.batch_execute_select([['Mean Girls', 2004], ['Goodfellas', 1977],
['The Prancing of the Lambs', 2005]])
puts("Items selected: #{response['responses'].length}\n")
print "\nDone!\n".green

new_step(4, 'Delete a batch of items from the movies table.')
sdk.batch_execute_write([['Mean Girls', 2004], ['Goodfellas', 1977], ['The
Prancing of the Lambs', 2005]])
print "\nDone!\n".green
```

```
  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- API-Details finden Sie BatchExecuteStatement in der AWS SDK für Ruby API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein Szenario aus, das eine Tabelle erstellt und PartiQL-Abfragen ausführt.

```
  table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
  scaffold = Scaffold.new(table_name)
  sdk = DynamoDBPartiQLSingle.new(table_name)

  new_step(1, 'Create a new DynamoDB table if none already exists.')
  unless scaffold.exists?(table_name)
    puts("\nNo such table: #{table_name}. Creating it...")
    scaffold.create_table(table_name)
    print "Done!\n".green
  end
```

```
  new_step(2, 'Populate DynamoDB table with movie data.')
  download_file = 'moviedata.json'
  puts("Downloading movie database to #{download_file}...")
  movie_data = scaffold.fetch_movie_data(download_file)
  puts("Writing movie data from #{download_file} into your table...")
  scaffold.write_batch(movie_data)
  puts("Records added: #{movie_data.length}.")
  print "Done!\n".green

  new_step(3, 'Select a single item from the movies table.')
  response = sdk.select_item_by_title('Star Wars')
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print response.items.first.to_s.yellow
  print "\n\nDone!\n".green

  new_step(4, 'Update a single item from the movies table.')
  puts "Let's correct the rating on The Big Lebowski to 10.0."
  sdk.update_rating_by_title('The Big Lebowski', 1998, 10.0)
  print "\nDone!\n".green

  new_step(5, 'Delete a single item from the movies table.')
  puts "Let's delete The Silence of the Lambs because it's just too scary."
  sdk.delete_item_by_title('The Silence of the Lambs', 1991)
  print "\nDone!\n".green

  new_step(6, 'Insert a new item into the movies table.')
  puts "Let's create a less-scary movie called The Prancing of the Lambs."
  sdk.insert_item('The Prancing of the Lambs', 2005, 'A movie about happy
 livestock.', 5.0)
  print "\nDone!\n".green

  new_step(7, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- API-Details finden Sie ExecuteStatement in der AWS SDK für Ruby API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet

die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Beispiele für Amazon EC2 unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon EC2 beginnen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
```

```
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
    instances = []

    paginator.each_page do |page|
      page.reservations.each do |reservation|
        reservation.instances.each do |instance|
          instances << instance
        end
      end
    end

    instances
  end

  # Prints details of the given EC2 instances.
  #
  # @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
 print.
  def print_instances(instances)
    instances.each do |instance|
      @logger.info("Instance ID: #{instance.instance_id}")
      @logger.info("Instance Type: #{instance.instance_type}")
      @logger.info("Public IP: #{instance.public_ip_address}")
      @logger.info("Public DNS Name: #{instance.public_dns_name}")
      @logger.info("\n")
    end
  end
end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
```

```
end
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK für Ruby API-Referenz.

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- API-Details finden Sie AllocateAddress in der AWS SDK für Ruby API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The assocation ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- API-Details finden Sie AssociateAddress in der AWS SDK für Ruby API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  true
```

```
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
```

```
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
```

```
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'

  puts '-' * 10
  puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.delete(filename)
  if File.exist?(filename)
    puts "Could not delete file at '#{filename}'. You must delete it yourself."
  else
    puts 'File deleted.'
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after deleting this key pair...'
  describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateKeyPair in der AWS SDK für Ruby API-Referenz.

`CreateRouteTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateRouteTable`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'
```

```
# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
```

```
    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
```

```
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if route_table_created_and_associated?(
    ec2_resource,
    vpc_id,
    subnet_id,
    gateway_id,
    destination_cidr_block,
    tag_key,
    tag_value
  )
    puts 'Route table created and associated.'
  else
    puts 'Route table not created or not associated.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateRouteTable in der AWS SDK für Ruby API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
```

```
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
```

```
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol " \
    "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
    "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
 perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
 perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
  print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

  print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

  print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
 perm.ip_ranges.count.positive?
  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups
```

```
  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:        #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?

  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
```

```
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http, to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
 description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
 from_port_http, to_port_http, cidr_ip_range_http)
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh, from_port_ssh,
 to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
 security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
```

```
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
  security_group_id = create_security_group(ec2_client, group_name, description,
 vpc_id)
  puts 'Could not create security group. Skipping this step.' if security_group_id
 == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
 to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
 ip_protocol, from_port, to_port,
                                               cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
end

def display_help
  puts 'Usage:   ruby ec2-ruby-example-security-group.rb ' \
    'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
    'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
    'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
    "my-security-group 'This is my security group.' vpc-6713dfEX " \
    "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp', '80',
 '80',
```

```
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK für Ruby API-Referenz.

`CreateSubnet`

Das folgende Codebeispiel zeigt die Verwendung`CreateSubnet`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
```

```
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
```

```
availability_zone = ''
tag_key = ''
tag_value = ''
region = ''
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage:   ruby ec2-ruby-example-create-subnet.rb ' \
    'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
    'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-6713dfEX'
  cidr_block = '10.0.0.0/24'
  availability_zone = 'us-west-2a'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  cidr_block = ARGV[1]
  availability_zone = ARGV[2]
  tag_key = ARGV[3]
  tag_value = ARGV[4]
  region = ARGV[5]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  puts 'Subnet created and tagged.'
else
  puts 'Subnet not created or not tagged.'
```

```
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateSubnet in der AWS SDK für Ruby API-Referenz.

`CreateVpc`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpc`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Creates a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
```

```
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
```

```
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
    tag_value
  )
    puts 'VPC created and tagged.'
  else
    puts 'VPC not created or not tagged.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateVpc in der AWS SDK für Ruby API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
```

```
# @example
#   list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
  ec2_resource = Aws::EC2::Resource.new(region: region)
  list_instance_ids_states(ec2_resource)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie DescribeInstances in der AWS SDK für Ruby API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print '  '
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
```

```
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print '  Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print "  State\n"
  print '-' * max_region_string_length
  print '  '
  print '-' * max_zone_string_length
  print '  '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print '  '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print '  '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts '  Messages for this zone:'
      zone.messages.each do |message|
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end
```

```
# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
 '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie DescribeRegions in der AWS SDK für Ruby API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- API-Details finden Sie ReleaseAddress in der AWS SDK für Ruby API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
      puts 'The instance is already running.'
      return true
    when 'terminated'
      puts 'Error starting instance: ' \
        'the instance is terminated, so you cannot start it.'
      return false
    end
  end

  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance started.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end
```

```
# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_started?(ec2_client, instance_id)

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie StartInstances in der AWS SDK für Ruby API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end
```

```
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie StopInstances in der AWS SDK für Ruby API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive? &&
     response.instance_statuses[0].instance_state.name == 'terminated'

    puts 'The instance is already terminated.'
    return true
  end

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
```

```
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:   ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)...'
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie TerminateInstances in der AWS SDK für Ruby API-Referenz.

Beispiele für Elastic Beanstalk unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Elastic Beanstalk Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeApplications`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeApplications`

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
```

```
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

  # Logs application details
  def log_application_details(application)
    @logger.info("Name:        #{application.application_name}")
    @logger.info("Description: #{application.description}")
  end

  # Lists and logs details of environments for a given application
  def list_environments(application_name)
    @eb_client.describe_environments(application_name:
 application_name).environments.each do |env|
      @logger.info("  Environment:  #{env.environment_name}")
      @logger.info("    URL:        #{env.cname}")
      @logger.info("    Health:     #{env.health}")
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Error listing environments for application #{application_name}:
 #{e.message}")
  end
end
```

- API-Details finden Sie DescribeApplications in der AWS SDK für Ruby API-Referenz.

`ListAvailableSolutionStacks`

Das folgende Codebeispiel zeigt die Verwendung`ListAvailableSolutionStacks`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end

  # Lists and logs Elastic Beanstalk solution stacks
  def list_stacks
    stacks = @eb_client.list_available_solution_stacks.solution_stacks
    orig_length = stacks.length
    filtered_length = 0

    stacks.each do |stack|
      if @filter.empty? || stack.downcase.include?(@filter)
        @logger.info(stack)
        filtered_length += 1
      end
    end

    log_summary(filtered_length, orig_length)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Error listing solution stacks: #{e.message}")
  end

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- API-Details finden Sie ListAvailableSolutionStacks in der AWS SDK für Ruby API-Referenz.

`UpdateApplication`

Das folgende Codebeispiel zeigt die Verwendung`UpdateApplication`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
```

```
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end

  # Uploads the ZIP file to the S3 bucket
  def upload_zip_to_s3(zip_file_name)
    zip_contents = File.read(zip_file_name)
    key = "#{@app_name}/#{zip_file_name}"
    @s3_client.put_object(body: zip_contents, bucket: fetch_bucket_name, key: key)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to upload ZIP file to S3: #{e.message}")
  end

  # Fetches the S3 bucket name from Elastic Beanstalk application versions
  def fetch_bucket_name
    app_versions = @eb_client.describe_application_versions(application_name:
@app_name)
    av = app_versions.application_versions.first
    av.source_bundle.s3_bucket
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch bucket name: #{e.message}")
    raise
  end

  # Creates a new application version and deploys it
  def create_and_deploy_new_application_version(zip_file_name)
    version_label = File.basename(zip_file_name, '.zip')
    @eb_client.create_application_version(
      process: false,
      application_name: @app_name,
      version_label: version_label,
      source_bundle: {
        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end
```

```
  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

  # Fetches the environment name of the application
  def fetch_environment_name
    envs = @eb_client.describe_environments(application_name: @app_name)
    envs.environments.first.environment_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to fetch environment name: #{e.message}")
    raise
  end
end
```

- API-Details finden Sie UpdateApplication in der AWS SDK für Ruby API-Referenz.

EventBridge Beispiele zur Verwendung des SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren EventBridge.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen und Auslösen einer Regel

Das folgende Codebeispiel zeigt, wie eine Regel in Amazon erstellt und ausgelöst wird EventBridge.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Funktionen in der richtigen Reihenfolge auf.

```
require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'
```

Überprüfen Sie, ob das angegebene Amazon Simple Notification Service (Amazon SNS)-Thema unter den für diese Funktion bereitgestellten Themen vorhanden ist.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
```

```
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Überprüfen Sie, ob das angegebene Thema unter den für den Aufrufer in Amazon SNS verfügbaren Themen vorhanden ist.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts 'Topic found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.topics.count.positive?

      if topic_found?(response.topics, topic_arn)
        puts 'Topic found.'
        return true
      end
    end
  end
  puts 'Topic not found.'
```

```
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Erstellen Sie ein Thema in Amazon SNS und abonnieren Sie dann eine E-Mail-Adresse, um Benachrichtigungen zu diesem Thema zu erhalten.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: 'email',
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts 'Subscription created with ARN ' \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    'and confirm the subscription to start receiving notification emails.'
  topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  'Error'
end
```

Prüfen Sie, ob die angegebene AWS Identity and Access Management (IAM-) Rolle zu den Rollen gehört, die für diese Funktion bereitgestellt wurden.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  false
end
```

Überprüfen Sie, ob die angegebene Rolle unter den für den Aufrufer in IAM verfügbaren Rollen vorhanden ist.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
```

```
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

      if role_found?(response.roles, role_arn)
        puts 'Role found.'
        return true
      end
    end
  end
  puts 'Role not found.'
  false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  false
end
```

Erstellen Sie eine Rolle in IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
```

```
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
    role_name: role_name
  )
  puts 'Access policy added to role.'
  response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts 'If the role was created, you must add the access policy ' \
    'to the role yourself, or delete the role yourself and try again.'
```

```
  'Error'
end
```

Überprüft, ob die angegebene EventBridge Regel zu den Regeln gehört, die für diese Funktion bereitgestellt wurden.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  false
end
```

Überprüft, ob die angegebene Regel zu den Regeln gehört, die dem Anrufer zur Verfügung stehen. EventBridge

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
```

```
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

      if rule_found?(response.rules, rule_name)
        puts 'Rule found.'
        return true
      end
    end
  end
  puts 'Rule not found.'
  false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  false
end
```

Erstellen Sie eine Regel in EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
```

```
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(
  cloudwatchevents_client,
  rule_name,
  rule_description,
  instance_state,
  role_arn,
  target_id,
  topic_arn
)
  puts "Creating rule with name '#{rule_name}'..."
  put_rule_response = cloudwatchevents_client.put_rule(
    name: rule_name,
    description: rule_description,
    event_pattern: {
      'source': [
        'aws.ec2'
      ],
      'detail-type': [
        'EC2 Instance State-change Notification'
      ],
      'detail': {
        'state': [
          instance_state
        ]
      }
    }.to_json,
```

```
    state: 'ENABLED',
    role_arn: role_arn
  )
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
      {
        id: target_id,
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
     put_targets_response.failed_entry_count.positive?
    puts 'Error(s) adding target to rule:'
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    false
  else
    true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts 'If the rule was created, you must add the target ' \
    'to the rule yourself, or delete the rule yourself and try again.'
  false
end
```

Prüfen Sie, ob die angegebene Protokollgruppe zu den Protokollgruppen gehört, die dem Anrufer in Amazon CloudWatch Logs zur Verfügung stehen.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
```

```
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

Erstellen Sie eine Protokollgruppe in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts 'Log group created.'
  true
```

```
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  false
end
```

Schreiben Sie ein Ereignis in einen Log-Stream in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @param log_stream_name [String] The name of the log stream within
#   the log group.
# @param message [String] The message to write to the log stream.
# @param sequence_token [String] If available, the sequence token from the
#   message that was written immediately before this message. This sequence
#   token is returned by Amazon CloudWatch Logs whenever you programmatically
#   write a message to the log stream.
# @return [String] The sequence token that is returned by
#   Amazon CloudWatch Logs after successfully writing the message to the
#   log stream.
# @example
#   puts log_event(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',
#     "Instance 'i-033c48ef067af3dEX' restarted.",
#     '49542672486831074009579604567656788214806863282469607 3EX'
#   )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
```

```
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end
```

Starten Sie eine Amazon Elastic Compute Cloud (Amazon EC2) -Instance neu und fügen Sie Informationen über die zugehörige Aktivität zu einem Log-Stream in CloudWatch Logs hinzu.

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
```

```
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
  )
  sequence_token = ''

  puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    'This might take a few minutes...'
  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts 'Instance stopped.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
  )

  puts 'Attempting to restart the instance. This might take a few minutes...'
  ec2_client.start_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
  puts 'Instance restarted.'
  sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
  )
```

```
  true
rescue StandardError => e
  puts 'Error creating log stream or stopping or restarting the instance: ' \
    "#{e.message}"
  log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Error stopping or starting instance '#{instance_id}': #{e.message}",
    sequence_token
  )
  false
end
```

Zeigt Informationen zur Aktivität für eine Regel in an EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
```

```
  end_time,
  period
)
  puts 'Attempting to display rule activity...'
  response = cloudwatch_client.get_metric_statistics(
    namespace: 'AWS/Events',
    metric_name: 'Invocations',
    dimensions: [
      {
        name: 'RuleName',
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ['Sum'],
    unit: 'Count'
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      'specified time period.'
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Zeigt Protokollinformationen für alle Log-Streams in einer CloudWatch Logs-Log-Gruppe an.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
```

```
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts 'Attempting to display log stream data for the log group ' \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: 'LastEventTime',
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
     describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts '-' * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
         get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts 'No log messages for this log stream.'
      end
    end
  end
rescue StandardError => e
  puts 'Error getting information about the log streams or their messages: ' \
    "#{e.message}"
end
```

Erinnern Sie den Anrufer daran, alle zugehörigen AWS Ressourcen, die er nicht mehr benötigt, manuell zu bereinigen.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts '-' * 10
  puts 'Some of the following AWS resources might still exist in your account.'
  puts 'If you no longer want to use this code example, then to clean up'
  puts 'your AWS account and avoid unexpected costs, you might want to'
  puts 'manually delete any of the following resources if they exist:'
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - PutEvents
  - PutRule

AWS Glue Beispiele zur Verwendung des SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für Ruby

> **Note**
>
> Da ist noch mehr dran GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-glue'
require 'logger'

# GlueManager is a class responsible for managing AWS Glue operations
# such as listing all Glue jobs in the current AWS account.
class GlueManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
```

```
  end

  # Lists and prints all Glue jobs in the current AWS account.
  def list_jobs
    @logger.info('Here are the Glue jobs in your account:')

    paginator = @client.get_jobs(max_results: 10)
    jobs = []

    paginator.each_page do |page|
      jobs.concat(page.jobs)
    end

    if jobs.empty?
      @logger.info("You don't have any Glue jobs.")
    else
      jobs.each do |job|
        @logger.info("- #{job.name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  glue_client = Aws::Glue::Client.new
  manager = GlueManager.new(glue_client)
  manager.list_jobs
end
```

- API-Details finden Sie ListJobs in der AWS SDK für Ruby API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog

- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Klasse, die die im Szenario verwendeten AWS Glue Funktionen umschließt.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
```

```
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
```

```
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
```

```
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
```

```
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

Erstellen Sie eine Klasse, die das Szenario ausführt.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end
```

```
  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end

  private

  def setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    new_step(1, 'Create a crawler')
    crawler = wrapper.get_crawler(crawler_name)
    unless crawler
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
db_prefix, data_source)
      puts "Successfully created #{crawler_name}."
    end
    wrapper.start_crawler(crawler_name)
    monitor_crawler(wrapper, crawler_name)
  end

  def monitor_crawler(wrapper, crawler_name)
    new_step(2, 'Monitor Crawler')
    crawler_state = nil
    until crawler_state == 'READY'
      custom_wait(15)
      crawler = wrapper.get_crawler(crawler_name)
      crawler_state = crawler[0]['state']
      print "Crawler status: #{crawler_state}".yellow
    end
  end

  def query_database(wrapper, _crawler_name, db_name)
    new_step(3, 'Query the database.')
    wrapper.get_database(db_name)
    puts "The crawler created database #{db_name}:"
    puts "Database contains tables: #{wrapper.get_tables(db_name).map { |t|
t['name'] }}"
  end

  def create_and_run_job(wrapper, job_script, job_name, db_name)
    new_step(4, 'Create and run job.')
    wrapper.upload_job_script(job_script, @glue_bucket)
```

```
    wrapper.create_job(job_name, 'ETL Job', @glue_service_role.arn, "s3://
#{@glue_bucket.name}/#{job_script}")
    run_job(wrapper, job_name, db_name)
  end

  def run_job(wrapper, job_name, db_name)
    new_step(5, 'Run the job.')
    wrapper.start_job_run(job_name, db_name, wrapper.get_tables(db_name)[0]['name'],
 @glue_bucket.name)
    job_run_status = nil
    until %w[SUCCEEDED FAILED STOPPED].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]['job_run_state']
      print "Job #{job_name} status: #{job_run_status}".yellow
    end
  end
end

def main
  banner('../../helpers/banner.txt')
  puts 'Starting AWS Glue demo...'

  # Load resource names from YAML.
  resource_names = YAML.load_file('resource_names.yaml')

  # Setup services and resources.
  iam_role = Aws::IAM::Resource.new(region: 'us-
east-1').role(resource_names['glue_service_role'])
  s3_bucket = Aws::S3::Resource.new(region: 'us-
east-1').bucket(resource_names['glue_bucket'])

  # Instantiate scenario and run.
  scenario = GlueCrawlerJobScenario.new(Aws::Glue::Client.new(region: 'us-east-1'),
 iam_role, s3_bucket, @logger)
  random_suffix = rand(10**4)
  scenario.run("crawler-#{random_suffix}", "db-#{random_suffix}", "prefix-
#{random_suffix}-", 's3://data_source',
               'job_script.py', "job-#{random_suffix}")

  puts 'Demo complete.'
end
```

Erstellen Sie ein ETL-Skript, das von AWS Glue zum Extrahieren, Transformieren und Laden von Daten während der Auftragsausführung verwendet wird.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
 to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
```

```
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler

- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
```

```
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, _db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie CreateCrawler in der AWS SDK für Ruby API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
      },
      glue_version: '3.0'
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie CreateJob in der AWS SDK für Ruby API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie DeleteCrawler in der AWS SDK für Ruby API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- API-Details finden Sie DeleteDatabase in der AWS SDK für Ruby API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- API-Details finden Sie DeleteJob in der AWS SDK für Ruby API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- API-Details finden Sie DeleteTable in der AWS SDK für Ruby API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil if
 not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie GetCrawler in der AWS SDK für Ruby API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
 if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get database #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie GetDatabase in der AWS SDK für Ruby API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- API-Details finden Sie GetJobRun in der AWS SDK für Ruby API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- API-Details finden Sie GetJobRuns in der AWS SDK für Ruby API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie GetTables in der AWS SDK für Ruby API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie ListJobs in der AWS SDK für Ruby API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie StartCrawler in der AWS SDK für Ruby API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
 simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods for
 interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- API-Details finden Sie StartJobRun in der AWS SDK für Ruby API-Referenz.

IAM-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von IAM Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- API-Details finden Sie ListPolicies in der AWS SDK für Ruby API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie einen IAM-Benutzer und eine Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets erteilt. Der Benutzer hat nur Rechte, um die Rolle anzunehmen. Nachdem Sie die Rolle übernommen haben, verwenden Sie temporäre Anmeldeinformationen, um Buckets für das Konto aufzulisten.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client
```

```
  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
```

```
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
```

```
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
```

```
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated into
a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
```

```
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are deleted
  # before the user is deleted.
```

```
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
 user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
```

```
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK für Ruby API-Referenz.

`AttachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachUserPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
```

```
  end
```

- API-Details finden Sie AttachUserPolicy in der AWS SDK für Ruby API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
```

```
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
```

```
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API-Details finden Sie CreateAccessKey in der AWS SDK für Ruby API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases
```

```
    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API-Details finden Sie CreateAccountAlias in der AWS SDK für Ruby API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API-Details finden Sie CreatePolicy in der AWS SDK für Ruby API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates a role and attaches policies to it.
  #
  # @param role_name [String] The name of the role.
  # @param assume_role_policy_document [Hash] The trust relationship policy
 document.
  # @param policy_arns [Array<String>] The ARNs of the policies to attach.
  # @return [String, nil] The ARN of the new role if successful, or nil if an error
 occurred.
  def create_role(role_name, assume_role_policy_document, policy_arns)
    response = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: assume_role_policy_document.to_json
    )
    role_arn = response.role.arn
```

```
    policy_arns.each do |policy_arn|
      @iam_client.attach_role_policy(
        role_name: role_name,
        policy_arn: policy_arn
      )
    end

    role_arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating role: #{e.message}")
    nil
  end
```

- API-Details finden Sie CreateRole in der AWS SDK für Ruby API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates a service-linked role
  #
  # @param service_name [String] The service name to create the role for.
  # @param description [String] The description of the service-linked role.
  # @param suffix [String] Suffix for customizing role name.
  # @return [String] The name of the created role
  def create_service_linked_role(service_name, description, suffix)
    response = @iam_client.create_service_linked_role(
      aws_service_name: service_name, description: description, custom_suffix:
suffix
    )
    role_name = response.role.role_name
    @logger.info("Created service-linked role #{role_name}.")
```

```
    role_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK für Ruby API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Creates a user and their login profile
  #
  # @param user_name [String] The name of the user
  # @param initial_password [String] The initial password for the user
  # @return [String, nil] The ID of the user if created, or nil if an error occurred
  def create_user(user_name, initial_password)
    response = @iam_client.create_user(user_name: user_name)
    @iam_client.wait_until(:user_exists, user_name: user_name)
    @iam_client.create_login_profile(
      user_name: user_name,
      password: initial_password,
      password_reset_required: true
    )
    @logger.info("User '#{user_name}' created successfully.")
    response.user.user_id
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- API-Details finden Sie CreateUser in der AWS SDK für Ruby API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
```

```
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
```

```
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK für Ruby API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases
```

```
    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API-Details finden Sie DeleteAccountAlias in der AWS SDK für Ruby API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes a role and its attached policies.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  # Detach and delete attached policies
  @iam_client.list_attached_role_policies(role_name: role_name).each do |response|
    response.attached_policies.each do |policy|
      @iam_client.detach_role_policy({
                                      role_name: role_name,
                                      policy_arn: policy.policy_arn
                                    })
      # Check if the policy is a customer managed policy (not AWS managed)
      unless policy.policy_arn.include?('aws:policy/')
        @iam_client.delete_policy({ policy_arn: policy.policy_arn })
        @logger.info("Deleted customer managed policy #{policy.policy_name}.")
      end
    end
  end

  # Delete the role
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Deleted role #{role_name}.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API-Details finden Sie DeleteRole in der AWS SDK für Ruby API-Referenz.

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
```

```
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API-Details finden Sie DeleteServerCertificate in der AWS SDK für Ruby API-Referenz.

`DeleteServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServiceLinkedRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes a service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id
    )
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)

    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
```

```
def handle_deletion_error(e, role_name)
  return if e.code == 'NoSuchEntity'

  @logger.error("Couldn't delete #{role_name}. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API-Details finden Sie DeleteServiceLinkedRole in der AWS SDK für Ruby API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Deletes a user and their associated resources
#
# @param user_name [String] The name of the user to delete
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
  user.each do |key|
    @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end

  @iam_client.delete_user(user_name: user_name)
  @logger.info("Deleted user '#{user_name}'.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error deleting user '#{user_name}': #{e.message}")
end
```

- API-Details finden Sie DeleteUser in der AWS SDK für Ruby API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- API-Details finden Sie DeleteUserPolicy in der AWS SDK für Ruby API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK für Ruby API-Referenz.

`DetachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachUserPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Detaches a policy from a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The ARN of the policy to detach
  # @return [Boolean] true if the policy was successfully detached, false otherwise
  def detach_user_policy(user_name, policy_arn)
    @iam_client.detach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
```

```
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
    false
  end
```

- API-Details finden Sie DetachUserPolicy in der AWS SDK für Ruby API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is: #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
```

```
    raise
  end
end
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK für Ruby API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end
```

- API-Details finden Sie GetPolicy in der AWS SDK für Ruby API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                              role_name: name
                            }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- API-Details finden Sie GetRole in der AWS SDK für Ruby API-Referenz.

`GetUser`

Das folgende Codebeispiel zeigt die Verwendung`GetUser`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- API-Details finden Sie GetUser in der AWS SDK für Ruby API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Zugriffsschlüssel aufgeführt, erstellt, deaktiviert und gelöscht.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end
```

```
  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API-Details finden Sie ListAccessKeys in der AWS SDK für Ruby API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Konto-Aliase auflisten, erstellen und löschen.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
```

```
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API-Details finden Sie ListAccountAliases in der AWS SDK für Ruby API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
```

```
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
```

```
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK für Ruby API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- API-Details finden Sie ListGroups in der AWS SDK für Ruby API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispielmodul werden Rollenrichtlinien aufgelistet, erstellt, angefügt und getrennt.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
```

```
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
```

```
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API-Details finden Sie ListPolicies in der AWS SDK für Ruby API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

- API-Details finden Sie ListRolePolicies in der AWS SDK für Ruby API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count

      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API-Details finden Sie ListRoles in der AWS SDK für Ruby API-Referenz.

Listet SAML-Anbieter auf

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der AWS SDK für Ruby -API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
```

```
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API-Details finden Sie ListServerCertificates in der AWS SDK für Ruby API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- API-Details finden Sie ListUsers in der AWS SDK für Ruby API-Referenz.

`PutUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutUserPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Creates an inline policy for a specified user.
```

```
  # @param username [String] The name of the IAM user.
  # @param policy_name [String] The name of the policy to create.
  # @param policy_document [String] The JSON policy document.
  # @return [Boolean]
  def create_user_policy(username, policy_name, policy_document)
    @iam_client.put_user_policy({
                                  user_name: username,
                                  policy_name: policy_name,
                                  policy_document: policy_document
                                })
    @logger.info("Policy #{policy_name} created for user #{username}.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't create policy #{policy_name} for user #{username}.
 Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- API-Details finden Sie PutUserPolicy in der AWS SDK für Ruby API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateServerCertificate`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Serverzertifikate auflisten, aktualisieren und löschen.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end
```

```
  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API-Details finden Sie UpdateServerCertificate in der AWS SDK für Ruby API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Updates an IAM user's name
  #
  # @param current_name [String] The current name of the user
  # @param new_name [String] The new name of the user
  def update_user_name(current_name, new_name)
    @iam_client.update_user(user_name: current_name, new_user_name: new_name)
    true
  rescue StandardError => e
    @logger.error("Error updating user name from '#{current_name}' to '#{new_name}':
 #{e.message}")
    false
```

```
end
```

- API-Details finden Sie UpdateUser in der AWS SDK für Ruby API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Kinesis Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'
```

```
def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
end
```

AWS KMS Beispiele mit SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren AWS KMS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
                           tags: [
                             {
                               tag_key: 'CreatedBy',
                               tag_value: 'ExampleUser'
                             }
                           ]
                         })

puts resp.key_metadata.key_id
```

- API-Details finden Sie CreateKey in der AWS SDK für Ruby API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
                        ciphertext_blob: blob_packed
                      })

puts 'Raw text: '
puts resp.plaintext
```

- Weitere API-Informationen finden Sie unter Decrypt in der AWS SDK für Ruby -API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
                        key_id: keyId,
                        plaintext: text
                      })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- Weitere API-Informationen finden Sie unter Encrypt in der AWS SDK für Ruby -API-Referenz.

`ReEncrypt`

Das folgende Codebeispiel zeigt die Verwendung`ReEncrypt`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
 '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596
sourceCiphertextBlob = [blob].pack('H*')

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-
ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                           ciphertext_blob: sourceCiphertextBlob,
                           destination_key_id: destinationKeyId
                         })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- API-Details finden Sie ReEncrypt in der AWS SDK für Ruby API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Lambda Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-lambda'

# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
```

```
  Aws::Lambda::Client.new
end

# Lists the Lambda functions in your AWS account, paginating the results if
 necessary
def list_lambda_functions
  lambda = lambda_client

  # Use a pagination iterator to list all functions
  functions = []
  lambda.list_functions.each_page do |page|
    functions.concat(page.functions)
  end

  # Print the name and ARN of each function
  functions.each do |function|
    puts "Function name: #{function.function_name}"
    puts "Function ARN: #{function.function_arn}"
    puts
  end

  puts "Total functions: #{functions.count}"
end

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- API-Details finden Sie ListFunctions in der AWS SDK für Ruby API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.

- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Richten Sie die erforderlichen IAM-Berechtigungen für eine Lambda-Funktion ein, die Protokolle schreiben kann.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

def create_iam_role(iam_role_name)
  role_policy = {
    'Version': '2012-10-17',
    'Statement': [
      {
        'Effect': 'Allow',
        'Principal': { 'Service': 'lambda.amazonaws.com' },
```

```
          'Action': 'sts:AssumeRole'
        }
      ]
    }
    role = @iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    @iam_client.attach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    wait_for_role_to_exist(iam_role_name)
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    sleep(10)
    [role, role_policy.to_json]
  end

  def destroy_iam_role(iam_role_name)
    @iam_client.detach_role_policy(
      {
        policy_arn: 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole',
        role_name: iam_role_name
      }
    )
    @iam_client.delete_role(role_name: iam_role_name)
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  end

  def wait_for_role_to_exist(iam_role_name)
    @iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  end
```

Definieren Sie einen Lambda-Handler, der eine als Aufrufparameter bereitgestellte Zahl inkrementiert.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
                   Logger::DEBUG
                 when 'info'
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Komprimieren (ZIP) Sie Ihre Lambda-Funktion in ein Bereitstellungspaket.

```
  # Creates a Lambda deployment package in .zip format.
  #
  # @param source_file: The name of the object, without suffix, for the Lambda file
 and zip.
  # @return: The deployment package.
  def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?('lambda_function.zip')
      File.delete('lambda_function.zip')
      @logger.debug('Deleting old zip: lambda_function.zip')
    end
```

```
    Zip::File.open('lambda_function.zip', create: true) do |zipfile|
      zipfile.add('lambda_function.rb', "#{source_file}.rb")
    end
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read('lambda_function.zip').to_s
  rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
  end
```

Erstellen Sie eine neue Lambda-Funktion.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
```

```
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Rufen Sie Ihre Lambda-Funktion mit optionalen Laufzeitparametern auf.

```
  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

Aktualisieren Sie die Konfiguration Ihrer Lambda-Funktion, um eine neue Umgebungsvariable einzufügen.

```
  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                     function_name: function_name,
                                                     environment: {
                                                       variables: {
                                                         'LOG_LEVEL' => log_level
                                                       }
                                                     }
                                                   })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
```

```
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

Aktualisieren Sie den Code Ihrer Lambda-Funktion mit einem anderen Bereitstellungspaket, das anderen Code enthält.

```
  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

Listen Sie alle vorhandenen Lambda-Funktionen mithilfe des eingebauten Paginators auf.

```
  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
```

```
    response['functions'].each do |function|
      functions.append(function['function_name'])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error listing functions:\n #{e.message}")
end
```

Löschen Sie eine bestimmte Lambda-Funktion.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print 'Done!'.green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
 code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
```

```
                                                        variables: {
                                                          'LOG_LEVEL' => 'info'
                                                        }
                                                      }
                                                    })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
 do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- API-Details finden Sie CreateFunction in der AWS SDK für Ruby API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end
```

```
  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print 'Done!'.green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- API-Details finden Sie DeleteFunction in der AWS SDK für Ruby API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
```

```
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- API-Details finden Sie GetFunction in der AWS SDK für Ruby API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
```

```
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für Ruby -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response['functions'].each do |function|
        functions.append(function['function_name'])
      end
    end
```

```
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error listing functions:\n #{e.message}")
  end
```

- API-Details finden Sie ListFunctions in der AWS SDK für Ruby API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
 contains
  # the code for the function.
  #
  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                            .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
```

```
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK für Ruby API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @cloudwatch_client = Aws::CloudWatchLogs::Client.new(region: 'us-east-1')
    @iam_client = Aws::IAM::Client.new(region: 'us-east-1')
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
```

```
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
                                                    function_name: function_name,
                                                    environment: {
                                                      variables: {
                                                        'LOG_LEVEL' => log_level
                                                      }
                                                    }
                                                  })
    @lambda_client.wait_until(:function_updated_v2, { function_name:
 function_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK für Ruby API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
```

```
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
```

```
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue => err
      $stderr.puts "An error occurred #{err}"
      raise err
    end
  end
  puts "Successfully processed #{event['Records'].length} records."
end
```

```
def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('UTF-8')
  # Placeholder for actual async work
  # You can use Ruby's asynchronous programming tools like async/await or fibers
 here.
  return data
end
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    return 'received empty event' if event['Records'].empty?

    event['Records'].each do |record|
      log_dynamodb_record(record)
    end

    "Records processed: #{event['Records'].length}"
  end

  def log_dynamodb_record(record)
    puts record['eventID']
    puts record['eventName']
    puts "DynamoDB Record: #{JSON.generate(record['dynamodb'])}"
  end
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'

def lambda_handler(event:, context:)
  event['events'].each do |record|
    log_document_db_event(record)
  end
  'OK'
end

def log_document_db_event(record)
  event_data = record['event'] || {}
  operation_type = event_data['operationType'] || 'Unknown'
  db = event_data.dig('ns', 'db') || 'Unknown'
  collection = event_data.dig('ns', 'coll') || 'Unknown'
  full_document = event_data['fullDocument'] || {}

  puts "Operation type: #{operation_type}"
  puts "db: #{db}"
  puts "collection: #{collection}"
  puts "Full document: #{JSON.pretty_generate(full_document)}"
end
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst

wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'aws-sdk'

def lambda_handler(event:, context:)
  batch_item_failures = []

  event['Records'].each do |record|
    begin
      puts "Processed Kinesis Event - EventID: #{record['eventID']}"
      record_data = get_record_data_async(record['kinesis'])
      puts "Record Data: #{record_data}"
      # TODO: Do interesting work based on the new data
    rescue StandardError => err
      puts "An error occurred #{err}"
      # Since we are working with streams, we can return the failed item
 immediately.
      # Lambda will immediately begin to retry processing from this failed item
 onwards.
      return { batchItemFailures: [{ itemIdentifier: record['kinesis']
['sequenceNumber'] }] }
    end
  end

  puts "Successfully processed #{event['Records'].length} records."
  { batchItemFailures: batch_item_failures }
end

def get_record_data_async(payload)
  data = Base64.decode64(payload['data']).force_encoding('utf-8')
  # Placeholder for actual async work
  sleep(1)
  data
```

```
end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Ruby.

```
def lambda_handler(event:, context:)
    records = event["Records"]
    cur_record_sequence_number = ""

    records.each do |record|
      begin
        # Process your record
        cur_record_sequence_number = record["dynamodb"]["SequenceNumber"]
      rescue StandardError => e
        # Return failed record's sequence number
        return {"batchItemFailures" => [{"itemIdentifier" =>
 cur_record_sequence_number}]}
      end
    end

    {"batchItemFailures" => []}
  end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'

def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

Beispiele für Amazon MSK unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
```

```
      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Beispiele für Amazon Polly unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- API-Details finden Sie DescribeVoices in der AWS SDK für Ruby API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- API-Details finden Sie ListLexicons in der AWS SDK für Ruby API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
```

```
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                    output_format: 'mp3',
                                    text: contents,
                                    voice_id: 'Joanna'
                                  })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- API-Details finden Sie SynthesizeSpeech in der AWS SDK für Ruby API-Referenz.

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Beispiele für Amazon RDS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Serverless-Beispiele

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon RDS beginnen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds'
require 'logger'

# RDSManager is a class responsible for managing RDS operations
# such as listing all RDS DB instances in the current AWS account.
class RDSManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all RDS DB instances in the current AWS account.
```

```
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Ruby - API-Referenz.

Aktionen

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
                                db_snapshot_identifier: id
                              })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der AWS SDK für Ruby -API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
                          "name": i.id,
                          "status": i.db_instance_status
                        })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der AWS SDK für Ruby - API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
```

```
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK für Ruby API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
                              "name": p.db_parameter_group_name,
                              "description": p.description
                            })
  end
```

```
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der AWS SDK für Ruby - API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-rds' # v2: require 'aws-sdk'

# List all Amazon Relational Database Service (Amazon RDS) DB instance
# snapshots.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return instance_snapshots [Array, nil] All instance snapshots, or nil if error.
def list_instance_snapshots(rds_resource)
  instance_snapshots = []
  rds_resource.db_snapshots.each do |s|
    instance_snapshots.append({
                                "id": s.snapshot_id,
                                "status": s.status
                              })
  end
  instance_snapshots
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instance snapshots:\n #{e.message}"
end
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der AWS SDK für Ruby - API-Referenz.

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

def lambda_handler(event:, context:)
  endpoint = ENV['DBEndpoint'] # Add the endpoint without https"
  port = ENV['Port']           # 3306
  user = ENV['DBUser']
  region = ENV['DBRegion']     # 'us-east-1'
  db_name = ENV['DBName']

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
```

```
    ENV['AWS_SESSION_TOKEN']
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

  token = rds_client.auth_token(
    endpoint: endpoint+ ':' + port,
    user_name: user,
    region: region
  )

  begin
    conn = Mysql2::Client.new(
      host: endpoint,
      username: user,
      password: token,
      port: port,
      database: db_name,
      sslca: '/var/task/global-bundle.pem',
      sslverify: true,
      enable_cleartext_plugin: true
    )
    a = 3
    b = 2
    result = conn.query("SELECT #{a} + #{b} AS sum").first['sum']
    puts result
    conn.close
    {
      statusCode: 200,
      body: result.to_json
    }
  rescue => e
    puts "Database connection failed due to #{e}"
  end
end
```

Beispiele für Amazon S3 unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
```

```
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- API-Details finden Sie ListBuckets in der AWS SDK für Ruby API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.

- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
```

```
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      puts('Enter a name for the downloaded file: ')
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
```

```
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
```

```
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
 __FILE__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects

- GetObject
- ListObjectsV2
- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Kopieren Sie ein Objekt.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
```

```
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Kopieren Sie ein Objekt und fügen Sie dem Zielobjekt eine serverseitige Verschlüsselung hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end
```

```
  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CopyObject in der AWS SDK für Ruby API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
```

```
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateBucket in der AWS SDK für Ruby API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
```

```
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie DeleteBucket in der AWS SDK für Ruby API-Referenz.

`DeleteBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketCors`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
```

```
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API-Details finden Sie DeleteBucketCors in der AWS SDK für Ruby API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK für Ruby API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API-Details finden Sie DeleteObjects in der AWS SDK für Ruby API-Referenz.

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketCors`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- API-Details finden Sie GetBucketCors in der AWS SDK für Ruby API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK für Ruby API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie ein Objekt ab.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data
```

```
  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rufen Sie ein Objekt ab und melden Sie seinen serverseitigen Verschlüsselungsstatus.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
```

```
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie GetObject in der AWS SDK für Ruby API-Referenz.

`HeadObject`

Das folgende Codebeispiel zeigt die Verwendung`HeadObject`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie HeadObject in der AWS SDK für Ruby API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
```

```
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie ListBuckets in der AWS SDK für Ruby API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
```

```
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK für Ruby API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
```

```
    false
  end

end
```

- API-Details finden Sie PutBucketCors in der AWS SDK für Ruby API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API-Details finden Sie PutBucketPolicy in der AWS SDK für Ruby API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
```

```
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie PutBucketWebsite in der AWS SDK für Ruby API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Laden Sie eine Datei mit einem verwalteten Uploader (Object.upload_file) hoch.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
```

```
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei hoch mit. Object.put

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
```

```
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Laden Sie eine Datei mit serverseitiger Verschlüsselung hoch Object.put und fügen Sie sie hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie PutObject in der AWS SDK für Ruby API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
 #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
 'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
```

```
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
```

```
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Beispiele für Amazon SES unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetIdentityVerificationAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetIdentityVerificationAttributes`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- API-Details finden Sie GetIdentityVerificationAttributes in der AWS SDK für Ruby API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                              identity_type: 'EmailAddress'
                            })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- API-Details finden Sie ListIdentities in der AWS SDK für Ruby API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')
```

```
# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- API-Details finden Sie SendEmail in der AWS SDK für Ruby API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- API-Details finden Sie VerifyEmailIdentity in der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon SES API v2 unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe der Amazon SES-API v2 Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
```

```
      destination: {
        to_addresses: [recipient_email]
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- API-Details finden Sie SendEmail in der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon SNS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to create an Amazon Simple Notification Service (SNS)
 topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new
```

```
  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- API-Details finden Sie CreateTopic in der AWS SDK für Ruby API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
 Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
```

```
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- API-Details finden Sie ListSubscriptions in der AWS SDK für Ruby API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
```

```
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- API-Details finden Sie ListTopics in der AWS SDK für Ruby API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
class SnsMessageSender
```

```
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für Ruby -API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- API-Details finden Sie SetTopicAttributes in der AWS SDK für Ruby API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
require 'aws-sdk-sns'
require 'logger'

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info('Subscription created successfully.')
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
```

```
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = 'email'
  endpoint = 'EMAIL_ADDRESS' # Should be replaced with a real email address
  topic_arn = 'TOPIC_ARN'    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info('Creating the subscription.')
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error('Subscription creation failed. Stopping program.')
    exit 1
  end
end
```

- Weitere Informationen finden Sie im AWS SDK für Ruby -Entwicklerhandbuch.
- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für Ruby -API-Referenz.

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Beispiele für Amazon SQS unter Verwendung von SDK für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für Ruby

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung`ChangeMessageVisibility`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages: 10
                                                      })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                     queue_url: queue_url,
                                     receipt_handle: message.receipt_handle,
                                     visibility_timeout: 30 # This message will not
 be visible for 30 seconds after first receipt.
                                   })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages: 10
                                                      })
```

```
  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- API-Details finden Sie ChangeMessageVisibility in der AWS SDK für Ruby API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
 Service (Amazon SQS).

require 'aws-sdk-sqs'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
```

```
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie CreateQueue in der AWS SDK für Ruby API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

- API-Details finden Sie DeleteQueue in der AWS SDK für Ruby API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
```

```
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ['All']
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- API-Details finden Sie ListQueues in der AWS SDK für Ruby API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)
  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )

  if response.messages.count.zero?
    puts 'No messages to receive, or all messages have already ' \
      'been previously received.'
    return
  end
```

```
  response.messages.each do |message|
    puts '-' * 20
    puts "Message body: #{message.body}"
    puts "Message ID:   #{message.message_id}"
  end
rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie ReceiveMessage in der AWS SDK für Ruby API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  message_body = 'This is my message.'

  sts_client = Aws::STS::Client.new(region: region)
```

```
  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending a message to the queue named '#{queue_name}'..."

  if message_sent?(sqs_client, queue_url, message_body)
    puts 'Message sent.'
  else
    puts 'Message not sent.'
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- API-Details finden Sie SendMessage in der AWS SDK für Ruby API-Referenz.

`SendMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`SendMessageBatch`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
```

```
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]
```

```
  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- API-Details finden Sie SendMessageBatch in der AWS SDK für Ruby API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Ruby

> **Note**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].each do |message|
    process_message(message)
  end
  puts "done"
end

def process_message(message)
  begin
    puts "Processed message #{message['body']}"
    # TODO: Do interesting work based on the new message
  rescue StandardError => err
    puts "An error occurred"
    raise err
  end
end
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
require 'json'
```

```
def lambda_handler(event:, context:)
  if event
    batch_item_failures = []
    sqs_batch_response = {}

    event["Records"].each do |record|
      begin
        # process message
      rescue StandardError => e
        batch_item_failures << {"itemIdentifier" => record['messageId']}
      end
    end

    sqs_batch_response["batchItemFailures"] = batch_item_failures
    return sqs_batch_response
  end
end
```

AWS STS Beispiele mit SDK für Ruby

Die folgenden Codebeispiele zeigen, wie Sie mithilfe von AWS SDK für Ruby with Aktionen ausführen und gängige Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für Ruby

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- API-Details finden Sie AssumeRole in der AWS SDK für Ruby API-Referenz.

Beispiele für Amazon Textract unter Verwendung von SDKs für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Textract Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:

- Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
- Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.
- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im GitHub Projekt unter.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Codebeispiele für Amazon Translate unter Verwendung von SDKs für Ruby

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Translate Aktionen ausführen und gängige Szenarien implementieren. AWS SDK für Ruby

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Anwendung zum Analysieren von Kundenfeedback

Das folgende Codebeispiel zeigt, wie Sie eine Anwendung erstellen, die Kundenkommentarkarten analysiert, sie aus der ursprünglichen Sprache übersetzt, die Stimmung ermittelt und auf der Grundlage des übersetzten Texts eine Audiodatei generiert.

SDK für Ruby

> Diese Beispielanwendung analysiert und speichert Kundenfeedback-Karten. Sie ist auf die Anforderungen eines fiktiven Hotels in New York City zugeschnitten. Das Hotel erhält Feedback von Gästen in Form von physischen Kommentarkarten in verschiedenen Sprachen. Dieses Feedback wird über einen Webclient in die App hochgeladen. Nachdem ein Bild einer Kommentarkarte hochgeladen wurde, werden folgende Schritte ausgeführt:
>
> - Der Text wird mithilfe von Amazon Textract aus dem Bild extrahiert.
> - Amazon Comprehend ermittelt die Stimmung und die Sprache des extrahierten Textes.

- Der extrahierte Text wird mithilfe von Amazon Translate ins Englische übersetzt.
- Amazon Polly generiert auf der Grundlage des extrahierten Texts eine Audiodatei.

Die vollständige App kann mithilfe des AWS CDK bereitgestellt werden. Den Quellcode und Anweisungen zur Bereitstellung finden Sie im Projekt unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Codebeispiele für SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie das AWS SDK für Rust mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für Rust-Entwicklerhandbuch — Mehr über die Verwendung von Rust mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- API-Gateway-Beispiele unter Verwendung von SDK für Rust
- Beispiele für API Gateway Management API unter Verwendung von SDK für Rust
- Beispiele für Application Auto Scaling unter Verwendung von SDK für Rust
- Aurora-Beispiele unter Verwendung von SDK für Rust
- Auto-Scaling-Beispiele unter Verwendung von SDK für Rust
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Rust
- Beispiele für Amazon Bedrock Agents Runtime unter Verwendung von SDK für Rust
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Rust
- Beispiele für Amazon Cognito Sync unter Verwendung von SDK für Rust
- Firehose-Beispiele unter Verwendung von SDK für Rust
- Beispiele für Amazon DocumentDB unter Verwendung von SDK für Rust
- DynamoDB-Beispiele unter Verwendung von SDK für Rust
- Beispiele für Amazon EBS unter Verwendung von SDK für Rust
- Beispiele für Amazon EC2 unter Verwendung von SDK für Rust
- Beispiele für Amazon ECR unter Verwendung von SDK für Rust
- Beispiele für Amazon ECS unter Verwendung von SDK für Rust
- Beispiele für Amazon EKS unter Verwendung von SDK für Rust
- AWS Glue Beispiele zur Verwendung des SDK für Rust
- IAM-Beispiele unter Verwendung von SDK für Rust
- AWS IoT Beispiele zur Verwendung des SDK für Rust
- Kinesis-Beispiele unter Verwendung von SDK für Rust
- AWS KMS Beispiele mit SDK für Rust
- Lambda-Beispiele unter Verwendung von SDK für Rust
- MediaLive Beispiele zur Verwendung des SDK für Rust
- MediaPackage Beispiele zur Verwendung des SDK für Rust
- Beispiele für Amazon MSK unter Verwendung von SDK für Rust
- Beispiele für Amazon Polly unter Verwendung von SDK für Rust

- Beispiele für Amazon RDS unter Verwendung von SDK für Rust
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Rust
- Beispiele für Amazon Rekognition unter Verwendung von SDK für Rust
- Beispiele für Route 53 unter Verwendung von SDK für Rust
- Beispiele für Amazon S3 unter Verwendung von SDK für Rust
- SageMaker KI-Beispiele, die SDK für Rust verwenden
- Beispiele für Secrets Manager unter Verwendung von SDK für Rust
- Beispiele für Amazon SES API v2 unter Verwendung von SDK für Rust
- Beispiele für Amazon SNS unter Verwendung von SDK für Rust
- Beispiele für Amazon SQS unter Verwendung von SDK für Rust
- AWS STS Beispiele mit SDK für Rust
- Codebeispiele für Systems Manager unter Verwendung von SDK für Rust
- Beispiele für Amazon Transcribe unter Verwendung von SDK für Rust

API-Gateway-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden AWS. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- AWS Beiträge der Gemeinschaft

Aktionen

`GetRestApis`

Das folgende Codebeispiel zeigt die Verwendung`GetRestApis`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt die REST-APIs von Amazon API Gateway in der Region an.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
    let resp = client.get_rest_apis().send().await?;

    for api in resp.items() {
        println!("ID:          {}", api.id().unwrap_or_default());
        println!("Name:        {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version:     {}", api.version().unwrap_or_default());
        println!(
            "Created:     {}",
            api.created_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie GetRestApis in der AWS SDK for Rust API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für API Gateway Management API unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit der API Gateway Management API Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PostToConnection`

Das folgende Codebeispiel zeigt die Verwendung`PostToConnection`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

    let endpoint_url = format!(
        "https://{api_id}.execute-api.{region}.amazonaws.com/{stage}",
        api_id = api_id,
        region = region,
        stage = stage
    );

    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let api_management_config = config::Builder::from(&shared_config)
        .endpoint_url(endpoint_url)
        .build();
    let client = Client::from_conf(api_management_config);
```

- API-Details finden Sie PostToConnection in der AWS SDK for Rust API-Referenz.

Beispiele für Application Auto Scaling unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Application Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeScalingPolicies`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingPolicies`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    println!("Auto Scaling Policies:");
    for policy in response.scaling_policies() {
        println!("{:?}\n", policy);
    }
    println!("Next token: {:?}", response.next_token());

    Ok(())
}
```

- API-Details finden Sie DescribeScalingPolicies in der AWS SDK for Rust API-Referenz.

Aurora-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Informationen zu APIs finden Sie unter DescribeDBClusters in der API-Referenz zum AWS - SDK für Rust.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.

- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Bibliothek, die die Szenarienpezifischen Funktionen für das Aurora-Szenario enthält.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str = "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
```

```
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) -> Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
```

```
        }
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
```

```
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
 ScenarioError> {
        let describe_db_engine_versions =
 self.rds.describe_db_engine_versions(DB_ENGINE).await;
        trace!(versions=?describe_db_engine_versions, "full list of versions");

        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
```

```
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn get_instance_classes(&self) -> Result<Vec<String>, ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o| o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
```

```
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }
```

```
    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
    // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
```

```
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) -> Result<Vec<AuroraScenarioParameter>,
ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   // Modify both the auto_increment_offset and auto_increment_increment parameters
in one call in the custom parameter group. Set their ParameterValue fields to a new
allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
```

```
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }

    // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
    // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
    // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
    // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
    pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
        if self.password.is_none() {
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
            ));
        }
        let create_db_cluster = self
            .rds
```

```
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
```

```
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }
```

```
            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until Status
== 'available'.
    pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
        let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
        let snapshot = self
```

```
            .rds
            .snapshot_cluster(id, format!("{id}_{name}").as_str())
            .await;
        match snapshot {
            Ok(output) => match output.db_cluster_snapshot {
                Some(snapshot) => Ok(snapshot),
                None => Err(ScenarioError::with("Missing Snapshot")),
            },
            Err(err) => Err(ScenarioError::new("Failed to create snapshot", &err)),
        }
    }

    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
```

```
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
```

```
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
```

```
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Tests für die Bibliothek, die Automocks für den RDS-Client-Wrapper nutzen.

```
use crate::rds::MockRdsImpl;

use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
```

```
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError, DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError, ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
```

```
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
```

```
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
```

```
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to retrieve
 DB Engine Versions"
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;
```

```
    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if message
 == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if message
 == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
```

```
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
```

```
    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
```

```
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(
                ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
```

```
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message ==
 "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
```

```
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
```

```
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
```

```
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
```

```
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
```

```
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Cluster
```

```
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
```

```
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}

#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Eine Binärdatei zum vollständigen Ausführen des Szenarien, wobei Inquirer verwendet wird, damit der Benutzer einige Entscheidungen treffen kann.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;
```

```
#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }

    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to the
 Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario, anyhow::Error> {
```

```
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(), version.as_str()).await;
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for engine:
{err}")),
    }
```

```
    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings) ->
 Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings, "auto_increment_increment",
 3);

    // Modify both the auto_increment_offset and auto_increment_increment parameters
 in one call in the custom parameter group. Set their ParameterValue fields to a new
 allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get just
 the modified parameters. rds.DescribeDbClusterParameters(Source='user')
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
 'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();
```

```
    let password = inquire::Text::new("Password for the database (minimum 8
 characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError> {
    // Create an Aurora DB cluster database cluster that contains a MySql database
 and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
 DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();
```

```
    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until Status
 == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();

    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the instance
 and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
```

```
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;

    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) -> u8
 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
```

```
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Ein Wrapper für den Service Amazon RDS, der Automocking für Tests ermöglicht.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError, DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
```

```
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError, DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError, ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;

#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
```

```
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
```

```
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
```

```
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
```

```
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
```

```
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }
}
```

Die Cargo.toml mit Abhängigkeiten, die in diesem Szenario verwendet werden.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
```

```
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
```

```
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);
```

```
        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
```

```
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
```

```
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });
```

```
    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
```

```
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
```

```
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
```

```
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
```

```
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter CreateDBCluster in der API-Referenz zum AWS -SDK für Rust.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
               db_cluster_parameter_group: None,
               ..
           }) => {
               return Err(ScenarioError::with(
                   "CreateDBClusterParameterGroup had empty response",
               ));
           }
           Err(error) => {
               if error.code() == Some("DBParameterGroupAlreadyExists") {
                   info!("Cluster Parameter Group already exists, nothing to do");
               } else {
                   return Err(ScenarioError::new(
                       "Could not create Cluster Parameter Group",
                       &error,
                   ));
               }
           }
```

```
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK for Rust API-Referenz.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
```

```
            return Err(ScenarioError::with(
                "Must set Secret Password before starting a cluster",
            ));
        }
        let create_db_cluster = self
            .rds
            .create_db_cluster(
                DB_CLUSTER_IDENTIFIER,
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_ENGINE,
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
```

```
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
```

```
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
```

```
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
```

```
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
```

```
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
```

```
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- API-Details finden Sie CreateDBClusterSnapshot in der AWS SDK for Rust API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
```

```
                self.engine_version.as_deref().expect("engine version"),
                self.username.as_deref().expect("username"),
                self.password
                    .replace(SecretString::new("".to_string()))
                    .expect("password"),
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
```

```
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
```

```
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
```

```
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
```

```
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
```

```
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
```

```
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
```

```
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
```

```
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter CreateDBInstance in der API-Referenz zum AWS -SDK für Rust.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
    let mut clean_up_errors: Vec<ScenarioError> = vec![];

    // Delete the instance. rds.DeleteDbInstance.
    let delete_db_instance = self
        .rds
        .delete_db_instance(
            self.db_instance_identifier
                .as_deref()
                .expect("instance identifier"),
        )
        .await;
    if let Err(err) = delete_db_instance {
        let identifier = self
            .db_instance_identifier
            .as_deref()
            .unwrap_or("Missing Instance Identifier");
        let message = format!("failed to delete db instance {identifier}");
        clean_up_errors.push(ScenarioError::new(message, &err));
    } else {
        // Wait for the instance to delete
        let waiter = Waiter::default();
        while waiter.sleep().await.is_ok() {
            let describe_db_instances = self.rds.describe_db_instances().await;
            if let Err(err) = describe_db_instances {
                clean_up_errors.push(ScenarioError::new(
                    "Failed to check instance state during deletion",
                    &err,
                ));
                break;
            }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
```

```
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
```

```
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));
```

```
    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
```

```
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
```

```
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
```

```
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Cluster
```

```
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DeleteDBCluster in der API-Referenz zum AWS -SDK für Rust.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
```

```
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
```

```
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
```

```
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
```

```
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
```

```
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
```

```
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
message == "Failed to check instance state during deletion");
```

```
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK for Rust API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
```

```
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
```

```
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
```

```
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
```

```
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
```

```
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
```

```
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DeleteDBInstance in der API-Referenz zum AWS -SDK für Rust.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) -> Result<Vec<AuroraScenarioParameter>,
ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
           .collect::<Vec<_>>();

       Ok(parameters)
   }

   pub async fn describe_db_cluster_parameters(
       &self,
       name: &str,
   ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
   {
       self.inner
           .describe_db_cluster_parameters()
           .db_cluster_parameter_group_name(name)
           .into_paginator()
```

```
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK for Rust API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql database
and uses the parameter group you created.
```

```
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine and engine
version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql', EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check for
DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(), ScenarioError>
{
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }
```

```
        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for ready");
```

```
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() == Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
```

```
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
```

```
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
```

```
                    "create db cluster error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message ==
 "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message ==
 "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message ==
 "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
```

```
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
```

```
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DescribeDBClusters in der API-Referenz zum AWS -SDK für Rust.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };
```

```
        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
 version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
 versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
```

```
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
```

```
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to retrieve
 DB Engine Versions"
    );
}
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for Rust API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
```

```
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances = self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in {status}");
                        continue;
                    }
                    None => {
```

```
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in {status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group. rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
```

```
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
```

```
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
Cluster
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
```

```
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
```

```
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _}) if
 message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first Describe
 Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second Describe
 Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Informationen zu APIs finden Sie unter DescribeDBInstances in der API-Referenz zum AWS -SDK für Rust.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn get_instance_classes(&self) -> Result<Vec<String>, ScenarioError> {
        let describe_orderable_db_instance_options_items = self
```

```
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o| o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
 classes", &err))
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

.db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
```

```
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                )),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK for Rust API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   // Modify both the auto_increment_offset and auto_increment_increment parameters
in one call in the custom parameter group. Set their ParameterValue fields to a new
allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
       }
```

```
        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
```

```
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(
                ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(), SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message ==
 "Failed to modify cluster parameter group");
}
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK for Rust API-Referenz.

Auto-Scaling-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt, wie Sie mit Auto Scaling beginnen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
```

```
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK for Rust API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Gruppe mit Amazon EC2 Auto Scaling mit einer Startvorlage und Availability Zones und erhalten Sie Informationen zu laufenden Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann die Ressourcen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }
```

```
    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step 0.
 Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match AutoScalingScenario::prepare_scenario(&shared_config).await
 {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");
```

```
    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
```

```
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
    )
    .await;

    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in the
group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
```

```
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
        );
    }

    // 10. DescribeScalingActivities: list the scaling activities that have occurred
 for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;
```

```
use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at a
 time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
```

```
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
```

```
            }
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
```

```
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) -> Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
```

```
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
 scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
        //   * Recommended: InstanceType='t1.micro', ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
            .create_launch_template()
```

```
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs (you
have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
            .min_size(1)
```

```
            .set_availability_zones(Some(availability_zones))
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning here
to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };

        let enable_metrics_collection = autoscaling
```

```
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
            (Ok(_), Ok(_)) => Ok(()),
```

```
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()
                    .auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting: {}",
                                    group.status().unwrap_or("unknown error")
                                )));
                                return Err(errors);
                            }
                        }
```

```
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing after
three rounds",
            )])
        }
    }

    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
```

```
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times must
be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
```

```
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
            trace!(
                "Waiting for no scaling found {} activities",
                activities.len()
```

```
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError> {
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using DescribeAutoScalingGroup's
instances property. However, this returns a Vec<Instance>, as opposed to a
Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|i|
i.instance_id.clone().unwrap_or_default()).filter(|id| !id.is_empty()).collect())

        // Alternatively, and for the sake of example, DescribeAutoScalingInstances
returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
```

```
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
                &err,
            ));
        }
        Ok(())
    }
```

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity}))").as_str(),
               &err,
           ));
       }
       Ok(())
   }

   pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
       // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
       let _ = self
           .autoscaling
           .disable_metrics_collection()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .send()
           .await;

       // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
       //   UpdateAutoScalingGroup with MinSize=0
       let update_group = self
           .autoscaling
           .update_auto_scaling_group()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .min_size(0)
           .desired_capacity(0)
           .send()
           .await;
       if let Err(err) = update_group {
```

```
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
```

```
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error> {
```

```
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK for Rust API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(), Error>
 {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
```

```
}
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK for Rust API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK for Rust API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
       // The direct way to list instances is by using DescribeAutoScalingGroup's
instances property. However, this returns a Vec<Instance>, as opposed to a
Vec<AutoScalingInstanceDetails>.
       // Ok(self.get_group().await?.instances.unwrap_or_default().map(|i|
i.instance_id.clone().unwrap_or_default()).filter(|id| !id.is_empty()).collect())

       // Alternatively, and for the sake of example, DescribeAutoScalingInstances
returns a list that can be filtered by the client.
       self.autoscaling
           .describe_auto_scaling_instances()
           .into_paginator()
           .items()
           .send()
           .try_collect()
           .await
           .map(|items| {
               items
                   .into_iter()
                   .filter(|i| {
                       i.auto_scaling_group_name.as_deref()
                           == Some(self.auto_scaling_group_name.as_str())
                   })
                   .map(|i| i.instance_id.unwrap_or_default())
                   .filter(|id| !id.is_empty())
```

```
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK for Rust API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
```

```
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times must
be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK for Rust API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK for Rust API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
```

```
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK for Rust API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity}))").as_str(),
               &err,
           ));
```

```
        }
        Ok(())
    }
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK for Rust API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
```

```
            } else {
                Ok(())
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK for Rust API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(), Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
}
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien
- Anthropic Claude

Szenarien

Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Szenario und die Logik für die Demo. Dies orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
```

```
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }

        Ok(())
    }
```

```
    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;

        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
```

```
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
```

```
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
```

```
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];

    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
{e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
{e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
{e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;
```

```
    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Dienstprogramme zum Drucken der Nachrichteninhaltsblöcke.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError> {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
    }
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock, ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;
```

```
// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
```

```
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}

#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
```

```
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Rust.

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
```

```
    let client = Client::new(&sdk_config);

    let response = client
        .converse()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    match response {
        Ok(output) => {
            let text = get_converse_output_text(output)?;
            println!("{}", text);
            Ok(())
        }
        Err(e) => Err(e
            .as_service_error()
            .map(BedrockConverseError::from)
            .unwrap_or_else(|| BedrockConverseError("Unknown service
 error".into()))),
    }
}

fn get_converse_output_text(output: ConverseOutput) -> Result<String,
 BedrockConverseError> {
    let text = output
        .output()
        .ok_or("no output")?
        .as_message()
        .map_err(|_| "output not a message")?
        .content()
        .first()
        .ok_or("no content in message")?
        .as_text()
        .map_err(|_| "content is not text")?
        .to_string();
    Ok(text)
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    operation::converse::{ConverseError, ConverseOutput},
    types::{ContentBlock, ConversationRole, Message},
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in one
 line.";

#[derive(Debug)]
struct BedrockConverseError(String);
impl std::fmt::Display for BedrockConverseError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseError {}
impl From<&str> for BedrockConverseError {
    fn from(value: &str) -> Self {
        BedrockConverseError(value.to_string())
    }
}
impl From<&ConverseError> for BedrockConverseError {
    fn from(value: &ConverseError) -> Self {
        BedrockConverseError::from(match value {
            ConverseError::ModelTimeoutException(_) => "Model took too long",
            ConverseError::ModelNotReadyException(_) => "Model is not ready",
            _ => "Unknown",
        })
    }
}
```

- Details zu API finden Sie unter Converse in der API-Referenz zum AWS SDK für Rust.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude und streamen Sie Antwort-Token mithilfe der ConverseStream Bedrock-API.

```
#[tokio::main]
async fn main() -> Result<(), BedrockConverseStreamError> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let response = client
        .converse_stream()
        .model_id(MODEL_ID)
        .messages(
            Message::builder()
                .role(ConversationRole::User)
                .content(ContentBlock::Text(USER_MESSAGE.to_string()))
                .build()
                .map_err(|_| "failed to build message")?,
        )
        .send()
        .await;

    let mut stream = match response {
        Ok(output) => Ok(output.stream),
```

```
        Err(e) => Err(BedrockConverseStreamError::from(
            e.as_service_error().unwrap(),
        )),
    }?;

    loop {
        let token = stream.recv().await;
        match token {
            Ok(Some(text)) => {
                let next = get_converse_output_text(text)?;
                print!("{}", next);
                Ok(())
            }
            Ok(None) => break,
            Err(e) => Err(e
                .as_service_error()
                .map(BedrockConverseStreamError::from)
                .unwrap_or(BedrockConverseStreamError(
                    "Unknown error receiving stream".into(),
                ))),
        }?
    }

    println!();

    Ok(())
}

fn get_converse_output_text(
    output: ConverseStreamOutputType,
) -> Result<String, BedrockConverseStreamError> {
    Ok(match output {
        ConverseStreamOutputType::ContentBlockDelta(event) => match event.delta() {
            Some(delta) => delta.as_text().cloned().unwrap_or_else(|_| "".into()),
            None => "".into(),
        },
        _ => "".into(),
    })
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::ProvideErrorMetadata,
    operation::converse_stream::ConverseStreamError,
    types::{
        error::ConverseStreamOutputError, ContentBlock, ConversationRole,
        ConverseStreamOutput as ConverseStreamOutputType, Message,
    },
    Client,
};

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

// Start a conversation with the user message.
const USER_MESSAGE: &str = "Describe the purpose of a 'hello world' program in one
 line.";

#[derive(Debug)]
struct BedrockConverseStreamError(String);
impl std::fmt::Display for BedrockConverseStreamError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Can't invoke '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl std::error::Error for BedrockConverseStreamError {}
impl From<&str> for BedrockConverseStreamError {
    fn from(value: &str) -> Self {
        BedrockConverseStreamError(value.into())
    }
}

impl From<&ConverseStreamError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamError) -> Self {
        BedrockConverseStreamError(
            match value {
                ConverseStreamError::ModelTimeoutException(_) => "Model took too
 long",
                ConverseStreamError::ModelNotReadyException(_) => "Model is not
 ready",
                _ => "Unknown",
            }
```

```
            .into(),
        )
    }
}

impl From<&ConverseStreamOutputError> for BedrockConverseStreamError {
    fn from(value: &ConverseStreamOutputError) -> Self {
        match value {
            ConverseStreamOutputError::ValidationException(ve) =>
 BedrockConverseStreamError(
                ve.message().unwrap_or("Unknown ValidationException").into(),
            ),
            ConverseStreamOutputError::ThrottlingException(te) =>
 BedrockConverseStreamError(
                te.message().unwrap_or("Unknown ThrottlingException").into(),
            ),
            value => BedrockConverseStreamError(
                value
                    .message()
                    .unwrap_or("Unknown StreamOutput exception")
                    .into(),
            ),
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Rust API-Referenz.

Szenario: Verwendung des Tools mit der Converse-API

Das folgende Codebeispiel zeigt, wie eine typische Interaktion zwischen einer Anwendung, einem Modell mit generativer KI und verbundenen Tools oder APIs aufgebaut wird, um Interaktionen zwischen der KI und der Außenwelt zu vermitteln. Verwendet wird das Beispiel der Verbindung einer externen Wetter-API mit dem KI-Modell, sodass Wetterinformationen in Echtzeit auf der Grundlage von Benutzereingaben bereitgestellt werden können.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Das primäre Szenario und die Logik für die Demo. Dies orchestriert die Kommunikation zwischen dem Benutzer, der Converse-API von Amazon Bedrock und einem Wettertool.

```
#[derive(Debug)]
#[allow(dead_code)]
struct InvokeToolResult(String, ToolResultBlock);
struct ToolUseScenario {
    client: Client,
    conversation: Vec<Message>,
    system_prompt: SystemContentBlock,
    tool_config: ToolConfiguration,
}

impl ToolUseScenario {
    fn new(client: Client) -> Self {
        let system_prompt = SystemContentBlock::Text(SYSTEM_PROMPT.into());
        let tool_config = ToolConfiguration::builder()
            .tools(Tool::ToolSpec(
                ToolSpecification::builder()
                    .name(TOOL_NAME)
                    .description(TOOL_DESCRIPTION)
                    .input_schema(ToolInputSchema::Json(make_tool_schema()))
                    .build()
                    .unwrap(),
            ))
            .build()
            .unwrap();

        ToolUseScenario {
            client,
            conversation: vec![],
            system_prompt,
            tool_config,
        }
```

```
    }

    async fn run(&mut self) -> Result<(), ToolUseScenarioError> {
        loop {
            let input = get_input().await?;
            if input.is_none() {
                break;
            }

            let message = Message::builder()
                .role(User)
                .content(ContentBlock::Text(input.unwrap()))
                .build()
                .map_err(ToolUseScenarioError::from)?;
            self.conversation.push(message);

            let response = self.send_to_bedrock().await?;

            self.process_model_response(response).await?;
        }

        Ok(())
    }

    async fn send_to_bedrock(&mut self) -> Result<ConverseOutput,
ToolUseScenarioError> {
        debug!("Sending conversation to bedrock");
        self.client
            .converse()
            .model_id(MODEL_ID)
            .set_messages(Some(self.conversation.clone()))
            .system(self.system_prompt.clone())
            .tool_config(self.tool_config.clone())
            .send()
            .await
            .map_err(ToolUseScenarioError::from)
    }

    async fn process_model_response(
        &mut self,
        mut response: ConverseOutput,
    ) -> Result<(), ToolUseScenarioError> {
        let mut iteration = 0;
```

```
        while iteration < MAX_RECURSIONS {
            iteration += 1;
            let message = if let Some(ref output) = response.output {
                if output.is_message() {
                    Ok(output.as_message().unwrap().clone())
                } else {
                    Err(ToolUseScenarioError(
                        "Converse Output is not a message".into(),
                    ))
                }
            } else {
                Err(ToolUseScenarioError("Missing Converse Output".into()))
            }?;

            self.conversation.push(message.clone());

            match response.stop_reason {
                StopReason::ToolUse => {
                    response = self.handle_tool_use(&message).await?;
                }
                StopReason::EndTurn => {
                    print_model_response(&message.content[0])?;
                    return Ok(());
                }
                _ => (),
            }
        }

        Err(ToolUseScenarioError(
            "Exceeded MAX_ITERATIONS when calling tools".into(),
        ))
    }

    async fn handle_tool_use(
        &mut self,
        message: &Message,
    ) -> Result<ConverseOutput, ToolUseScenarioError> {
        let mut tool_results: Vec<ContentBlock> = vec![];

        for block in &message.content {
            match block {
                ContentBlock::Text(_) => print_model_response(block)?,
                ContentBlock::ToolUse(tool) => {
                    let tool_response = self.invoke_tool(tool).await?;
```

```
                    tool_results.push(ContentBlock::ToolResult(tool_response.1));
                }
                _ => (),
            };
        }

        let message = Message::builder()
            .role(User)
            .set_content(Some(tool_results))
            .build()?;
        self.conversation.push(message);

        self.send_to_bedrock().await
    }

    async fn invoke_tool(
        &mut self,
        tool: &ToolUseBlock,
    ) -> Result<InvokeToolResult, ToolUseScenarioError> {
        match tool.name() {
            TOOL_NAME => {
                println!(
                    "\x1b[0;90mExecuting tool: {TOOL_NAME} with input: {:?}...
\x1b[0m",
                    tool.input()
                );
                let content = fetch_weather_data(tool).await?;
                println!(
                    "\x1b[0;90mTool responded with {:?}\x1b[0m",
                    content.content()
                );
                Ok(InvokeToolResult(tool.tool_use_id.clone(), content))
            }
            _ => Err(ToolUseScenarioError(format!(
                "The requested tool with name {} does not exist",
                tool.name()
            ))),
        }
    }
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
```

```
    let sdk_config = aws_config::defaults(BehaviorVersion::latest())
        .region(CLAUDE_REGION)
        .load()
        .await;
    let client = Client::new(&sdk_config);

    let mut scenario = ToolUseScenario::new(client);

    header();
    if let Err(err) = scenario.run().await {
        println!("There was an error running the scenario! {}", err.0)
    }
    footer();
}
```

Das in der Demo verwendete Wettertool. Dieses Skript definiert die Werkzeugspezifikation und implementiert die Logik zum Abrufen von Wetterdaten mithilfe der Open-Meteo API.

```
const ENDPOINT: &str = "https://api.open-meteo.com/v1/forecast";
async fn fetch_weather_data(
    tool_use: &ToolUseBlock,
) -> Result<ToolResultBlock, ToolUseScenarioError> {
    let input = tool_use.input();
    let latitude = input
        .as_object()
        .unwrap()
        .get("latitude")
        .unwrap()
        .as_string()
        .unwrap();
    let longitude = input
        .as_object()
        .unwrap()
        .get("longitude")
        .unwrap()
        .as_string()
        .unwrap();
    let params = [
        ("latitude", latitude),
        ("longitude", longitude),
        ("current_weather", "true"),
    ];
```

```
    debug!("Calling {ENDPOINT} with {params:?}");

    let response = reqwest::Client::new()
        .get(ENDPOINT)
        .query(&params)
        .send()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error requesting weather:
 {e:?}")))?
        .error_for_status()
        .map_err(|e| ToolUseScenarioError(format!("Failed to request weather:
 {e:?}")))?;

    debug!("Response: {response:?}");

    let bytes = response
        .bytes()
        .await
        .map_err(|e| ToolUseScenarioError(format!("Error reading response:
 {e:?}")))?;

    let result = String::from_utf8(bytes.to_vec())
        .map_err(|_| ToolUseScenarioError("Response was not utf8".into()))?;

    Ok(ToolResultBlock::builder()
        .tool_use_id(tool_use.tool_use_id())
        .content(ToolResultContentBlock::Text(result))
        .build()?)
}
```

Dienstprogramme zum Drucken der Nachrichteninhaltsblöcke.

```
fn print_model_response(block: &ContentBlock) -> Result<(), ToolUseScenarioError> {
    if block.is_text() {
        let text = block.as_text().unwrap();
        println!("\x1b[0;90mThe model's response:\x1b[0m\n{text}");
        Ok(())
    } else {
        Err(ToolUseScenarioError(format!(
            "Content block is not text ({block:?})"
        )))
```

```
    }
}
```

Verwenden von Anweisungen, des Fehlerdienstprogramms und von Konstanten.

```
use std::{collections::HashMap, io::stdin};

use aws_config::BehaviorVersion;
use aws_sdk_bedrockruntime::{
    error::{BuildError, SdkError},
    operation::converse::{ConverseError, ConverseOutput},
    types::{
        ContentBlock, ConversationRole::User, Message, StopReason,
 SystemContentBlock, Tool,
        ToolConfiguration, ToolInputSchema, ToolResultBlock, ToolResultContentBlock,
        ToolSpecification, ToolUseBlock,
    },
    Client,
};
use aws_smithy_runtime_api::http::Response;
use aws_smithy_types::Document;
use tracing::debug;

// Set the model ID, e.g., Claude 3 Haiku.
const MODEL_ID: &str = "anthropic.claude-3-haiku-20240307-v1:0";
const CLAUDE_REGION: &str = "us-east-1";

const SYSTEM_PROMPT: &str = "You are a weather assistant that provides current
 weather data for user-specified locations using only
the Weather_Tool, which expects latitude and longitude. Infer the coordinates from
 the location yourself.
If the user provides coordinates, infer the approximate location and refer to it in
 your response.
To use the tool, you strictly apply the provided tool specification.

- Explain your step-by-step process, and give brief updates before each step.
- Only use the Weather_Tool for data. Never guess or make up information.
- Repeat the tool use for subsequent requests if necessary.
- If the tool errors, apologize, explain weather is unavailable, and suggest other
 options.
- Report temperatures in °C (°F) and wind in km/h (mph). Keep weather reports
 concise. Sparingly use
```

```
  emojis where appropriate.
- Only respond to weather queries. Remind off-topic users of your purpose.
- Never claim to search online, access external data, or use tools besides
 Weather_Tool.
- Complete the entire process until you have all required data before sending the
 complete response.
";

// The maximum number of recursive calls allowed in the tool_use_demo function.
// This helps prevent infinite loops and potential performance issues.
const MAX_RECURSIONS: i8 = 5;

const TOOL_NAME: &str = "Weather_Tool";
const TOOL_DESCRIPTION: &str =
    "Get the current weather for a given location, based on its WGS84 coordinates.";
fn make_tool_schema() -> Document {
    Document::Object(HashMap::<String, Document>::from([
        ("type".into(), Document::String("object".into())),
        (
            "properties".into(),
            Document::Object(HashMap::from([
                (
                    "latitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String("Geographical WGS84 latitude of the
 location.".into()),
                        ),
                    ])),
                ),
                (
                    "longitude".into(),
                    Document::Object(HashMap::from([
                        ("type".into(), Document::String("string".into())),
                        (
                            "description".into(),
                            Document::String(
                                "Geographical WGS84 longitude of the
 location.".into(),
                            ),
                        ),
                    ])),
```

```
                ),
            ])),
        ),
        (
            "required".into(),
            Document::Array(vec![
                Document::String("latitude".into()),
                Document::String("longitude".into()),
            ]),
        ),
    ]))
}

#[derive(Debug)]
struct ToolUseScenarioError(String);
impl std::fmt::Display for ToolUseScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "Tool use error with '{}'. Reason: {}", MODEL_ID, self.0)
    }
}
impl From<&str> for ToolUseScenarioError {
    fn from(value: &str) -> Self {
        ToolUseScenarioError(value.into())
    }
}
impl From<BuildError> for ToolUseScenarioError {
    fn from(value: BuildError) -> Self {
        ToolUseScenarioError(value.to_string().clone())
    }
}
impl From<SdkError<ConverseError, Response>> for ToolUseScenarioError {
    fn from(value: SdkError<ConverseError, Response>) -> Self {
        ToolUseScenarioError(match value.as_service_error() {
            Some(value) => value.meta().message().unwrap_or("Unknown").into(),
            None => "Unknown".into(),
        })
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Rust.

Beispiele für Amazon Bedrock Agents Runtime unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Bedrock Agents Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`InvokeAgent`

Das folgende Codebeispiel zeigt die Verwendung`InvokeAgent`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_config::{BehaviorVersion, SdkConfig};
use aws_sdk_bedrockagentruntime::{
    self as bedrockagentruntime,
    types::{error::ResponseStreamError, ResponseStream},
};
#[allow(unused_imports)]
use mockall::automock;

const BEDROCK_AGENT_ID: &str = "AJBHXXILZN";
```

```
const BEDROCK_AGENT_ALIAS_ID: &str = "AVKP1ITZAA";
const BEDROCK_AGENT_REGION: &str = "us-east-1";

#[cfg(not(test))]
pub use EventReceiverImpl as EventReceiver;
#[cfg(test)]
pub use MockEventReceiverImpl as EventReceiver;

pub struct EventReceiverImpl {
    inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
        ResponseStream,
        ResponseStreamError,
    >,
}

#[cfg_attr(test, automock)]
impl EventReceiverImpl {
    #[allow(dead_code)]
    pub fn new(
        inner: aws_sdk_bedrockagentruntime::primitives::event_stream::EventReceiver<
            ResponseStream,
            ResponseStreamError,
        >,
    ) -> Self {
        Self { inner }
    }

    pub async fn recv(
        &mut self,
    ) -> Result<
        Option<ResponseStream>,
        aws_sdk_bedrockagentruntime::error::SdkError<
            ResponseStreamError,
            aws_smithy_types::event_stream::RawMessage,
        >,
    > {
        self.inner.recv().await
    }
}

#[tokio::main]
async fn main() -> Result<(), Box<bedrockagentruntime::Error>> {
    let result = invoke_bedrock_agent("I need help.".to_string(),
 "123".to_string()).await?;
```

```
    println!("{}", result);
    Ok(())
}

async fn invoke_bedrock_agent(
    prompt: String,
    session_id: String,
) -> Result<String, bedrockagentruntime::Error> {
    let sdk_config: SdkConfig = aws_config::defaults(BehaviorVersion::latest())
        .region(BEDROCK_AGENT_REGION)
        .load()
        .await;
    let bedrock_client = bedrockagentruntime::Client::new(&sdk_config);

    let command_builder = bedrock_client
        .invoke_agent()
        .agent_id(BEDROCK_AGENT_ID)
        .agent_alias_id(BEDROCK_AGENT_ALIAS_ID)
        .session_id(session_id)
        .input_text(prompt);

    let response = command_builder.send().await?;

    let response_stream = response.completion;

    let event_receiver = EventReceiver::new(response_stream);

    process_agent_response_stream(event_receiver).await
}

async fn process_agent_response_stream(
    mut event_receiver: EventReceiver,
) -> Result<String, bedrockagentruntime::Error> {
    let mut full_agent_text_response = String::new();

    while let Some(event_result) = event_receiver.recv().await? {
        match event_result {
            ResponseStream::Chunk(chunk) => {
                if let Some(bytes) = chunk.bytes {
                    match String::from_utf8(bytes.into_inner()) {
                        Ok(text_chunk) => {
                            full_agent_text_response.push_str(&text_chunk);
                        }
                        Err(e) => {
```

```
                            eprintln!("UTF-8 decoding error for chunk: {}", e);
                        }
                    }
                }
            }
            _ => {
                panic!("received an unhandled event type from Bedrock stream",);
            }
        }
    }
    Ok(full_agent_text_response)
}

#[cfg(test)]
mod test {

    use super::*;

    #[tokio::test]
    async fn test_process_agent_response_stream() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Chunk(
                    aws_sdk_bedrockagentruntime::types::PayloadPart::builder()
                        .set_bytes(Some(aws_smithy_types::Blob::new(vec![
                            116, 101, 115, 116, 32, 99, 111, 109, 112, 108, 101,
 116, 105, 111, 110,
                        ])))
                        .build(),
                ),
            ))
        });

        // end the stream
        mock.expect_recv().times(1).returning(|| Ok(None));

        let response = process_agent_response_stream(mock).await.unwrap();

        assert_eq!("test completion", response);
    }

    #[tokio::test]
```

```
    #[should_panic(expected = "received an unhandled event type from Bedrock
 stream")]
    async fn test_process_agent_response_stream_error() {
        let mut mock = MockEventReceiverImpl::default();
        mock.expect_recv().times(1).returning(|| {
            Ok(Some(
                aws_sdk_bedrockagentruntime::types::ResponseStream::Trace(

 aws_sdk_bedrockagentruntime::types::TracePart::builder().build(),
                ),
            ))
        });

        let _ = process_agent_response_stream(mock).await.unwrap();
    }
}
```

- API-Details finden Sie InvokeAgent in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListUserPools`

Das folgende Codebeispiel zeigt die Verwendung`ListUserPools`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API-Details finden Sie ListUserPools in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon Cognito Sync unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Cognito Sync Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListIdentityPoolUsage`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentityPoolUsage`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
```

```
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- API-Details finden Sie ListIdentityPoolUsage in der AWS SDK for Rust API-Referenz.

Firehose-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Firehose Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutRecordBatch`

Das folgende Codebeispiel zeigt die Verwendung`PutRecordBatch`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn put_record_batch(
    client: &Client,
    stream: &str,
    data: Vec<Record>,
) -> Result<PutRecordBatchOutput, SdkError<PutRecordBatchError>> {
    client
        .put_record_batch()
        .delivery_stream_name(stream)
        .set_records(Some(data))
        .send()
        .await
}
```

- API-Details finden Sie PutRecordBatch in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon DocumentDB unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon DocumentDB Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(), Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;
```

```
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

DynamoDB-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren Teams erstellt wurden und verwaltet werden. AWS Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Aktionen

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<CreateTableOutput, Error> {
```

```
    let a_name: String = key.into();
    let table_name: String = table.into();

    let ad = AttributeDefinition::builder()
        .attribute_name(&a_name)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .map_err(Error::BuildError)?;

    let ks = KeySchemaElement::builder()
        .attribute_name(&a_name)
        .key_type(KeyType::Hash)
        .build()
        .map_err(Error::BuildError)?;

    let create_table_response = client
        .create_table()
        .table_name(table_name)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await;

    match create_table_response {
        Ok(out) => {
            println!("Added table {} with key {}", table, key);
            Ok(out)
        }
        Err(e) => {
            eprintln!("Got an error creating table:");
            eprintln!("{}", e);
            Err(Error::unhandled(e))
        }
    }
}
```

- API-Details finden Sie CreateTable in der AWS SDK for Rust API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_item(
    client: &Client,
    table: &str,
    key: &str,
    value: &str,
) -> Result<DeleteItemOutput, Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(out) => {
            println!("Deleted item from table");
            Ok(out)
        }
        Err(e) => Err(Error::unhandled(e)),
    }
}
```

- API-Details finden Sie DeleteItem in der AWS SDK for Rust API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_table(client: &Client, table: &str) -> Result<DeleteTableOutput,
 Error> {
    let resp = client.delete_table().table_name(table).send().await;

    match resp {
        Ok(out) => {
            println!("Deleted table");
            Ok(out)
        }
        Err(e) => Err(Error::Unhandled(e.into())),
    }
}
```

- API-Details finden Sie DeleteTable in der AWS SDK for Rust API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_tables(client: &Client) -> Result<Vec<String>, Error> {
    let paginator = client.list_tables().into_paginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;
```

```
    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(table_names)
}
```

Finden Sie heraus, ob eine Tabelle vorhanden ist.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;

    match table_list {
        Ok(list) => Ok(list.table_names().contains(&table.into())),
        Err(e) => Err(e.into()),
    }
}
```

- API-Details finden Sie ListTables in der AWS SDK for Rust API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn add_item(client: &Client, item: Item, table: &String) ->
 Result<ItemOut, Error> {
```

```
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");

    let resp = request.send().await?;

    let attributes = resp.attributes().unwrap();

    let username = attributes.get("username").cloned();
    let first_name = attributes.get("first_name").cloned();
    let last_name = attributes.get("last_name").cloned();
    let age = attributes.get("age").cloned();
    let p_type = attributes.get("p_type").cloned();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        username, first_name, last_name, age, p_type
    );

    Ok(ItemOut {
        p_type,
        age,
        username,
        first_name,
        last_name,
    })
}
```

- API-Details finden Sie PutItem in der AWS SDK for Rust API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Finden Sie die Filme, die im angegebenen Jahr gedreht wurden.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy", AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Rust.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_items(client: &Client, table: &str, page_size: Option<i32>) ->
 Result<(), Error> {
    let page_size = page_size.unwrap_or(10);
    let items: Result<Vec<_>, _> = client
        .scan()
        .table_name(table)
        .limit(page_size)
        .into_paginator()
        .items()
        .send()
        .collect()
        .await;

    println!("Items in table (up to {page_size}):");
    for item in items? {
        println!("   {:?}", item);
    }

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Rust.

Szenarien

Herstellen einer Verbindung mit einer lokalen Instance

Das folgende Codebeispiel zeigt, wie eine Endpunkt-URL überschrieben wird, um eine Verbindung zu einer lokalen Entwicklungsbereitstellung von DynamoDB und einem AWS SDK herzustellen.

Weitere Informationen finden Sie unter Lokale Version von DynamoDB.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Lists your tables from a local DynamoDB instance by setting the SDK Config's
/// endpoint_url and test_credentials.
#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();

    let config = aws_config::defaults(aws_config::BehaviorVersion::latest())
        .test_credentials()
        // DynamoDB run locally uses port 8000 by default.
        .endpoint_url("http://localhost:8000")
        .load()
        .await;
    let dynamodb_local_config =
 aws_sdk_dynamodb::config::Builder::from(&config).build();

    let client = aws_sdk_dynamodb::Client::from_conf(dynamodb_local_config);

    let list_resp = client.list_tables().send().await;
    match list_resp {
        Ok(resp) => {
            println!("Found {} tables", resp.table_names().len());
            for name in resp.table_names() {
                println!("  {}", name);
            }
```

```
        }
        Err(err) => eprintln!("Failed to list local dynamodb tables: {err:?}"),
    }
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_table(
    client: &Client,
    table: &str,
    key: &str,
) -> Result<(), SdkError<CreateTableError>> {
    let ad = AttributeDefinition::builder()
        .attribute_name(key)
        .attribute_type(ScalarAttributeType::S)
        .build()
        .expect("creating AttributeDefinition");

    let ks = KeySchemaElement::builder()
        .attribute_name(key)
        .key_type(KeyType::Hash)
        .build()
        .expect("creating KeySchemaElement");

    match client
        .create_table()
        .table_name(table)
        .key_schema(ks)
        .attribute_definitions(ad)
        .billing_mode(BillingMode::PayPerRequest)
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn add_item(client: &Client, item: Item) -> Result<(),
 SdkError<ExecuteStatementError>> {
    match client
```

```
        .execute_statement()
        .statement(format!(
            r#"INSERT INTO "{}" VALUE {{
                "{}": ?,
                "acount_type": ?,
                "age": ?,
                "first_name": ?,
                "last_name": ?
        }} "#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![
            AttributeValue::S(item.utype),
            AttributeValue::S(item.age),
            AttributeValue::S(item.first_name),
            AttributeValue::S(item.last_name),
        ]))
        .send()
        .await
    {
        Ok(_) => Ok(()),
        Err(e) => Err(e),
    }
}

async fn query_item(client: &Client, item: Item) -> bool {
    match client
        .execute_statement()
        .statement(format!(
            r#"SELECT * FROM "{}" WHERE "{}" = ?"#,
            item.table, item.key
        ))
        .set_parameters(Some(vec![AttributeValue::S(item.value)]))
        .send()
        .await
    {
        Ok(resp) => {
            if !resp.items().is_empty() {
                println!("Found a matching entry in the table:");
                println!("{:?}", resp.items.unwrap_or_default().pop());
                true
            } else {
                println!("Did not find a match.");
                false
```

```
            }
        }
        Err(e) => {
            println!("Got an error querying table:");
            println!("{}", e);
            process::exit(1);
        }
    }
}

async fn remove_item(client: &Client, table: &str, key: &str, value: String) ->
 Result<(), Error> {
    client
        .execute_statement()
        .statement(format!(r#"DELETE FROM "{table}" WHERE "{key}" = ?"#))
        .set_parameters(Some(vec![AttributeValue::S(value)]))
        .send()
        .await?;

    println!("Deleted item.");

    Ok(())
}

async fn remove_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;

    Ok(())
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK for Rust API-Referenz.

EXIF- und andere Bildinformationen speichern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.

- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
```

```
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
```

```
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;

    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);
```

```
    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!("Successfully processed {} record(s)", records.len());
```

```
    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

Beispiele für Amazon EBS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon EBS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CompleteSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CompleteSnapshot`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;

    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been
 transferred to Amazon S3.");
```

```
    println!("Use the get-snapshot-state code example to get the state of the
 snapshot.");

    Ok(())
}
```

- API-Details finden Sie CompleteSnapshot in der AWS SDK for Rust API-Referenz.

`PutSnapshotBlock`

Das folgende Codebeispiel zeigt die Verwendung`PutSnapshotBlock`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_block(
    client: &Client,
    id: &str,
    idx: usize,
    block: Vec<u8>,
    checksum: &str,
) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie PutSnapshotBlock in der AWS SDK for Rust API-Referenz.

`StartSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`StartSnapshot`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;

    Ok(snapshot.snapshot_id.unwrap())
}
```

- API-Details finden Sie StartSnapshot in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon EC2 unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon EC2.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids: Vec<String>)
 {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
```

```
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Rust API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die EC2InstanceScenario Implementierung enthält Logik, um das gesamte Beispiel auszuführen.

```
//! Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
 Cloud
//! (Amazon EC2) to do the following:
//!
//! * Create a key pair that is used to secure SSH communication between your
 computer and
//!   an EC2 instance.
//! * Create a security group that acts as a virtual firewall for your EC2 instances
 to
//!   control incoming and outgoing traffic.
//! * Find an Amazon Machine Image (AMI) and a compatible instance type.
//! * Create an instance that is created from the instance type and AMI you select,
 and
//!   is configured to use the security group and key pair created in this example.
//! * Stop and restart the instance.
//! * Create an Elastic IP address and associate it as a consistent IP address for
 your instance.
//! * Connect to your instance with SSH, using both its public IP address and your
 Elastic IP
//!   address.
//! * Clean up all of the resources created by this example.

use std::net::Ipv4Addr;

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
 security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
```

```
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }

    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
        self.stop_and_start_instance().await?;
        Ok(())
    }

    /// 1. Creates an RSA key pair and saves its private key data as a .pem file in
 secure
    ///    temporary storage. The private key data is deleted after the example
 completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
 connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;
```

```
        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
                );
            }
        }

        Ok(())
    }

    /// 1. Creates a security group for the default VPC.
    /// 2. Adds an inbound rule to allow SSH. The SSH rule allows only
    ///    inbound traffic from the current computer’s public IPv4 address.
    /// 3. Displays information about the security group.
    ///
    /// This function uses <http://checkip.amazonaws.com> to get the current public
IP
    /// address of the computer that is running the example. This method works in
most
    /// cases. However, depending on how your computer connects to the internet, you
    /// might have to manually add your public IP address to the security group by
using
    /// the AWS Management Console.
    pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
        println!("Let's create a security group to manage access to your
instance.");
        let group_name = self.util.prompt_security_group_name()?;

        self.security_group_manager
```

```
            .create(
                &self.ec2,
                &group_name,
                "Security group for example: get started with instances.",
            )
            .await?;

        println!(
            "Created security group {} in your default VPC {}.",
            self.security_group_manager.group_name(),
            self.security_group_manager
                .vpc_id()
                .unwrap_or("(unknown vpc)")
        );

        let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
        let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
            EC2Error::new(format!(
                "Failed to convert response {} to IP Address: {e:?}",
                check_ip
            ))
        })?;

        println!("Your public IP address seems to be {current_ip_address}");
        if self.util.should_add_to_security_group() {
            match self
                .security_group_manager
                .authorize_ingress(&self.ec2, current_ip_address)
                .await
            {
                Ok(_) => println!("Security group rules updated"),
                Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
            }
        }
        println!("{}", self.security_group_manager);

        Ok(())
    }

    /// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager. Specifying
the
    ///    '/aws/service/ami-amazon-linux-latest' path returns only the latest AMIs.
```

```
    /// 2. Gets and displays information about the available AMIs and lets you
select one.
    /// 3. Gets a list of instance types that are compatible with the selected AMI
and
    ///    lets you select one.
    /// 4. Creates an instance with the previously created key pair and security
group,
    ///    and the selected AMI and instance type.
    /// 5. Waits for the instance to be running and then displays its information.
    pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
        let ami = self.find_image().await?;

        let instance_types = self
            .ec2
            .list_instance_types(&ami.0)
            .await
            .map_err(|e| e.add_message("Could not find instance types"))?;
        println!(
            "There are several instance types that support the {} architecture of
the image.",
            ami.0
                .architecture
                .as_ref()
                .ok_or_else(|| EC2Error::new(format!("Missing architecture in {:?}",
ami.0)))?
        );
        let instance_type = self.util.select_instance_type(instance_types)?;

        println!("Creating your instance and waiting for it to start...");
        self.instance_manager
            .create(
                &self.ec2,
                ami.0
                    .image_id()
                    .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
                instance_type,
                self.key_pair_manager.key_pair(),
                self.security_group_manager
                    .security_group()
                    .map(|sg| vec![sg])
                    .ok_or_else(|| EC2Error::new("Could not find security group"))?,
            )
            .await
            .map_err(|e| e.add_message("Scenario failed to create instance"))?;
```

```
        while let Err(err) = self
            .ec2
            .wait_for_instance_ready(self.instance_manager.instance_id(), None)
            .await
        {
            println!("{err}");
            if !self.util.should_continue_waiting() {
                return Err(err);
            }
        }

        println!("Your instance is ready:\n{}", self.instance_manager);

        self.display_ssh_info();

        Ok(())
    }

    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
            .into_iter()
            .filter(|param| param.name().is_some_and(|name| name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }

    // 1. Stops the instance and waits for it to stop.
    // 2. Starts the instance and waits for it to start.
```

```
    // 3. Displays information about the instance.
    // 4. Displays an SSH connection string. When an Elastic IP address is
associated
    //    with the instance, the IP address stays consistent when the instance stops
    //    and starts.
    pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
        println!("Let's stop and start your instance to see what changes.");
        println!("Stopping your instance and waiting until it's stopped...");
        self.instance_manager.stop(&self.ec2).await?;
        println!("Your instance is stopped. Restarting...");
        self.instance_manager.start(&self.ec2).await?;
        println!("Your instance is running.");
        println!("{}", self.instance_manager);
        if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
            println!("Every time your instance is restarted, its public IP address
changes.");
        } else {
            println!(
                "Because you have associated an Elastic IP with your instance, you
can connect by using a consistent IP address after the instance restarts."
            );
        }
        self.display_ssh_info();
        Ok(())
    }

    /// 1. Allocates an Elastic IP address and associates it with the instance.
    /// 2. Displays an SSH connection string that uses the Elastic IP address.
    async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
        self.elastic_ip_manager.allocate(&self.ec2).await?;
        println!(
            "Allocated static Elastic IP address: {}",
            self.elastic_ip_manager.public_ip()
        );

        self.elastic_ip_manager
            .associate(&self.ec2, self.instance_manager.instance_id())
            .await?;
        println!("Associated your Elastic IP with your instance.");
        println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
        self.display_ssh_info();
        Ok(())
    }
```

```
    /// Displays an SSH connection string that can be used to connect to a running
    /// instance.
    fn display_ssh_info(&self) {
        let ip_addr = if self.elastic_ip_manager.has_allocation() {
            self.elastic_ip_manager.public_ip()
        } else {
            self.instance_manager.instance_ip()
        };
        let key_file_path = self.key_pair_manager.key_file_path().unwrap();
        println!("To connect, open another command prompt and run the following
command:");
        println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
        let _ = self.util.enter_to_continue();
    }

    /// 1. Disassociate and delete the previously created Elastic IP.
    /// 2. Terminate the previously created instance.
    /// 3. Delete the previously created security group.
    /// 4. Delete the previously created key pair.
    pub async fn clean_up(self) {
        println!("Let's clean everything up. This example created these
resources:");
        println!(
            "\tKey pair: {}",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .unwrap_or("(unknown key pair)")
        );
        println!(
            "\tSecurity group: {}",
            self.security_group_manager.group_name()
        );
        println!(
            "\tInstance: {}",
            self.instance_manager.instance_display_name()
        );
        if self.util.should_clean_resources() {
            if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.instance_manager.delete(&self.ec2).await {
                eprintln!("{err}")
```

```
            }
            if let Err(err) = self.security_group_manager.delete(&self.ec2).await {
                eprintln!("{err}");
            }
            if let Err(err) = self.key_pair_manager.delete(&self.ec2,
 &self.util).await {
                eprintln!("{err}");
            }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
("--------------------------------------------------------------------------------");
    println!(
        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
 instances demo."
    );
    println!
("--------------------------------------------------------------------------------");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
("--------------------------------------------------------------------------------");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
("--------------------------------------------------------------------------------");
}
```

Die EC2Impl-Struktur dient als Automock-Punkt für Tests und ihre Funktionen umschließen die EC2-SDK-Aufrufe.

```
use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
 KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
 String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
```

```
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
            .delete_key_pair()
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }

    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
```

```
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }

    /// Find a single security group, by ID. Returns Err if multiple groups are
found.
    pub async fn describe_security_group(
        &self,
        group_id: &str,
    ) -> Result<Option<SecurityGroup>, EC2Error> {
        let group_id: String = group_id.into();
        let describe_output = self
            .client
            .describe_security_groups()
            .group_ids(&group_id)
            .send()
            .await?;

        let mut groups = describe_output.security_groups.unwrap_or_default();

        match groups.len() {
            0 => Ok(None),
            1 => Ok(Some(groups.remove(0))),
            _ => Err(EC2Error::new(format!(
                "Expected single group for {group_id}"
            ))),
        }
    }

    /// Add an ingress rule to a security group explicitly allowing IPv4 address
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
```

```
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }

    pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
 EC2Error> {
        tracing::info!("Deleting security group {group_id}");
        self.client
            .delete_security_group()
            .group_id(group_id)
            .send()
            .await?;
        Ok(())
    }

    pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
 EC2Error> {
        let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
        let output = self
            .client
            .describe_images()
```

```
            .set_image_ids(Some(image_ids))
            .send()
            .await?;

        let images = output.images.unwrap_or_default();
        if images.is_empty() {
            Err(EC2Error::new("No images for selected AMIs"))
        } else {
            Ok(images)
        }
    }

    /// List instance types that match an image's architecture and are free tier
eligible.
    pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
        let architecture = format!(
            "{}",
            image.architecture().ok_or_else(|| EC2Error::new(format!(
                "Image {:?} does not have a listed architecture",
                image.image_id()
            )))?
        );
        let free_tier_eligible_filter = Filter::builder()
            .name("free-tier-eligible")
            .values("false")
            .build();
        let supported_architecture_filter = Filter::builder()
            .name("processor-info.supported-architecture")
            .values(architecture)
            .build();
        let response = self
            .client
            .describe_instance_types()
            .filters(free_tier_eligible_filter)
            .filters(supported_architecture_filter)
            .send()
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
```

```
            .collect())
    }

    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
```

```
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }

    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
        let response = self
            .client
```

```
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }

    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;
```

```
        tracing::info!("Stopped instance.");

        Ok(())
    }

    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        Ok(())
    }

    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn delete_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
```

```
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }

    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
        self.client
            .allocate_address()
            .domain(DomainType::Vpc)
            .send()
            .await
            .map_err(EC2Error::from)
    }

    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }
```

```
    pub async fn associate_ip_address(
        &self,
        allocation_id: &str,
        instance_id: &str,
    ) -> Result<AssociateAddressOutput, EC2Error> {
        let response = self
            .client
            .associate_address()
            .allocation_id(allocation_id)
            .instance_id(instance_id)
            .send()
            .await?;
        Ok(response)
    }

    pub async fn disassociate_ip_address(&self, association_id: &str) -> Result<(),
 EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(format!(
            "{}: {}",
            value
                .code()
```

```
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for EC2Error {}

impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

Die SSM-Struktur dient als Automock-Punkt für Tests und ihre Funktionen umschließen die SSM-SDK-Aufrufe.

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
```

```
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error> {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
}
```

Das Szenario verwendet mehrere „Manager“-Strukturen, um den Zugriff auf Ressourcen zu verwalten, die im Verlauf des Szenarios erstellt und gelöscht werden.

```
use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
 associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}
```

```
impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
        "0.0.0.0"
    }

    pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
        let allocation = ec2.allocate_ip_address().await?;
        self.elastic_ip = Some(allocation);
        Ok(())
    }

    pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
 EC2Error> {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                let association = ec2.associate_ip_address(allocation_id,
 instance_id).await?;
                self.association = Some(association);
                return Ok(());
            }
        }
        Err(EC2Error::new("No ip address allocation to associate"))
    }

    pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(association) = &self.association {
            if let Some(association_id) = association.association_id() {
                ec2.disassociate_ip_address(association_id).await?;
            }
        }
        self.association = None;
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                ec2.deallocate_ip_address(allocation_id).await?;
            }
```

```
        }
        self.elastic_ip = None;
        Ok(())
    }
}


use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};

use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
#[derive(Debug, Default)]
pub struct InstanceManager {
    instance: Option<Instance>,
}

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
 Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
```

```
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{} ({})", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }

    /// Start the managed EC2 instance, if present.
    pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.start_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    /// Stop the managed EC2 instance, if present.
    pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.stop_instance(self.instance_id()).await?;
        }
        Ok(())
    }
```

```
    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }

    /// Terminate and delete the managed EC2 instance, if present.
    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.delete_instance(self.instance_id()).await?;
        }
        Ok(())
    }
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}", instance.instance_id().unwrap_or("(Unknown)"))?;
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            )?;
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            )?;
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            )?;
            writeln!(f, "\tVPC ID: {}", instance.vpc_id().unwrap_or("(Unknown)"))?;
            writeln!(
```

```
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            )?;
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                        .map(|isn| format!("{isn}"))
                        .unwrap_or("(Unknown)".to_string())
                })
                .unwrap_or("(Unknown)".to_string());
            writeln!(f, "\tState: {instance_state}")?;
        } else {
            writeln!(f, "\tNo loaded instance")?;
        }
        Ok(())
    }
}


use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }
```

```
    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
        &self.key_file_dir
    }

    /// Creates a key pair that can be used to securely connect to an EC2 instance.
    /// The returned key pair contains private key information that cannot be
retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }

    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
```

```
        if let Some(key_name) = self.key_pair.key_name() {
            ec2.delete_key_pair(key_name).await?;
            if let Some(key_path) = self.key_file_path() {
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }

    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }
}

impl Default for KeyPairManager {
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}


use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
```

```
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
                .map_err(|e| e.add_message("Couldn't create security group"))?,
        );

        Ok(())
    }

    pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.authorize_security_group_ssh_ingress(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
                vec![ip_address],
            )
            .await?;
        };

        Ok(())
    }

    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.delete_security_group(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            )
            .await?;
        };

        Ok(())
    }

    pub fn group_name(&self) -> &str {
```

```
        &self.group_name
    }

    pub fn vpc_id(&self) -> Option<&str> {
        self.security_group.as_ref().and_then(|sg| sg.vpc_id())
    }

    pub fn security_group(&self) -> Option<&SecurityGroup> {
        self.security_group.as_ref()
    }
}

impl std::fmt::Display for SecurityGroupManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.security_group {
            Some(sg) => {
                writeln!(
                    f,
                    "Security group: {}",
                    sg.group_name().unwrap_or("(unknown group)")
                )?;
                writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
 id)"))?;
                writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
 vpc)"))?;
                if !sg.ip_permissions().is_empty() {
                    writeln!(f, "\tInbound Permissions:")?;
                    for permission in sg.ip_permissions() {
                        writeln!(f, "\t\t{permission:?}")?;
                    }
                }
                Ok(())
            }
            None => writeln!(f, "No security group loaded."),
        }
    }
}
```

Der Haupteinstiegspunkt für das Szenario.

```
use ec2_code_examples::{
```

```
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
    let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
    let util = UtilImpl {};
    let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
    run(scenario).await;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances

- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
       self.client
           .allocate_address()
           .domain(DomainType::Vpc)
           .send()
           .await
           .map_err(EC2Error::from)
   }
```

- API-Details finden Sie AllocateAddress in der AWS SDK for Rust API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
    Ok(response)
}
```

- API-Details finden Sie AssociateAddress in der AWS SDK for Rust API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Add an ingress rule to a security group explicitly allowing IPv4 address
```

```
    /// as {ip}/32 over TCP port 22.
    pub async fn authorize_security_group_ssh_ingress(
        &self,
        group_id: &str,
        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
                            .build()
                    })
                    .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK for Rust API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rust-Implementierung, die create_key_pair des EC2-Clients aufruft und das zurückgegebene Material extrahiert.

```
   pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
       tracing::info!("Creating key pair {name}");
       let output = self.client.create_key_pair().key_name(name).send().await?;
       let info = KeyPairInfo::builder()
           .set_key_name(output.key_name)
           .set_key_fingerprint(output.key_fingerprint)
           .set_key_pair_id(output.key_pair_id)
           .build();
       let material = output
           .key_material
           .ok_or_else(|| EC2Error::new("Create Key Pair has no key material"))?;
       Ok((info, material))
   }
```

Eine Funktion, die die create_key-Implementierung aufruft und den privaten PEM-Schlüssel sicher speichert.

```
   /// Creates a key pair that can be used to securely connect to an EC2 instance.
   /// The returned key pair contains private key information that cannot be
retrieved
   /// again. The private key data is stored as a .pem file.
   ///
   /// :param key_name: The name of the key pair to create.
   pub async fn create(
       &mut self,
       ec2: &EC2,
       util: &Util,
       key_name: String,
   ) -> Result<KeyPairInfo, EC2Error> {
       let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
           self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
           e.add_message(format!("Couldn't create key {key_name}"))
       })?;

       let path = self.key_file_dir.join(format!("{key_name}.pem"));
```

```
        // Save the key_pair information immediately, so it can get cleaned up if
write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }
```

- API-Details finden Sie CreateKeyPair in der AWS SDK for Rust API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn create_security_group(
        &self,
        name: &str,
        description: &str,
    ) -> Result<SecurityGroup, EC2Error> {
        tracing::info!("Creating security group {name}");
        let create_output = self
            .client
            .create_security_group()
            .group_name(name)
            .description(description)
            .send()
            .await
            .map_err(EC2Error::from)?;

        let group_id = create_output
            .group_id
```

```
            .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

        let group = self
            .describe_security_group(&group_id)
            .await?
            .ok_or_else(|| {
                EC2Error::new(format!("Could not find security group with id
{group_id}"))
            })?;

        tracing::info!("Created security group {name} as {group_id}");

        Ok(group)
    }
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK for Rust API-Referenz.

`CreateTags`

Das folgende Codebeispiel zeigt die Verwendung`CreateTags`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

In diesem Beispiel wird nach dem Erstellen einer Instance das Tag „Name“ angewendet.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
```

```
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
```

```
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- API-Details finden Sie CreateTags in der AWS SDK for Rust API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Wrapper um delete_key, der auch den zugrunde liegenden privaten PEM-Schlüssel entfernt.

```
    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
        if let Some(key_name) = self.key_pair.key_name() {
            ec2.delete_key_pair(key_name).await?;
            if let Some(key_path) = self.key_file_path() {
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }
```

```
    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
```

```
            .delete_key_pair()
            .key_name(key_name)
            .send()
            .await?;
        Ok(())
    }
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK for Rust API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
        tracing::info!("Deleting security group {group_id}");
        self.client
            .delete_security_group()
            .group_id(group_id)
            .send()
            .await?;
        Ok(())
    }
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK for Rust API-Referenz.

`DeleteSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSnapshot`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- API-Details finden Sie DeleteSnapshot in der AWS SDK for Rust API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
 EC2Error> {
        let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
        let output = self
            .client
            .describe_images()
            .set_image_ids(Some(image_ids))
            .send()
```

```
            .await?;

        let images = output.images.unwrap_or_default();
        if images.is_empty() {
            Err(EC2Error::new("No images for selected AMIs"))
        } else {
            Ok(images)
        }
    }
```

Verwenden Sie die Funktion list_images mit SSM zur Einschränkung auf Grundlage Ihrer Umgebung. Weitere Informationen zu SSM finden Sie unter https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ssm_GetParameters_section.html.

```
    async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
        let params: Vec<Parameter> = self
            .ssm
            .list_path("/aws/service/ami-amazon-linux-latest")
            .await
            .map_err(|e| e.add_message("Could not find parameters for available
images"))?
            .into_iter()
            .filter(|param| param.name().is_some_and(|name| name.contains("amzn2")))
            .collect();
        let amzn2_images: Vec<ScenarioImage> = self
            .ec2
            .list_images(params)
            .await
            .map_err(|e| e.add_message("Could not find images"))?
            .into_iter()
            .map(ScenarioImage::from)
            .collect();
        println!("We will now create an instance from an Amazon Linux 2 AMI");
        let ami = self.util.select_scenario_image(amzn2_images)?;
        Ok(ami)
    }
```

- API-Details finden Sie DescribeImages in der AWS SDK for Rust API-Referenz.

`DescribeInstanceStatus`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceStatus`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str = Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider = RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}:", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "  Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("    Event ID:     {}",
 event.instance_event_id().unwrap());
                println!("    Description:  {}", event.description().unwrap());
                println!("    Event code:   {}", event.code().unwrap().as_ref());
                println!();
            }
        }
    }

    Ok(())
}
```

- API-Details finden Sie DescribeInstanceStatus in der AWS SDK for Rust API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   /// List instance types that match an image's architecture and are free tier
eligible.
   pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
       let architecture = format!(
           "{}",
           image.architecture().ok_or_else(|| EC2Error::new(format!(
               "Image {:?} does not have a listed architecture",
               image.image_id()
           )))?
       );
       let free_tier_eligible_filter = Filter::builder()
           .name("free-tier-eligible")
           .values("false")
           .build();
       let supported_architecture_filter = Filter::builder()
           .name("processor-info.supported-architecture")
           .values(architecture)
           .build();
       let response = self
           .client
           .describe_instance_types()
           .filters(free_tier_eligible_filter)
           .filters(supported_architecture_filter)
           .send()
```

```
            .await?;

        Ok(response
            .instance_types
            .unwrap_or_default()
            .into_iter()
            .filter_map(|iti| iti.instance_type)
            .collect())
    }
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK for Rust API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die Details einer EC2-Instance ab.

```
    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
```

```
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }
```

Nachdem Sie eine EC2-Instance erstellt haben, rufen Sie deren Details ab und speichern Sie diese.

```
    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,
        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }
```

- API-Details finden Sie DescribeInstances in der AWS SDK for Rust API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK for Rust API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- API-Details finden Sie DescribeRegions in der AWS SDK for Rust API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids: Vec<String>)
 {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
        }
    }
```

```
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Rust API-Referenz.

`DescribeSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSnapshots`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Zeigt den Status eines Snapshots an.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots();
```

```
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:       {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();

    Ok(())
}
```

- API-Details finden Sie DescribeSnapshots in der AWS SDK for Rust API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn disassociate_ip_address(&self, association_id: &str) -> Result<(),
EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
```

```
            .send()
            .await?;
        Ok(())
    }
```

- API-Details finden Sie DisassociateAddress in der AWS SDK for Rust API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }
```

```
    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
```

```
        Ok(())
    }
```

Waiters: Warten, bis sich die Instance in den Status „Angehalten“ und „Bereit“ befindet, mithilfe der Waiters-API. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn wait_for_instance_stopped(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_stopped()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to stop.",
                    exceeded.max_wait().as_secs(),
```

```
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- API-Details finden Sie RebootInstances in der AWS SDK for Rust API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
        self.client
            .release_address()
            .allocation_id(allocation_id)
            .send()
            .await?;
        Ok(())
    }
```

- API-Details finden Sie ReleaseAddress in der AWS SDK for Rust API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn create_instance<'a>(
        &self,
        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
```

```
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

        match response {
            Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
            Err(err) => {
                tracing::info!("Error applying tags to {instance_id}: {err:?}");
                return Err(err.into());
            }
        }

        tracing::info!("Instance is created.");

        Ok(instance_id.to_string())
    }
```

- API-Details finden Sie RunInstances in der AWS SDK for Rust API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Starten Sie eine EC2-Instance mithilfe der Instance-ID.

```
    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }
```

Warten Sie mit der Waiters-API, bis sich die Instance in den Status „Bereit” und „OK” befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    /// Wait for an instance to be ready and status ok (default wait 60 seconds)
    pub async fn wait_for_instance_ready(
        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- API-Details finden Sie StartInstances in der AWS SDK for Rust API-Referenz.

StopInstances

Das folgende Codebeispiel zeigt die VerwendungStopInstances.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;

    tracing::info!("Stopped instance.");

    Ok(())
}
```

Warten Sie mithilfe der Waiters API, bis sich eine Instance im Status „Angehalten“ befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;
```

```
        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }
```

- API-Details finden Sie StopInstances in der AWS SDK for Rust API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn delete_instance(&self, instance_id: &str) -> Result<(), EC2Error> {
        tracing::info!("Deleting instance with id {instance_id}");
        self.stop_instance(instance_id).await?;
        self.client
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await?;
        self.wait_for_instance_terminated(instance_id).await?;
        tracing::info!("Terminated instance with id {instance_id}");
        Ok(())
    }
```

Warten Sie mithilfe der Waiters-API, bis sich eine Instance im Status „Beendet“ befindet. Die Verwendung der Waiters-API erfordert in der Rust-Datei die Anweisung use aws_sdk_ec2::client::Waiters.

```
    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client
            .wait_until_instance_terminated()
            .instance_ids(instance_id)
            .wait(Duration::from_secs(60))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to terminate.",
                    exceeded.max_wait().as_secs(),
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }
```

- API-Details finden Sie TerminateInstances in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon ECR unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon ECR Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRepositories`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(), aws_sdk_ecr::Error>
 {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN:  {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- API-Details finden Sie DescribeRepositories in der AWS SDK for Rust API-Referenz.

`ListImages`

Das folgende Codebeispiel zeigt die Verwendung`ListImages`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_images(
```

```
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;

    let images = rsp.image_ids();

    println!("found {} images", images.len());

    for image in images {
        println!(
            "image: {}:{}",
            image.image_tag().unwrap(),
            image.image_digest().unwrap()
        );
    }

    Ok(())
}
```

- API-Details finden Sie ListImages in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon ECS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon ECS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),
 aws_sdk_ecs::Error> {
    let cluster = client.create_cluster().cluster_name(name).send().await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- API-Details finden Sie CreateCluster in der AWS SDK for Rust API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_cluster(
    client: &aws_sdk_ecs::Client,
    name: &str,
```

```
) -> Result<(), aws_sdk_ecs::Error> {
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- API-Details finden Sie DeleteCluster in der AWS SDK for Rust API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(),
 aws_sdk_ecs::Error> {
    let resp = client.list_clusters().send().await?;

    let cluster_arns = resp.cluster_arns();
    println!("Found {} clusters:", cluster_arns.len());

    let clusters = client
        .describe_clusters()
        .set_clusters(Some(cluster_arns.into()))
        .send()
        .await?;

    for cluster in clusters.clusters() {
        println!("  ARN:  {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
    }

    Ok(())
}
```

- API-Details finden Sie DescribeClusters in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon EKS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon EKS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
```

```
    let cluster = client
        .create_cluster()
        .name(name)
        .role_arn(arn)
        .resources_vpc_config(
            VpcConfigRequest::builder()
                .set_subnet_ids(Some(subnet_ids))
                .build(),
        )
        .send()
        .await?;
    println!("cluster created: {:?}", cluster);

    Ok(())
}
```

- API-Details finden Sie CreateCluster in der AWS SDK for Rust API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- API-Details finden Sie DeleteCluster in der AWS SDK for Rust API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
```

```
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- API-Details finden Sie ListJobs in der AWS SDK for Rust API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen und starten Sie einen Crawler, der einen öffentlichen Amazon Simple Storage Service (Amazon S3) -Bucket crawlt und eine Metadaten-Datenbank generiert, die die gefundenen CSV-formatted Daten beschreibt.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler = glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

Erstellen und führen Sie einen Job aus, der CSV-Daten aus dem Amazon S3-Quell-Bucket extrahiert, sie durch Entfernen und Umbenennen von Feldern transformiert und die JSON-formatted Ausgabe in einen anderen Amazon S3-Bucket lädt.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just started
job".into()))?
            .to_string();
```

Löscht alle Ressourcen, die von der Demo erstellt wurden.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCrawler`

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- API-Details finden Sie CreateCrawler in der AWS SDK for Rust API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating job".into())
        })?;
```

- API-Details finden Sie CreateJob in der AWS SDK for Rust API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API-Details finden Sie DeleteCrawler in der AWS SDK for Rust API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API-Details finden Sie DeleteDatabase in der AWS SDK for Rust API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- API-Details finden Sie DeleteJob in der AWS SDK for Rust API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        }
```

- API-Details finden Sie DeleteTable in der AWS SDK for Rust API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- API-Details finden Sie GetCrawler in der AWS SDK for Rust API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let database = glue
```

```
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
database".into()))?;
```

- API-Details finden Sie GetDatabase in der AWS SDK for Rust API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };
```

```
        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- API-Details finden Sie GetJobRun in der AWS SDK for Rust API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- API-Details finden Sie GetTables in der AWS SDK for Rust API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
        }
```

- API-Details finden Sie ListJobs in der AWS SDK for Rust API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let start_crawler = glue.start_crawler().name(self.crawler()).send().await;
```

```
        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- API-Details finden Sie StartCrawler in der AWS SDK for Rust API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die Verwendung`StartJobRun`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
```

```
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just started
job".into()))?
            .to_string();
```

- API-Details finden Sie StartJobRun in der AWS SDK for Rust API-Referenz.

IAM-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt die ersten Schritte mit IAM.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Von src/bin /hello.rs.

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

Von src/iam -service-lib.rs.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
```

```
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- API-Details finden Sie ListPolicies in der AWS SDK for Rust API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client as
 iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document) =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };
```

```
    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));

    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}", "iam_demo_user_",
 uuid)).await?;
```

```
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
```

```
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
```

```
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
```

```
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK for Rust API-Referenz.

`AttachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachUserPolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
```

```
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie AttachUserPolicy in der AWS SDK for Rust API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
```

```
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- API-Details finden Sie CreateAccessKey in der AWS SDK for Rust API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- API-Details finden Sie CreatePolicy in der AWS SDK for Rust API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- API-Details finden Sie CreateRole in der AWS SDK for Rust API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput, SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK for Rust API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- API-Details finden Sie CreateUser in der AWS SDK for Rust API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
```

```
            }
        }
    }
    Ok(())
}
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK for Rust API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- API-Details finden Sie DeletePolicy in der AWS SDK for Rust API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError> {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- API-Details finden Sie DeleteRole in der AWS SDK for Rust API-Referenz.

`DeleteServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServiceLinkedRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_service_linked_role(
    client: &iamClient,
```

```
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie DeleteServiceLinkedRole in der AWS SDK for Rust API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
```

```
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
}
```

- API-Details finden Sie DeleteUser in der AWS SDK for Rust API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;
```

```
    Ok(())
}
```

- API-Details finden Sie DeleteUserPolicy in der AWS SDK for Rust API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK for Rust API-Referenz.

`DetachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachUserPolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie DetachUserPolicy in der AWS SDK for Rust API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn get_account_password_policy(
```

```
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput, SdkError<GetAccountPasswordPolicyError>>
 {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK for Rust API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- API-Details finden Sie GetRole in der AWS SDK for Rust API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput, SdkError<ListAttachedRolePoliciesError>>
 {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK for Rust API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API-Details finden Sie ListGroups in der AWS SDK for Rust API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- API-Details finden Sie ListPolicies in der AWS SDK for Rust API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API-Details finden Sie ListRolePolicies in der AWS SDK for Rust API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
```

```
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API-Details finden Sie ListRoles in der AWS SDK for Rust API-Referenz.

Listet SAML-Anbieter auf

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- Weitere API-Informationen finden Sie unter ListSAMLProviders in der API-Referenz zum AWS -SDK für Rust.

ListUsers

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API-Details finden Sie ListUsers in der AWS SDK for Rust API-Referenz.

AWS IoT Beispiele zur Verwendung des SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren AWS IoT.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error> {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK for Rust API-Referenz.

`ListThings`

Das folgende Codebeispiel zeigt die Verwendung`ListThings`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- API-Details finden Sie ListThings in der AWS SDK for Rust API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Kinesis Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`CreateStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateStream`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- API-Details finden Sie CreateStream in der AWS SDK for Rust API-Referenz.

`DeleteStream`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStream`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client.delete_stream().stream_name(stream).send().await?;

    println!("Deleted stream.");

    Ok(())
}
```

- API-Details finden Sie DeleteStream in der AWS SDK for Rust API-Referenz.

`DescribeStream`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStream`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();
```

```
    println!("Stream description:");
    println!("  Name:              {}:", desc.stream_name());
    println!("  Status:            {:?}", desc.stream_status());
    println!("  Open shards:       {:?}", desc.shards.len());
    println!("  Retention (hours): {}", desc.retention_period_hours());
    println!("  Encryption:        {:?}", desc.encryption_type.unwrap());

    Ok(())
}
```

- API-Details finden Sie DescribeStream in der AWS SDK for Rust API-Referenz.

`ListStreams`

Das folgende Codebeispiel zeigt die Verwendung`ListStreams`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names;
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());

    Ok(())
}
```

- API-Details finden Sie ListStreams in der AWS SDK for Rust API-Referenz.

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung`PutRecord`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) ->
 Result<(), Error> {
    let blob = Blob::new(data);

    client
        .put_record()
        .data(blob)
        .partition_key(key)
        .stream_name(stream)
        .send()
        .await?;

    println!("Put data into stream.");

    Ok(())
}
```

- API-Details finden Sie PutRecord in der AWS SDK for Rust API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error> {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
```

```
            }
        }
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
```

```
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS KMS Beispiele mit SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren AWS KMS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp.key_metadata.as_ref().unwrap().key_id();

    println!("Key: {}", id);

    Ok(())
}
```

- API-Details finden Sie CreateKey in der AWS SDK for Rust API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),
 Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base 64
 characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Decrypt in der API-Referenz zum AWS -SDK für Rust.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Encrypt in der API-Referenz zum AWS -SDK für Rust.

`GenerateDataKey`

Das folgende Codebeispiel zeigt die Verwendung`GenerateDataKey`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API-Details finden Sie GenerateDataKey in der AWS SDK for Rust API-Referenz.

`GenerateDataKeyWithoutPlaintext`

Das folgende Codebeispiel zeigt die Verwendung`GenerateDataKeyWithoutPlaintext`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API-Details finden Sie GenerateDataKeyWithoutPlaintext in der AWS SDK for Rust API-Referenz.

`GenerateRandom`

Das folgende Codebeispiel zeigt die Verwendung`GenerateRandom`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API-Details finden Sie GenerateRandom in der AWS SDK for Rust API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- API-Details finden Sie ListKeys in der AWS SDK for Rust API-Referenz.

`ReEncrypt`

Das folgende Codebeispiel zeigt die Verwendung`ReEncrypt`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- API-Details finden Sie ReEncrypt in der AWS SDK for Rust API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

AWS Community-Beiträge sind Beispiele, die von mehreren AWS Teams erstellt wurden und verwaltet werden. Verwenden Sie den Mechanismus, der in den verknüpften Repositorys zur Verfügung steht, um Feedback zu geben.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele
- AWS Beiträge der Gemeinschaft

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.

- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Cargo.toml mit Abhängigkeiten, die in diesem Szenario verwendet werden.

```
[package]
name = "lambda-code-examples"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-ec2 = { version = "1.3.0" }
aws-sdk-iam = { version = "1.3.0" }
aws-sdk-lambda = { version = "1.3.0" }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
tracing = "0.1.37"
serde_json = "1.0.94"
```

```
anyhow = "1.0.71"
uuid = { version = "1.3.3", features = ["v4"] }
lambda_runtime = "0.8.0"
serde = "1.0.164"
```

Eine Sammlung von Hilfsprogrammen, die das Aufrufen von Lambda für dieses Szenario optimieren. Diese Datei src/ations ist.rs in der Kiste.

```
use anyhow::anyhow;
use aws_sdk_iam::operation::{create_role::CreateRoleError,
 delete_role::DeleteRoleOutput};
use aws_sdk_lambda::{
    operation::{
        delete_function::DeleteFunctionOutput, get_function::GetFunctionOutput,
        invoke::InvokeOutput, list_functions::ListFunctionsOutput,
        update_function_code::UpdateFunctionCodeOutput,
        update_function_configuration::UpdateFunctionConfigurationOutput,
    },
    primitives::ByteStream,
    types::{Environment, FunctionCode, LastUpdateStatus, State},
};
use aws_sdk_s3::{
    error::ErrorMetadata,
    operation::{delete_bucket::DeleteBucketOutput,
 delete_object::DeleteObjectOutput},
    types::CreateBucketConfiguration,
};
use aws_smithy_types::Blob;
use serde::{ser::SerializeMap, Serialize};
use std::{fmt::Display, path::PathBuf, str::FromStr, time::Duration};
use tracing::{debug, info, warn};

/* Operation describes  */
#[derive(Clone, Copy, Debug, Serialize)]
pub enum Operation {
    #[serde(rename = "plus")]
    Plus,
    #[serde(rename = "minus")]
    Minus,
    #[serde(rename = "times")]
    Times,
```

```
    #[serde(rename = "divided-by")]
    DividedBy,
}

impl FromStr for Operation {
    type Err = anyhow::Error;

    fn from_str(s: &str) -> Result<Self, Self::Err> {
        match s {
            "plus" => Ok(Operation::Plus),
            "minus" => Ok(Operation::Minus),
            "times" => Ok(Operation::Times),
            "divided-by" => Ok(Operation::DividedBy),
            _ => Err(anyhow!("Unknown operation {s}")),
        }
    }
}

impl Display for Operation {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match self {
            Operation::Plus => write!(f, "plus"),
            Operation::Minus => write!(f, "minus"),
            Operation::Times => write!(f, "times"),
            Operation::DividedBy => write!(f, "divided-by"),
        }
    }
}

/**
 * InvokeArgs will be serialized as JSON and sent to the AWS Lambda handler.
 */
#[derive(Debug)]
pub enum InvokeArgs {
    Increment(i32),
    Arithmetic(Operation, i32, i32),
}

impl Serialize for InvokeArgs {
    fn serialize<S>(&self, serializer: S) -> Result<S::Ok, S::Error>
    where
        S: serde::Serializer,
    {
        match self {
```

```
            InvokeArgs::Increment(i) => serializer.serialize_i32(*i),
            InvokeArgs::Arithmetic(o, i, j) => {
                let mut map: S::SerializeMap = serializer.serialize_map(Some(3))?;
                map.serialize_key(&"op".to_string())?;
                map.serialize_value(&o.to_string())?;
                map.serialize_key(&"i".to_string())?;
                map.serialize_value(&i)?;
                map.serialize_key(&"j".to_string())?;
                map.serialize_value(&j)?;
                map.end()
            }
        }
    }
}

/** A policy document allowing Lambda to execute this function on the account's
 behalf. */
const ROLE_POLICY_DOCUMENT: &str = r#"{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "lambda.amazonaws.com" },
            "Action": "sts:AssumeRole"
        }
    ]
}"#;

/**
 * A LambdaManager gathers all the resources necessary to run the Lambda example
 scenario.
 * This includes instantiated aws_sdk clients and details of resource names.
 */
pub struct LambdaManager {
    iam_client: aws_sdk_iam::Client,
    lambda_client: aws_sdk_lambda::Client,
    s3_client: aws_sdk_s3::Client,
    lambda_name: String,
    role_name: String,
    bucket: String,
    own_bucket: bool,
}
```

```
// These unit type structs provide nominal typing on top of String parameters for
 LambdaManager::new
pub struct LambdaName(pub String);
pub struct RoleName(pub String);
pub struct Bucket(pub String);
pub struct OwnBucket(pub bool);

impl LambdaManager {
    pub fn new(
        iam_client: aws_sdk_iam::Client,
        lambda_client: aws_sdk_lambda::Client,
        s3_client: aws_sdk_s3::Client,
        lambda_name: LambdaName,
        role_name: RoleName,
        bucket: Bucket,
        own_bucket: OwnBucket,
    ) -> Self {
        Self {
            iam_client,
            lambda_client,
            s3_client,
            lambda_name: lambda_name.0,
            role_name: role_name.0,
            bucket: bucket.0,
            own_bucket: own_bucket.0,
        }
    }

    /**
     * Load the AWS configuration from the environment.
     * Look up lambda_name and bucket if none are given, or generate a random name
 if not present in the environment.
     * If the bucket name is provided, the caller needs to have created the bucket.
     * If the bucket name is generated, it will be created.
     */
    pub async fn load_from_env(lambda_name: Option<String>, bucket: Option<String>)
 -> Self {
        let sdk_config = aws_config::load_from_env().await;
        let lambda_name = LambdaName(lambda_name.unwrap_or_else(|| {
            std::env::var("LAMBDA_NAME").unwrap_or_else(|_|
 "rust_lambda_example".to_string())
        }));
        let role_name = RoleName(format!("{}_role", lambda_name.0));
        let (bucket, own_bucket) =
```

```
            match bucket {
                Some(bucket) => (Bucket(bucket), false),
                None => (
                    Bucket(std::env::var("LAMBDA_BUCKET").unwrap_or_else(|_| {
                        format!("rust-lambda-example-{}", uuid::Uuid::new_v4())
                    })),
                    true,
                ),
            };

        let s3_client = aws_sdk_s3::Client::new(&sdk_config);

        if own_bucket {
            info!("Creating bucket for demo: {}", bucket.0);
            s3_client
                .create_bucket()
                .bucket(bucket.0.clone())
                .create_bucket_configuration(
                    CreateBucketConfiguration::builder()

.location_constraint(aws_sdk_s3::types::BucketLocationConstraint::from(
                            sdk_config.region().unwrap().as_ref(),
                        ))
                        .build(),
                )
                .send()
                .await
                .unwrap();
        }

        Self::new(
            aws_sdk_iam::Client::new(&sdk_config),
            aws_sdk_lambda::Client::new(&sdk_config),
            s3_client,
            lambda_name,
            role_name,
            bucket,
            OwnBucket(own_bucket),
        )
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
```

```
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }

    /**
     * Create a function, uploading from a zip file.
     */
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
 anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
```

```
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Create an IAM execution role for the managed Lambda function.
     * If the role already exists, use that instead.
     */
    async fn create_role(&self) -> Result<aws_sdk_iam::types::Role, CreateRoleError>
{
        info!("Creating execution role for function");
        let get_role = self
            .iam_client
            .get_role()
            .role_name(self.role_name.clone())
            .send()
            .await;
        if let Ok(get_role) = get_role {
            if let Some(role) = get_role.role {
                return Ok(role);
            }
        }

        let create_role = self
            .iam_client
            .create_role()
```

```
            .role_name(self.role_name.clone())
            .assume_role_policy_document(ROLE_POLICY_DOCUMENT)
            .send()
            .await;

        match create_role {
            Ok(create_role) => match create_role.role {
                Some(role) => Ok(role),
                None => Err(CreateRoleError::generic(
                    ErrorMetadata::builder()
                        .message("CreateRole returned empty success")
                        .build(),
                )),
            },
            Err(err) => Err(err.into_service_error()),
        }
    }

    /**
     * Poll `is_function_ready` with a 1-second delay. It returns when the function
is ready or when there's an error checking the function's state.
     */
    pub async fn wait_for_function_ready(&self) -> Result<(), anyhow::Error> {
        info!("Waiting for function");
        while !self.is_function_ready(None).await? {
            info!("Function is not ready, sleeping 1s");
            tokio::time::sleep(Duration::from_secs(1)).await;
        }
        Ok(())
    }

    /**
     * Check if a Lambda function is ready to be invoked.
     * A Lambda function is ready for this scenario when its state is active and its
LastUpdateStatus is Successful.
     * Additionally, if a sha256 is provided, the function must have that as its
current code hash.
     * Any missing properties or failed requests will be reported as an Err.
     */
    async fn is_function_ready(
        &self,
        expected_code_sha256: Option<&str>,
    ) -> Result<bool, anyhow::Error> {
        match self.get_function().await {
```

```
            Ok(func) => {
                if let Some(config) = func.configuration() {
                    if let Some(state) = config.state() {
                        info!(?state, "Checking if function is active");
                        if !matches!(state, State::Active) {
                            return Ok(false);
                        }
                    }
                    match config.last_update_status() {
                        Some(last_update_status) => {
                            info!(?last_update_status, "Checking if function is
ready");
                            match last_update_status {
                                LastUpdateStatus::Successful => {
                                    // continue
                                }
                                LastUpdateStatus::Failed |
LastUpdateStatus::InProgress => {
                                    return Ok(false);
                                }
                                unknown => {
                                    warn!(
                                        status_variant = unknown.as_str(),
                                        "LastUpdateStatus unknown"
                                    );
                                    return Err(anyhow!(
                                        "Unknown LastUpdateStatus, fn config is
{config:?}"
                                    ));
                                }
                            }
                        }
                        None => {
                            warn!("Missing last update status");
                            return Ok(false);
                        }
                    };
                    if expected_code_sha256.is_none() {
                        return Ok(true);
                    }
                    if let Some(code_sha256) = config.code_sha256() {
                        return Ok(code_sha256 ==
expected_code_sha256.unwrap_or_default());
                    }
```

```
                }
            }
            Err(e) => {
                warn!(?e, "Could not get function while waiting");
            }
        }
        Ok(false)
    }

    /** Get the Lambda function with this Manager's name. */
    pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error> {
        info!("Getting lambda function");
        self.lambda_client
            .get_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput, anyhow::Error>
{
        info!("Listing lambda functions");
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
```

```
    }

    /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
    pub async fn update_function_code(
        &self,
        zip_file: PathBuf,
        key: String,
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;
```

```
        self.wait_for_function_ready().await?;

        Ok(updated)
    }

    /** Delete a function and its role, and if possible or necessary, its associated
code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
```

```
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }

    pub async fn cleanup(
        &self,
        location: Option<String>,
    ) -> (
        (
            Result<DeleteFunctionOutput, anyhow::Error>,
            Result<DeleteRoleOutput, anyhow::Error>,
            Option<Result<DeleteObjectOutput, anyhow::Error>>,
        ),
        Option<Result<DeleteBucketOutput, anyhow::Error>>,
    ) {
        let delete_function = self.delete_function(location).await;

        let delete_bucket = if self.own_bucket {
            info!("Deleting bucket {}", self.bucket);
            if delete_function.2.is_none() ||
delete_function.2.as_ref().unwrap().is_ok() {
                Some(
                    self.s3_client
                        .delete_bucket()
                        .bucket(self.bucket.clone())
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                None
            }
        } else {
            info!("No bucket to clean up");
            None
        };

        (delete_function, delete_bucket)
```

```
    }
}

/**
 * Testing occurs primarily as an integration test running the `scenario` bin
 successfully.
 * Each action relies deeply on the internal workings and state of Amazon Simple
 Storage Service (Amazon S3), Lambda, and IAM working together.
 * It is therefore infeasible to mock the clients to test the individual actions.
 */
#[cfg(test)]
mod test {
    use super::{InvokeArgs, Operation};
    use serde_json::json;

    /** Make sure that the JSON output of serializing InvokeArgs is what's expected
 by the calculator. */
    #[test]
    fn test_serialize() {
        assert_eq!(json!(InvokeArgs::Increment(5)), 5);
        assert_eq!(
            json!(InvokeArgs::Arithmetic(Operation::Plus, 5, 7)).to_string(),
            r#"{"op":"plus","i":5,"j":7}"#.to_string(),
        );
    }
}
```

Eine Binärdatei, um das Szenario vom Anfang bis Ende auszuführen, wobei Befehlszeilen-Flags verwendet werden, um einige Verhaltensweisen zu steuern. Diese Datei ist src/bin /scenario.rs in der Kiste.

```
/*
## Service actions

Service actions wrap the SDK call, taking a client and any specific parameters
 necessary for the call.

* CreateFunction
* GetFunction
* ListFunctions
* Invoke
```

```
* UpdateFunctionCode
* UpdateFunctionConfiguration
* DeleteFunction

## Scenario
A scenario runs at a command prompt and prints output to the user on the result
 of each service action. A scenario can run in one of two ways: straight through,
 printing out progress as it goes, or as an interactive question/answer script.

## Getting started with functions

Use an SDK to manage AWS Lambda functions: create a function, invoke it, update its
 code, invoke it again, view its output and logs, and delete it.

This scenario uses two Lambda handlers:
_Note: Handlers don't use AWS SDK API calls._

The increment handler is straightforward:

1. It accepts a number, increments it, and returns the new value.
2. It performs simple logging of the result.

The arithmetic handler is more complex:
1. It accepts a set of actions ['plus', 'minus', 'times', 'divided-by'] and two
 numbers, and returns the result of the calculation.
2. It uses an environment variable to control log level (such as DEBUG, INFO,
 WARNING, ERROR).
It logs a few things at different levels, such as:
    * DEBUG: Full event data.
    * INFO: The calculation result.
    * WARN~ING~: When a divide by zero error occurs.
    * This will be the typical `RUST_LOG` variable.


The steps of the scenario are:

1. Create an AWS Identity and Access Management (IAM) role that meets the following
 requirements:
    * Has an assume_role policy that grants 'lambda.amazonaws.com' the
 'sts:AssumeRole' action.
    * Attaches the 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' managed role.
    * _You must wait for ~10 seconds after the role is created before you can use
 it!_
```

```
2. Create a function (CreateFunction) for the increment handler by packaging it as a
 zip and doing one of the following:
    * Adding it with CreateFunction Code.ZipFile.
    * --or--
    * Uploading it to Amazon Simple Storage Service (Amazon S3) and adding it with
 CreateFunction Code.S3Bucket/S3Key.
    * _Note: Zipping the file does not have to be done in code._
    * If you have a waiter, use it to wait until the function is active. Otherwise,
 call GetFunction until State is Active.
3. Invoke the function with a number and print the result.
4. Update the function (UpdateFunctionCode) to the arithmetic handler by packaging
 it as a zip and doing one of the following:
    * Adding it with UpdateFunctionCode ZipFile.
    * --or--
    * Uploading it to Amazon S3 and adding it with UpdateFunctionCode S3Bucket/
S3Key.
5. Call GetFunction until Configuration.LastUpdateStatus is 'Successful' (or
 'Failed').
6. Update the environment variable by calling UpdateFunctionConfiguration and pass
 it a log level, such as:
    * Environment={'Variables': {'RUST_LOG': 'TRACE'}}
7. Invoke the function with an action from the list and a couple of values. Include
 LogType='Tail' to get logs in the result. Print the result of the calculation and
 the log.
8. [Optional] Invoke the function to provoke a divide-by-zero error and show the log
 result.
9. List all functions for the account, using pagination (ListFunctions).
10. Delete the function (DeleteFunction).
11. Delete the role.

Each step should use the function created in Service Actions to abstract calling the
 SDK.
 */

use aws_sdk_lambda::{operation::invoke::InvokeOutput, types::Environment};
use clap::Parser;
use std::{collections::HashMap, path::PathBuf};
use tracing::{debug, info, warn};
use tracing_subscriber::EnvFilter;

use lambda_code_examples::actions::{
    InvokeArgs::{Arithmetic, Increment},
    LambdaManager, Operation,
};
```

```
#[derive(Debug, Parser)]
pub struct Opt {
    /// The AWS Region.
    #[structopt(short, long)]
    pub region: Option<String>,

    // The bucket to use for the FunctionCode.
    #[structopt(short, long)]
    pub bucket: Option<String>,

    // The name of the Lambda function.
    #[structopt(short, long)]
    pub lambda_name: Option<String>,

    // The number to increment.
    #[structopt(short, long, default_value = "12")]
    pub inc: i32,

    // The left operand.
    #[structopt(long, default_value = "19")]
    pub num_a: i32,

    // The right operand.
    #[structopt(long, default_value = "23")]
    pub num_b: i32,

    // The arithmetic operation.
    #[structopt(short, long, default_value = "plus")]
    pub operation: Operation,

    #[structopt(long)]
    pub cleanup: Option<bool>,

    #[structopt(long)]
    pub no_cleanup: Option<bool>,
}

fn code_path(lambda: &str) -> PathBuf {
    PathBuf::from(format!("../target/lambda/{lambda}/bootstrap.zip"))
}

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
```

```
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}

async fn main_block(
    opt: &Opt,
    manager: &LambdaManager,
    code_location: String,
) -> Result<(), anyhow::Error> {
    let invoke = manager.invoke(Increment(opt.inc)).await?;
    log_invoke_output(&invoke, "Invoked function configured as increment");

    let update_code = manager
        .update_function_code(code_path("arithmetic"), code_location.clone())
        .await?;

    let code_sha256 = update_code.code_sha256().unwrap_or("Unknown SHA");
    info!(?code_sha256, "Updated function code with arithmetic.zip");

    let arithmetic_args = Arithmetic(opt.operation, opt.num_a, opt.num_b);
    let invoke = manager.invoke(arithmetic_args).await?;
    log_invoke_output(&invoke, "Invoked function configured as arithmetic");

    let update = manager
        .update_function_configuration(
            Environment::builder()
                .set_variables(Some(HashMap::from([(
                    "RUST_LOG".to_string(),
                    "trace".to_string(),
                )])))
                .build(),
        )
        .await?;
    let updated_environment = update.environment();
    info!(?updated_environment, "Updated function configuration");
```

```
    let invoke = manager
        .invoke(Arithmetic(opt.operation, opt.num_a, opt.num_b))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with increased logging",
    );

    let invoke = manager
        .invoke(Arithmetic(Operation::DividedBy, opt.num_a, 0))
        .await?;
    log_invoke_output(
        &invoke,
        "Invoked function configured as arithmetic with divide by zero",
    );

    Ok::<(), anyhow::Error>(())
}

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt()
        .without_time()
        .with_file(true)
        .with_line_number(true)
        .with_env_filter(EnvFilter::from_default_env())
        .init();

    let opt = Opt::parse();
    let manager = LambdaManager::load_from_env(opt.lambda_name.clone(),
 opt.bucket.clone()).await;

    let key = match manager.create_function(code_path("increment")).await {
        Ok(init) => {
            info!(?init, "Created function, initially with increment.zip");
            let run_block = main_block(&opt, &manager, init.clone()).await;
            info!(?run_block, "Finished running example, cleaning up");
            Some(init)
        }
        Err(err) => {
            warn!(?err, "Error happened when initializing function");
            None
        }
    };
```

```
    if Some(false) == opt.cleanup || Some(true) == opt.no_cleanup {
        info!("Skipping cleanup")
    } else {
        let delete = manager.cleanup(key).await;
        info!(?delete, "Deleted function & cleaned up resources");
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFunction`

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /**
     * Create a function, uploading from a zip file.
     */
```

```
    pub async fn create_function(&self, zip_file: PathBuf) -> Result<String,
 anyhow::Error> {
        let code = self.prepare_function(zip_file, None).await?;

        let key = code.s3_key().unwrap().to_string();

        let role = self.create_role().await.map_err(|e| anyhow!(e))?;

        info!("Created iam role, waiting 15s for it to become active");
        tokio::time::sleep(Duration::from_secs(15)).await;

        info!("Creating lambda function {}", self.lambda_name);
        let _ = self
            .lambda_client
            .create_function()
            .function_name(self.lambda_name.clone())
            .code(code)
            .role(role.arn())
            .runtime(aws_sdk_lambda::types::Runtime::Providedal2)
            .handler("_unused")
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        self.lambda_client
            .publish_version()
            .function_name(self.lambda_name.clone())
            .send()
            .await?;

        Ok(key)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
```

```
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;

        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- API-Details finden Sie CreateFunction in der AWS SDK for Rust API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Delete a function and its role, and if possible or necessary, its associated
code object and bucket. */
    pub async fn delete_function(
        &self,
        location: Option<String>,
```

```
    ) -> (
        Result<DeleteFunctionOutput, anyhow::Error>,
        Result<DeleteRoleOutput, anyhow::Error>,
        Option<Result<DeleteObjectOutput, anyhow::Error>>,
    ) {
        info!("Deleting lambda function {}", self.lambda_name);
        let delete_function = self
            .lambda_client
            .delete_function()
            .function_name(self.lambda_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        info!("Deleting iam role {}", self.role_name);
        let delete_role = self
            .iam_client
            .delete_role()
            .role_name(self.role_name.clone())
            .send()
            .await
            .map_err(anyhow::Error::from);

        let delete_object: Option<Result<DeleteObjectOutput, anyhow::Error>> =
            if let Some(location) = location {
                info!("Deleting object {location}");
                Some(
                    self.s3_client
                        .delete_object()
                        .bucket(self.bucket.clone())
                        .key(location)
                        .send()
                        .await
                        .map_err(anyhow::Error::from),
                )
            } else {
                info!(?location, "Skipping delete object");
                None
            };

        (delete_function, delete_role, delete_object)
    }
```

- API-Details finden Sie DeleteFunction in der AWS SDK for Rust API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/** Get the Lambda function with this Manager's name. */
pub async fn get_function(&self) -> Result<GetFunctionOutput, anyhow::Error> {
    info!("Getting lambda function");
    self.lambda_client
        .get_function()
        .function_name(self.lambda_name.clone())
        .send()
        .await
        .map_err(anyhow::Error::from)
}
```

- API-Details finden Sie GetFunction in der AWS SDK for Rust API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Invoke the lambda function using calculator InvokeArgs. */
    pub async fn invoke(&self, args: InvokeArgs) -> Result<InvokeOutput,
 anyhow::Error> {
        info!(?args, "Invoking {}", self.lambda_name);
        let payload = serde_json::to_string(&args)?;
        debug!(?payload, "Sending payload");
        self.lambda_client
            .invoke()
            .function_name(self.lambda_name.clone())
            .payload(Blob::new(payload))
            .send()
            .await
            .map_err(anyhow::Error::from)
    }

fn log_invoke_output(invoke: &InvokeOutput, message: &str) {
    if let Some(payload) = invoke.payload().cloned() {
        let payload = String::from_utf8(payload.into_inner());
        info!(?payload, message);
    } else {
        info!("Could not extract payload")
    }
    if let Some(logs) = invoke.log_result() {
        debug!(?logs, "Invoked function logs")
    } else {
        debug!("Invoked function had no logs")
    }
}
```

- Weitere API-Informationen finden Sie unter Aufrufen in der API-Referenz zum AWS -SDK für Rust.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** List all Lambda functions in the current Region. */
    pub async fn list_functions(&self) -> Result<ListFunctionsOutput, anyhow::Error>
{
        info!("Listing lambda functions");
        self.lambda_client
            .list_functions()
            .send()
            .await
            .map_err(anyhow::Error::from)
    }
```

- API-Details finden Sie ListFunctions in der AWS SDK for Rust API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Given a Path to a zip file, update the function's code and wait for the
update to finish. */
    pub async fn update_function_code(
        &self,
        zip_file: PathBuf,
        key: String,
```

```
    ) -> Result<UpdateFunctionCodeOutput, anyhow::Error> {
        let function_code = self.prepare_function(zip_file, Some(key)).await?;

        info!("Updating code for {}", self.lambda_name);
        let update = self
            .lambda_client
            .update_function_code()
            .function_name(self.lambda_name.clone())
            .s3_bucket(self.bucket.clone())
            .s3_key(function_code.s3_key().unwrap().to_string())
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(update)
    }

    /**
     * Upload function code from a path to a zip file.
     * The zip file must have an AL2 Linux-compatible binary called `bootstrap`.
     * The easiest way to create such a zip is to use `cargo lambda build --output-
format Zip`.
     */
    async fn prepare_function(
        &self,
        zip_file: PathBuf,
        key: Option<String>,
    ) -> Result<FunctionCode, anyhow::Error> {
        let body = ByteStream::from_path(zip_file).await?;

        let key = key.unwrap_or_else(|| format!("{}_code", self.lambda_name));

        info!("Uploading function code to s3://{}/{}", self.bucket, key);
        let _ = self
            .s3_client
            .put_object()
            .bucket(self.bucket.clone())
            .key(key.clone())
            .body(body)
            .send()
            .await?;
```

```
        Ok(FunctionCode::builder()
            .s3_bucket(self.bucket.clone())
            .s3_key(key)
            .build())
    }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK for Rust API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    /** Update the environment for a function. */
    pub async fn update_function_configuration(
        &self,
        environment: Environment,
    ) -> Result<UpdateFunctionConfigurationOutput, anyhow::Error> {
        info!(
            ?environment,
            "Updating environment for {}", self.lambda_name
        );
        let updated = self
            .lambda_client
            .update_function_configuration()
            .function_name(self.lambda_name.clone())
            .environment(environment)
            .send()
            .await
            .map_err(anyhow::Error::from)?;

        self.wait_for_function_ready().await?;

        Ok(updated)
```

```
    }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK for Rust API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
```

```
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
```

```
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Aufrufen einer Lambda-Funktion über einen Kinesis-Auslöser

Das folgende Codebeispiel veranschaulicht, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem Kinesis-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die Kinesis-Nutzlast ab, dekodiert von Base64 und protokolliert den Datensatzinhalt.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Kinesis-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::kinesis::KinesisEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) -> Result<(), Error> {
    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    event.payload.records.iter().for_each(|record| {
        tracing::info!("EventId:
 {}",record.event_id.as_deref().unwrap_or_default());

        let record_data = std::str::from_utf8(&record.kinesis.data);

        match record_data {
            Ok(data) => {
                // log the record data
                tracing::info!("Data: {}", data);
            }
            Err(e) => {
                tracing::error!("Error: {}", e);
            }
        }
```

```
    });

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Aufrufen einer Lambda-Funktion über einen DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DynamoDB-Stream ausgelöstes Ereignis empfängt. Die Funktion ruft die DynamoDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines DynamoDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
```

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<Event>) ->Result<(), Error> {

    let records = &event.payload.records;
    tracing::info!("event payload: {:?}",records);
    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_dynamo_dbrecord(record);
    }

    tracing::info!("Dynamo db records processed");

    // Prepare the response
    Ok(())

}

fn log_dynamo_dbrecord(record: &EventRecord)-> Result<(), Error>{
    tracing::info!("EventId: {}", record.event_id);
    tracing::info!("EventName: {}", record.event_name);
    tracing::info!("DynamoDB Record: {:?}", record.change );
    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
```

```
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Aufrufen einer Lambda-Funktion über einen Amazon DocumentDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein durch den Empfang von Datensätzen aus einem DocumentDB-Änderungsstream ausgelöstes Ereignis empfängt. Die Funktion ruft die DocumentDB-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon DocumentDB-Ereignisses mit Lambda unter Verwendung von Rust.

```
use lambda_runtime::{service_fn, tracing, Error, LambdaEvent};
use aws_lambda_events::{
    event::documentdb::{DocumentDbEvent, DocumentDbInnerEvent},
   };


// Built with the following dependencies:
//lambda_runtime = "0.11.1"
//serde_json = "1.0"
//tokio = { version = "1", features = ["macros"] }
```

```
//tracing = { version = "0.1", features = ["log"] }
//tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }
//aws_lambda_events = "0.15.0"

async fn function_handler(event: LambdaEvent<DocumentDbEvent>) ->Result<(), Error> {

    tracing::info!("Event Source ARN: {:?}", event.payload.event_source_arn);
    tracing::info!("Event Source: {:?}", event.payload.event_source);

    let records = &event.payload.events;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(());
    }

    for record in records{
        log_document_db_event(record);
    }

    tracing::info!("Document db records processed");

    // Prepare the response
    Ok(())

}

fn log_document_db_event(record: &DocumentDbInnerEvent)-> Result<(), Error>{
    tracing::info!("Change Event: {:?}", record.event);

    Ok(())

}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
    .with_max_level(tracing::Level::INFO)
    .with_target(false)
    .without_time()
    .init();

    let func = service_fn(function_handler);
```

```
    lambda_runtime::run(func).await?;
    Ok(())

}
```

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error> {
```

```
    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");
```

```
    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket name
 to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features =
 ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
```

```
}
```

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
```

```
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Kinesis-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem Kinesis-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei Kinesis-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::kinesis::KinesisEvent,
    kinesis::KinesisEventRecord,
    streams::{KinesisBatchItemFailure, KinesisEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<KinesisEvent>) ->
 Result<KinesisEventResponse, Error> {
    let mut response = KinesisEventResponse {
        batch_item_failures: vec![],
    };

    if event.payload.records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
```

```
    }

    for record in &event.payload.records {
        tracing::info!(
            "EventId: {}",
            record.event_id.as_deref().unwrap_or_default()
        );

        let record_processing_result = process_record(record);

        if record_processing_result.is_err() {
            response.batch_item_failures.push(KinesisBatchItemFailure {
                item_identifier: record.kinesis.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }

    tracing::info!(
        "Successfully processed {} records",
        event.payload.records.len()
    );

    Ok(response)
}

fn process_record(record: &KinesisEventRecord) -> Result<(), Error> {
    let record_data = std::str::from_utf8(record.kinesis.data.as_slice());

    if let Some(err) = record_data.err() {
        tracing::error!("Error: {}", err);
        return Err(Error::from(err));
    }

    let record_data = record_data.unwrap_or_default();

    // do something interesting with the data
    tracing::info!("Data: {}", record_data);

    Ok(())
```

```
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem DynamoDB-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse von einem DynamoDB-Stream empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von DynamoDB-Batchelementfehlern mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::{
    event::dynamodb::{Event, EventRecord, StreamRecord},
    streams::{DynamoDbBatchItemFailure, DynamoDbEventResponse},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Process the stream record
fn process_record(record: &EventRecord) -> Result<(), Error> {
    let stream_record: &StreamRecord = &record.change;
```

```
    // process your stream record here...
    tracing::info!("Data: {:?}", stream_record);

    Ok(())
}

/// Main Lambda handler here...
async fn function_handler(event: LambdaEvent<Event>) ->
 Result<DynamoDbEventResponse, Error> {
    let mut response = DynamoDbEventResponse {
        batch_item_failures: vec![],
    };

    let records = &event.payload.records;

    if records.is_empty() {
        tracing::info!("No records found. Exiting.");
        return Ok(response);
    }

    for record in records {
        tracing::info!("EventId: {}", record.event_id);

        // Couldn't find a sequence number
        if record.change.sequence_number.is_none() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: Some("".to_string()),
            });
            return Ok(response);
        }

        // Process your record here...
        if process_record(record).is_err() {
            response.batch_item_failures.push(DynamoDbBatchItemFailure {
                item_identifier: record.change.sequence_number.clone(),
            });
            /* Since we are working with streams, we can return the failed item
 immediately.
            Lambda will immediately begin to retry processing from this failed item
 onwards. */
            return Ok(response);
        }
    }
```

```
    tracing::info!("Successfully processed {} record(s)", records.len());

    Ok(response)
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
```

```
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<SqsBatchResponse,
 Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

AWS Beiträge der Gemeinschaft

Erstellen und Testen einer Serverless-Anwendung

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung mithilfe von API Gateway mit Lambda und DynamoDB erstellt und getestet wird.

SDK für Rust

Es wird gezeigt, wie eine Serverless-Anwendung, bestehend aus einem API Gateway mit Lambda und DynamoDB, mithilfe des Rust SDK erstellt und getestet wird.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda

MediaLive Beispiele zur Verwendung des SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren MediaLive.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListInputs`

Das folgende Codebeispiel zeigt die Verwendung`ListInputs`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Ihre MediaLive Eingabenamen und ARNs in der Region auf.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListInputs in der AWS SDK for Rust API-Referenz.

MediaPackage Beispiele zur Verwendung des SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren MediaPackage.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListChannels`

Das folgende Codebeispiel zeigt die Verwendung`ListChannels`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Kanal-ARNs und Beschreibungen auf.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();

        println!("  Description : {}", description);
        println!("  ARN :         {}", arn);
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListChannels in der AWS SDK for Rust API-Referenz.

`ListOriginEndpoints`

Das folgende Codebeispiel zeigt die Verwendung`ListOriginEndpoints`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Listen Sie Endpunktbeschreibungen und URLs auf.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
        println!("  URL :        {}", endpoint_url);
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListOriginEndpoints in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon MSK unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon MSK Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-MSK-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Datensätzen von einem Amazon MSK-Cluster ausgelöst wird. Die Funktion ruft die MSK-Nutzdaten ab und protokolliert den Inhalt des Datensatzes.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines Amazon MSK-Ereignisses mit Lambda unter Verwendung von Rust.

```
use aws_lambda_events::event::kafka::KafkaEvent;
use lambda_runtime::{run, service_fn, tracing, Error, LambdaEvent};
use base64::prelude::*;
use serde_json::{Value};
use tracing::{info};

/// Pre-Requisites:
/// 1. Install Cargo Lambda - see https://www.cargo-lambda.info/guide/getting-
started.html
/// 2. Add packages tracing, tracing-subscriber, serde_json, base64
///
/// This is the main body for the function.
/// Write your code inside it.
/// There are some code example in the following URLs:
/// - https://github.com/awslabs/aws-lambda-rust-runtime/tree/main/examples
/// - https://github.com/aws-samples/serverless-rust-demo/

async fn function_handler(event: LambdaEvent<KafkaEvent>) -> Result<Value, Error> {

    let payload = event.payload.records;

    for (_name, records) in payload.iter() {

        for record in records {

         let record_text = record.value.as_ref().ok_or("Value is None")?;
         info!("Record: {}", &record_text);

         // perform Base64 decoding
         let record_bytes = BASE64_STANDARD.decode(record_text)?;
         let message = std::str::from_utf8(&record_bytes)?;

         info!("Message: {}", message);
```

```
        }

    }

    Ok(().into())
}

#[tokio::main]
async fn main() -> Result<(), Error> {

    // required to enable CloudWatch error logging by the runtime
    tracing::init_default_subscriber();
    info!("Setup CW subscriber!");

    run(service_fn(function_handler)).await
}
```

Beispiele für Amazon Polly unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Polly Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_voices().send().await?;

    println!("Voices:");

    let voices = resp.voices();
    for voice in voices {
        println!("  Name:     {}", voice.name().unwrap_or("No name!"));
        println!(
            "  Language: {}",
            voice.language_name().unwrap_or("No language!")
        );

        println!();
    }

    println!("Found {} voices", voices.len());

    Ok(())
}
```

- API-Details finden Sie DescribeVoices in der AWS SDK for Rust API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons();

    for lexicon in lexicons {
        println!("  Name:     {}", lexicon.name().unwrap_or_default());
        println!(
            "  Language: {:?}\n",
            lexicon
                .attributes()
                .as_ref()
                .map(|attrib| attrib
                    .language_code
                    .as_ref()
                    .expect("languages must have language codes"))
                .expect("languages must have attributes")
        );
    }

    println!();
    println!("Found {} lexicons.", lexicons.len());
    println!();

    Ok(())
}
```

- API-Details finden Sie ListLexicons in der AWS SDK for Rust API-Referenz.

`PutLexicon`

Das folgende Codebeispiel zeigt die Verwendung`PutLexicon`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
 Result<(), Error> {
    let content = format!("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
    <lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-lexicon
\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
    xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://
www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\"
    alphabet=\"ipa\" xml:lang=\"en-US\">
    <lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
    </lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- API-Details finden Sie PutLexicon in der AWS SDK for Rust API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- API-Details finden Sie SynthesizeSpeech in der AWS SDK for Rust API-Referenz.

Szenarien

Text in Sprache und zurück in Text konvertieren

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Klartext-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, laden Sie die Audiodatei in einen Amazon S3-Bucket hoch, verwenden Sie Amazon Transcribe, um diese Audiodatei in Text zu konvertieren, und zeigen Sie den Text an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Beispiele für Amazon RDS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Serverless-Beispiele

Serverless-Beispiele

Herstellen einer Verbindung mit einer Amazon-RDS-Datenbank in einer Lambda-Funktion

Die folgenden Codebeispiele veranschaulichen, wie eine Lambda-Funktion implementiert wird, die eine Verbindung zu einer RDS-Datenbank herstellt. Die Funktion stellt eine einfache Datenbankanfrage und gibt das Ergebnis zurück.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Herstellen einer Verbindung zu einer Amazon-RDS-Datenbank in einer Lambda-Funktion mit Rust.

```
use aws_config::BehaviorVersion;
use aws_credential_types::provider::ProvideCredentials;
use aws_sigv4::{
    http_request::{sign, SignableBody, SignableRequest, SigningSettings},
    sign::v4,
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use serde_json::{json, Value};
use sqlx::postgres::PgConnectOptions;
use std::env;
use std::time::{Duration, SystemTime};

const RDS_CERTS: &[u8] = include_bytes!("global-bundle.pem");

async fn generate_rds_iam_token(
    db_hostname: &str,
    port: u16,
    db_username: &str,
) -> Result<String, Error> {
    let config = aws_config::load_defaults(BehaviorVersion::v2024_03_28()).await;

    let credentials = config
        .credentials_provider()
```

```
        .expect("no credentials provider found")
        .provide_credentials()
        .await
        .expect("unable to load credentials");
    let identity = credentials.into();
    let region = config.region().unwrap().to_string();

    let mut signing_settings = SigningSettings::default();
    signing_settings.expires_in = Some(Duration::from_secs(900));
    signing_settings.signature_location =
 aws_sigv4::http_request::SignatureLocation::QueryParams;

    let signing_params = v4::SigningParams::builder()
        .identity(&identity)
        .region(&region)
        .name("rds-db")
        .time(SystemTime::now())
        .settings(signing_settings)
        .build()?;

    let url = format!(
        "https://{db_hostname}:{port}/?Action=connect&DBUser={db_user}",
        db_hostname = db_hostname,
        port = port,
        db_user = db_username
    );

    let signable_request =
        SignableRequest::new("GET", &url, std::iter::empty(),
 SignableBody::Bytes(&[]))
            .expect("signable request");

    let (signing_instructions, _signature) =
        sign(signable_request, &signing_params.into())?.into_parts();

    let mut url = url::Url::parse(&url).unwrap();
    for (name, value) in signing_instructions.params() {
        url.query_pairs_mut().append_pair(name, &value);
    }

    let response = url.to_string().split_off("https://".len());

    Ok(response)
}
```

```
#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(handler)).await
}

async fn handler(_event: LambdaEvent<Value>) -> Result<Value, Error> {
    let db_host = env::var("DB_HOSTNAME").expect("DB_HOSTNAME must be set");
    let db_port = env::var("DB_PORT")
        .expect("DB_PORT must be set")
        .parse::<u16>()
        .expect("PORT must be a valid number");
    let db_name = env::var("DB_NAME").expect("DB_NAME must be set");
    let db_user_name = env::var("DB_USERNAME").expect("DB_USERNAME must be set");

    let token = generate_rds_iam_token(&db_host, db_port, &db_user_name).await?;

    let opts = PgConnectOptions::new()
        .host(&db_host)
        .port(db_port)
        .username(&db_user_name)
        .password(&token)
        .database(&db_name)
        .ssl_root_cert_from_pem(RDS_CERTS.to_vec())
        .ssl_mode(sqlx::postgres::PgSslMode::Require);

    let pool = sqlx::postgres::PgPoolOptions::new()
        .connect_with(opts)
        .await?;

    let result: i32 = sqlx::query_scalar("SELECT $1 + $2")
        .bind(3)
        .bind(2)
        .fetch_one(&pool)
        .await?;

    println!("Result: {:?}", result);

    Ok(json!({
        "statusCode": 200,
        "content-type": "text/plain",
        "body": format!("The selected sum is: {result}")
    }))
}
```

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
```

```
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);

    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

- API-Details finden Sie ExecuteStatement in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon Rekognition unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Rekognition Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Gesichter in einem Bild erkennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

EXIF- und andere Bildinformationen speichern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Beispiele für Route 53 unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Route 53 Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListHostedZones`

Das folgende Codebeispiel zeigt die Verwendung`ListHostedZones`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),
 aws_sdk_route53::Error> {
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;

    println!(
        "Number of hosted zones in region : {}",
        hosted_zone_count.hosted_zone_count(),
    );

    let hosted_zones = client.list_hosted_zones().send().await?;

    println!("Zones:");

    for hz in hosted_zones.hosted_zones() {
        let zone_name = hz.name();
        let zone_id = hz.id();
```

```
        println!("  ID :   {}", zone_id);
        println!("  Name : {}", zone_name);
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListHostedZones in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon S3 unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien
- Serverless-Beispiele

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt, wie Sie mit Amazon S3 beginnen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
 present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
```

```
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }

    // Prepare a ByteStream around the file, and upload the object using that
ByteStream.
    let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
        .await
        .map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to create bytestream for {filepath:?} ({err:?})"
            ))
        })?;
    let resp = client
        .put_object()
        .bucket(bucket)
        .key(key)
        .body(body)
        .send()
        .await?;

    println!(
        "Upload success. Version: {:?}",
        resp.version_id()
            .expect("S3 Object upload missing version ID")
    );

    // Retrieve the just-uploaded object.
    let resp = client.get_object().bucket(bucket).key(key).send().await?;
    println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
    println!("version: {}", resp.version_id().unwrap_or("(missing)"));

    Ok(())
```

```
}
```

S3ExampleError Nebenkosten.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
```

```
}
```

- API-Details finden Sie ListBuckets in der AWS SDK for Rust API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die binäre Crate-Datei, die das Szenario ausführt.

```
#![allow(clippy::result_large_err)]

//!  Purpose
//!  Shows how to use the AWS SDK for Rust to get started using
//!  Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects into
 and out of it,
//!  and delete all resources at the end of the demo.
//!
//!  This example follows the steps in "Getting started with Amazon S3" in the
 Amazon S3
//!  user guide.
```

```
//!  - https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name, key,
 target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
 &key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
 &key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
 &target_key)
            .await?;
```

```
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

Allgemeine Aktionen, die im Szenario verwendet werden.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
```

```
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
        .map_err(S3ExampleError::from)
}

/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
```

```
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object} with
 etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}

pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) -> Result<(),
 S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
```

```
        }
    }

    Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
```

```
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CompleteMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CompleteMultipartUpload`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
```

```
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

- API-Details finden Sie CompleteMultipartUpload in der AWS SDK for Rust API-Referenz.

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object} with
 etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
 aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- API-Details finden Sie CopyObject in der AWS SDK for Rust API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
 S3ExampleError> {
    let constraint =
 aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
 task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
 se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
```

```
    })
}
```

- API-Details finden Sie CreateBucket in der AWS SDK for Rust API-Referenz.

`CreateMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultipartUpload`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
```

```
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- API-Details finden Sie CreateMultipartUpload in der AWS SDK for Rust API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

- API-Details finden Sie DeleteBucket in der AWS SDK for Rust API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    // There are no modeled errors to handle when deleting an object.

    Ok(())
}
```

- API-Details finden Sie DeleteObject in der AWS SDK for Rust API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building object
 keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for delete_object:
 {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object input
 {err:?}"))
                })?,
        )
        .send()
        .await?;
    Ok(())
}
```

- API-Details finden Sie DeleteObjects in der AWS SDK for Rust API-Referenz.

`GetBucketLocation`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketLocation`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
```

```
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- API-Details finden Sie GetBucketLocation in der AWS SDK for Rust API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
```

```
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
 {err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
}
```

- API-Details finden Sie GetObject in der AWS SDK for Rust API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();
```

```
    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- API-Details finden Sie ListBuckets in der AWS SDK for Rust API-Referenz.

`ListObjectVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectVersions`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListObjectVersions in der AWS SDK for Rust API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) -> Result<(),
 S3ExampleError> {
    let mut response = client
        .list_objects_v2()
```

```
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK for Rust API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
```

```
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
 aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- API-Details finden Sie PutObject in der AWS SDK for Rust API-Referenz.

`UploadPart`

Das folgende Codebeispiel zeigt die Verwendung`UploadPart`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
```

```
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
```

```
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;
```

```
    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();
```

```
    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;
```

- API-Details finden Sie UploadPart in der AWS SDK for Rust API-Referenz.

Szenarien

Text in Sprache und zurück in Text konvertieren

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Klartext-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, laden Sie die Audiodatei in einen Amazon S3-Bucket hoch, verwenden Sie Amazon Transcribe, um diese Audiodatei in Text zu konvertieren, und zeigen Sie den Text an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie Vorsignieranforderungen für GET-S3-Objekte.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Erstellen Sie Vorsignieranforderungen für GET- und PUT-S3-Objekte.

```
async fn put_object(
    client: &Client,
```

```
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
 std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Gesichter in einem Bild erkennen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

Abrufen eines Objekts aus einem Bucket abrufen, wenn es geändert wurde

Das folgende Codebeispiel zeigt, wie Sie Daten aus einem Objekt in einem S3 Bucket lesen, jedoch nur, wenn dieser Bucket seit dem letzten Abruf nicht geändert wurde.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
```

```
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };
```

```
    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => match err {
            SdkError::ServiceError(err) => {
```

```
                // Get the raw HTTP response. If its status is 304, the
                // object has not changed. This is the expected code path.
                let http = err.raw();
                match http.status().as_u16() {
                    // If the HTTP status is 304: Not Modified, return a
                    // tuple containing the values of the HTTP
                    // `last-modified` and `etag` headers.
                    304 => (
                        Ok(DateTime::from_str(
                            http.headers().get("last-modified").unwrap(),
                            DateTimeFormat::HttpDate,
                        )
                        .unwrap()),
                        http.headers().get("etag").map(|t| t.into()).unwrap(),
                    ),
                    // Any other HTTP status code is returned as an
                    // `SdkError::ServiceError`.
                    _ => (Err(SdkError::ServiceError(err)), String::new()),
                }
            }
            // Any other kind of error is returned in a tuple containing the
            // error and an empty string.
            _ => (Err(err), String::new()),
        },
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
```

```
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- API-Details finden Sie GetObject in der AWS SDK for Rust API-Referenz.

EXIF- und andere Bildinformationen speichern

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Modul- und Integrationstest mit einem SDK

Das folgende Codebeispiel zeigt Beispiele für bewährte Verfahren beim Schreiben von Einheiten- und Integrationstests mithilfe eines AWS SDK.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Cargo.toml zum Testen von Beispielen.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
```

```
path = "src/main.rs"
```

Beispiel für Modultests mit automock und einem Service-Wrapper

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
```

```
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
```

```
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
```

```
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}
```

Beispiel für Integrationstests unter Verwendung von StaticReplayClient.

```
use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }
```

```
        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/response_1.xml")))
                    .unwrap(),
            );
```

```
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
```

```
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
    }
}
```

Hoch- oder Herunterladen großer Dateien

Das folgende Codebeispiel veranschaulicht, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter Hochladen eines Objekts mit Multipart-Upload.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
```

```
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        return Err(S3ExampleError::new("Bad file size."));
    }
```

```
    if chunk_count > MAX_CHUNKS {
        return Err(S3ExampleError::new(
            "Too many chunks! Try increasing your chunk size.",
        ));
    }

    let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();

        // Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

    // upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
```

```
    let completed_multipart_upload: CompletedMultipartUpload =
 CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await?;

    let data: GetObjectOutput =
        s3_code_examples::download_object(&client, &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_code_examples::clear_bucket(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_code_examples::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;
```

```
    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket name
 to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

SageMaker KI-Beispiele, die SDK für Rust verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit SageMaker KI Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListNotebookInstances`

Das folgende Codebeispiel zeigt die Verwendung`ListNotebookInstances`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name();

        println!("  Name :          {}", n_name.unwrap_or("Unknown"));
        println!("  Status :        {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }
```

```
    Ok(())
}
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK for Rust API-Referenz.

`ListTrainingJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTrainingJobs`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_jobs(client: &Client) -> Result<(), Error> {
    let job_details = client.list_training_jobs().send().await?;

    println!("Jobs:");

    for j in job_details.training_job_summaries() {
        let name = j.training_job_name().unwrap_or("Unknown");
        let creation_time = j.creation_time().expect("creation
 time").to_chrono_utc()?;
        let training_end_time = j
            .training_end_time()
            .expect("Training end time")
            .to_chrono_utc()?;

        let status = j.training_job_status().expect("training status");
        let duration = training_end_time - creation_time;

        println!("  Name:               {}", name);
        println!(
            "  Creation date/time: {}",
            creation_time.format("%Y-%m-%d@%H:%M:%S")
        );
        println!("  Duration (seconds): {}", duration.num_seconds());
```

```
        println!("  Status:             {:?}", status);

        println!();
    }

    Ok(())
}
```

- API-Details finden Sie ListTrainingJobs in der AWS SDK for Rust API-Referenz.

Beispiele für Secrets Manager unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Secrets Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- API-Details finden Sie GetSecretValue in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon SES API v2 unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon SES API v2 Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateContact`

Das folgende Codebeispiel zeigt die Verwendung`CreateContact`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(), Error>
 {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

- API-Details finden Sie CreateContact in der AWS SDK for Rust API-Referenz.

`CreateContactList`

Das folgende Codebeispiel zeigt die Verwendung`CreateContactList`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
```

```
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- API-Details finden Sie CreateContactList in der AWS SDK for Rust API-Referenz.

`CreateEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailIdentity`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }
```

- API-Details finden Sie CreateEmailIdentity in der AWS SDK for Rust API-Referenz.

`CreateEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailTemplate`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
```

```
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }
```

- API-Details finden Sie CreateEmailTemplate in der AWS SDK for Rust API-Referenz.

`DeleteContactList`

Das folgende Codebeispiel zeigt die Verwendung`DeleteContactList`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }
```

- API-Details finden Sie DeleteContactList in der AWS SDK for Rust API-Referenz.

`DeleteEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailIdentity`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }
```

- API-Details finden Sie DeleteEmailIdentity in der AWS SDK for Rust API-Referenz.

`DeleteEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailTemplate`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
```

```
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- API-Details finden Sie DeleteEmailTemplate in der AWS SDK for Rust API-Referenz.

`GetEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`GetEmailIdentity`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Bestimmt, ob eine E-Mail-Adresse überprüft wurde.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }
```

```
    Ok(())
}
```

- API-Details finden Sie GetEmailIdentity in der AWS SDK for Rust API-Referenz.

`ListContactLists`

Das folgende Codebeispiel zeigt die Verwendung`ListContactLists`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
}
```

- API-Details finden Sie ListContactLists in der AWS SDK for Rust API-Referenz.

`ListContacts`

Das folgende Codebeispiel zeigt die Verwendung`ListContacts`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    println!("Contacts:");

    for contact in resp.contacts() {
        println!("  {}", contact.email_address().unwrap_or_default());
    }

    Ok(())
}
```

- API-Details finden Sie ListContacts in der AWS SDK for Rust API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht an alle Mitglieder der Kontaktliste.

```
async fn send_message(
    client: &Client,
    list: &str,
    from: &str,
    subject: &str,
    message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts();

    let cs: Vec<String> = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default().to_string())
        .collect();

    let mut dest: Destination = Destination::builder().build();
    dest.to_addresses = Some(cs);
    let subject_content = Content::builder()
        .data(subject)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body_content = Content::builder()
        .data(message)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
```

```
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
        .send()
        .await?;

    println!("Email sent to list");

    Ok(())
}
```

Sendet mithilfe einer Vorlage eine Nachricht an alle Mitglieder der Kontaktliste.

```
            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
```

```
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}: {}",
email, e)),
            }
```

- API-Details finden Sie SendEmail in der AWS SDK for Rust API-Referenz.

Szenarien

Newsletter-Szenario

Das folgende Codebeispiel zeigt, wie Sie das Newsletter-Szenario von Amazon SES API v2 ausführen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        match self
            .client
            .create_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list created successfully.")?,
            Err(e) => match e.into_service_error() {
```

```
                CreateContactListError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Contact list already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating contact list: {}", e)),
            },
        }

            match self
                .client
                .create_contact()
                .contact_list_name(CONTACT_LIST_NAME)
                .email_address(email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Contact created for {}", email)?,
                Err(e) => match e.into_service_error() {
                    CreateContactError::AlreadyExistsException(_) => writeln!(
                        self.stdout,
                        "Contact already exists for {}, skipping creation.",
                        email
                    )?,
                    e => return Err(anyhow!("Error creating contact for {}: {}",
email, e)),
                },
            }

        let contacts: Vec<Contact> = match self
            .client
            .list_contacts()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(list_contacts_output) => {
                list_contacts_output.contacts.unwrap().into_iter().collect()
            }
            Err(e) => {
                return Err(anyhow!(
                    "Error retrieving contact list {}: {}",
                    CONTACT_LIST_NAME,
```

```
                    e
                ))
            }
        };

            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
```

```
                Err(e) => return Err(anyhow!("Error sending newsletter to {}: {}",
 email, e)),
            }

        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }

        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
```

```
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }

        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }

            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }

        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
```

```
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für Rust.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Beispiele für Amazon SNS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien
- Serverless-Beispiele

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- API-Details finden Sie CreateTopic in der AWS SDK for Rust API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- API-Details finden Sie ListTopics in der AWS SDK for Rust API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
```

```
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für Rust.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für Rust.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Serverless-Beispiele

Eine Lambda-Funktion über einen Amazon-SNS-Trigger aufrufen

Im folgenden Codebeispiel wird die Implementierung einer Lambda-Funktion veranschaulicht, die ein Ereignis empfängt, das durch das Empfangen von Nachrichten aus einem SNS-Thema ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SNS-Ereignisses mit Lambda unter Verwendung von Rust

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features =
 ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Beispiele für Amazon SQS unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Serverless-Beispiele

Aktionen

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Rufen Sie die erste Amazon-SQS-Warteschlange ab, die in der Region aufgeführt ist.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to proceed.")
        .to_string())
```

```
}
```

- API-Details finden Sie ListQueues in der AWS SDK for Rust API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
 client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}
```

- API-Details finden Sie ReceiveMessage in der AWS SDK for Rust API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
 Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // and message_group_id or configure the queue for
 ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:#?}", rsp);

    Ok(())
}
```

- API-Details finden Sie SendMessage in der AWS SDK for Rust API-Referenz.

Serverless-Beispiele

Aufrufen einer Lambda-Funktion über einen Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine Lambda-Funktion implementiert wird, die ein Ereignis empfängt, das durch den Empfang von Nachrichten aus einer SQS-Warteschlange ausgelöst wird. Die Funktion ruft die Nachrichten aus dem Ereignisparameter ab und protokolliert den Inhalt jeder Nachricht.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Nutzen eines SQS-Ereignisses mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sqs::SqsEvent;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<(), Error> {
    event.payload.records.iter().for_each(|record| {
        // process the record
        tracing::info!("Message body: {}",
 record.body.as_deref().unwrap_or_default())
    });

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        // disable printing the name of the module in every log line.
        .with_target(false)
        // disabling time is handy because CloudWatch will add the ingestion time.
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Melden von Batch-Elementfehlern für Lambda-Funktionen mit einem Amazon-SQS-Auslöser

Das folgende Codebeispiel zeigt, wie eine teilweise Batch-Antwort für Lambda-Funktionen implementiert wird, die Ereignisse aus einer SQS-Warteschlange empfangen. Die Funktion meldet die Batch-Elementfehler in der Antwort und signalisiert Lambda, diese Nachrichten später erneut zu versuchen.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit Serverless-Beispielen.

Melden von Fehlern bei SQS-Batchelementen mit Lambda unter Verwendung von Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::{
    event::sqs::{SqsBatchResponse, SqsEvent},
    sqs::{BatchItemFailure, SqsMessage},
};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

async fn process_record(_: &SqsMessage) -> Result<(), Error> {
    Err(Error::from("Error processing message"))
}

async fn function_handler(event: LambdaEvent<SqsEvent>) -> Result<SqsBatchResponse,
 Error> {
    let mut batch_item_failures = Vec::new();
    for record in event.payload.records {
        match process_record(&record).await {
            Ok(_) => (),
            Err(_) => batch_item_failures.push(BatchItemFailure {
                item_identifier: record.message_id.unwrap(),
            }),
        }
    }

    Ok(SqsBatchResponse {
```

```
        batch_item_failures,
    })
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    run(service_fn(function_handler)).await
}
```

AWS STS Beispiele mit SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust with Aktionen ausführen und allgemeine Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}", e.user_id().unwrap_or_default());
            println!("Account:               {}", e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- API-Details finden Sie AssumeRole in der AWS SDK for Rust API-Referenz.

Codebeispiele für Systems Manager unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Systems Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DescribeParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeParameters`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;

    for param in resp.parameters() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- API-Details finden Sie DescribeParameters in der AWS SDK for Rust API-Referenz.

`GetParameter`

Das folgende Codebeispiel zeigt die Verwendung`GetParameter`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error> {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
```

- API-Details finden Sie GetParameter in der AWS SDK for Rust API-Referenz.

`PutParameter`

Das folgende Codebeispiel zeigt die Verwendung`PutParameter`.

SDK für Rust

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
```

```
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());

    Ok(())
}
```

- API-Details finden Sie PutParameter in der AWS SDK for Rust API-Referenz.

Beispiele für Amazon Transcribe unter Verwendung von SDK für Rust

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Rust mit Amazon Transcribe Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Text in Sprache und zurück in Text konvertieren

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.

- Zeigen Sie den Text an.

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, laden Sie die Audiodatei in einen Amazon S3-Bucket hoch, verwenden Sie Amazon Transcribe, um diese Audiodatei in Text zu konvertieren, und zeigen Sie den Text an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. GitHub

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Codebeispiele für SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie das AWS SDK für SAP ABAP mit verwenden. AWS

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für SAP ABAP Developer Guide — Weitere Informationen zur Verwendung von SAP ABAP mit. AWS

- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- ACM-Beispiele, die das SDK für SAP ABAP verwenden
- Beispiele für API-Gateways, die das SDK für SAP ABAP verwenden
- Beispiele für Application Recovery Controller, die das SDK für SAP ABAP verwenden
- Aurora-Beispiele, die das SDK für SAP ABAP verwenden
- Auto Scaling-Beispiele unter Verwendung des SDK für SAP ABAP
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für SAP ABAP
- Beispiele für Amazon Bedrock Agents Runtime unter Verwendung von SDK für SAP ABAP
- CloudFront Beispiele zur Verwendung des SDK für SAP ABAP
- CloudWatch Beispiele mit SDK für SAP ABAP
- CloudWatch Loggt Beispiele mit SDK für SAP ABAP
- Beispiele für Amazon Cognito Identity Provider, die das SDK für SAP ABAP verwenden
- Amazon Comprehend-Beispiele zur Verwendung des SDK für SAP ABAP
- AWS Config Beispiele zur Verwendung des SDK für SAP ABAP
- AWS Control Tower Beispiele zur Verwendung des SDK für SAP ABAP
- Firehose-Beispiele, die das SDK für SAP ABAP verwenden
- DynamoDB-Beispiele unter Verwendung von SDK für SAP ABAP
- Beispiele für Amazon EC2 unter Verwendung von SDK für SAP ABAP
- Amazon ECR-Beispiele, die das SDK für SAP ABAP verwenden
- Amazon EMR-Beispiele mit SDK für SAP ABAP
- EventBridge Scheduler-Beispiele, die das SDK für SAP ABAP verwenden
- AWS Glue Beispiele zur Verwendung des SDK für SAP ABAP
- HealthImaging Beispiele zur Verwendung des SDK für SAP ABAP
- HealthLake Beispiele zur Verwendung des SDK für SAP ABAP
- IAM-Beispiele, die das SDK für SAP ABAP verwenden
- AWS IoT Beispiele zur Verwendung des SDK für SAP ABAP

- AWS IoT data Beispiele zur Verwendung des SDK für SAP ABAP
- AWS IoT SiteWise Beispiele zur Verwendung des SDK für SAP ABAP
- Beispiele für Amazon Keyspaces, die das SDK für SAP ABAP verwenden
- Kinesis-Beispiele unter Verwendung von SDK für SAP ABAP
- AWS KMS Beispiele zur Verwendung des SDK für SAP ABAP
- Lamdba-Beispiele unter Verwendung von SDK für SAP ABAP
- Beispiele für Organisationen, die SDK für SAP ABAP verwenden
- Amazon Pinpoint-Beispiele, die das SDK für SAP ABAP verwenden
- Amazon Pinpoint SMS- und Voice-API-Beispiele unter Verwendung des SDK für SAP ABAP
- Beispiele für Amazon Polly, die das SDK für SAP ABAP verwenden
- Amazon RDS-Beispiele, die das SDK für SAP ABAP verwenden
- Amazon Redshift-Beispiele, die das SDK für SAP ABAP verwenden
- Amazon Rekognition-Beispiele, die das SDK für SAP ABAP verwenden
- Beispiele für Amazon S3 unter Verwendung von SDK für SAP ABAP
- Beispiele für Amazon S3 Control mit SDK für SAP ABAP
- SageMaker KI-Beispiele, die das SDK für SAP ABAP verwenden
- Secrets Manager-Beispiele, die SDK für SAP ABAP verwenden
- Amazon SES-Beispiele, die das SDK für SAP ABAP verwenden
- Beispiele für Amazon SES API v2 unter Verwendung des SDK für SAP ABAP
- Beispiele für Amazon SNS unter Verwendung von SDK für SAP ABAP
- Beispiele für Amazon SQS unter Verwendung von SDK für SAP ABAP
- Beispiele für Step Functions mit SDK für SAP ABAP
- Beispiele für Systems Manager, die das SDK für SAP ABAP verwenden
- Beispiele für Amazon Textract unter Verwendung von SDK für SAP ABAP
- Amazon Transcribe-Beispiele mithilfe des SDK für SAP ABAP
- Beispiele für Amazon Translate unter Verwendung von SDK für SAP ABAP

ACM-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit ACM Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AddTagsToCertificate`

Das folgende Codebeispiel zeigt die Verwendung. `AddTagsToCertificate`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->addtagstocertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags added to certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
      CATCH /aws1/cx_acmtoomanytagsex.
        MESSAGE 'Too many tags for certificate.' TYPE 'I'.
```

```
    ENDTRY.
```

- API-Details finden Sie AddTagsToCertificate in der AWS SDK for SAP ABAP API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        lo_acm->deletecertificate( iv_certificatearn = iv_certificate_arn ).
        MESSAGE 'Certificate deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmresourceinuseex.
        MESSAGE 'Certificate is in use and cannot be deleted.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteCertificate in der AWS SDK for SAP ABAP API-Referenz.

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->describecertificate( iv_certificatearn =
 iv_certificate_arn ).
        MESSAGE 'Certificate details retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeCertificate in der AWS SDK for SAP ABAP API-Referenz.

`GetCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        oo_result = lo_acm->getcertificate( iv_certificatearn =
 iv_certificate_arn ).
```

```
        MESSAGE 'Certificate body and chain retrieved.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acmrequestinprgssex.
        MESSAGE 'Certificate request is in progress.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie GetCertificate in der AWS SDK for SAP ABAP API-Referenz.

`ImportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ImportCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass certificate chain if it's provided (it's optional)
        IF iv_certificate_chain IS NOT INITIAL.
          DATA(lo_result) = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
            iv_certificatechain = iv_certificate_chain
          ).
        ELSE.
          lo_result = lo_acm->importcertificate(
            iv_certificate = iv_certificate
            iv_privatekey = iv_private_key
          ).
        ENDIF.
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate imported successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidparameterex.
```

```
      MESSAGE 'Invalid parameter provided.' TYPE 'I'.
    CATCH /aws1/cx_acmlimitexceededex.
      MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie ImportCertificate in der AWS SDK for SAP ABAP API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_acm->listcertificates(
        iv_maxitems = iv_max_items
        it_certificatestatuses = it_statuses
        io_includes = io_includes
      ).
      MESSAGE 'Certificates listed successfully.' TYPE 'I'.
    CATCH /aws1/cx_acminvalidargsex.
      MESSAGE 'Invalid arguments provided.' TYPE 'I'.
    CATCH /aws1/cx_acmvalidationex.
      MESSAGE 'Validation error occurred.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie ListCertificates in der AWS SDK for SAP ABAP API-Referenz.

`ListTagsForCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        DATA(lo_result) = lo_acm->listtagsforcertificate(
          iv_certificatearn = iv_certificate_arn
        ).
        ot_tags = lo_result->get_tags( ).
        MESSAGE 'Certificate tags retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListTagsForCertificate in der AWS SDK for SAP ABAP API-Referenz.

`RemoveTagsFromCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTagsFromCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
```

```
        lo_acm->removetagsfromcertificate(
          iv_certificatearn = iv_certificate_arn
          it_tags = it_tags
        ).
        MESSAGE 'Tags removed from certificate successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidtagex.
        MESSAGE 'Invalid tag provided.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie RemoveTagsFromCertificate in der AWS SDK for SAP ABAP API-Referenz.

`RequestCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RequestCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_domain_name = 'example.com'
        " iv_validation_method = 'DNS' or 'EMAIL'
        DATA(lo_result) = lo_acm->requestcertificate(
          iv_domainname = iv_domain_name
          it_subjectalternativenames = COND #( WHEN it_alternate_domains IS NOT
INITIAL
                                               THEN it_alternate_domains )
          iv_validationmethod = iv_validation_method
        ).
        ov_certificate_arn = lo_result->get_certificatearn( ).
        MESSAGE 'Certificate requested successfully.' TYPE 'I'.
```

```
      CATCH /aws1/cx_acminvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
      CATCH /aws1/cx_acmlimitexceededex.
        MESSAGE 'Certificate limit exceeded.' TYPE 'I'.
      CATCH /aws1/cx_acminvdomvationoptsex.
        MESSAGE 'Invalid domain validation options.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie RequestCertificate in der AWS SDK for SAP ABAP API-Referenz.

`ResendValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung`ResendValidationEmail`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_certificate_arn = 'arn:aws:acm:region:123456789012:certificate/
certificate-id'
        " iv_domain = 'example.com'
        " iv_validation_domain = 'example.com'
        lo_acm->resendvalidationemail(
          iv_certificatearn = iv_certificate_arn
          iv_domain = iv_domain
          iv_validationdomain = iv_validation_domain
        ).
        MESSAGE 'Validation email resent successfully.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidarnex.
        MESSAGE 'The certificate ARN is not valid.' TYPE 'I'.
      CATCH /aws1/cx_acmresourcenotfoundex.
        MESSAGE 'Certificate not found.' TYPE 'I'.
      CATCH /aws1/cx_acminvalidstateex.
        MESSAGE 'Certificate is not in a valid state.' TYPE 'I'.
      CATCH /aws1/cx_acminvdomvationoptsex.
```

```
        MESSAGE 'Invalid domain validation options.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ResendValidationEmail in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für API-Gateways, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit API Gateway Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDeployment`

Das folgende Codebeispiel zeigt die Verwendung`CreateDeployment`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->createdeployment(
          iv_restapiid = iv_rest_api_id
```

```
        iv_stagename = iv_stage_name
        iv_description = 'Deployment created by ABAP SDK' ).
      DATA(lv_deployment_id) = oo_result->get_id( ).
      MESSAGE 'Deployment created with ID: ' && lv_deployment_id TYPE 'I'.
    CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
      MESSAGE lo_not_found->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_not_found.
    CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
      MESSAGE lo_too_many->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many.
  ENDTRY.
```

- API-Details finden Sie CreateDeployment in der AWS SDK for SAP ABAP API-Referenz.

`CreateResource`

Das folgende Codebeispiel zeigt die Verwendung`CreateResource`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_agw->createresource(
        iv_restapiid = iv_rest_api_id
        iv_parentid = iv_parent_id
        iv_pathpart = iv_resource_path ).
      DATA(lv_resource_id) = oo_result->get_id( ).
      MESSAGE 'Resource created with ID: ' && lv_resource_id TYPE 'I'.
    CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
```

```
      MESSAGE lo_not_found->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_not_found.
    CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
      MESSAGE lo_too_many->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many.
  ENDTRY.
```

- API-Details finden Sie CreateResource in der AWS SDK for SAP ABAP API-Referenz.

`CreateRestApi`

Das folgende Codebeispiel zeigt die Verwendung`CreateRestApi`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_agw->createrestapi(
        iv_name = iv_api_name
        iv_description = 'Sample REST API created by ABAP SDK' ).
      DATA(lv_api_id) = oo_result->get_id( ).
      MESSAGE 'REST API created with ID: ' && lv_api_id TYPE 'I'.
    CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
      MESSAGE lo_too_many->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many.
    CATCH /aws1/cx_agwunauthorizedex INTO DATA(lo_unauthorized).
      MESSAGE lo_unauthorized->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_unauthorized.
  ENDTRY.
```

- API-Details finden Sie CreateRestApi in der AWS SDK for SAP ABAP API-Referenz.

`DeleteRestApi`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRestApi`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_agw->deleterestapi(
      iv_restapiid = iv_rest_api_id ).
    MESSAGE 'REST API deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- API-Details finden Sie DeleteRestApi in der AWS SDK for SAP ABAP API-Referenz.

`GetResources`

Das folgende Codebeispiel zeigt die Verwendung`GetResources`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->getresources(
          iv_restapiid = iv_rest_api_id ).
        DATA(lt_resources) = oo_result->get_items( ).
        DATA(lv_count) = lines( lt_resources ).
        MESSAGE 'Found ' && lv_count && ' resources' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- API-Details finden Sie GetResources in der AWS SDK for SAP ABAP API-Referenz.

`GetRestApis`

Das folgende Codebeispiel zeigt die Verwendung`GetRestApis`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->getrestapis( ).
        DATA(lt_apis) = oo_result->get_items( ).
        DATA(lv_count) = lines( lt_apis ).
        MESSAGE 'Found ' && lv_count && ' REST APIs' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
```

```
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- API-Details finden Sie GetRestApis in der AWS SDK for SAP ABAP API-Referenz.

`PutIntegration`

Das folgende Codebeispiel zeigt die Verwendung`PutIntegration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->putintegration(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_type = 'AWS_PROXY'
          iv_integrationhttpmethod = 'POST'
          iv_uri = iv_integration_uri ).
        MESSAGE 'Integration configured for method' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- API-Details finden Sie PutIntegration in der AWS SDK for SAP ABAP API-Referenz.

`PutIntegrationResponse`

Das folgende Codebeispiel zeigt die Verwendung`PutIntegrationResponse`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_agw->putintegrationresponse(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_statuscode = '200' ).
    MESSAGE 'Integration response configured for status 200' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- API-Details finden Sie PutIntegrationResponse in der AWS SDK for SAP ABAP API-Referenz.

`PutMethod`

Das folgende Codebeispiel zeigt die Verwendung`PutMethod`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_agw->putmethod(
      iv_restapiid = iv_rest_api_id
      iv_resourceid = iv_resource_id
      iv_httpmethod = iv_http_method
      iv_authorizationtype = 'NONE' ).
    MESSAGE 'Method ' && iv_http_method && ' added to resource' TYPE 'I'.
  CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
    MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
  CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
    MESSAGE lo_not_found->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_not_found.
  CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
    MESSAGE lo_too_many->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_too_many.
ENDTRY.
```

- API-Details finden Sie PutMethod in der AWS SDK for SAP ABAP API-Referenz.

`PutMethodResponse`

Das folgende Codebeispiel zeigt die Verwendung`PutMethodResponse`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_agw->putmethodresponse(
          iv_restapiid = iv_rest_api_id
          iv_resourceid = iv_resource_id
          iv_httpmethod = iv_http_method
          iv_statuscode = '200' ).
        MESSAGE 'Method response configured for status 200' TYPE 'I'.
      CATCH /aws1/cx_agwbadrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_agwnotfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_agwtoomanyrequestsex INTO DATA(lo_too_many).
        MESSAGE lo_too_many->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many.
    ENDTRY.
```

- API-Details finden Sie PutMethodResponse in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Application Recovery Controller, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Application Recovery Controller Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung`GetRoutingControlState`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
```

```
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.

          " Create session for this region
          lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

          " Create client with the specific endpoint
          lo_client = create_recovery_client(
            iv_endpoint = lv_url
            iv_region   = lv_region
            io_session  = lo_session ).

          " Try to get the routing control state
          oo_result = lo_client->getroutingcontrolstate(
            iv_routingcontrolarn = iv_routing_control_arn ).

          " If successful, return the result
          RETURN.

        CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
          " This endpoint is temporarily unavailable, try the next one
          lo_exception = lo_endpoint_ex.
          CONTINUE.

        CATCH /aws1/cx_r5vaccessdeniedex
              /aws1/cx_r5vinternalserverex
              /aws1/cx_r5vresourcenotfoundex
              /aws1/cx_r5vthrottlingex
              /aws1/cx_r5vvalidationex
              /aws1/cx_rt_generic INTO lo_exception.
          " For other errors, re-raise immediately
          RAISE EXCEPTION lo_exception.
      ENDTRY.
    ENDLOOP.

    " If we get here, all endpoints failed - re-raise the last exception
    IF lo_exception IS BOUND.
      RAISE EXCEPTION lo_exception.
    ENDIF.
```

- API-Details finden Sie GetRoutingControlState in der AWS SDK for SAP ABAP API-Referenz.

`UpdateRoutingControlState`

Das folgende Codebeispiel zeigt die Verwendung`UpdateRoutingControlState`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA lo_exception TYPE REF TO /aws1/cx_rt_generic.
    DATA lo_session TYPE REF TO /aws1/cl_rt_session_base.
    DATA lo_client TYPE REF TO /aws1/if_r5v.
    DATA lt_endpoints TYPE TABLE OF string.
    DATA lv_endpoint TYPE string.
    DATA lv_region TYPE /aws1/rt_region_id.

    " Parse the comma-separated cluster endpoints
    " Expected format: "https://endpoint1.com|us-west-2,https://endpoint2.com|us-
east-1"
    SPLIT iv_cluster_endpoints AT ',' INTO TABLE lt_endpoints.

    " As a best practice, shuffle cluster endpoints to distribute load
    " For more information, see https://docs.aws.amazon.com/r53recovery/latest/dg/
route53-arc-best-practices.html#route53-arc-best-practices.regional
    " For simplicity, we'll try them in order (shuffling can be added if needed)

    " Try each endpoint in order
    LOOP AT lt_endpoints INTO lv_endpoint.
      TRY.
          " Parse endpoint and region from the format "url|region"
          DATA(lv_pos) = find( val = lv_endpoint sub = '|' ).
          IF lv_pos > 0.
            DATA(lv_url) = substring( val = lv_endpoint len = lv_pos ).
            lv_region = substring( val = lv_endpoint off = lv_pos + 1 ).
          ELSE.
            " If no region specified, use default
            lv_url = lv_endpoint.
            lv_region = 'us-east-1'.
          ENDIF.
```

```
        " Create session for this region
        lo_session = /aws1/cl_rt_session_aws=>create( cv_pfl ).

        " Create client with the specific endpoint
        lo_client = create_recovery_client(
          iv_endpoint = lv_url
          iv_region   = lv_region
          io_session  = lo_session ).

        " Try to update the routing control state
        oo_result = lo_client->updateroutingcontrolstate(
          iv_routingcontrolarn     = iv_routing_control_arn
          iv_routingcontrolstate   = iv_routing_control_state
          it_safetyrulestooverride = it_safety_rules_override ).

        " If successful, return the result
        RETURN.

      CATCH /aws1/cx_r5vendpttmpyunavailex INTO DATA(lo_endpoint_ex).
        " This endpoint is temporarily unavailable, try the next one
        lo_exception = lo_endpoint_ex.
        CONTINUE.

      CATCH /aws1/cx_r5vaccessdeniedex
            /aws1/cx_r5vconflictexception
            /aws1/cx_r5vinternalserverex
            /aws1/cx_r5vresourcenotfoundex
            /aws1/cx_r5vthrottlingex
            /aws1/cx_r5vvalidationex
            /aws1/cx_rt_generic INTO lo_exception.
        " For other errors, re-raise immediately
        RAISE EXCEPTION lo_exception.
    ENDTRY.
  ENDLOOP.

  " If we get here, all endpoints failed - re-raise the last exception
  IF lo_exception IS BOUND.
    RAISE EXCEPTION lo_exception.
  ENDIF.
```

- API-Details finden Sie UpdateRoutingControlState in der AWS SDK for SAP ABAP API-Referenz.

Aurora-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aurora Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_output) = lo_rds->createdbclusterparamgroup(
      iv_dbclusterparamgroupname = iv_param_group_name
      iv_dbparametergroupfamily = iv_param_group_family
      iv_description = iv_description
    ).
```

```
      oo_result = lo_output->get_dbclusterparametergroup( ).
    CATCH /aws1/cx_rdsdbparmgralrexfault.
      " Re-raise exception - parameter group already exists
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbparmgralrexfault.
    CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
      " Re-raise exception - quota exceeded
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrquotaexcd00.
  ENDTRY.
```

- API-Details finden Sie CreateDBClusterParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_rds->deletedbclusterparamgroup(
        iv_dbclusterparamgroupname = iv_param_group_name
      ).
    CATCH /aws1/cx_rdsdbprmgrnotfndfault.
      " Re-raise exception - parameter group not found
      RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
    CATCH /aws1/cx_rdsinvdbprmgrstatef00.
      " Re-raise exception - invalid state
      RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
  ENDTRY.
```

- API-Details finden Sie DeleteDBClusterParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_output) = lo_rds->describedbclusterparamgroups(
      iv_dbclusterparamgroupname = iv_param_group_name
    ).
    DATA(lt_param_groups) = lo_output->get_dbclusterparametergroups( ).
    IF lines( lt_param_groups ) > 0.
      oo_result = lt_param_groups[ 1 ].
    ENDIF.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
ENDTRY.
```

- API-Details finden Sie DescribeDBClusterParameterGroups in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        DATA lv_marker TYPE /aws1/rdsstring VALUE ''.
        DATA lt_all_parameters TYPE /aws1/cl_rdsparameter=>tt_parameterslist.

        DO.
          DATA(lo_output) = lo_rds->describedbclusterparameters(
            iv_dbclusterparamgroupname = iv_param_group_name
            iv_source = iv_source
            iv_marker = lv_marker
          ).

          LOOP AT lo_output->get_parameters( ) INTO DATA(lo_param).
            IF iv_name_prefix IS INITIAL OR
               lo_param->get_parametername( ) CP |{ iv_name_prefix }*|.
              APPEND lo_param TO lt_all_parameters.
            ENDIF.
          ENDLOOP.

          lv_marker = lo_output->get_marker( ).
          IF lv_marker IS INITIAL.
            EXIT.
          ENDIF.
        ENDDO.

        ot_parameters = lt_all_parameters.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        " Re-raise exception - parameter group not found
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
    ENDTRY.
```

- API-Details finden Sie DescribeDBClusterParameters in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_engine                 = 'mysql'
   " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter group
family)
   TRY.
       oo_result = lo_rds->describedbengineversions(
         iv_engine                 = iv_engine
         iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
       DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
       MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
   ENDTRY.
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for SAP ABAP API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_engine        = 'mysql'
   " iv_engineversion = '8.0.35'
   TRY.
       oo_result = lo_rds->descrorderabledbinstoptions(
```

```
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result->get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.| TYPE
'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK for SAP ABAP API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_rds->modifydbclusterparamgroup(
          iv_dbclusterparamgroupname = iv_param_group_name
          it_parameters = it_update_parameters
        ).
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        " Re-raise exception - parameter group not found
        RAISE EXCEPTION TYPE /aws1/cx_rdsdbprmgrnotfndfault.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        " Re-raise exception - invalid state
        RAISE EXCEPTION TYPE /aws1/cx_rdsinvdbprmgrstatef00.
    ENDTRY.
```

- API-Details finden Sie ModifyDBClusterParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

Auto Scaling-Beispiele unter Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Auto Scaling Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateAutoScalingGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA lo_launch_template TYPE REF TO /aws1/cl_asclaunchtemplatespec.

" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_launch_template_name = 'my-launch-template'
" Example: iv_min_size = 1
" Example: iv_max_size = 3
" Example: iv_vpc_zone_identifier = 'subnet-12345,subnet-67890' (for VPC)

TRY.
    " Create launch template specification
```

```
        lo_launch_template = NEW /aws1/cl_asclaunchtemplatespec(
          iv_launchtemplatename = iv_launch_template_name
          iv_version = '$Default' ).

        " Create the Auto Scaling group
        " Use VPCZoneIdentifier for VPC subnets, or AvailabilityZones for EC2-
Classic
        IF iv_vpc_zone_identifier IS NOT INITIAL.
          " VPC-based deployment - use subnet IDs
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            iv_vpczoneidentifier = iv_vpc_zone_identifier
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ELSE.
          " EC2-Classic or default VPC - use availability zones
          ao_asc->createautoscalinggroup(
            iv_autoscalinggroupname = iv_group_name
            it_availabilityzones = it_group_zones
            io_launchtemplate = lo_launch_template
            iv_minsize = iv_min_size
            iv_maxsize = iv_max_size ).
        ENDIF.

        " Wait for the group to be created (simplified - in production use proper
 polling)
        WAIT UP TO 10 SECONDS.

        MESSAGE 'Auto Scaling group created successfully' TYPE 'I'.

      CATCH /aws1/cx_ascalreadyexistsfault INTO DATA(lo_already_exists).
        RAISE EXCEPTION lo_already_exists.
      CATCH /aws1/cx_asclimitexceededfault INTO DATA(lo_limit_exceeded).
        RAISE EXCEPTION lo_limit_exceeded.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- API-Details finden Sie CreateAutoScalingGroup in der AWS SDK for SAP ABAP API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " Example: iv_group_name = 'my-auto-scaling-group'

   TRY.
       ao_asc->deleteautoscalinggroup(
         iv_autoscalinggroupname = iv_group_name ).

       " Wait for the group to be deleted (simplified - in production use proper
polling)
       WAIT UP TO 10 SECONDS.

       MESSAGE 'Auto Scaling group deleted successfully' TYPE 'I'.

     CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
       RAISE EXCEPTION lo_activity_in_progress.
     CATCH /aws1/cx_ascresourceinusefault INTO DATA(lo_resource_in_use).
       RAISE EXCEPTION lo_resource_in_use.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- API-Details finden Sie DeleteAutoScalingGroup in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_group_names TYPE /aws1/
cl_ascautoscgroupnames_w=>tt_autoscalinggroupnames.
    DATA lo_group_name TYPE REF TO /aws1/cl_ascautoscgroupnames_w.

    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        " Build group names parameter
        CREATE OBJECT lo_group_name
          EXPORTING
            iv_value = iv_group_name.
        APPEND lo_group_name TO lt_group_names.

        " Describe the Auto Scaling group
        DATA(lo_output) = ao_asc->describeautoscalinggroups(
          it_autoscalinggroupnames = lt_group_names ).

        " Return the first (and only) group in the result
        DATA(lt_groups) = lo_output->get_autoscalinggroups( ).
        IF lines( lt_groups ) > 0.
          READ TABLE lt_groups INDEX 1 INTO DATA(lo_group).
          oo_output = lo_group.
        ENDIF.

        MESSAGE 'Auto Scaling group information retrieved successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- API-Details finden Sie DescribeAutoScalingGroups in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " Example: it_instance_ids contains a list of instance IDs

   TRY.
       DATA(lo_output) = ao_asc->describeautoscalinginstances(
         it_instanceids = it_instance_ids ).

       ot_output = lo_output->get_autoscalinginstances( ).

       MESSAGE 'Auto Scaling instances information retrieved successfully' TYPE
'I'.

     CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
       RAISE EXCEPTION lo_contention.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
       RAISE EXCEPTION lo_generic_exception.
   ENDTRY.
```

- API-Details finden Sie DescribeAutoScalingInstances in der AWS SDK for SAP ABAP API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'

TRY.
    DATA(lo_output) = ao_asc->describescalingactivities(
      iv_autoscalinggroupname = iv_group_name ).

    ot_output = lo_output->get_activities( ).

    MESSAGE 'Scaling activities retrieved successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- API-Details finden Sie DescribeScalingActivities in der AWS SDK for SAP ABAP API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: iv_group_name = 'my-auto-scaling-group'

    TRY.
        ao_asc->disablemetricscollection(
          iv_autoscalinggroupname = iv_group_name ).

        MESSAGE 'Metrics collection disabled successfully' TYPE 'I'.

      CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
        RAISE EXCEPTION lo_contention.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
        RAISE EXCEPTION lo_generic_exception.
    ENDTRY.
```

- API-Details finden Sie DisableMetricsCollection in der AWS SDK for SAP ABAP API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Example: iv_group_name = 'my-auto-scaling-group'
    " Example: it_metrics contains list of metrics like 'GroupMinSize',
 'GroupMaxSize', etc.

    TRY.
        ao_asc->enablemetricscollection(
          iv_autoscalinggroupname = iv_group_name
          it_metrics = it_metrics
          iv_granularity = '1Minute' ).

        MESSAGE 'Metrics collection enabled successfully' TYPE 'I'.
```

```
    CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
      RAISE EXCEPTION lo_contention.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- API-Details finden Sie EnableMetricsCollection in der AWS SDK for SAP ABAP API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  " Example: iv_group_name = 'my-auto-scaling-group'
  " Example: iv_capacity = 2

  TRY.
      ao_asc->setdesiredcapacity(
        iv_autoscalinggroupname = iv_group_name
        iv_desiredcapacity = iv_capacity
        iv_honorcooldown = abap_false ).

      MESSAGE 'Desired capacity set successfully' TYPE 'I'.

    CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
      RAISE EXCEPTION lo_activity_in_progress.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
      RAISE EXCEPTION lo_generic_exception.
  ENDTRY.
```

- API-Details finden Sie SetDesiredCapacity in der AWS SDK for SAP ABAP API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_instance_id = 'i-1234567890abcdef0'
" Example: iv_decrease_capacity = abap_true

TRY.
    DATA(lo_output) = ao_asc->terminateinstinautoscgroup(
      iv_instanceid = iv_instance_id
      iv_shoulddecrementdesiredcap = iv_decrease_capacity ).

    oo_output = lo_output->get_activity( ).

    MESSAGE 'Instance terminated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- API-Details finden Sie TerminateInstanceInAutoScalingGroup in der AWS SDK for SAP ABAP API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Example: iv_group_name = 'my-auto-scaling-group'
" Example: iv_max_size = 5

TRY.
    ao_asc->updateautoscalinggroup(
      iv_autoscalinggroupname = iv_group_name
      iv_maxsize = iv_max_size
      iv_minsize = iv_min_size ).

    MESSAGE 'Auto Scaling group updated successfully' TYPE 'I'.

  CATCH /aws1/cx_ascresrccontionfault INTO DATA(lo_contention).
    RAISE EXCEPTION lo_contention.
  CATCH /aws1/cx_ascscaactivityinprg00 INTO DATA(lo_activity_in_progress).
    RAISE EXCEPTION lo_activity_in_progress.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_exception).
    RAISE EXCEPTION lo_generic_exception.
ENDTRY.
```

- API-Details finden Sie UpdateAutoScalingGroup in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Bedrock Runtime Aktionen ausführen und gängige Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Stable Image Core

Stable Image Core

InvokeModel

Das folgende Codebeispiel zeigt, wie Stability.ai Stable Image Core auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Stable Diffusion.

```
    "Stable Image Core Input Parameters should be in a format like this:
*   {
*     "prompt": "Draw a dolphin with a mustache, photorealistic",
*     "aspect_ratio": "1:1",
*     "seed": 0,
*     "output_format": "png"
*   }
    DATA: BEGIN OF ls_input,
            prompt        TYPE /aws1/rt_shape_string,
            aspect_ratio  TYPE /aws1/rt_shape_string,
            seed          TYPE /aws1/rt_shape_integer,
            output_format TYPE /aws1/rt_shape_string,
          END OF ls_input.

    ls_input-prompt = iv_prompt.
    ls_input-aspect_ratio = '1:1'.
    ls_input-seed = 0. "or better, choose a random integer.
    ls_input-output_format = 'png'.

    DATA(lv_json) = /ui2/cl_json=>serialize(
      data = ls_input
```

```
                pretty_name   = /ui2/cl_json=>pretty_mode-low_case ).

    TRY.
        DATA(lo_response) = lo_bdr->invokemodel(
          iv_body = /aws1/cl_rt_util=>string_to_xstring( lv_json )
          iv_modelid = 'stability.stable-image-core-v1:1'
          iv_accept = 'application/json'
          iv_contenttype = 'application/json' ).

        "Stable Image Core Result Format:
*       {
*         "seeds": ["0"],
*         "finish_reasons": [null],
*         "images": ["iVBORw0KGgoAAAANSUhEUgAAAgAAA...."]
*       }
        DATA: BEGIN OF ls_response,
                images TYPE STANDARD TABLE OF /aws1/rt_shape_string,
              END OF ls_response.

        /ui2/cl_json=>deserialize(
          EXPORTING jsonx = lo_response->get_body( )
                    pretty_name = /ui2/cl_json=>pretty_mode-camel_case
          CHANGING  data  = ls_response ).
        IF ls_response-images IS NOT INITIAL.
          DATA(lv_image) =
 cl_http_utility=>if_http_utility~decode_x_base64( ls_response-images[ 1 ] ).
        ENDIF.
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

Rufen Sie das Stability.ai Stable Image Core Foundation-Modell auf, um Images mithilfe des L2-High-Level-Clients zu generieren.

```
    TRY.
        DATA(lo_bdr_l2_sd) = /aws1/
cl_bdr_l2_factory=>create_stable_diffusion_xl_1( lo_bdr ).
```

```
        " iv_prompt contains a prompt like 'Show me a picture of a unicorn reading
 an enterprise financial report'.
        DATA(lv_image) = lo_bdr_l2_sd->text_to_image( iv_prompt ).
      CATCH /aws1/cx_bdraccessdeniedex INTO DATA(lo_ex).
        WRITE / lo_ex->get_text( ).
        WRITE / |Don't forget to enable model access at https://
console.aws.amazon.com/bedrock/home?#/modelaccess|.

    ENDTRY.
```

- Einzelheiten zur API finden Sie InvokeModel in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Bedrock Agents Runtime unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Bedrock Agents Runtime Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`InvokeAgent`

Das folgende Codebeispiel zeigt die Verwendung. `InvokeAgent`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_result) = lo_bdz->invokeagent(
  iv_agentid      = iv_agentid
    iv_agentaliasid = iv_agentaliasid
    iv_enabletrace  = abap_true
    iv_sessionid    = CONV #( cl_system_uuid=>create_uuid_c26_static( ) )
    iv_inputtext    = |Let's play "rock, paper, scissors".  I choose rock.| ).
DATA(lo_stream) = lo_result->get_completion( ).
TRY.
    " loop while there are still events in the stream
    WHILE lo_stream->/aws1/if_rt_stream_reader~data_available( ) = abap_true.
      DATA(lo_evt) = lo_stream->read( ).
      " each /AWS1/CL_BDZRESPONSESTREAM_EV event contains exactly one member
      " all others are INITIAL.  For each event, process the non-initial
      " member if desired
      IF lo_evt->get_chunk( ) IS NOT INITIAL.
        " Process a Chunk event
        DATA(lv_xstr) = lo_evt->get_chunk( )->get_bytes( ).
        DATA(lv_answer) = /aws1/cl_rt_util=>xstring_to_string( lv_xstr ).
        " the answer says something like "I chose paper, so you lost"
      ELSEIF lo_evt->get_files( ) IS NOT INITIAL.
        " process a Files event if desired
      ELSEIF lo_evt->get_returncontrol( ) IS NOT INITIAL.
        " process a ReturnControl event if desired
      ELSEIF lo_evt->get_trace( ) IS NOT INITIAL.
        " process a Trace event if desired
      ENDIF.
    ENDWHILE.
    " the stream of events can possibly contain an exception
    " which will be raised to break the loop
    " catch /AWS1/CX_BDZACCESSDENIEDEX.
    " catch /AWS1/CX_BDZINTERNALSERVEREX.
    " catch /AWS1/CX_BDZMODELNOTREADYEX.
    " catch /AWS1/CX_BDZVALIDATIONEX.
```

```
        " catch /AWS1/CX_BDZTHROTTLINGEX.
        " catch /AWS1/CX_BDZDEPENDENCYFAILEDEX.
        " catch /AWS1/CX_BDZBADGATEWAYEX.
        " catch /AWS1/CX_BDZRESOURCENOTFOUNDEX.
        " catch /AWS1/CX_BDZSERVICEQUOTAEXCDEX.
        " catch /AWS1/CX_BDZCONFLICTEXCEPTION.
    ENDTRY.
```

- API-Details finden Sie InvokeAgent in der AWS SDK for SAP ABAP API-Referenz.

CloudFront Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. CloudFront

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListDistributions`

Das folgende Codebeispiel zeigt die Verwendung`ListDistributions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_fnt->listdistributions( ). " oo_result is returned for
 testing purposes. "
        MESSAGE 'Retrieved list of CloudFront distributions.' TYPE 'I'.
      CATCH /aws1/cx_fntinvalidargument.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListDistributions in der AWS SDK for SAP ABAP API-Referenz.

`UpdateDistribution`

Das folgende Codebeispiel zeigt die Verwendung`UpdateDistribution`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Get the current distribution configuration and ETag "
        DATA(lo_distribution_config_result) = lo_fnt->getdistributionconfig( iv_id =
 iv_distribution_id ).
        DATA(lo_old_config) = lo_distribution_config_result-
>get_distributionconfig( ).
        DATA(lv_etag) = lo_distribution_config_result->get_etag( ).

        " Create a new distribution config with the updated comment "
        " Since the config object is immutable, we need to create a new one with all
 existing values "
        DATA(lo_new_config) = NEW /aws1/cl_fntdistributionconfig(
          iv_callerreference = lo_old_config->get_callerreference( )
          io_aliases = lo_old_config->get_aliases( )
          iv_defaultrootobject = lo_old_config->get_defaultrootobject( )
          io_origins = lo_old_config->get_origins( )
          io_origingroups = lo_old_config->get_origingroups( )
          io_defaultcachebehavior = lo_old_config->get_defaultcachebehavior( )
```

```
        io_cachebehaviors = lo_old_config->get_cachebehaviors( )
        io_customerrorresponses = lo_old_config->get_customerrorresponses( )
        iv_comment = iv_comment
        io_logging = lo_old_config->get_logging( )
        iv_priceclass = lo_old_config->get_priceclass( )
        iv_enabled = lo_old_config->get_enabled( )
        io_viewercertificate = lo_old_config->get_viewercertificate( )
        io_restrictions = lo_old_config->get_restrictions( )
        iv_webaclid = lo_old_config->get_webaclid( )
        iv_httpversion = lo_old_config->get_httpversion( )
        iv_isipv6enabled = lo_old_config->get_isipv6enabled( ) ).

      " Update the distribution with the modified configuration "
      lo_fnt->updatedistribution(
        io_distributionconfig = lo_new_config
        iv_id = iv_distribution_id
        iv_ifmatch = lv_etag ).
      MESSAGE 'CloudFront distribution updated successfully.' TYPE 'I'.
    CATCH /aws1/cx_fntnosuchdistribution.
      MESSAGE 'Distribution does not exist.' TYPE 'E'.
    CATCH /aws1/cx_fntpreconditionfailed.
      MESSAGE 'Precondition failed - ETag mismatch.' TYPE 'E'.
    CATCH /aws1/cx_fntinvalidifmatchvrs.
      MESSAGE 'Invalid If-Match version.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie UpdateDistribution in der AWS SDK for SAP ABAP API-Referenz.

CloudWatch Beispiele mit SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. CloudWatch

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAlarms`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_cwt->deletealarms(
      it_alarmnames = it_alarm_names ).
    MESSAGE 'Alarms deleted.' TYPE 'I'.
  CATCH /aws1/cx_cwtresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteAlarms in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAlarms`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarms`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwt->describealarms(                 " oo_result is returned
 for testing purposes. "
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeAlarms in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwt->describealarmsformetric(          " oo_result is
returned for testing purposes. "
          iv_namespace   = iv_namespace
          iv_metricname  = iv_metric_name
```

```
          it_dimensions  = it_dimensions
          iv_statistic   = iv_statistic
          iv_period      = iv_period
          iv_unit        = iv_unit ).
        MESSAGE 'Alarms for metric retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeAlarmsForMetric in der AWS SDK for SAP ABAP API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Disables actions on the specified alarm. "
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DisableAlarmActions in der AWS SDK for SAP ABAP API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie EnableAlarmActions in der AWS SDK for SAP ABAP API-Referenz.

`GetMetricStatistics`

Das folgende Codebeispiel zeigt die Verwendung`GetMetricStatistics`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cwt->getmetricstatistics(              " oo_result is
returned for testing purposes. "
          iv_namespace   = iv_namespace
          iv_metricname  = iv_metric_name
          iv_starttime   = iv_start_time
          iv_endtime     = iv_end_time
          iv_period      = iv_period
          it_statistics  = it_statistics ).
        MESSAGE 'Metric statistics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetMetricStatistics in der AWS SDK for SAP ABAP API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "The following list-metrics example displays the metrics for Amazon CloudWatch."
    TRY.
        oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
          iv_namespace = iv_namespace ).
        DATA(lt_metrics) = oo_result->get_metrics( ).
        MESSAGE 'Metrics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListMetrics in der AWS SDK for SAP ABAP API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                = iv_alarm_name
      iv_comparisonoperator       = iv_comparison_operator
      iv_evaluationperiods        = iv_evaluation_periods
      iv_metricname               = iv_metric_name
      iv_namespace                = iv_namespace
      iv_statistic                = iv_statistic
      iv_threshold                = iv_threshold
      iv_actionsenabled           = iv_actions_enabled
      iv_alarmdescription         = iv_alarm_description
      iv_unit                     = iv_unit
      iv_period                   = iv_period
      it_dimensions               = it_dimensions ).
    MESSAGE 'Alarm created.' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie PutMetricAlarm in der AWS SDK for SAP ABAP API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

"Create metric data object.
DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
  iv_metricname = iv_metric_name
  iv_value      = iv_value
  iv_unit       = iv_unit ).

INSERT lo_metricdatum INTO TABLE lt_metricdata.

TRY.
    lo_cwt->putmetricdata(
      iv_namespace   = iv_namespace
      it_metricdata  = lt_metricdata ).
    MESSAGE 'Metric data added.' TYPE 'I'.
  CATCH /aws1/cx_cwtinvparamvalueex.
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
ENDTRY.
```

Fügen Sie eine Reihe von Daten in eine CloudWatch Metrik ein.

```
DATA lt_metricdata TYPE /aws1/cl_cwtmetricdatum=>tt_metricdata.

"Create metric data object with values and counts.
DATA(lo_metricdatum) = NEW /aws1/cl_cwtmetricdatum(
  iv_metricname = iv_metric_name
  iv_timestamp  = iv_timestamp
  iv_unit       = iv_unit
  it_values     = it_values
  it_counts     = it_counts ).
```

```
    INSERT lo_metricdatum INTO TABLE lt_metricdata.

    TRY.
        lo_cwt->putmetricdata(
          iv_namespace   = iv_namespace
          it_metricdata  = lt_metricdata ).
        MESSAGE 'Metric data set added.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutMetricData in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Erste Schritte mit Alarmen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Alarm.
- Deaktivieren Sie Alarmaktionen.
- Beschreiben Sie einen Alarm.
- Löschen Sie einen Alarm.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
    DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

    "Create an alarm"
    TRY.
        lo_cwt->putmetricalarm(
```

```
          iv_alarmname                 = iv_alarm_name
          iv_comparisonoperator        = iv_comparison_operator
          iv_evaluationperiods         = iv_evaluation_periods
          iv_metricname                = iv_metric_name
          iv_namespace                 = iv_namespace
          iv_statistic                 = iv_statistic
          iv_threshold                 = iv_threshold
          iv_actionsenabled            = iv_actions_enabled
          iv_alarmdescription          = iv_alarm_description
          iv_unit                      = iv_unit
          iv_period                    = iv_period
          it_dimensions                = it_dimensions ).
        MESSAGE 'Alarm created' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.

    "Create an ABAP internal table for the created alarm."
    lo_alarmname = NEW #( iv_value = iv_alarm_name ).
    INSERT lo_alarmname INTO TABLE lt_alarmnames.

    "Disable alarm actions."
    TRY.
        lo_cwt->disablealarmactions(
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception->av_err_code }"
 - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
          it_alarmnames                = lt_alarmnames ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.
```

```
    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound.
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

CloudWatch Loggt Beispiele mit SDK für SAP ABAP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP with CloudWatch Logs Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetQueryResults`

Das folgende Codebeispiel zeigt die Verwendung`GetQueryResults`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_cwl->getqueryresults(
         iv_queryid = iv_query_id ).

       " Display query status and result count
       DATA(lv_status) = oo_result->get_status( ).
       DATA(lt_results) = oo_result->get_results( ).
       DATA(lv_result_count) = lines( lt_results ).

       MESSAGE |Query status: { lv_status }. Retrieved { lv_result_count } log
event(s).| TYPE 'I'.
     CATCH /aws1/cx_cwlinvalidparameterex.
       MESSAGE 'Invalid parameter.' TYPE 'E'.
     CATCH /aws1/cx_cwlresourcenotfoundex.
       MESSAGE 'Resource not found.' TYPE 'E'.
     CATCH /aws1/cx_cwlserviceunavailex.
       MESSAGE 'Service unavailable.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetQueryResults in der AWS SDK for SAP ABAP API-Referenz.

`StartQuery`

Das folgende Codebeispiel zeigt die Verwendung`StartQuery`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_log_group_name = '/aws/lambda/my-function'
        " iv_query_string = 'fields @timestamp, @message | sort @timestamp desc |
limit 20'
        " iv_start_time and iv_end_time must be in Unix epoch milliseconds (ms since
Jan 1, 1970 00:00:00 UTC)
        oo_result = lo_cwl->startquery(
          iv_loggroupname = iv_log_group_name
          iv_starttime    = iv_start_time
          iv_endtime      = iv_end_time
          iv_querystring  = iv_query_string
          iv_limit        = iv_limit ).

        " Display the query ID for tracking
        DATA(lv_query_id) = oo_result->get_queryid( ).
        MESSAGE |Query started successfully with ID: { lv_query_id }| TYPE 'I'.
      CATCH /aws1/cx_cwlinvalidparameterex.
        MESSAGE 'Invalid parameter.' TYPE 'E'.
      CATCH /aws1/cx_cwllimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cwlmalformedqueryex.
        MESSAGE 'Malformed query.' TYPE 'E'.
      CATCH /aws1/cx_cwlresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cwlserviceunavailex.
        MESSAGE 'Service unavailable.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie StartQuery in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Cognito Identity Provider, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Cognito Identity Provider Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung. `AdminInitiateAuth`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Set up authentication parameters
        DATA(lt_auth_params) = VALUE /aws1/
cl_cgpauthparamstype_w=>tt_authparameterstype(
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'USERNAME'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_user_name ) ) )
          ( VALUE /aws1/cl_cgpauthparamstype_w=>ts_authparameterstype_maprow(
              key = 'PASSWORD'
              value = NEW /aws1/cl_cgpauthparamstype_w( iv_password ) ) )
        ).

        " Add SECRET_HASH if provided
        IF iv_secret_hash IS NOT INITIAL.
          INSERT VALUE #(
            key = 'SECRET_HASH'
            value = NEW /aws1/cl_cgpauthparamstype_w( iv_secret_hash )
          ) INTO TABLE lt_auth_params.
        ENDIF.

        oo_result = lo_cgp->admininitiateauth(
```

```
          iv_userpoolid = iv_user_pool_id
          iv_clientid = iv_client_id
          iv_authflow = 'ADMIN_USER_PASSWORD_AUTH'
          it_authparameters = lt_auth_params
        ).

        DATA(lv_challenge) = oo_result->get_challengename( ).

        IF lv_challenge IS INITIAL.
          MESSAGE 'User successfully signed in.' TYPE 'I'.
        ELSE.
          MESSAGE |Authentication challenge required: { lv_challenge }.| TYPE 'I'.
        ENDIF.

      CATCH /aws1/cx_cgpusernotfoundex INTO DATA(lo_user_ex).
        MESSAGE |User { iv_user_name } not found.| TYPE 'E'.

      CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
        MESSAGE 'Not authorized. Check credentials.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK for SAP ABAP API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Build challenge responses
        DATA(lt_challenge_responses) = VALUE /aws1/
cl_cgpchallengerspstyp00=>tt_challengeresponsestype(
          ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
              key = 'USERNAME'
```

```
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_user_name ) ) )
        ( VALUE /aws1/cl_cgpchallengerspstyp00=>ts_challengerspstype_maprow(
            key = 'SOFTWARE_TOKEN_MFA_CODE'
            value = NEW /aws1/cl_cgpchallengerspstyp00( iv_mfa_code ) ) )
      ).

      " Add SECRET_HASH if provided
      IF iv_secret_hash IS NOT INITIAL.
        INSERT VALUE #(
          key = 'SECRET_HASH'
          value = NEW /aws1/cl_cgpchallengerspstyp00( iv_secret_hash )
        ) INTO TABLE lt_challenge_responses.
      ENDIF.

      DATA(lo_result) = lo_cgp->adminrespondtoauthchallenge(
        iv_userpoolid = iv_user_pool_id
        iv_clientid = iv_client_id
        iv_challengename = 'SOFTWARE_TOKEN_MFA'
        it_challengeresponses = lt_challenge_responses
        iv_session = iv_session
      ).

      oo_auth_result = lo_result->get_authenticationresult( ).

      IF oo_auth_result IS BOUND.
        MESSAGE 'MFA challenge completed successfully.' TYPE 'I'.
      ELSE.
        " Another challenge might be required
        DATA(lv_next_challenge) = lo_result->get_challengename( ).
        MESSAGE |Additional challenge required: { lv_next_challenge }.| TYPE 'I'.
      ENDIF.

    CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
      MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

    CATCH /aws1/cx_cgpexpiredcodeex INTO DATA(lo_expired_ex).
      MESSAGE 'MFA code has expired.' TYPE 'E'.

    CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
      MESSAGE 'Not authorized. Check MFA configuration.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK for SAP ABAP API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_cgp->associatesoftwaretoken(
      iv_session = iv_session
    ).

    ov_secret_code = lo_result->get_secretcode( ).

    MESSAGE 'MFA secret code generated successfully.' TYPE 'I'.

  CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE 'Session not found or expired.' TYPE 'E'.

  CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
    MESSAGE 'Not authorized to associate software token.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK for SAP ABAP API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_cgp->listusers(
      iv_userpoolid = iv_user_pool_id
    ).

    ot_users = lo_result->get_users( ).

    MESSAGE |Found { lines( ot_users ) } users in the pool.| TYPE 'I'.

  CATCH /aws1/cx_cgpresourcenotfoundex INTO DATA(lo_ex).
    MESSAGE |User pool { iv_user_pool_id } not found.| TYPE 'E'.

  CATCH /aws1/cx_cgpnotauthorizedex INTO DATA(lo_auth_ex).
    MESSAGE 'Not authorized to list users.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListUsers in der AWS SDK for SAP ABAP API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      DATA(lo_result) = lo_cgp->verifysoftwaretoken(
        iv_session = iv_session
        iv_usercode = iv_user_code
      ).

      ov_status = lo_result->get_status( ).

      IF ov_status = 'SUCCESS'.
        MESSAGE 'MFA token verified successfully.' TYPE 'I'.
      ELSE.
        MESSAGE |MFA verification status: { ov_status }.| TYPE 'I'.
      ENDIF.

    CATCH /aws1/cx_cgpcodemismatchex INTO DATA(lo_code_ex).
      MESSAGE 'Invalid MFA code provided.' TYPE 'E'.

    CATCH /aws1/cx_cgpenbsoftwaretokmf00 INTO DATA(lo_enabled_ex).
      MESSAGE 'Software token MFA is already enabled.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK for SAP ABAP API-Referenz.

Amazon Comprehend-Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Comprehend Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung. `CreateDocumentClassifier`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->createdocumentclassifier(
      iv_documentclassifiername = iv_classifier_name
      iv_languagecode = iv_language_code
      io_inputdataconfig = NEW /aws1/cl_cpddocclifierinpdat00(
        iv_s3uri = iv_training_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
      iv_mode = iv_mode
    ).
    MESSAGE 'Document classifier creation started.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanytagsex.
    MESSAGE 'Too many tags.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateDocumentClassifier in der AWS SDK for SAP ABAP API-Referenz.

`DeleteDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDocumentClassifier`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->deletedocumentclassifier(
      iv_documentclassifierarn = iv_classifier_arn
    ).
    MESSAGE 'Document classifier deleted.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourcenotfoundex.
    MESSAGE 'Resource not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdresourceinuseex.
    MESSAGE 'Resource in use.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteDocumentClassifier in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDocumentClassificationJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDocumentClassificationJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->describedocclassificationjob(
      iv_jobid = iv_job_id
    ).
    MESSAGE 'Document classification job described.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdjobnotfoundex.
    MESSAGE 'Job not found.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DescribeDocumentClassificationJob in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDocumentClassifier`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDocumentClassifier`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      oo_result = lo_cpd->describedocumentclassifier(
        iv_documentclassifierarn = iv_classifier_arn
      ).
      MESSAGE 'Document classifier described.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribeDocumentClassifier in der AWS SDK for SAP ABAP API-Referenz.

`DescribeTopicsDetectionJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTopicsDetectionJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_cpd->describetopicsdetectionjob(
        iv_jobid = iv_job_id
      ).
      MESSAGE 'Topics detection job described.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdjobnotfoundex.
      MESSAGE 'Job not found.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
```

```
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribeTopicsDetectionJob in der AWS SDK for SAP ABAP API-Referenz.

`DetectDominantLanguage`

Das folgende Codebeispiel zeigt die Verwendung`DetectDominantLanguage`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_cpd->detectdominantlanguage( iv_text = iv_text ).
      MESSAGE 'Languages detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetectDominantLanguage in der AWS SDK for SAP ABAP API-Referenz.

`DetectEntities`

Das folgende Codebeispiel zeigt die Verwendung`DetectEntities`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->detectentities(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Entities detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetectEntities in der AWS SDK for SAP ABAP API-Referenz.

`DetectKeyPhrases`

Das folgende Codebeispiel zeigt die Verwendung`DetectKeyPhrases`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      oo_result = lo_cpd->detectkeyphrases(
        iv_text = iv_text
        iv_languagecode = iv_language_code
      ).
      MESSAGE 'Key phrases detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdunsuppedlanguageex.
      MESSAGE 'Unsupported language.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetectKeyPhrases in der AWS SDK for SAP ABAP API-Referenz.

`DetectPiiEntities`

Das folgende Codebeispiel zeigt die Verwendung`DetectPiiEntities`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_cpd->detectpiientities(
        iv_text = iv_text
        iv_languagecode = iv_language_code
      ).
      MESSAGE 'PII entities detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdunsuppedlanguageex.
      MESSAGE 'Unsupported language.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
```

```
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetectPiiEntities in der AWS SDK for SAP ABAP API-Referenz.

`DetectSentiment`

Das folgende Codebeispiel zeigt die Verwendung`DetectSentiment`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_cpd->detectsentiment(
        iv_text = iv_text
        iv_languagecode = iv_language_code
      ).
      MESSAGE 'Sentiment detected.' TYPE 'I'.
    CATCH /aws1/cx_cpdtextsizelmtexcdex.
      MESSAGE 'Text size exceeds limit.' TYPE 'E'.
    CATCH /aws1/cx_cpdunsuppedlanguageex.
      MESSAGE 'Unsupported language.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetectSentiment in der AWS SDK for SAP ABAP API-Referenz.

`DetectSyntax`

Das folgende Codebeispiel zeigt die Verwendung`DetectSyntax`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->detectsyntax(
      iv_text = iv_text
      iv_languagecode = iv_language_code
    ).
    MESSAGE 'Syntax tokens detected.' TYPE 'I'.
  CATCH /aws1/cx_cpdtextsizelmtexcdex.
    MESSAGE 'Text size exceeds limit.' TYPE 'E'.
  CATCH /aws1/cx_cpdunsuppedlanguageex.
    MESSAGE 'Unsupported language.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetectSyntax in der AWS SDK for SAP ABAP API-Referenz.

`ListDocumentClassificationJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListDocumentClassificationJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      oo_result = lo_cpd->listdocclassificationjobs( ).
      MESSAGE 'Document classification jobs listed.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidfilterex.
      MESSAGE 'Invalid filter.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListDocumentClassificationJobs in der AWS SDK for SAP ABAP API-Referenz.

`ListDocumentClassifiers`

Das folgende Codebeispiel zeigt die Verwendung`ListDocumentClassifiers`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_cpd->listdocumentclassifiers( ).
      MESSAGE 'Document classifiers listed.' TYPE 'I'.
    CATCH /aws1/cx_cpdinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
    CATCH /aws1/cx_cpdtoomanyrequestsex.
      MESSAGE 'Too many requests.' TYPE 'E'.
    CATCH /aws1/cx_cpdinvalidfilterex.
      MESSAGE 'Invalid filter.' TYPE 'E'.
    CATCH /aws1/cx_cpdinternalserverex.
      MESSAGE 'Internal server error occurred.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListDocumentClassifiers in der AWS SDK for SAP ABAP API-Referenz.

`ListTopicsDetectionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTopicsDetectionJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->listtopicsdetectionjobs( ).
    MESSAGE 'Topics detection jobs listed.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdinvalidfilterex.
    MESSAGE 'Invalid filter.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListTopicsDetectionJobs in der AWS SDK for SAP ABAP API-Referenz.

`StartDocumentClassificationJob`

Das folgende Codebeispiel zeigt die Verwendung`StartDocumentClassificationJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_cpd->startdocclassificationjob(
          iv_jobname = iv_job_name
          iv_documentclassifierarn = iv_classifier_arn
          io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
            iv_s3uri = iv_input_s3_uri
            iv_inputformat = iv_input_format
          )
          io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
            iv_s3uri = iv_output_s3_uri
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
        ).
        MESSAGE 'Document classification job started.' TYPE 'I'.
      CATCH /aws1/cx_cpdinvalidrequestex.
        MESSAGE 'Invalid request.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanyrequestsex.
        MESSAGE 'Too many requests.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'E'.
      CATCH /aws1/cx_cpdresourceunavailex.
        MESSAGE 'Resource unavailable.' TYPE 'E'.
      CATCH /aws1/cx_cpdkmskeyvalidationex.
        MESSAGE 'KMS key validation error.' TYPE 'E'.
      CATCH /aws1/cx_cpdtoomanytagsex.
        MESSAGE 'Too many tags.' TYPE 'E'.
      CATCH /aws1/cx_cpdresrclimitexcdex.
        MESSAGE 'Resource limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_cpdinternalserverex.
        MESSAGE 'Internal server error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie StartDocumentClassificationJob in der AWS SDK for SAP ABAP API-Referenz.

`StartTopicsDetectionJob`

Das folgende Codebeispiel zeigt die Verwendung`StartTopicsDetectionJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_cpd->starttopicsdetectionjob(
      iv_jobname = iv_job_name
      io_inputdataconfig = NEW /aws1/cl_cpdinputdataconfig(
        iv_s3uri = iv_input_s3_uri
        iv_inputformat = iv_input_format
      )
      io_outputdataconfig = NEW /aws1/cl_cpdoutputdataconfig(
        iv_s3uri = iv_output_s3_uri
      )
      iv_dataaccessrolearn = iv_data_access_role_arn
    ).
    MESSAGE 'Topics detection job started.' TYPE 'I'.
  CATCH /aws1/cx_cpdinvalidrequestex.
    MESSAGE 'Invalid request.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanyrequestsex.
    MESSAGE 'Too many requests.' TYPE 'E'.
  CATCH /aws1/cx_cpdkmskeyvalidationex.
    MESSAGE 'KMS key validation error.' TYPE 'E'.
  CATCH /aws1/cx_cpdtoomanytagsex.
    MESSAGE 'Too many tags.' TYPE 'E'.
  CATCH /aws1/cx_cpdresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_cpdinternalserverex.
    MESSAGE 'Internal server error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie StartTopicsDetectionJob in der AWS SDK for SAP ABAP API-Referenz.

AWS Config Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS Config

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteConfigRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteConfigRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
lo_cfs->deleteconfigrule( iv_rule_name ).
MESSAGE 'Deleted AWS Config rule.' TYPE 'I'.
```

- API-Details finden Sie DeleteConfigRule in der AWS SDK for SAP ABAP API-Referenz.

`DescribeConfigRules`

Das folgende Codebeispiel zeigt die Verwendung`DescribeConfigRules`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA(lo_result) = lo_cfs->describeconfigrules(
      it_configrulenames = VALUE /aws1/cl_cfsconfigrulenames_w=>tt_configrulenames(
        ( NEW /aws1/cl_cfsconfigrulenames_w( iv_rule_name ) )
      )
    ).
    ot_cfg_rules = lo_result->get_configrules( ).
    MESSAGE 'Retrieved AWS Config rule data.' TYPE 'I'.
```

- API-Details finden Sie DescribeConfigRules in der AWS SDK for SAP ABAP API-Referenz.

`PutConfigRule`

Das folgende Codebeispiel zeigt die Verwendung`PutConfigRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create a config rule for S3 bucket public read prohibition
    lo_cfs->putconfigrule(
      io_configrule = NEW /aws1/cl_cfsconfigrule(
        iv_configrulename = iv_rule_name
        iv_description = |S3 Public Read Prohibited Bucket Rule|
        io_scope = NEW /aws1/cl_cfsscope(
          it_complianceresourcetypes = VALUE /aws1/
cl_cfscplncresrctypes_w=>tt_complianceresourcetypes(
```

```
            ( NEW /aws1/cl_cfscplncresrctypes_w( |AWS::S3::Bucket| ) )
          )
        )
        io_source = NEW /aws1/cl_cfssource(
          iv_owner = |AWS|
          iv_sourceidentifier = |S3_BUCKET_PUBLIC_READ_PROHIBITED|
        )
        iv_inputparameters = '{}'
        iv_configrulestate = |ACTIVE|
      )
    ).
    MESSAGE 'Created AWS Config rule.' TYPE 'I'.
```

- API-Details finden Sie PutConfigRule in der AWS SDK for SAP ABAP API-Referenz.

AWS Control Tower Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS Control Tower

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DisableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`DisableBaseline`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Disable the baseline
        DATA(lo_output) = io_ctt->disablebaseline(
          iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
        ).

        DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

        " Wait for operation to complete
        DATA lv_status TYPE /aws1/cttbaselineopstatus.
        DO 100 TIMES.
          lv_status = get_baseline_operation(
            io_ctt = io_ctt
            iv_operation_id = lv_operation_id
          ).

          DATA(lv_msg) = |Baseline operation status: { lv_status }|.
          MESSAGE lv_msg TYPE 'I'.

          IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
            EXIT.
          ENDIF.

          " Wait 30 seconds
          WAIT UP TO 30 SECONDS.
        ENDDO.

        ov_operation_id = lv_operation_id.
        MESSAGE 'Baseline disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_cttconflictexception INTO DATA(lo_conflict).
        " Log conflict but don't fail - return empty operation ID
        DATA(lv_msg2) = |Conflict disabling baseline: { lo_conflict->get_text( ) }.
Skipping disable step.|.
        MESSAGE lv_msg2 TYPE 'I'.
```

```
      CLEAR ov_operation_id.
  ENDTRY.
```

- API-Details finden Sie DisableBaseline in der AWS SDK for SAP ABAP API-Referenz.

`DisableControl`

Das folgende Codebeispiel zeigt die Verwendung`DisableControl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  " Disable the control
  DATA(lo_output) = io_ctt->disablecontrol(
    iv_controlidentifier = iv_control_arn
    iv_targetidentifier  = iv_target_identifier
  ).

  DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

  " Wait for operation to complete
  DATA lv_status TYPE /aws1/cttcontrolopstatus.
  DO 100 TIMES.
    lv_status = get_control_operation(
      io_ctt = io_ctt
      iv_operation_id = lv_operation_id
    ).

    DATA(lv_msg) = |Control operation status: { lv_status }|.
    MESSAGE lv_msg TYPE 'I'.

    IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
      EXIT.
    ENDIF.

    " Wait 30 seconds
```

```
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control disabled successfully.' TYPE 'I'.
```

- API-Details finden Sie DisableControl in der AWS SDK for SAP ABAP API-Referenz.

`EnableBaseline`

Das folgende Codebeispiel zeigt die Verwendung`EnableBaseline`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Prepare parameters for enabling baseline
    DATA lt_parameters TYPE /aws1/
cl_cttenbdbaselineparam=>tt_enabledbaselineparameters.

    " Add Identity Center baseline parameter if provided
    IF iv_identity_center_baseline IS NOT INITIAL.
      " Create a JSON document with the baseline ARN value
      DATA(lv_json) = |\{ "IdentityCenterEnabledBaselineArn":
 "{ iv_identity_center_baseline }" \}|.
      DATA(lo_param) = NEW /aws1/cl_cttenbdbaselineparam(
        iv_key = 'IdentityCenterEnabledBaselineArn'
        io_value = /aws1/cl_rt_document=>from_json_str( lv_json )
      ).
      APPEND lo_param TO lt_parameters.
    ENDIF.

    " Enable the baseline
    DATA(lo_output) = io_ctt->enablebaseline(
      iv_baselineidentifier = iv_baseline_identifier
      iv_baselineversion    = iv_baseline_version
```

```
    iv_targetidentifier   = iv_target_identifier
    it_parameters         = lt_parameters
  ).

  DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

  " Wait for operation to complete
  DATA lv_status TYPE /aws1/cttbaselineopstatus.
  DO 100 TIMES.
    lv_status = get_baseline_operation(
      io_ctt = io_ctt
      iv_operation_id = lv_operation_id
    ).

    DATA(lv_msg) = |Baseline operation status: { lv_status }|.
    MESSAGE lv_msg TYPE 'I'.

    IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
      EXIT.
    ENDIF.

    " Wait 30 seconds
    WAIT UP TO 30 SECONDS.
  ENDDO.

  ov_enabled_baseline_arn = lo_output->get_arn( ).
  MESSAGE 'Baseline enabled successfully.' TYPE 'I'.
```

- API-Details finden Sie EnableBaseline in der AWS SDK for SAP ABAP API-Referenz.

`EnableControl`

Das folgende Codebeispiel zeigt die Verwendung`EnableControl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Enable the control
    DATA(lo_output) = io_ctt->enablecontrol(
      iv_controlidentifier = iv_control_arn
      iv_targetidentifier  = iv_target_identifier
    ).

    DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

    " Wait for operation to complete
    DATA lv_status TYPE /aws1/cttcontrolopstatus.
    DO 100 TIMES.
      lv_status = get_control_operation(
        io_ctt = io_ctt
        iv_operation_id = lv_operation_id
      ).

      DATA(lv_msg) = |Control operation status: { lv_status }|.
      MESSAGE lv_msg TYPE 'I'.

      IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
        EXIT.
      ENDIF.

      " Wait 30 seconds
      WAIT UP TO 30 SECONDS.
    ENDDO.

    ov_operation_id = lv_operation_id.
    MESSAGE 'Control enabled successfully.' TYPE 'I'.
```

- API-Details finden Sie EnableControl in der AWS SDK for SAP ABAP API-Referenz.

`GetBaselineOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetBaselineOperation`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_output) = io_ctt->getbaselineoperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_baselineoperation( )->get_status( ).
```

- API-Details finden Sie GetBaselineOperation in der AWS SDK for SAP ABAP API-Referenz.

`GetControlOperation`

Das folgende Codebeispiel zeigt die Verwendung`GetControlOperation`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_output) = io_ctt->getcontroloperation(
  iv_operationidentifier = iv_operation_id
).

ov_status = lo_output->get_controloperation( )->get_status( ).
```

- API-Details finden Sie GetControlOperation in der AWS SDK for SAP ABAP API-Referenz.

`ListBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListBaselines`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA lt_baselines TYPE /aws1/cl_cttbaselinesummary=>tt_baselines.
DATA lv_nexttoken TYPE /aws1/cttstring.

" List all baselines using pagination
DO.
  DATA(lo_output) = io_ctt->listbaselines(
    iv_nexttoken = lv_nexttoken
  ).

  APPEND LINES OF lo_output->get_baselines( ) TO lt_baselines.

  lv_nexttoken = lo_output->get_nexttoken( ).
  IF lv_nexttoken IS INITIAL.
    EXIT.
  ENDIF.
ENDDO.

ot_baselines = lt_baselines.
MESSAGE 'Listed baselines successfully.' TYPE 'I'.
```

- API-Details finden Sie ListBaselines in der AWS SDK for SAP ABAP API-Referenz.

`ListEnabledBaselines`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledBaselines`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_enabled_baselines TYPE /aws1/
cl_cttenbdbaselinesumm=>tt_enabledbaselines.
    DATA lv_nexttoken TYPE /aws1/cttlstenbdbaselinesnex00.

    " List all enabled baselines using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledbaselines(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_enabledbaselines( ) TO lt_enabled_baselines.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_baselines = lt_enabled_baselines.
    MESSAGE 'Listed enabled baselines successfully.' TYPE 'I'.
```

- API-Details finden Sie ListEnabledBaselines in der AWS SDK for SAP ABAP API-Referenz.

`ListEnabledControls`

Das folgende Codebeispiel zeigt die Verwendung`ListEnabledControls`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_enabled_controls TYPE /aws1/
cl_cttenabledcontrolsumm=>tt_enabledcontrols.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all enabled controls using pagination
    DO.
      DATA(lo_output) = io_ctt->listenabledcontrols(
        iv_targetidentifier = iv_target_identifier
        iv_nexttoken        = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_enabledcontrols( ) TO lt_enabled_controls.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_enabled_controls = lt_enabled_controls.
    MESSAGE 'Listed enabled controls successfully.' TYPE 'I'.
```

- API-Details finden Sie ListEnabledControls in der AWS SDK for SAP ABAP API-Referenz.

`ListLandingZones`

Das folgende Codebeispiel zeigt die Verwendung`ListLandingZones`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_landing_zones TYPE /aws1/
cl_cttlandingzonesummary=>tt_landingzonesummaries.
    DATA lv_nexttoken TYPE /aws1/cttstring.

    " List all landing zones using pagination
    DO.
      DATA(lo_output) = io_ctt->listlandingzones(
        iv_nexttoken = lv_nexttoken
      ).

      APPEND LINES OF lo_output->get_landingzones( ) TO lt_landing_zones.

      lv_nexttoken = lo_output->get_nexttoken( ).
      IF lv_nexttoken IS INITIAL.
        EXIT.
      ENDIF.
    ENDDO.

    ot_landing_zones = lt_landing_zones.
    MESSAGE 'Listed landing zones successfully.' TYPE 'I'.
```

- API-Details finden Sie ListLandingZones in der AWS SDK for SAP ABAP API-Referenz.

`ResetEnabledBaseline`

Das folgende Codebeispiel zeigt die Verwendung`ResetEnabledBaseline`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Reset the enabled baseline
DATA(lo_output) = io_ctt->resetenabledbaseline(
  iv_enabledbaselineidentifier = iv_enabled_baseline_identifier
).

DATA(lv_operation_id) = lo_output->get_operationidentifier( ).

" Wait for operation to complete
DATA lv_status TYPE /aws1/cttbaselineopstatus.
DO 100 TIMES.
  lv_status = get_baseline_operation(
    io_ctt = io_ctt
    iv_operation_id = lv_operation_id
  ).

  DATA(lv_msg) = |Baseline operation status: { lv_status }|.
  MESSAGE lv_msg TYPE 'I'.

  IF lv_status = 'SUCCEEDED' OR lv_status = 'FAILED'.
    EXIT.
  ENDIF.

  " Wait 30 seconds
  WAIT UP TO 30 SECONDS.
ENDDO.

ov_operation_id = lv_operation_id.
MESSAGE 'Baseline reset successfully.' TYPE 'I'.
```

- API-Details finden Sie ResetEnabledBaseline in der AWS SDK for SAP ABAP API-Referenz.

Firehose-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Firehose Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung. `PutRecord`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_record) = NEW /aws1/cl_frhrecord( iv_data = iv_data ).

        DATA(lo_result) = lo_frh->putrecord(
          iv_deliverystreamname = iv_deliv_stream_name
          io_record             = lo_record ).

        MESSAGE 'Record sent to Firehose delivery stream.' TYPE 'I'.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
```

```
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutRecord in der AWS SDK for SAP ABAP API-Referenz.

`PutRecordBatch`

Das folgende Codebeispiel zeigt die Verwendung`PutRecordBatch`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_frh->putrecordbatch(
          iv_deliverystreamname = iv_deliv_stream_name
          it_records            = it_records ).

        DATA(lv_failed_count) = lo_result->get_failedputcount( ).

        IF lv_failed_count > 0.
          MESSAGE |{ lv_failed_count } records failed to send.| TYPE 'I'.
        ELSE.
          MESSAGE 'All records sent successfully to Firehose delivery stream.' TYPE
'I'.
        ENDIF.
      CATCH /aws1/cx_frhresourcenotfoundex.
        MESSAGE 'Delivery stream not found.' TYPE 'E'.
      CATCH /aws1/cx_frhinvalidargumentex.
        MESSAGE 'Invalid argument provided.' TYPE 'E'.
      CATCH /aws1/cx_frhserviceunavailex.
        MESSAGE 'Service temporarily unavailable.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutRecordBatch in der AWS SDK for SAP ABAP API-Referenz.

DynamoDB-Beispiele unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit DynamoDB Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create an Amazon Dynamo DB table.

    TRY.
        DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
        DATA(lo_dyn) = /aws1/cl_dyn_factory=>create( lo_session ).
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        DATA(oo_result) = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
 returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " It throws exception if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
```

```
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
 { lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.

    " Describe table
    TRY.
        DATA(lo_table) = lo_dyn->describetable( iv_tablename = iv_table_name ).
        DATA(lv_tablename) = lo_table->get_table( )->ask_tablename( ).
        MESSAGE 'The table name is ' && lv_tablename TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table does not exist' TYPE 'E'.
    ENDTRY.

    " Put items into the table.
    TRY.
        DATA(lo_resp_putitem) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1975' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.5' }| ) ) )
          ) ).
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s = 'Star
 Wars' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1978' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '8.1' }| ) ) )
          ) ).
```

```
        lo_resp_putitem = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = VALUE /aws1/
cl_dynattributevalue=>tt_putiteminputattributemap(
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Speed' ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '1994' }| ) ) )
            ( VALUE /aws1/cl_dynattributevalue=>ts_putiteminputattrmap_maprow(
              key = 'rating' value = NEW /aws1/cl_dynattributevalue( iv_n = |
{ '7.9' }| ) ) )
          ) ).
        " TYPE REF TO /AWSEX/CL_AWS1_dyn_PUT_ITEM_OUTPUT
        MESSAGE '3 rows inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Get item from table.
    TRY.
        DATA(lo_resp_getitem) = lo_dyn->getitem(
          iv_tablename                = iv_table_name
          it_key                      = VALUE /aws1/cl_dynattributevalue=>tt_key(
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 'Jaws' ) ) )
           ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
             key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n =
 '1975' ) ) )
          ) ).
        DATA(lt_attr) = lo_resp_getitem->get_item( ).
        DATA(lo_title) = lt_attr[ key = 'title' ]-value.
        DATA(lo_year) = lt_attr[ key = 'year' ]-value.
        DATA(lo_rating) = lt_attr[ key = 'year' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
```

```
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Query item from table.
    TRY.
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
              ( NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) ).
        DATA(lt_keyconditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
            it_attributevaluelist = lt_attributelist
            iv_comparisonoperator = |EQ|
          ) ) ) ).
        DATA(lo_query_result) = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_keyconditions ).
        DATA(lt_items) = lo_query_result->get_items( ).
        READ TABLE lo_query_result->get_items( ) INTO DATA(lt_item) INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Scan items from table.
    TRY.
        DATA(lo_scan_result) = lo_dyn->scan( iv_tablename = iv_table_name ).
        lt_items = lo_scan_result->get_items( ).
        " Read the first item and display the attributes.
        READ TABLE lo_query_result->get_items( ) INTO lt_item INDEX 1.
        lo_title = lt_item[ key = 'title' ]-value.
        lo_year = lt_item[ key = 'year' ]-value.
        lo_rating = lt_item[ key = 'rating' ]-value.
        MESSAGE 'Movie name is: ' && lo_title->get_s( ) TYPE 'I'.
        MESSAGE 'Movie year is: ' && lo_year->get_n( ) TYPE 'I'.
        MESSAGE 'Movie rating is: ' && lo_rating->get_n( ) TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
```

```
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.

    " Update items from table.
    TRY.
        DATA(lt_attributeupdates) = VALUE /aws1/
cl_dynattrvalueupdate=>tt_attributeupdates(
          ( VALUE /aws1/cl_dynattrvalueupdate=>ts_attributeupdates_maprow(
          key = 'rating' value = NEW /aws1/cl_dynattrvalueupdate(
            io_value  = NEW /aws1/cl_dynattributevalue( iv_n = '7.6' )
            iv_action = |PUT| ) ) ) ).
        DATA(lt_key) = VALUE /aws1/cl_dynattributevalue=>tt_key(
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'year' value = NEW /aws1/cl_dynattributevalue( iv_n = '1975' ) ) )
          ( VALUE /aws1/cl_dynattributevalue=>ts_key_maprow(
            key = 'title' value = NEW /aws1/cl_dynattributevalue( iv_s =
 '1980' ) ) ) ).
        DATA(lo_resp) = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = lt_key
          it_attributeupdates = lt_attributeupdates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.

    " Delete table.
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted as it is in use' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lt_keyschema) = VALUE /aws1/cl_dynkeyschemaelement=>tt_keyschema(
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'year'
                                              iv_keytype = 'HASH' ) )
          ( NEW /aws1/cl_dynkeyschemaelement( iv_attributename = 'title'
                                              iv_keytype = 'RANGE' ) ) ).
        DATA(lt_attributedefinitions) = VALUE /aws1/
cl_dynattributedefn=>tt_attributedefinitions(
```

```
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'year'
                                           iv_attributetype = 'N' ) )
          ( NEW /aws1/cl_dynattributedefn( iv_attributename = 'title'
                                           iv_attributetype = 'S' ) ) ).

        " Adjust read/write capacities as desired.
        DATA(lo_dynprovthroughput)  = NEW /aws1/cl_dynprovthroughput(
          iv_readcapacityunits = 5
          iv_writecapacityunits = 5 ).
        oo_result = lo_dyn->createtable(
          it_keyschema = lt_keyschema
          iv_tablename = iv_table_name
          it_attributedefinitions = lt_attributedefinitions
          io_provisionedthroughput = lo_dynprovthroughput ).
        " Table creation can take some time. Wait till table exists before
returning.
        lo_dyn->get_waiter( )->tableexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'DynamoDB Table' && iv_table_name && 'created.' TYPE 'I'.
        " This exception can happen if the table already exists.
      CATCH /aws1/cx_dynresourceinuseex INTO DATA(lo_resourceinuseex).
        DATA(lv_error) = |"{ lo_resourceinuseex->av_err_code }" -
{ lo_resourceinuseex->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateTable in der AWS SDK for SAP ABAP API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_resp) = lo_dyn->deleteitem(
          iv_tablename                = iv_table_name
          it_key                      = it_key_input ).
        MESSAGE 'Deleted one item.' TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteItem in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_dyn->deletetable( iv_tablename = iv_table_name ).
        " Wait till the table is actually deleted.
        lo_dyn->get_waiter( )->tablenotexists(
          iv_max_wait_time = 200
          iv_tablename     = iv_table_name ).
        MESSAGE 'Table ' && iv_table_name && ' deleted.' TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table ' && iv_table_name && ' does not exist' TYPE 'E'.
      CATCH /aws1/cx_dynresourceinuseex.
        MESSAGE 'The table cannot be deleted since it is in use' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteTable in der AWS SDK for SAP ABAP API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_dyn->describetable( iv_tablename = iv_table_name ).
    DATA(lv_tablename) = oo_result->get_table( )->ask_tablename( ).
    DATA(lv_tablearn) = oo_result->get_table( )->ask_tablearn( ).
    DATA(lv_tablestatus) = oo_result->get_table( )->ask_tablestatus( ).
    DATA(lv_itemcount) = oo_result->get_table( )->ask_itemcount( ).
    MESSAGE 'The table name is ' && lv_tablename
        && '. The table ARN is ' && lv_tablearn
        && '. The tablestatus is ' && lv_tablestatus
        && '. Item count is ' && lv_itemcount TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table ' && lv_tablename && ' does not exist' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DescribeTable in der AWS SDK for SAP ABAP API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_item = lo_dyn->getitem(
      iv_tablename                = iv_table_name
      it_key                      = it_key ).
    DATA(lt_attr) = oo_item->get_item( ).
    DATA(lo_title) = lt_attr[ key = 'title' ]-value.
    DATA(lo_year) = lt_attr[ key = 'year' ]-value.
    DATA(lo_rating) = lt_attr[ key = 'rating' ]-value.
    MESSAGE 'Movie name is: ' && lo_title->get_s( )
      && 'Movie year is: ' && lo_year->get_n( )
      && 'Moving rating is: ' && lo_rating->get_n( ) TYPE 'I'.
  CATCH /aws1/cx_dynresourcenotfoundex.
    MESSAGE 'The table or index does not exist' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetItem in der AWS SDK for SAP ABAP API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        oo_result = lo_dyn->listtables( ).
        " You can loop over the oo_result to get table properties like this.
        LOOP AT oo_result->get_tablenames( ) INTO DATA(lo_table_name).
          DATA(lv_tablename) = lo_table_name->get_value( ).
        ENDLOOP.
        DATA(lv_tablecount) = lines( oo_result->get_tablenames( ) ).
        MESSAGE 'Found ' && lv_tablecount && ' tables' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListTables in der AWS SDK for SAP ABAP API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_resp) = lo_dyn->putitem(
          iv_tablename = iv_table_name
          it_item      = it_item ).
        MESSAGE '1 row inserted into DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
 TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutItem in der AWS SDK for SAP ABAP API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Query movies for a given year .
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_year }| ) ) ).
        DATA(lt_key_conditions) = VALUE /aws1/cl_dyncondition=>tt_keyconditions(
          ( VALUE /aws1/cl_dyncondition=>ts_keyconditions_maprow(
          key = 'year'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |EQ|
          ) ) ) ).
        oo_result = lo_dyn->query(
          iv_tablename = iv_table_name
          it_keyconditions = lt_key_conditions ).
        DATA(lt_items) = oo_result->get_items( ).
        "You can loop over the results to get item attributes.
        LOOP AT lt_items INTO DATA(lt_item).
          DATA(lo_title) = lt_item[ key = 'title' ]-value.
          DATA(lo_year) = lt_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_result->get_count( ).
        MESSAGE 'Item count is: ' && lv_count TYPE 'I'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
```

```
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Abfragen in der API-Referenz für das AWS SDK für SAP ABAP.

Scan

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Scan movies for rating greater than or equal to the rating specified
        DATA(lt_attributelist) = VALUE /aws1/
cl_dynattributevalue=>tt_attributevaluelist(
            ( NEW /aws1/cl_dynattributevalue( iv_n = |{ iv_rating }| ) ) ).
        DATA(lt_filter_conditions) = VALUE /aws1/
cl_dyncondition=>tt_filterconditionmap(
          ( VALUE /aws1/cl_dyncondition=>ts_filterconditionmap_maprow(
          key = 'rating'
          value = NEW /aws1/cl_dyncondition(
          it_attributevaluelist = lt_attributelist
          iv_comparisonoperator = |GE|
          ) ) ) ).
        oo_scan_result = lo_dyn->scan( iv_tablename = iv_table_name
          it_scanfilter = lt_filter_conditions ).
        DATA(lt_items) = oo_scan_result->get_items( ).
        LOOP AT lt_items INTO DATA(lo_item).
          " You can loop over to get individual attributes.
          DATA(lo_title) = lo_item[ key = 'title' ]-value.
          DATA(lo_year) = lo_item[ key = 'year' ]-value.
        ENDLOOP.
        DATA(lv_count) = oo_scan_result->get_count( ).
        MESSAGE 'Found ' && lv_count && ' items' TYPE 'I'.
```

```
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Scan in der API-Referenz für das AWS -SDK für SAP ABAP.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_output = lo_dyn->updateitem(
          iv_tablename        = iv_table_name
          it_key              = it_item_key
          it_attributeupdates = it_attribute_updates ).
        MESSAGE '1 item updated in DynamoDB Table' && iv_table_name TYPE 'I'.
      CATCH /aws1/cx_dyncondalcheckfaile00.
        MESSAGE 'A condition specified in the operation could not be evaluated.'
TYPE 'E'.
      CATCH /aws1/cx_dynresourcenotfoundex.
        MESSAGE 'The table or index does not exist' TYPE 'E'.
      CATCH /aws1/cx_dyntransactconflictex.
        MESSAGE 'Another transaction is using the item' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie UpdateItem in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon EC2 unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon EC2 Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung. `AllocateAddress`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).   " oo_result is
 returned for testing purposes. "
        MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AllocateAddress in der AWS SDK for SAP ABAP API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->associateaddress(                        " oo_result is
 returned for testing purposes. "
            iv_allocationid = iv_allocation_id
            iv_instanceid = iv_instance_id ).
        MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AssociateAddress in der AWS SDK for SAP ABAP API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create IP permissions for SSH access (port 22)
    " iv_cidr_ip = '192.0.2.0/24'
    DATA lt_ip_permissions TYPE /aws1/cl_ec2ippermission=>tt_ippermissionlist.
    DATA(lo_ip_permission) = NEW /aws1/cl_ec2ippermission(
      iv_ipprotocol = 'tcp'
      iv_fromport = 22
      iv_toport = 22
      it_ipranges = VALUE /aws1/cl_ec2iprange=>tt_iprangelist(
        ( NEW /aws1/cl_ec2iprange( iv_cidrip = iv_cidr_ip ) )
      )
    ).
    APPEND lo_ip_permission TO lt_ip_permissions.

    TRY.
        oo_result = lo_ec2->authsecuritygroupingress(             " oo_result is
 returned for testing purposes. "
          iv_groupid = iv_group_id
          it_ippermissions = lt_ip_permissions ).
        MESSAGE 'Authorized ingress rule for security group.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK for SAP ABAP API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
              " oo_result is returned for testing purposes. "
        MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateKeyPair in der AWS SDK for SAP ABAP API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->createsecuritygroup(                 " oo_result is
 returned for testing purposes. "
          iv_description = 'Security group example'
          iv_groupname = iv_security_group_name
          iv_vpcid = iv_vpc_id ).
        MESSAGE 'Security group created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK for SAP ABAP API-Referenz.

`CreateVpc`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpc`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_cidr_block = '10.0.0.0/16'
    TRY.
        oo_result = lo_ec2->createvpc( iv_cidrblock = iv_cidr_block ).            "
 oo_result is returned for testing purposes. "
        DATA(lv_vpc_id) = oo_result->get_vpc( )->get_vpcid( ).
        MESSAGE 'Created VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateVpc in der AWS SDK for SAP ABAP API-Referenz.

`CreateVpcEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`CreateVpcEndpoint`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_vpc_id = 'vpc-abc123'
```

```
    " iv_service_name = 'com.amazonaws.region.service'
    TRY.
        oo_result = lo_ec2->createvpcendpoint(             " oo_result is returned
 for testing purposes. "
          iv_vpcid = iv_vpc_id
          iv_servicename = iv_service_name
          it_routetableids = it_route_table_ids ).
        DATA(lv_vpc_endpoint_id) = oo_result->get_vpcendpoint( )-
>get_vpcendpointid( ).
        MESSAGE 'Created VPC endpoint.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateVpcEndpoint in der AWS SDK for SAP ABAP API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
        MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK for SAP ABAP API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).
        MESSAGE 'Security group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK for SAP ABAP API-Referenz.

`DeleteVpc`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVpc`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        lo_ec2->deletevpc( iv_vpcid = iv_vpc_id ).
        MESSAGE 'Deleted VPC.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteVpc in der AWS SDK for SAP ABAP API-Referenz.

`DeleteVpcEndpoints`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVpcEndpoints`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->deletevpcendpoints( it_vpcendpointids = it_vpc_endpoint_ids ).
        MESSAGE 'Deleted VPC endpoint(s).' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteVpcEndpoints in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAddresses`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAddresses`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeaddresses( ).                       " oo_result
 is returned for testing purposes. "
        DATA(lt_addresses) = oo_result->get_addresses( ).
        MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeAddresses in der AWS SDK for SAP ABAP API-Referenz.

`DescribeAvailabilityZones`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAvailabilityZones`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeavailabilityzones( ).                       "
 oo_result is returned for testing purposes. "
        DATA(lt_zones) = oo_result->get_availabilityzones( ).
        MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeAvailabilityZones in der AWS SDK for SAP ABAP API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeimages( it_imageids = it_image_ids ).
  " oo_result is returned for testing purposes. "
        DATA(lt_images) = oo_result->get_images( ).
        MESSAGE 'Retrieved information about Amazon Machine Images (AMIs).' TYPE
 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeImages in der AWS SDK for SAP ABAP API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create filters for architecture and instance type patterns
    " iv_architecture = 'x86_64'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'processor-info.supported-architecture'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_architecture ) )
      )
    ) TO lt_filters.
    " Filter for instance type patterns like '*.micro', '*.small'
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'instance-type'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.micro' ) )
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.small' ) )
      )
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeinstancetypes( it_filters = lt_filters ).
      " oo_result is returned for testing purposes. "
        DATA(lt_instance_types) = oo_result->get_instancetypes( ).
        MESSAGE 'Retrieved information about EC2 instance types.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK for SAP ABAP API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeinstances( ).                       " oo_result
 is returned for testing purposes. "

        " Retrieving details of EC2 instances. "
        DATA: lv_istance_id    TYPE /aws1/ec2string,
              lv_status        TYPE /aws1/ec2instancestatename,
              lv_instance_type TYPE /aws1/ec2instancetype,
              lv_image_id      TYPE /aws1/ec2string.
        LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
          LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_istance_id = lo_instance->get_instanceid( ).
            lv_status = lo_instance->get_state( )->get_name( ).
            lv_instance_type = lo_instance->get_instancetype( ).
            lv_image_id = lo_instance->get_imageid( ).
          ENDLOOP.
        ENDLOOP.
        MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeInstances in der AWS SDK for SAP ABAP API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describekeypairs( ).                        " oo_result
 is returned for testing purposes. "
        DATA(lt_key_pairs) = oo_result->get_keypairs( ).
        MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK for SAP ABAP API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ec2->describeregions( ).                         " oo_result
 is returned for testing purposes. "
        DATA(lt_regions) = oo_result->get_regions( ).
        MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeRegions in der AWS SDK for SAP ABAP API-Referenz.

`DescribeRouteTables`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRouteTables`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create filter for VPC ID
    " iv_vpc_id = 'vpc-abc123'
    DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
    APPEND NEW /aws1/cl_ec2filter(
      iv_name = 'vpc-id'
      it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_vpc_id ) )
      )
    ) TO lt_filters.

    TRY.
        oo_result = lo_ec2->describeroutetables( it_filters = lt_filters ).
    " oo_result is returned for testing purposes. "
        DATA(lt_route_tables) = oo_result->get_routetables( ).
        MESSAGE 'Retrieved information about route tables.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
```

```
    ENDTRY.
```

- API-Details finden Sie DescribeRouteTables in der AWS SDK for SAP ABAP API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA lt_group_ids TYPE /aws1/cl_ec2groupidstrlist_w=>tt_groupidstringlist.
        APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
 lt_group_ids.
        oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
      " oo_result is returned for testing purposes. "
        DATA(lt_security_groups) = oo_result->get_securitygroups( ).
        MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for SAP ABAP API-Referenz.

`MonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`MonitorInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to monitor
 the instance without actually making the request. "
        lo_ec2->monitorinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to monitor this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
          " DryRun is set to false to enable detailed monitoring. "
          lo_ec2->monitorinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to monitor this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to enable detailed monitoring failed. User does not have
 the permissions to monitor the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- API-Details finden Sie MonitorInstances in der AWS SDK for SAP ABAP API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to reboot
 the instance without actually making the request. "
        lo_ec2->rebootinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to reboot this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
          " DryRun is set to false to make a reboot request. "
          lo_ec2->rebootinstances(
             it_instanceids = lt_instance_ids
             iv_dryrun = abap_false ).
          MESSAGE 'Instance rebooted.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to reboot this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
```

```
          MESSAGE 'Dry run to reboot instance failed. User does not have permissions
 to reboot the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- API-Details finden Sie RebootInstances in der AWS SDK for SAP ABAP API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
        MESSAGE 'Elastic IP address released.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ReleaseAddress in der AWS SDK for SAP ABAP API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create tags for resource created during instance launch. "
    DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
    DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
    ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
      iv_resourcetype = 'instance'
      it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
      ) ).
    APPEND ls_tagspecifications TO lt_tagspecifications.

    TRY.
        " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
        oo_result = lo_ec2->runinstances(                           " oo_result is
 returned for testing purposes. "
          iv_imageid = iv_ami_id
          iv_instancetype = 't3.micro'
          iv_maxcount = 1
          iv_mincount = 1
          it_tagspecifications = lt_tagspecifications
          iv_subnetid = iv_subnet_id ).
        MESSAGE 'EC2 instance created.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie RunInstances in der AWS SDK for SAP ABAP API-Referenz.

StartInstances

Das folgende Codebeispiel zeigt die VerwendungStartInstances.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to start
 the instance without actually making the request. "
        lo_ec2->startinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to start this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
          " DryRun is set to false to start instance. "
          oo_result = lo_ec2->startinstances(           " oo_result is returned for
 testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to start instance failed. User does not have permissions
 to start the instance.' TYPE 'E'.
        ELSE.
```

```
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- API-Details finden Sie StartInstances in der AWS SDK for SAP ABAP API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
    APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
 lt_instance_ids.

    "Perform dry run"
    TRY.
        " DryRun is set to true. This checks for the required permissions to stop
 the instance without actually making the request. "
        lo_ec2->stopinstances(
          it_instanceids = lt_instance_ids
          iv_dryrun = abap_true ).
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is `DryRunOperation`, then you have the
 required permissions to stop this instance. "
        IF lo_exception->av_err_code = 'DryRunOperation'.
          MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
          " DryRun is set to false to stop instance. "
          oo_result = lo_ec2->stopinstances(           " oo_result is returned for
 testing purposes. "
```

```
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false ).
          MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
          " If the error code returned is `UnauthorizedOperation`, then you don't
 have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
          MESSAGE 'Dry run to stop instance failed. User does not have permissions
 to stop the instance.' TYPE 'E'.
        ELSE.
          DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
          MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- API-Details finden Sie StopInstances in der AWS SDK for SAP ABAP API-Referenz.

Amazon ECR-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon ECR Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateRepository`

Das folgende Codebeispiel zeigt die Verwendung. `CreateRepository`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->createrepository(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_repository_uri) = oo_result->get_repository( )-
>get_repositoryuri( ).
        MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryalrexex.
        " If repository already exists, retrieve it
        DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.
        APPEND NEW /aws1/cl_ecrrepositorynamels00( iv_value = iv_repository_name )
 TO lt_repo_names.
        DATA(lo_describe_result) = lo_ecr->describerepositories( it_repositorynames
 = lt_repo_names ).
        DATA(lt_repos) = lo_describe_result->get_repositories( ).
        IF lines( lt_repos ) > 0.
          READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
          oo_result = NEW /aws1/cl_ecrcrerepositoryrsp( io_repository = lo_repo ).
          MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
        ENDIF.
    ENDTRY.
```

- API-Details finden Sie CreateRepository in der AWS SDK for SAP ABAP API-Referenz.

`DeleteRepository`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRepository`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        lo_ecr->deleterepository(
          iv_repositoryname = iv_repository_name
          iv_force = abap_true ).
        MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteRepository in der AWS SDK for SAP ABAP API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " it_image_ids = VALUE #( ( NEW /aws1/cl_ecrimageidentifier( iv_imagetag =
'latest' ) ) )
        IF it_image_ids IS NOT INITIAL.
          oo_result = lo_ecr->describeimages(
```

```
            iv_repositoryname = iv_repository_name
            it_imageids = it_image_ids ).
        ELSE.
          oo_result = lo_ecr->describeimages(
            iv_repositoryname = iv_repository_name ).
        ENDIF.
        DATA(lt_image_details) = oo_result->get_imagedetails( ).
        MESSAGE |Found { lines( lt_image_details ) } images in repository.| TYPE
 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrimagenotfoundex.
        MESSAGE 'Image not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrinvalidparameterex.
        MESSAGE 'Invalid parameter provided.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeImages in der AWS SDK for SAP ABAP API-Referenz.

`DescribeRepositories`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRepositories`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " it_repository_names = VALUE #( ( NEW /aws1/
cl_ecrrepositorynamels00( iv_value = 'my-repository' ) ) )
        oo_result = lo_ecr->describerepositories(
          it_repositorynames = it_repository_names ).
        DATA(lt_repositories) = oo_result->get_repositories( ).
        MESSAGE |Found { lines( lt_repositories ) } repositories.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
```

```
    ENDTRY.
```

- API-Details finden Sie DescribeRepositories in der AWS SDK for SAP ABAP API-Referenz.

GetAuthorizationToken

Das folgende Codebeispiel zeigt die VerwendungGetAuthorizationToken.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ecr->getauthorizationtoken( ).
        DATA(lt_auth_data) = oo_result->get_authorizationdata( ).
        IF lines( lt_auth_data ) > 0.
          READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
          DATA(lv_token) = lo_auth_data->get_authorizationtoken( ).
          MESSAGE 'Authorization token retrieved.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_ecrserverexception.
        MESSAGE 'Server exception occurred.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie GetAuthorizationToken in der AWS SDK for SAP ABAP API-Referenz.

GetRepositoryPolicy

Das folgende Codebeispiel zeigt die VerwendungGetRepositoryPolicy.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        oo_result = lo_ecr->getrepositorypolicy(
          iv_repositoryname = iv_repository_name ).
        DATA(lv_policy_text) = oo_result->get_policytext( ).
        MESSAGE 'Repository policy retrieved.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrrepositoryplynot00.
        MESSAGE 'Repository policy not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie GetRepositoryPolicy in der AWS SDK for SAP ABAP API-Referenz.

`PutLifeCyclePolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutLifeCyclePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
```

```
        lo_ecr->putlifecyclepolicy(
          iv_repositoryname = iv_repository_name
          iv_lifecyclepolicytext = iv_lifecycle_policy_text ).
        MESSAGE |Lifecycle policy set for repository { iv_repository_name }.| TYPE
'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
      CATCH /aws1/cx_ecrvalidationex.
        MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie PutLifeCyclePolicy in der AWS SDK for SAP ABAP API-Referenz.

`SetRepositoryPolicy`

Das folgende Codebeispiel zeigt die Verwendung`SetRepositoryPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_repository_name = 'my-repository'
        " iv_policy_text = '{"Version":"2012-10-17",       "Statement":[...]}'
        lo_ecr->setrepositorypolicy(
          iv_repositoryname = iv_repository_name
          iv_policytext = iv_policy_text ).
        MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
      CATCH /aws1/cx_ecrrepositorynotfndex.
        MESSAGE 'Repository not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie SetRepositoryPolicy in der AWS SDK for SAP ABAP API-Referenz.

Amazon EMR-Beispiele mit SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon EMR Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AddJobFlowSteps`

Das folgende Codebeispiel zeigt die Verwendung. `AddJobFlowSteps`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Build args list for Spark submit
    DATA lt_args TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( 'spark-submit' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( '--deploy-mode' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( 'cluster' ) TO lt_args.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_script_uri ) TO lt_args.
    APPEND LINES OF it_script_args TO lt_args.

    " Create step configuration
    DATA(lo_hadoop_jar_step) = NEW /aws1/cl_emrhadoopjarstepcfg(
      iv_jar = 'command-runner.jar'
```

```
          it_args = lt_args
        ).

        DATA(lo_step_config) = NEW /aws1/cl_emrstepconfig(
          iv_name = iv_name
          iv_actiononfailure = 'CONTINUE'
          io_hadoopjarstep = lo_hadoop_jar_step
        ).

        DATA lt_steps TYPE /aws1/cl_emrstepconfig=>tt_stepconfiglist.
        APPEND lo_step_config TO lt_steps.

        DATA(lo_result) = lo_emr->addjobflowsteps(
          iv_jobflowid = iv_cluster_id
          it_steps = lt_steps
        ).

        " Get first step ID
        DATA(lt_step_ids) = lo_result->get_stepids( ).
        READ TABLE lt_step_ids INDEX 1 INTO DATA(lo_step_id_obj).
        IF sy-subrc = 0.
          ov_step_id = lo_step_id_obj->get_value( ).
          MESSAGE |Step added with ID { ov_step_id }| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AddJobFlowSteps in der AWS SDK for SAP ABAP API-Referenz.

`DescribeCluster`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCluster`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_emr->describecluster(
          iv_clusterid = iv_cluster_id
        ).
        DATA(lo_cluster) = oo_result->get_cluster( ).
        DATA(lv_cluster_name) = lo_cluster->get_name( ).
        MESSAGE |Retrieved cluster information for { lv_cluster_name }| TYPE 'I'.
      CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
      CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
        lv_error = lo_invalid_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeCluster in der AWS SDK for SAP ABAP API-Referenz.

`DescribeStep`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStep`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_emr->describestep(
          iv_clusterid = iv_cluster_id
          iv_stepid = iv_step_id
        ).
        DATA(lo_step) = oo_result->get_step( ).
        DATA(lv_step_name) = lo_step->get_name( ).
        MESSAGE |Retrieved step information for { lv_step_name }| TYPE 'I'.
      CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
        DATA(lv_error) = lo_internal_error->if_message~get_text( ).
        MESSAGE lv_error TYPE 'E'.
```

```
    CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
      lv_error = lo_invalid_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribeStep in der AWS SDK for SAP ABAP API-Referenz.

`ListSteps`

Das folgende Codebeispiel zeigt die Verwendung`ListSteps`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_emr->liststeps(
        iv_clusterid = iv_cluster_id
      ).
      DATA(lt_steps) = oo_result->get_steps( ).
      DATA(lv_step_count) = lines( lt_steps ).
      MESSAGE |Retrieved { lv_step_count } steps for cluster| TYPE 'I'.
    CATCH /aws1/cx_emrinternalserverex INTO DATA(lo_internal_error).
      DATA(lv_error) = lo_internal_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
    CATCH /aws1/cx_emrinvalidrequestex INTO DATA(lo_invalid_error).
      lv_error = lo_invalid_error->if_message~get_text( ).
      MESSAGE lv_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListSteps in der AWS SDK for SAP ABAP API-Referenz.

`RunJobFlow`

Das folgende Codebeispiel zeigt die Verwendung`RunJobFlow`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create instances configuration
    DATA(lo_instances) = NEW /aws1/cl_emrjobflowinstsconfig(
      iv_masterinstancetype = 'm5.xlarge'
      iv_slaveinstancetype = 'm5.xlarge'
      iv_instancecount = 3
      iv_keepjobflowalivewhennos00 = iv_keep_alive
      iv_emrmanagedmastersecgroup = iv_primary_sec_grp
      iv_emrmanagedslavesecgroup = iv_secondary_sec_grp
    ).

    DATA(lo_result) = lo_emr->runjobflow(
      iv_name = iv_name
      iv_loguri = iv_log_uri
      iv_releaselabel = 'emr-5.30.1'
      io_instances = lo_instances
      it_steps = it_steps
      it_applications = it_applications
      iv_jobflowrole = iv_job_flow_role
      iv_servicerole = iv_service_role
      iv_ebsrootvolumesize = 10
      iv_visibletoallusers = abap_true
    ).

    ov_cluster_id = lo_result->get_jobflowid( ).
    MESSAGE 'EMR cluster created successfully.' TYPE 'I'.
  CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
  CATCH /aws1/cx_emrclientexc INTO DATA(lo_client_error).
    lv_error = lo_client_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie RunJobFlow in der AWS SDK for SAP ABAP API-Referenz.

`TerminateJobFlows`

Das folgende Codebeispiel zeigt die Verwendung`TerminateJobFlows`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA lt_cluster_ids TYPE /aws1/cl_emrxmlstringlist_w=>tt_xmlstringlist.
    APPEND NEW /aws1/cl_emrxmlstringlist_w( iv_cluster_id ) TO lt_cluster_ids.

    lo_emr->terminatejobflows(
      it_jobflowids = lt_cluster_ids
    ).
    MESSAGE 'EMR cluster terminated successfully.' TYPE 'I'.
  CATCH /aws1/cx_emrinternalservererr INTO DATA(lo_internal_error).
    DATA(lv_error) = lo_internal_error->if_message~get_text( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie TerminateJobFlows in der AWS SDK for SAP ABAP API-Referenz.

EventBridge Scheduler-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Scheduler Aktionen ausführen und gängige Szenarien implementieren. EventBridge

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateSchedule`

Das folgende Codebeispiel zeigt die Verwendung. `CreateSchedule`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Constants for time calculations
    DATA lv_start_date TYPE /aws1/scdstartdate.
    DATA lv_end_date TYPE /aws1/scdenddate.
    DATA lv_start_timestamp TYPE timestamp.
    DATA lv_end_timestamp TYPE timestamp.
    DATA lv_hours_to_run TYPE i VALUE 1.

    " Get current timestamp
    GET TIME STAMP FIELD lv_start_timestamp.

    " Add 1 hour to the current timestamp using CL_ABAP_TSTMP
    lv_end_timestamp = cl_abap_tstmp=>add(
      tstmp = lv_start_timestamp
      secs = lv_hours_to_run * 3600 ).

    " Convert timestamps to decimal format for AWS API
    lv_start_date = lv_start_timestamp.
    lv_end_date = lv_end_timestamp.

    " Prepare flexible time window configuration
```

```
DATA lo_flexible_time_window TYPE REF TO /aws1/cl_scdflexibletimewindow.
IF iv_use_flexible_time_win = abap_true.
  " iv_use_flexible_time_win = ABAP_TRUE
  " Example: Set MaximumWindowInMinutes to 10 for flexible window
  lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
    iv_mode = 'FLEXIBLE'
    iv_maximumwindowinminutes = 10 ).
ELSE.
  lo_flexible_time_window = NEW /aws1/cl_scdflexibletimewindow(
    iv_mode = 'OFF' ).
ENDIF.

" Prepare target configuration
" Example iv_target_arn = 'arn:aws:sqs:us-east-1:123456789012:my-queue'
" Example iv_role_arn = 'arn:aws:iam::123456789012:role/SchedulerRole'
" Example iv_input = '{"message": "Hello from EventBridge Scheduler"}'
DATA(lo_target) = NEW /aws1/cl_scdtarget(
  iv_arn = iv_target_arn
  iv_rolearn = iv_role_arn
  iv_input = iv_input ).

" Set action after completion if needed
DATA lv_action_after_completion TYPE /aws1/scdactionaftercompletion.
IF iv_delete_after_completion = abap_true.
  " iv_delete_after_completion = ABAP_TRUE
  lv_action_after_completion = 'DELETE'.
ELSE.
  lv_action_after_completion = 'NONE'.
ENDIF.

" Create the schedule
" Example iv_name = 'my-schedule'
" Example iv_schedule_expression = 'rate(15 minutes)'
" Example iv_schedule_group_name = 'my-schedule-group'
DATA(lo_result) = lo_scd->createschedule(
  iv_name = iv_name
  iv_scheduleexpression = iv_schedule_expression
  iv_groupname = iv_schedule_group_name
  io_target = lo_target
  io_flexibletimewindow = lo_flexible_time_window
  iv_startdate = lv_start_date
  iv_enddate = lv_end_date
  iv_actionaftercompletion = lv_action_after_completion ).
```

```
        ov_schedule_arn = lo_result->get_schedulearn( ).
        MESSAGE 'Schedule created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error creating schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie CreateSchedule in der AWS SDK for SAP ABAP API-Referenz.

`CreateScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateScheduleGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        DATA(lo_result) = lo_scd->createschedulegroup(
          iv_name = iv_name ).

        ov_schedule_group_arn = lo_result->get_schedulegrouparn( ).
        MESSAGE 'Schedule group created successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdconflictexception INTO DATA(lo_conflict_ex).
        DATA(lv_error) = |Conflict creating schedule group: { lo_conflict_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
```

```
        DATA(lv_generic_error) = |Error creating schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie CreateScheduleGroup in der AWS SDK for SAP ABAP API-Referenz.

`DeleteSchedule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSchedule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule'
        " Example iv_schedule_group_name = 'my-schedule-group'
        lo_scd->deleteschedule(
          iv_name = iv_name
          iv_groupname = iv_schedule_group_name ).
        MESSAGE 'Schedule deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteSchedule in der AWS SDK for SAP ABAP API-Referenz.

`DeleteScheduleGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteScheduleGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example iv_name = 'my-schedule-group'
        lo_scd->deleteschedulegroup(
          iv_name = iv_name ).
        MESSAGE 'Schedule group deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_scdresourcenotfoundex INTO DATA(lo_not_found_ex).
        DATA(lv_error) = |Schedule group not found: { lo_not_found_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_generic_ex).
        DATA(lv_generic_error) = |Error deleting schedule group: { lo_generic_ex-
>if_message~get_text( ) }|.
        MESSAGE lv_generic_error TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteScheduleGroup in der AWS SDK for SAP ABAP API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS Glue

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_database_name = 'my-database'
        " iv_table_prefix = 'test_'
        " iv_s3_target = 's3://example-bucket/data/'

        DATA(lt_s3_targets) = VALUE /aws1/cl_glus3target=>tt_s3targetlist(
          ( NEW /aws1/cl_glus3target( iv_path = iv_s3_target ) ) ).

        DATA(lo_targets) = NEW /aws1/cl_glucrawlertargets(
          it_s3targets = lt_s3_targets ).

        lo_glu->createcrawler(
          iv_name = iv_crawler_name
          iv_role = iv_role_arn
          iv_databasename = iv_database_name
          iv_tableprefix = iv_table_prefix
          io_targets = lo_targets ).
        MESSAGE 'Crawler created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
```

```
        MESSAGE 'Crawler already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateCrawler in der AWS SDK for SAP ABAP API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_description = 'ETL job for data transformation'
        " iv_role_arn = 'arn:aws:iam::123456789012:role/AWSGlueServiceRole-Test'
        " iv_script_location = 's3://example-bucket/scripts/my-script.py'

        DATA(lo_command) = NEW /aws1/cl_glujobcommand(
          iv_name = 'glueetl'
          iv_scriptlocation = iv_script_location
          iv_pythonversion = '3' ).

        lo_glu->createjob(
          iv_name = iv_job_name
          iv_description = iv_description
          iv_role = iv_role_arn
```

```
          io_command = lo_command
          iv_glueversion = '3.0' ).
        MESSAGE 'Job created successfully.' TYPE 'I'.
      CATCH /aws1/cx_glualreadyexistsex.
        MESSAGE 'Job already exists.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
      CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
        DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
        MESSAGE lv_limit_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateJob in der AWS SDK for SAP ABAP API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        lo_glu->deletecrawler( iv_name = iv_crawler_name ).
        MESSAGE 'Crawler deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_glucrawlerrunningex.
        MESSAGE 'Crawler is currently running.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Crawler does not exist.' TYPE 'E'.
```

```
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
    CATCH /aws1/cx_gluschdrtransingex.
      MESSAGE 'Scheduler is transitioning.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DeleteCrawler in der AWS SDK for SAP ABAP API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " iv_database_name = 'my-database'
      lo_glu->deletedatabase( iv_name = iv_database_name ).
      MESSAGE 'Database deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Database does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DeleteDatabase in der AWS SDK for SAP ABAP API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_job_name = 'my-etl-job'
    lo_glu->deletejob( iv_jobname = iv_job_name ).
    MESSAGE 'Job deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteJob in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_database_name = 'my-database'
        " iv_table_name = 'my-table'
        lo_glu->deletetable(
          iv_databasename = iv_database_name
          iv_name = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Table or database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteTable in der AWS SDK for SAP ABAP API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_crawler_name = 'my-crawler'
        oo_result = lo_glu->getcrawler( iv_name = iv_crawler_name ).
        DATA(lo_crawler) = oo_result->get_crawler( ).
        MESSAGE 'Crawler information retrieved.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
```

```
      MESSAGE 'Crawler does not exist.' TYPE 'I'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie GetCrawler in der AWS SDK for SAP ABAP API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " iv_database_name = 'my-database'
      oo_result = lo_glu->getdatabase( iv_name = iv_database_name ).
      DATA(lo_database) = oo_result->get_database( ).
      MESSAGE 'Database information retrieved.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Database does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
      DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie GetDatabase in der AWS SDK for SAP ABAP API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_job_name = 'my-etl-job'
    " iv_run_id = 'jr_abcd1234567890abcdef1234567890abcdef12345678'
    oo_result = lo_glu->getjobrun(
      iv_jobname = iv_job_name
      iv_runid = iv_run_id ).
    DATA(lo_job_run) = oo_result->get_jobrun( ).
    MESSAGE 'Job run information retrieved.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Job or job run does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetJobRun in der AWS SDK for SAP ABAP API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_job_name = 'my-etl-job'
    oo_result = lo_glu->getjobruns( iv_jobname = iv_job_name ).
    DATA(lt_job_runs) = oo_result->get_jobruns( ).
    MESSAGE 'Job runs retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_gluentitynotfoundex.
    MESSAGE 'Job does not exist.' TYPE 'E'.
  CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
    DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
    MESSAGE lv_invalid_error TYPE 'E'.
  CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
    DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
    MESSAGE lv_internal_error TYPE 'E'.
  CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
    DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
    MESSAGE lv_timeout_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetJobRuns in der AWS SDK for SAP ABAP API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_database_name = 'my-database'
        oo_result = lo_glu->gettables( iv_databasename = iv_database_name ).
        DATA(lt_tables) = oo_result->get_tablelist( ).
        MESSAGE 'Tables retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Database does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
      CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
        DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
        MESSAGE lv_internal_error TYPE 'E'.
      CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
        DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
        MESSAGE lv_timeout_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetTables in der AWS SDK for SAP ABAP API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_glu->listjobs( ).
        DATA(lt_job_names) = oo_result->get_jobnames( ).
        MESSAGE 'Job list retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'No jobs found.' TYPE 'I'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
```

```
      MESSAGE lv_invalid_error TYPE 'E'.
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListJobs in der AWS SDK for SAP ABAP API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " iv_crawler_name = 'my-crawler'
      lo_glu->startcrawler( iv_name = iv_crawler_name ).
      MESSAGE 'Crawler started successfully.' TYPE 'I'.
    CATCH /aws1/cx_glucrawlerrunningex.
      MESSAGE 'Crawler is already running.' TYPE 'I'.
    CATCH /aws1/cx_gluentitynotfoundex.
      MESSAGE 'Crawler does not exist.' TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie StartCrawler in der AWS SDK for SAP ABAP API-Referenz.

StartJobRun

Das folgende Codebeispiel zeigt die VerwendungStartJobRun.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'my-etl-job'
        " iv_input_database = 'my-database'
        " iv_input_table = 'my-table'
        " iv_output_bucket_url = 's3://example-output-bucket/'

        DATA lt_arguments TYPE /aws1/cl_glugenericmap_w=>tt_genericmap.
        lt_arguments = VALUE #(
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_database'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_database ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--input_table'
            value = NEW /aws1/cl_glugenericmap_w( iv_value = iv_input_table ) ) )
          ( VALUE /aws1/cl_glugenericmap_w=>ts_genericmap_maprow(
            key = '--output_bucket_url'
            value = NEW /aws1/cl_glugenericmap_w( iv_value =
iv_output_bucket_url ) ) ) ).

        DATA(oo_result) = lo_glu->startjobrun(
          iv_jobname = iv_job_name
          it_arguments = lt_arguments ).
        ov_job_run_id = oo_result->get_jobrunid( ).
        MESSAGE 'Job run started successfully.' TYPE 'I'.
      CATCH /aws1/cx_gluconcurrentrunsex00.
        MESSAGE 'Maximum concurrent runs exceeded.' TYPE 'E'.
      CATCH /aws1/cx_gluentitynotfoundex.
        MESSAGE 'Job does not exist.' TYPE 'E'.
      CATCH /aws1/cx_gluinvalidinputex INTO DATA(lo_invalid_ex).
        DATA(lv_invalid_error) = lo_invalid_ex->if_message~get_longtext( ).
        MESSAGE lv_invalid_error TYPE 'E'.
```

```
    CATCH /aws1/cx_gluinternalserviceex INTO DATA(lo_internal_ex).
      DATA(lv_internal_error) = lo_internal_ex->if_message~get_longtext( ).
      MESSAGE lv_internal_error TYPE 'E'.
    CATCH /aws1/cx_gluoperationtimeoutex INTO DATA(lo_timeout_ex).
      DATA(lv_timeout_error) = lo_timeout_ex->if_message~get_longtext( ).
      MESSAGE lv_timeout_error TYPE 'E'.
    CATCH /aws1/cx_gluresrcnumlmtexcdex INTO DATA(lo_limit_ex).
      DATA(lv_limit_error) = lo_limit_ex->if_message~get_longtext( ).
      MESSAGE lv_limit_error TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie StartJobRun in der AWS SDK for SAP ABAP API-Referenz.

HealthImaging Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. HealthImaging

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CopyImageSet`

Das folgende Codebeispiel zeigt die Verwendung`CopyImageSet`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_source_image_set_id = '12345678901234567890123456789012345678 90'
```

```
        " iv_source_version_id = '1'
        " iv_destination_image_set_id =
'12345678901234567890123456789012345678 90' (optional)
        " iv_destination_version_id = '1' (optional)
        " iv_force = abap_false
        DATA(lo_source_info) = NEW /aws1/cl_migcpsrcimagesetinf00(
          iv_latestversionid = iv_source_version_id ).
        DATA(lo_copy_info) = NEW /aws1/cl_migcpimagesetinfmtion(
          io_sourceimageset = lo_source_info ).
        IF iv_destination_image_set_id IS NOT INITIAL AND
           iv_destination_version_id IS NOT INITIAL.
          DATA(lo_dest_info) = NEW /aws1/cl_migcopydstimageset(
            iv_imagesetid = iv_destination_image_set_id
            iv_latestversionid = iv_destination_version_id ).
          lo_copy_info = NEW /aws1/cl_migcpimagesetinfmtion(
            io_sourceimageset = lo_source_info
            io_destinationimageset = lo_dest_info ).
        ENDIF.
        oo_result = lo_mig->copyimageset(
          iv_datastoreid = iv_datastore_id
          iv_sourceimagesetid = iv_source_image_set_id
          io_copyimagesetinformation = lo_copy_info
          iv_force = iv_force ).
        DATA(lo_dest_props) = oo_result->get_dstimagesetproperties( ).
        DATA(lv_new_id) = lo_dest_props->get_imagesetid( ).
        MESSAGE |Image set copied with new ID: { lv_new_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie CopyImageSet in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`CreateDatastore`

Das folgende Codebeispiel zeigt die Verwendung`CreateDatastore`.

SDK für SAP ABAP

```
TRY.
    " iv_datastore_name = 'my-datastore-name'
    oo_result = lo_mig->createdatastore( iv_datastorename = iv_datastore_name ).
    DATA(lv_datastore_id) = oo_result->get_datastoreid( ).
    MESSAGE 'Data store created.' TYPE 'I'.
  CATCH /aws1/cx_migaccessdeniedex.
    MESSAGE 'Access denied.' TYPE 'I'.
  CATCH /aws1/cx_migconflictexception.
    MESSAGE 'Conflict. Data store may already exist.' TYPE 'I'.
  CATCH /aws1/cx_miginternalserverex.
    MESSAGE 'Internal server error.' TYPE 'I'.
  CATCH /aws1/cx_migservicequotaexcdex.
    MESSAGE 'Service quota exceeded.' TYPE 'I'.
  CATCH /aws1/cx_migthrottlingex.
    MESSAGE 'Request throttled.' TYPE 'I'.
  CATCH /aws1/cx_migvalidationex.
    MESSAGE 'Validation error.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie CreateDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteDatastore`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatastore`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->deletedatastore( iv_datastoreid = iv_datastore_id ).
        MESSAGE 'Data store deleted.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict. Data store may contain resources.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Data store not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`DeleteImageSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteImageSet`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
```

```
      oo_result = lo_mig->deleteimageset(
        iv_datastoreid = iv_datastore_id
        iv_imagesetid = iv_image_set_id ).
      MESSAGE 'Image set deleted.' TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Image set not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie DeleteImageSet in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`GetDICOMImportJob`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_job_id = '12345678901234567890123456789012'
      oo_result = lo_mig->getdicomimportjob(
        iv_datastoreid = iv_datastore_id
        iv_jobid = iv_job_id ).
      DATA(lo_job_props) = oo_result->get_jobproperties( ).
      DATA(lv_job_status) = lo_job_props->get_jobstatus( ).
      MESSAGE |Job status: { lv_job_status }.| TYPE 'I'.
```

```
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Job not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie GetDICOMImportJob in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDatastore`

Das folgende Codebeispiel zeigt die Verwendung`GetDatastore`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      oo_result = lo_mig->getdatastore( iv_datastoreid = iv_datastore_id ).
      DATA(lo_properties) = oo_result->get_datastoreproperties( ).
      DATA(lv_name) = lo_properties->get_datastorename( ).
      DATA(lv_status) = lo_properties->get_datastorestatus( ).
      MESSAGE 'Data store properties retrieved.' TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
```

```
      MESSAGE 'Data store not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie GetDatastore in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt die Verwendung`GetImageFrame`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_image_set_id = '12345678901234567890123456789012345678 90'
      " iv_image_frame_id = '12345678901234567890123456789012345678 90'
      oo_result = lo_mig->getimageframe(
        iv_datastoreid = iv_datastore_id
        iv_imagesetid = iv_image_set_id
        io_imageframeinformation = NEW /aws1/cl_migimageframeinfmtion(
          iv_imageframeid = iv_image_frame_id ) ).
      DATA(lv_frame_blob) = oo_result->get_imageframeblob( ).
      MESSAGE 'Image frame retrieved.' TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Image frame not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
```

```
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie GetImageFrame in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSet`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSet`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_image_set_id = '12345678901234567890123456789012345678 90'
      " iv_version_id = '1' (optional)
      IF iv_version_id IS NOT INITIAL.
        oo_result = lo_mig->getimageset(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_versionid = iv_version_id ).
      ELSE.
        oo_result = lo_mig->getimageset(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
      ENDIF.
      DATA(lv_state) = oo_result->get_imagesetstate( ).
      MESSAGE |Image set retrieved with state: { lv_state }.| TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Image set not found.' TYPE 'I'.
```

```
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie GetImageSet in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`GetImageSetMetadata`.

SDK für SAP ABAP

```
  TRY.
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_image_set_id = '12345678901234567890123456789012345678 90'
      " iv_version_id = '1' (optional)
      IF iv_version_id IS NOT INITIAL.
        oo_result = lo_mig->getimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_versionid = iv_version_id ).
      ELSE.
        oo_result = lo_mig->getimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
      ENDIF.
      DATA(lv_metadata_blob) = oo_result->get_imagesetmetadatablob( ).
      MESSAGE 'Image set metadata retrieved.' TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie GetImageSetMetadata in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDICOMImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListDICOMImportJobs`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->listdicomimportjobs( iv_datastoreid = iv_datastore_id ).
        DATA(lt_jobs) = oo_result->get_jobsummaries( ).
        DATA(lv_count) = lines( lt_jobs ).
        MESSAGE |Found { lv_count } DICOM import jobs.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
```

```
    ENDTRY.
```

- API-Details finden Sie ListDICOMImportJobs in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListDatastores`.

SDK für SAP ABAP

```
    TRY.
        oo_result = lo_mig->listdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastoresummaries( ).
        DATA(lv_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_count } data stores.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListDatastores in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListImageSetVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListImageSetVersions`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->listimagesetversions(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id ).
        DATA(lt_versions) = oo_result->get_imagesetpropertieslist( ).
        DATA(lv_count) = lines( lt_versions ).
        MESSAGE |Found { lv_count } image set versions.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListImageSetVersions in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        oo_result = lo_mig->listtagsforresource( iv_resourcearn = iv_resource_arn ).
        DATA(lt_tags) = oo_result->get_tags( ).
        DATA(lv_count) = lines( lt_tags ).
        MESSAGE |Found { lv_count } tags for resource.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListTagsForResource in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt die Verwendung`SearchImageSets`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        oo_result = lo_mig->searchimagesets(
          iv_datastoreid = iv_datastore_id
          io_searchcriteria = io_search_criteria ).
```

```
      DATA(lt_imagesets) = oo_result->get_imagesetsmetadatasums( ).
      DATA(lv_count) = lines( lt_imagesets ).
      MESSAGE |Found { lv_count } image sets.| TYPE 'I'.
    CATCH /aws1/cx_migaccessdeniedex.
      MESSAGE 'Access denied.' TYPE 'I'.
    CATCH /aws1/cx_migconflictexception.
      MESSAGE 'Conflict error.' TYPE 'I'.
    CATCH /aws1/cx_miginternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
    CATCH /aws1/cx_migresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'I'.
    CATCH /aws1/cx_migthrottlingex.
      MESSAGE 'Request throttled.' TYPE 'I'.
    CATCH /aws1/cx_migvalidationex.
      MESSAGE 'Validation error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie SearchImageSets in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartDICOMImportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartDICOMImportJob`.

SDK für SAP ABAP

```
  TRY.
      " iv_job_name = 'import-job-1'
      " iv_datastore_id = '12345678901234567890123456789012345678 90'
      " iv_role_arn = 'arn:aws:iam::123456789012:role/ImportJobRole'
      " iv_input_s3_uri = 's3://my-bucket/input/'
      " iv_output_s3_uri = 's3://my-bucket/output/'
      oo_result = lo_mig->startdicomimportjob(
        iv_jobname = iv_job_name
        iv_datastoreid = iv_datastore_id
        iv_dataaccessrolearn = iv_role_arn
```

```
          iv_inputs3uri = iv_input_s3_uri
          iv_outputs3uri = iv_output_s3_uri ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |DICOM import job started with ID: { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie StartDICOMImportJob in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags ).
```

```
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie TagResource in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für SAP ABAP

```
    TRY.
        " iv_resource_arn = 'arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012'
        lo_mig->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Resource not found.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
```

```
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie UntagResource in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`UpdateImageSetMetadata`

Das folgende Codebeispiel zeigt die Verwendung`UpdateImageSetMetadata`.

SDK für SAP ABAP

```
    TRY.
        " iv_datastore_id = '12345678901234567890123456789012345678 90'
        " iv_image_set_id = '12345678901234567890123456789012345678 90'
        " iv_latest_version_id = '1'
        " iv_force = abap_false
        oo_result = lo_mig->updateimagesetmetadata(
          iv_datastoreid = iv_datastore_id
          iv_imagesetid = iv_image_set_id
          iv_latestversionid = iv_latest_version_id
          io_updateimagesetmetupdates = io_metadata_updates
          iv_force = iv_force ).
        DATA(lv_new_version) = oo_result->get_latestversionid( ).
        MESSAGE |Image set metadata updated to version: { lv_new_version }.| TYPE
'I'.
      CATCH /aws1/cx_migaccessdeniedex.
        MESSAGE 'Access denied.' TYPE 'I'.
      CATCH /aws1/cx_migconflictexception.
        MESSAGE 'Conflict error.' TYPE 'I'.
      CATCH /aws1/cx_miginternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
      CATCH /aws1/cx_migresourcenotfoundex.
        MESSAGE 'Image set not found.' TYPE 'I'.
```

```
      CATCH /aws1/cx_migservicequotaexcdex.
        MESSAGE 'Service quota exceeded.' TYPE 'I'.
      CATCH /aws1/cx_migthrottlingex.
        MESSAGE 'Request throttled.' TYPE 'I'.
      CATCH /aws1/cx_migvalidationex.
        MESSAGE 'Validation error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie UpdateImageSetMetadata in der API-Referenz zum AWS SDK für SAP ABAP.

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

HealthLake Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. HealthLake

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`Erstellen Sie einen FHIR-Datenspeicher`

Das folgende Codebeispiel zeigt die Verwendung. `CreateFHIRDatastore`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_name = 'MyHealthLakeDataStore'
        oo_result = lo_hll->createfhirdatastore(
          iv_datastorename = iv_datastore_name
          iv_datastoretypeversion = 'R4'
        ).
        MESSAGE 'Data store created successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllinternalserverex INTO DATA(lo_internal_ex).
        lv_error = |Internal server error: { lo_internal_ex->av_err_code }-
{ lo_internal_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- API-Details finden Sie unter createFhirDataStore in der AWS SDK for SAP ABAP API-Referenz.

`Löschen Sie FhirDataStore`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteFHIRDatastore`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->deletefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        MESSAGE 'Data store deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        DATA(lv_error) = |Access denied: { lo_access_ex->av_err_code }-
{ lo_access_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
      CATCH /aws1/cx_hllconflictexception INTO DATA(lo_conflict_ex).
        lv_error = |Conflict error: { lo_conflict_ex->av_err_code }-
{ lo_conflict_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie unter DeleteFhirDataStore in der AWS SDK for SAP ABAP API-Referenz.

`DescribeFHIRDatastore`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeFHIRDatastore`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirdatastore(
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lo_datastore_properties) = oo_result->get_datastoreproperties( ).
        IF lo_datastore_properties IS BOUND.
          DATA(lv_datastore_name) = lo_datastore_properties->get_datastorename( ).
          DATA(lv_datastore_status) = lo_datastore_properties-
>get_datastorestatus( ).
          MESSAGE 'Data store described successfully.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API-Details finden Sie unter DescribeFhirDataStore in der AWS SDK for SAP ABAP API-Referenz.

`DescribeFHIRExportJob`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeFHIRExportJob`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirexportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_export_job_properties) = oo_result->get_exportjobproperties( ).
        IF lo_export_job_properties IS BOUND.
          DATA(lv_job_status) = lo_export_job_properties->get_jobstatus( ).
          MESSAGE |Export job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API-Details finden Sie DescribeFHIRExportJob in der AWS SDK for SAP ABAP API-Referenz.

`DescribeFHIRImportJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeFHIRImportJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        " iv_job_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        oo_result = lo_hll->describefhirimportjob(
          iv_datastoreid = iv_datastore_id
          iv_jobid = iv_job_id
        ).
        DATA(lo_import_job_properties) = oo_result->get_importjobproperties( ).
        IF lo_import_job_properties IS BOUND.
          DATA(lv_job_status) = lo_import_job_properties->get_jobstatus( ).
          MESSAGE |Import job status: { lv_job_status }.| TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        DATA(lv_error) = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        lv_error = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
    ENDTRY.
```

- API-Details finden Sie DescribeFHIRImportJob in der AWS SDK for SAP ABAP API-Referenz.

`ListFHIRDatastores`

Das folgende Codebeispiel zeigt die Verwendung`ListFHIRDatastores`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_hll->listfhirdatastores( ).
        DATA(lt_datastores) = oo_result->get_datastorepropertieslist( ).
        DATA(lv_datastore_count) = lines( lt_datastores ).
        MESSAGE |Found { lv_datastore_count } data store(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
    ENDTRY.
```

- API-Details finden Sie unter ListFhirDataStores in der AWS SDK for SAP ABAP API-Referenz.

`ListFHIRExportJobs`

Das folgende Codebeispiel zeigt die Verwendung. `ListFHIRExportJobs`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirexportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_export_jobs) = oo_result->get_exportjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_export_jobs ).
        MESSAGE |Found { lv_job_count } export job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie ListFHIRExportJobs in der AWS SDK for SAP ABAP API-Referenz.

`ListFHIRImportJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListFHIRImportJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_datastore_id = 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        IF iv_submitted_after IS NOT INITIAL.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
            iv_submittedafter = iv_submitted_after
          ).
        ELSE.
          oo_result = lo_hll->listfhirimportjobs(
            iv_datastoreid = iv_datastore_id
          ).
        ENDIF.
        DATA(lt_import_jobs) = oo_result->get_importjobpropertieslist( ).
        DATA(lv_job_count) = lines( lt_import_jobs ).
        MESSAGE |Found { lv_job_count } import job(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie ListFHIRImportJobs in der AWS SDK for SAP ABAP API-Referenz.

`ListTagsForResource`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForResource`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        DATA(lo_result) = lo_hll->listtagsforresource(
          iv_resourcearn = iv_resource_arn
        ).
        ot_tags = lo_result->get_tags( ).
        DATA(lv_tag_count) = lines( ot_tags ).
        MESSAGE |Found { lv_tag_count } tag(s).| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie ListTagsForResource in der AWS SDK for SAP ABAP API-Referenz.

`StartFHIRExportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartFHIRExportJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'MyExportJob'
        " iv_output_s3_uri = 's3://my-bucket/export/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
```

```
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeExportRole'
        oo_result = lo_hll->startfhirexportjob(
          iv_jobname = iv_job_name
          io_outputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Export job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API-Details finden Sie StartFHIRExportJob in der AWS SDK for SAP ABAP API-Referenz.

`StartFHIRImportJob`

Das folgende Codebeispiel zeigt die Verwendung`StartFHIRImportJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_job_name = 'MyImportJob'
        " iv_input_s3_uri = 's3://my-bucket/import/data.ndjson'
        " iv_job_output_s3_uri = 's3://my-bucket/import/output/'
        " iv_kms_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012'
        " iv_data_access_role_arn = 'arn:aws:iam::123456789012:role/
HealthLakeImportRole'
        oo_result = lo_hll->startfhirimportjob(
          iv_jobname = iv_job_name
          io_inputdataconfig = NEW /aws1/cl_hllinputdataconfig( iv_s3uri =
 iv_input_s3_uri )
          io_joboutputdataconfig = NEW /aws1/cl_hlloutputdataconfig(
            io_s3configuration = NEW /aws1/cl_hlls3configuration(
              iv_s3uri = iv_job_output_s3_uri
              iv_kmskeyid = iv_kms_key_id
            )
          )
          iv_dataaccessrolearn = iv_data_access_role_arn
          iv_datastoreid = iv_datastore_id
        ).
        DATA(lv_job_id) = oo_result->get_jobid( ).
        MESSAGE |Import job started with ID { lv_job_id }.| TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllthrottlingex INTO DATA(lo_throttling_ex).
        lv_error = |Throttling error: { lo_throttling_ex->av_err_code }-
{ lo_throttling_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_throttling_ex.
      CATCH /aws1/cx_hllaccessdeniedex INTO DATA(lo_access_ex).
```

```
        lv_error = |Access denied: { lo_access_ex->av_err_code }-{ lo_access_ex-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_access_ex.
    ENDTRY.
```

- API-Details finden Sie StartFHIRImportJob in der AWS SDK for SAP ABAP API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->tagresource(
          iv_resourcearn = iv_resource_arn
          it_tags = it_tags
        ).
        MESSAGE 'Resource tagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie TagResource in der AWS SDK for SAP ABAP API-Referenz.

`UntagResource`

Das folgende Codebeispiel zeigt die Verwendung`UntagResource`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_resource_arn = 'arn:aws:healthlake:us-east-1:123456789012:datastore/
fhir/a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'
        lo_hll->untagresource(
          iv_resourcearn = iv_resource_arn
          it_tagkeys = it_tag_keys
        ).
        MESSAGE 'Resource untagged successfully.' TYPE 'I'.
      CATCH /aws1/cx_hllvalidationex INTO DATA(lo_validation_ex).
        DATA(lv_error) = |Validation error: { lo_validation_ex->av_err_code }-
{ lo_validation_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_validation_ex.
      CATCH /aws1/cx_hllresourcenotfoundex INTO DATA(lo_notfound_ex).
        lv_error = |Resource not found: { lo_notfound_ex->av_err_code }-
{ lo_notfound_ex->av_err_msg }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_notfound_ex.
    ENDTRY.
```

- API-Details finden Sie UntagResource in der AWS SDK for SAP ABAP API-Referenz.

IAM-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit IAM Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung. `AttachRolePolicy`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->attachrolepolicy(
      iv_rolename = iv_role_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy attached to role successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role or policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK for SAP ABAP API-Referenz.

`AttachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachUserPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->attachuserpolicy(
      iv_username = iv_user_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy attached to user successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User or policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie AttachUserPolicy in der AWS SDK for SAP ABAP API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createaccesskey(
          iv_username = iv_user_name ).
        MESSAGE 'Access key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Maximum number of access keys reached.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateAccessKey in der AWS SDK for SAP ABAP API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->createaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Account alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Account alias limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateAccountAlias in der AWS SDK for SAP ABAP API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->createpolicy(
      iv_policyname = iv_policy_name
      iv_policydocument = iv_policy_document
      iv_description = iv_description ).
    MESSAGE 'Policy created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'Policy already exists.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.
    MESSAGE 'Policy document is malformed.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy limit exceeded.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreatePolicy in der AWS SDK for SAP ABAP API-Referenz.

`CreatePolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicyVersion`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_policydocument = iv_policy_document
          iv_setasdefault = iv_set_as_default ).
        MESSAGE 'Policy version created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Policy version limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreatePolicyVersion in der AWS SDK for SAP ABAP API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->createrole(
          iv_rolename = iv_role_name
          iv_assumerolepolicydocument = iv_assume_role_policy_document ).
        MESSAGE 'Role created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Role already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Assume role policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Role limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateRole in der AWS SDK for SAP ABAP API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listpolicyversions(
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK for SAP ABAP API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->createuser(
      iv_username = iv_user_name ).
    MESSAGE 'User created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'User already exists.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Limit exceeded for IAM users.' TYPE 'E'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Entity does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateUser in der AWS SDK for SAP ABAP API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->deleteaccesskey(
      iv_accesskeyid = iv_access_key_id
      iv_username = iv_user_name ).
    MESSAGE 'Access key deleted successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Access key or user does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK for SAP ABAP API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_iam->deleteaccountalias(
        iv_accountalias = iv_account_alias ).
      MESSAGE 'Account alias deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Account alias does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DeleteAccountAlias in der AWS SDK for SAP ABAP API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deletepolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Policy cannot be deleted due to attachments.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeletePolicy in der AWS SDK for SAP ABAP API-Referenz.

`DeletePolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicyVersion`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_iam->deletepolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Policy version deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Cannot delete default policy version.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeletePolicyVersion in der AWS SDK for SAP ABAP API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->deleterole( iv_rolename = iv_role_name ).
    MESSAGE 'Role deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamdeleteconflictex.
    MESSAGE 'Role cannot be deleted due to attached resources.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteRole in der AWS SDK for SAP ABAP API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->deleteuser( iv_username = iv_user_name ).
    MESSAGE 'User deleted successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'User does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iamdeleteconflictex.
      MESSAGE 'User cannot be deleted due to attached resources.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DeleteUser in der AWS SDK for SAP ABAP API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_iam->detachrolepolicy(
        iv_rolename = iv_role_name
        iv_policyarn = iv_policy_arn ).
      MESSAGE 'Policy detached from role successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Role or policy does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK for SAP ABAP API-Referenz.

`DetachUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachUserPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->detachuserpolicy(
      iv_username = iv_user_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy detached from user successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User or policy does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetachUserPolicy in der AWS SDK for SAP ABAP API-Referenz.

`GenerateCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung`GenerateCredentialReport`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->generatecredentialreport( ).
    MESSAGE 'Credential report generation started.' TYPE 'I'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Report generation limit exceeded.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when generating credential report.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GenerateCredentialReport in der AWS SDK for SAP ABAP API-Referenz.

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung`GetAccessKeyLastUsed`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->getaccesskeylastused(
      iv_accesskeyid = iv_access_key_id ).
    MESSAGE 'Retrieved access key last used information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetAccessKeyLastUsed in der AWS SDK for SAP ABAP API-Referenz.

`GetAccountAuthorizationDetails`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountAuthorizationDetails`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getaccountauthdetails( ).
        MESSAGE 'Retrieved account authorization details.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting account authorization details.' TYPE
'E'.
    ENDTRY.
```

- API-Details finden Sie GetAccountAuthorizationDetails in der AWS SDK for SAP ABAP API-Referenz.

`GetAccountPasswordPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountPasswordPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getaccountpasswordpolicy( ).
        MESSAGE 'Retrieved account password policy.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'No password policy exists.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting password policy.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetAccountPasswordPolicy in der AWS SDK for SAP ABAP API-Referenz.

`GetAccountSummary`

Das folgende Codebeispiel zeigt die Verwendung`GetAccountSummary`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->getaccountsummary( ).
    MESSAGE 'Retrieved account summary.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting account summary.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetAccountSummary in der AWS SDK for SAP ABAP API-Referenz.

`GetCredentialReport`

Das folgende Codebeispiel zeigt die Verwendung`GetCredentialReport`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->getcredentialreport( ).
    MESSAGE 'Retrieved credential report.' TYPE 'I'.
  CATCH /aws1/cx_iamcredrptnotpresen00.
    MESSAGE 'Credential report not present.' TYPE 'E'.
  CATCH /aws1/cx_iamcredrptexpiredex.
    MESSAGE 'Credential report expired.' TYPE 'E'.
  CATCH /aws1/cx_iamcredrptnotreadyex.
    MESSAGE 'Credential report not ready.' TYPE 'E'.
  CATCH /aws1/cx_iamservicefailureex.
```

```
        MESSAGE 'Service failure when getting credential report.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetCredentialReport in der AWS SDK for SAP ABAP API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getpolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetPolicy in der AWS SDK for SAP ABAP API-Referenz.

`GetPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicyVersion`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Retrieved policy version information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'Invalid input provided.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetPolicyVersion in der AWS SDK for SAP ABAP API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->getrole( iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved role information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetRole in der AWS SDK for SAP ABAP API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listaccesskeys(
      iv_username = iv_user_name ).
    MESSAGE 'Retrieved access key list.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListAccessKeys in der AWS SDK for SAP ABAP API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listaccountaliases( ).
    MESSAGE 'Retrieved account alias list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing account aliases.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListAccountAliases in der AWS SDK for SAP ABAP API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listattachedrolepolicies(
      iv_rolename = iv_role_name ).
    MESSAGE 'Retrieved attached policy list for role.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK for SAP ABAP API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listgroups( ).
    MESSAGE 'Retrieved group list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
```

```
        MESSAGE 'Service failure when listing groups.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListGroups in der AWS SDK for SAP ABAP API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listpolicies( iv_scope = iv_scope ).
        MESSAGE 'Retrieved policy list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policies.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListPolicies in der AWS SDK for SAP ABAP API-Referenz.

`ListPolicyVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicyVersions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListPolicyVersions in der AWS SDK for SAP ABAP API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_iam->listrolepolicies(
          iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved inline policy list for role.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListRolePolicies in der AWS SDK for SAP ABAP API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listroles( ).
    MESSAGE 'Retrieved role list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing roles.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListRoles in der AWS SDK for SAP ABAP API-Referenz.

`Listet SAML-Anbieter auf`

Das folgende Codebeispiel zeigt die Verwendung. `ListSAMLProviders`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listsamlproviders( ).
    MESSAGE 'Retrieved SAML provider list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing SAML providers.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie unter ListSamlProviders in der AWS SDK for SAP ABAP API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung. `ListUsers`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_iam->listusers( ).
    MESSAGE 'Retrieved user list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing users.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListUsers in der AWS SDK for SAP ABAP API-Referenz.

`SetDefaultPolicyVersion`

Das folgende Codebeispiel zeigt die Verwendung`SetDefaultPolicyVersion`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->setdefaultpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_versionid = iv_version_id ).
    MESSAGE 'Default policy version set successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Policy or version does not exist.' TYPE 'E'.
    CATCH /aws1/cx_iaminvalidinputex.
      MESSAGE 'Invalid input provided.' TYPE 'E'.
    CATCH /aws1/cx_iamlimitexceededex.
      MESSAGE 'Limit exceeded.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie SetDefaultPolicyVersion in der AWS SDK for SAP ABAP API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_iam->updateaccesskey(
        iv_accesskeyid = iv_access_key_id
        iv_status = iv_status
        iv_username = iv_user_name ).
      MESSAGE 'Access key updated successfully.' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
      MESSAGE 'Access key or user does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie UpdateAccessKey in der AWS SDK for SAP ABAP API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_iam->updateuser(
      iv_username = iv_user_name
      iv_newusername = iv_new_user_name ).
    MESSAGE 'User updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'New user name already exists.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie UpdateUser in der AWS SDK for SAP ABAP API-Referenz.

AWS IoT Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP with AWS IoT Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        lo_iot->attachthingprincipal(
          iv_thingname = iv_thing_name
          iv_principal = iv_principal ).
        MESSAGE |Principal attached to IoT thing '{ iv_thing_name }'.| TYPE 'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
        MESSAGE |Resource not found when attaching principal to
'{ iv_thing_name }'.| TYPE 'I'.
        RAISE EXCEPTION lo_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
        MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_svc_ex.
    ENDTRY.
```

- API-Details finden Sie AttachThingPrincipal in der AWS SDK for SAP ABAP API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    oo_result = lo_iot->createkeysandcertificate( iv_setasactive = abap_true ).
    MESSAGE |Certificate created: { oo_result->get_certificateid( ) }| TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- API-Details finden Sie CreateKeysAndCertificate in der AWS SDK for SAP ABAP API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
```

```
        oo_result = lo_iot->creatething(
          iv_thingname = iv_thing_name ).
        MESSAGE |IoT thing created: { oo_result->get_thingname( ) } ARN:
{ oo_result->get_thingarn( ) }| TYPE 'I'.
      CATCH /aws1/cx_iotresrcalrdyexistsex.
        MESSAGE |IoT thing '{ iv_thing_name }' already exists.| TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie CreateThing in der AWS SDK for SAP ABAP API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        " Build the SNS action that will receive messages matching the rule.
        DATA lo_sns_action TYPE REF TO /aws1/cl_iotsnsaction.
        CREATE OBJECT lo_sns_action
          EXPORTING
            iv_targetarn = iv_sns_action_arn
            iv_rolearn   = iv_role_arn.

        DATA lo_action TYPE REF TO /aws1/cl_iotaction.
        CREATE OBJECT lo_action
          EXPORTING
            io_sns = lo_sns_action.
```

```
      DATA lt_actions TYPE /aws1/cl_iotaction=>tt_actionlist.
      APPEND lo_action TO lt_actions.

      " iv_topic = 'my/iot/topic' - The MQTT topic pattern to match
      DATA lo_payload TYPE REF TO /aws1/cl_iottopicrulepayload.
      CREATE OBJECT lo_payload
        EXPORTING
          iv_sql     = |SELECT * FROM '{ iv_topic }'|
          it_actions = lt_actions.

      lo_iot->createtopicrule(
        iv_rulename         = iv_rule_name
        io_topicrulepayload = lo_payload ).
      MESSAGE |IoT topic rule created: { iv_rule_name }| TYPE 'I'.
    CATCH /aws1/cx_iotresrcalrdyexistsex.
      MESSAGE |Topic rule '{ iv_rule_name }' already exists.| TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
      MESSAGE lo_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- API-Details finden Sie CreateTopicRule in der AWS SDK for SAP ABAP API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
  DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
  DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
  TRY.
```

```
      " Certificates must be deactivated before they can be deleted.
      lo_iot->updatecertificate(
        iv_certificateid = iv_certificate_id
        iv_newstatus     = 'INACTIVE' ).
      lo_iot->deletecertificate( iv_certificateid = iv_certificate_id ).
      MESSAGE |Certificate deleted: { iv_certificate_id }| TYPE 'I'.
    CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
      MESSAGE |Certificate '{ iv_certificate_id }' not found.| TYPE 'I'.
      RAISE EXCEPTION lo_ex.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
      MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_svc_ex.
  ENDTRY.
```

- API-Details finden Sie DeleteCertificate in der AWS SDK for SAP ABAP API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
  DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
  DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
  TRY.
      lo_iot->deletething( iv_thingname = iv_thing_name ).
      MESSAGE |IoT thing deleted: { iv_thing_name }| TYPE 'I'.
    CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
      MESSAGE |IoT thing '{ iv_thing_name }' not found.| TYPE 'I'.
      RAISE EXCEPTION lo_ex.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
      MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_svc_ex.
  ENDTRY.
```

- API-Details finden Sie DeleteThing in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopicRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
TRY.
    lo_iot->deletetopicrule( iv_rulename = iv_rule_name ).
    MESSAGE |IoT topic rule deleted: { iv_rule_name }| TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- API-Details finden Sie DeleteTopicRule in der AWS SDK for SAP ABAP API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    " iv_endpoint_type = 'iot:Data-ATS' - Endpoint type for data operations
    TRY.
        DATA(lo_result) = lo_iot->describeendpoint( iv_endpointtype =
iv_endpoint_type ).
        ov_endpoint_address = lo_result->get_endpointaddress( ).
        MESSAGE |Endpoint address: { ov_endpoint_address }| TYPE 'I'.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
        MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie DescribeEndpoint in der AWS SDK for SAP ABAP API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        lo_iot->detachthingprincipal(
          iv_thingname = iv_thing_name
          iv_principal = iv_principal ).
        MESSAGE |Principal detached from IoT thing '{ iv_thing_name }'.| TYPE 'I'.
      CATCH /aws1/cx_iotresourcenotfoundex INTO DATA(lo_ex).
```

```
        MESSAGE |Resource not found when detaching principal from
'{ iv_thing_name }'.| TYPE 'I'.
        RAISE EXCEPTION lo_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_svc_ex).
        MESSAGE lo_svc_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_svc_ex.
    ENDTRY.
```

- API-Details finden Sie DetachThingPrincipal in der AWS SDK for SAP ABAP API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    TRY.
        " Collect all certificates by following the pagination marker.
        DATA lv_marker TYPE /aws1/iotmarker.
        DATA lv_count  TYPE i.

        DO.
          oo_result = lo_iot->listcertificates( iv_marker = lv_marker ).
          lv_count  = lv_count + lines( oo_result->get_certificates( ) ).
          lv_marker = oo_result->get_nextmarker( ).
          IF lv_marker IS INITIAL.
            EXIT.
          ENDIF.
        ENDDO.

        MESSAGE |Retrieved { lv_count } IoT certificates.| TYPE 'I'.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
```

```
      MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
      RAISE EXCEPTION lo_throttle_ex.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
      MESSAGE lo_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- API-Details finden Sie ListCertificates in der AWS SDK for SAP ABAP API-Referenz.

`ListThings`

Das folgende Codebeispiel zeigt die Verwendung`ListThings`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
  DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
  DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
  TRY.
      " Collect all things by following the pagination token.
      DATA lv_nexttoken TYPE /aws1/iotnexttoken.
      DATA lv_count     TYPE i.

      DO.
        oo_result    = lo_iot->listthings( iv_nexttoken = lv_nexttoken ).
        lv_count     = lv_count + lines( oo_result->get_things( ) ).
        lv_nexttoken = oo_result->get_nexttoken( ).
        IF lv_nexttoken IS INITIAL.
          EXIT.
        ENDIF.
      ENDDO.

      MESSAGE |Retrieved { lv_count } IoT things.| TYPE 'I'.
    CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
      MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
```

```
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie ListThings in der AWS SDK for SAP ABAP API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
    " iv_query_string = 'thingName:MyThing*' - Fleet indexing query string
    TRY.
        oo_result = lo_iot->searchindex( iv_querystring = iv_query_string ).
        MESSAGE |Found { lines( oo_result->get_things( ) ) } IoT things matching
query.| TYPE 'I'.
      CATCH /aws1/cx_iotindexnotreadyex INTO DATA(lo_idx_ex).
        MESSAGE 'Fleet indexing is not ready. Enable indexing with
UpdateIndexingConfiguration first.' TYPE 'I'.
        RAISE EXCEPTION lo_idx_ex.
      CATCH /aws1/cx_iotthrottlingex INTO DATA(lo_throttle_ex).
        MESSAGE 'Request throttled. Please try again later.' TYPE 'I'.
        RAISE EXCEPTION lo_throttle_ex.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
        MESSAGE lo_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie SearchIndex in der AWS SDK for SAP ABAP API-Referenz.

`UpdateIndexingConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateIndexingConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   CONSTANTS cv_pfl TYPE /aws1/rt_profile_id VALUE 'ZCODE_DEMO'.
   DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
   DATA(lo_iot) = /aws1/cl_iot_factory=>create( lo_session ).
   TRY.
       DATA lo_idx_conf TYPE REF TO /aws1/cl_iotthingindexingconf.
       CREATE OBJECT lo_idx_conf
         EXPORTING
           iv_thingindexingmode = 'REGISTRY'.
       lo_iot->updateindexingconfiguration(
         io_thingindexingconf = lo_idx_conf ).
       MESSAGE 'IoT thing indexing configuration updated to REGISTRY mode.' TYPE
'I'.
     CATCH /aws1/cx_rt_service_generic INTO DATA(lo_ex).
       MESSAGE lo_ex->get_text( ) TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- API-Details finden Sie UpdateIndexingConfiguration in der AWS SDK for SAP ABAP API-Referenz.

AWS IoT data Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS IoT data

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`GetThingShadow`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_iop->getthingshadow( iv_thingname = iv_thing_name ).

    " Convert xstring payload to JSON string
    DATA(lv_payload) = lo_result->get_payload( ).
    ov_shadow = /aws1/cl_rt_util=>xstring_to_string( lv_payload ).
    MESSAGE 'Thing shadow retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_iopresourcenotfoundex.
    MESSAGE 'Thing shadow not found.' TYPE 'E'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
    DATA(lv_error) = |{ lo_exception->get_text( ) }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetThingShadow in der AWS SDK for SAP ABAP API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Convert JSON string to xstring for payload
    DATA(lv_payload) = /aws1/cl_rt_util=>string_to_xstring( iv_shadow_state ).

    lo_iop->updatethingshadow(
      iv_thingname = iv_thing_name
      iv_payload = lv_payload ).
    MESSAGE 'Thing shadow updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iopresourcenotfoundex.
    MESSAGE 'Thing not found.' TYPE 'E'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_exception).
    DATA(lv_error) = |{ lo_exception->get_text( ) }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie UpdateThingShadow in der AWS SDK for SAP ABAP API-Referenz.

AWS IoT SiteWise Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS IoT SiteWise

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo AWS IoT SiteWise

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT SiteWise beginnen.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_ios->listassetmodels( ). " oo_result is returned for testing
purposes. "
       DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
       MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list asset models.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie ListAssetModels in der AWS SDK for SAP ABAP API-Referenz.

Aktionen

`BatchPutAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchPutAssetPropertyValue`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ios->batchputassetpropertyvalue(
      it_entries = it_entries
    ).
    MESSAGE 'Data sent to IoT SiteWise asset successfully.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Asset does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie BatchPutAssetPropertyValue in der AWS SDK for SAP ABAP API-Referenz.

`CreateAsset`

Das folgende Codebeispiel zeigt die Verwendung`CreateAsset`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->createasset(
      iv_assetname = iv_asset_name
      iv_assetmodelid = iv_asset_model_id
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise asset created' TYPE 'I'.
```

```
    CATCH /aws1/cx_iosresourcenotfoundex.
      MESSAGE 'Asset model does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie CreateAsset in der AWS SDK for SAP ABAP API-Referenz.

`CreateAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`CreateAssetModel`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_ios->createassetmodel(
        iv_assetmodelname = iv_asset_model_name
        iv_assetmodeldescription = 'This is a sample asset model description.'
        it_assetmodelproperties = it_properties
      ). " oo_result is returned for testing purposes. "
      MESSAGE 'IoT SiteWise asset model created' TYPE 'I'.
    CATCH /aws1/cx_iosresrcalrdyexistsex.
      MESSAGE 'Asset model already exists.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie CreateAssetModel in der AWS SDK for SAP ABAP API-Referenz.

`CreateGateway`

Das folgende Codebeispiel zeigt die Verwendung`CreateGateway`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ios->creategateway(
      iv_gatewayname = iv_gateway_name
      io_gatewayplatform = NEW /aws1/cl_iosgatewayplatform(
        io_greengrassv2 = NEW /aws1/cl_iosgreengrassv2(
          iv_coredevicethingname = iv_core_device_thing_name
        )
      )
      it_tags = VALUE /aws1/cl_iostagmap_w=>tt_tagmap(
        (
          VALUE /aws1/cl_iostagmap_w=>ts_tagmap_maprow(
            key = 'Environment'
            value = NEW /aws1/cl_iostagmap_w( 'Production' )
          )
        )
      )
    ). " oo_result is returned for testing purposes. "
    MESSAGE 'IoT SiteWise gateway created' TYPE 'I'.
  CATCH /aws1/cx_iosresrcalrdyexistsex.
    MESSAGE 'Gateway already exists.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateGateway in der AWS SDK for SAP ABAP API-Referenz.

`DeleteAsset`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAsset`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ios->deleteasset(
      iv_assetid = iv_asset_id
    ).
    MESSAGE 'IoT SiteWise asset deleted.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to delete asset.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteAsset in der AWS SDK for SAP ABAP API-Referenz.

`DeleteAssetModel`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAssetModel`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ios->deleteassetmodel(
      iv_assetmodelid = iv_asset_model_id
    ).
    MESSAGE 'IoT SiteWise asset model deleted.' TYPE 'I'.
  CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to delete asset model.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteAssetModel in der AWS SDK for SAP ABAP API-Referenz.

`DeleteGateway`

Das folgende Codebeispiel zeigt die Verwendung`DeleteGateway`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ios->deletegateway(
      iv_gatewayid = iv_gateway_id
    ).
    MESSAGE 'IoT SiteWise gateway deleted.' TYPE 'I'.
  CATCH /aws1/cx_iosresourcenotfoundex.
    MESSAGE 'Gateway does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteGateway in der AWS SDK for SAP ABAP API-Referenz.

`DescribeGateway`

Das folgende Codebeispiel zeigt die Verwendung`DescribeGateway`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->describegateway(
          iv_gatewayid = iv_gateway_id
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'Retrieved gateway description.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Gateway does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeGateway in der AWS SDK for SAP ABAP API-Referenz.

`GetAssetPropertyValue`

Das folgende Codebeispiel zeigt die Verwendung`GetAssetPropertyValue`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->getassetpropertyvalue(
          iv_assetid = iv_asset_id
          iv_propertyid = iv_property_id
        ). " oo_result is returned for testing purposes. "
        MESSAGE 'Retrieved asset property value.' TYPE 'I'.
      CATCH /aws1/cx_iosresourcenotfoundex.
        MESSAGE 'Asset or property does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetAssetPropertyValue in der AWS SDK for SAP ABAP API-Referenz.

`ListAssetModelProperties`

Das folgende Codebeispiel zeigt die Verwendung`ListAssetModelProperties`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->listassetmodelproperties(
          iv_assetmodelid = iv_asset_model_id
        ). " oo_result is returned for testing purposes. "
        DATA(lt_properties) = oo_result->get_assetmodelpropertysums( ).
        MESSAGE 'Retrieved list of asset model properties.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list asset model properties.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListAssetModelProperties in der AWS SDK for SAP ABAP API-Referenz.

`ListAssetModels`

Das folgende Codebeispiel zeigt die Verwendung`ListAssetModels`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_ios->listassetmodels( ). " oo_result is returned for testing
purposes. "
        DATA(lt_asset_models) = oo_result->get_assetmodelsummaries( ).
        MESSAGE 'Retrieved list of asset models.' TYPE 'I'.
```

```
    CATCH /aws1/cx_rt_generic.
      MESSAGE 'Unable to list asset models.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListAssetModels in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Keyspaces, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Keyspaces Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateKeyspace`

Das folgende Codebeispiel zeigt die Verwendung. `CreateKeyspace`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_kys->createkeyspace(
```

```
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysconflictexception.
        MESSAGE 'Keyspace already exists.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateKeyspace in der AWS SDK for SAP ABAP API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Define schema with columns
        DATA(lt_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'title' iv_type = 'text' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'year' iv_type = 'int' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'release_date' iv_type =
 'timestamp' ) )
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'plot' iv_type = 'text' ) )
        ).

        " Define partition keys
        DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'year' ) )
          ( NEW /aws1/cl_kyspartitionkey( iv_name = 'title' ) )
```

```
        ).

        " Create schema definition
        DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
          it_allcolumns = lt_columns
          it_partitionkeys = lt_partition_keys ).

        " Enable point-in-time recovery
        DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
          iv_status = 'ENABLED' ).

        oo_result = lo_kys->createtable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          io_schemadefinition = lo_schema
          io_pointintimerecovery = lo_pitr ).
        MESSAGE 'Table created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateTable in der AWS SDK for SAP ABAP API-Referenz.

`DeleteKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyspace`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_kys->deletekeyspace(
          iv_keyspacename = iv_keyspace_name ).
```

```
        MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteKeyspace in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_kys->deletetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
        MESSAGE 'Table deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteTable in der AWS SDK for SAP ABAP API-Referenz.

`GetKeyspace`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyspace`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->getkeyspace(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Keyspace does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetKeyspace in der AWS SDK for SAP ABAP API-Referenz.

`GetTable`

Das folgende Codebeispiel zeigt die Verwendung`GetTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->gettable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name ).
```

```
        MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_kysresourcenotfoundex.
        MESSAGE 'Table does not exist.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetTable in der AWS SDK for SAP ABAP API-Referenz.

`ListKeyspaces`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyspaces`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->listkeyspaces(
          iv_maxresults = iv_max_results ).
        MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListKeyspaces in der AWS SDK for SAP ABAP API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->listtables(
          iv_keyspacename = iv_keyspace_name ).
        MESSAGE 'Tables listed successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListTables in der AWS SDK for SAP ABAP API-Referenz.

`RestoreTable`

Das folgende Codebeispiel zeigt die Verwendung`RestoreTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kys->restoretable(
          iv_sourcekeyspacename = iv_source_keyspace_name
          iv_sourcetablename = iv_source_table_name
          iv_targetkeyspacename = iv_target_keyspace_name
          iv_targettablename = iv_target_table_name
          iv_restoretimestamp = iv_restore_timestamp ).
```

```
        MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie RestoreTable in der AWS SDK for SAP ABAP API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTable`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Add a new column to track watched movies
        DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
          ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'watched' iv_type =
 'boolean' ) )
        ).

        oo_result = lo_kys->updatetable(
          iv_keyspacename = iv_keyspace_name
          iv_tablename = iv_table_name
          it_addcolumns = lt_add_columns ).
        MESSAGE 'Table updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie UpdateTable in der AWS SDK for SAP ABAP API-Referenz.

Kinesis-Beispiele unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Kinesis Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Stream und fügen Sie einen Datensatz ein.
- Erstellen Sie einen Shard-Iterator.
- Lesen Sie den Datensatz und bereinigen Sie anschließend die Ressourcen.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
   DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
   DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
   DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

   "Create stream."
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
   ENDTRY.

   "Wait for stream to becomes active."
   lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
   lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
   WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
     IF sy-index = 30.
       EXIT.               "maximum 5 minutes"
     ENDIF.
     WAIT UP TO 10 SECONDS.
     lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
     lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
   ENDWHILE.

   "Create record."
   TRY.
       lo_record_result = lo_kns->putrecord(
           iv_streamname = iv_stream_name
           iv_data       = iv_data
           iv_partitionkey = iv_partition_key ).
       MESSAGE 'Record created.' TYPE 'I'.
```

```
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knskmsaccessdeniedex.
       MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
     CATCH /aws1/cx_knskmsdisabledex.
       MESSAGE 'KMS key used is disabled.' TYPE 'E'.
     CATCH /aws1/cx_knskmsinvalidstateex.
       MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
     CATCH /aws1/cx_knskmsnotfoundex.
       MESSAGE 'KMS key used is not found.' TYPE 'E'.
     CATCH /aws1/cx_knskmsoptinrequired.
       MESSAGE 'KMS key option is required.' TYPE 'E'.
     CATCH /aws1/cx_knskmsthrottlingex.
       MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_knsresourcenotfoundex.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.

   "Create a shard iterator in order to read the record."
   TRY.
       lo_sharditerator = lo_kns->getsharditerator(
         iv_shardid = lo_record_result->get_shardid( )
         iv_sharditeratortype = iv_sharditeratortype
         iv_streamname = iv_stream_name ).
       MESSAGE 'Shard iterator created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knsprovthruputexcdex.
       MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
   ENDTRY.

   "Read the record."
   TRY.
       oo_result = lo_kns->getrecords(                    " oo_result is returned
for testing purposes. "
           iv_sharditerator   = lo_sharditerator->get_sharditerator( ) ).
```

```
        MESSAGE 'Shard iterator created.' TYPE 'I'.
      CATCH /aws1/cx_knsexpirediteratorex.
        MESSAGE 'Iterator expired.' TYPE 'E'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.

    "Delete stream."
    TRY.
        lo_kns->deletestream(
            iv_streamname = iv_stream_name ).
        MESSAGE 'Stream deleted.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceeded
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourceinuseex.
        MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - CreateStream

- DeleteStream
- GetRecords
- GetShardIterator
- PutRecord

Aktionen

`CreateStream`

Das folgende Codebeispiel zeigt die Verwendung`CreateStream`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_kns->createstream(
           iv_streamname = iv_stream_name
           iv_shardcount = iv_shard_count ).
       MESSAGE 'Stream created.' TYPE 'I'.
     CATCH /aws1/cx_knsinvalidargumentex.
       MESSAGE 'The specified argument was not valid.' TYPE 'E'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreateStream in der AWS SDK for SAP ABAP API-Referenz.

`DeleteStream`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStream`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_kns->deletestream(
           iv_streamname = iv_stream_name ).
       MESSAGE 'Stream deleted.' TYPE 'I'.
     CATCH /aws1/cx_knslimitexceededex.
       MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
     CATCH /aws1/cx_knsresourceinuseex.
       MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie DeleteStream in der AWS SDK for SAP ABAP API-Referenz.

`DescribeStream`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStream`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
```

```
        oo_result = lo_kns->describestream(
            iv_streamname = iv_stream_name ).
        DATA(lt_stream_description) = oo_result->get_streamdescription( ).
        MESSAGE 'Streams retrieved.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeStream in der AWS SDK for SAP ABAP API-Referenz.

`GetRecords`

Das folgende Codebeispiel zeigt die Verwendung`GetRecords`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->getrecords(             " oo_result is returned for
testing purposes. "
            iv_sharditerator = iv_shard_iterator ).
        DATA(lt_records) = oo_result->get_records( ).
        MESSAGE 'Record retrieved.' TYPE 'I'.
      CATCH /aws1/cx_knsexpirediteratorex.
        MESSAGE 'Iterator expired.' TYPE 'E'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
```

```
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
        MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetRecords in der AWS SDK for SAP ABAP API-Referenz.

`ListStreams`

Das folgende Codebeispiel zeigt die Verwendung`ListStreams`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->liststreams(        " oo_result is returned for testing
purposes. "
            "Set Limit to specify that a maximum of streams should be returned."
            iv_limit = iv_limit ).
        DATA(lt_streams) = oo_result->get_streamnames( ).
        MESSAGE 'Streams listed.' TYPE 'I'.
      CATCH /aws1/cx_knslimitexceededex.
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
```

```
    ENDTRY.
```

- API-Details finden Sie ListStreams in der AWS SDK for SAP ABAP API-Referenz.

`PutRecord`

Das folgende Codebeispiel zeigt die Verwendung`PutRecord`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->putrecord(            " oo_result is returned for
testing purposes. "
            iv_streamname = iv_stream_name
            iv_data       = iv_data
            iv_partitionkey = iv_partition_key ).
        MESSAGE 'Record created.' TYPE 'I'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_knskmsaccessdeniedex.
        MESSAGE 'You do not have permission to perform this AWS KMS action.' TYPE
'E'.
      CATCH /aws1/cx_knskmsdisabledex.
        MESSAGE 'KMS key used is disabled.' TYPE 'E'.
      CATCH /aws1/cx_knskmsinvalidstateex.
        MESSAGE 'KMS key used is in an invalid state. ' TYPE 'E'.
      CATCH /aws1/cx_knskmsnotfoundex.
        MESSAGE 'KMS key used is not found.' TYPE 'E'.
      CATCH /aws1/cx_knskmsoptinrequired.
        MESSAGE 'KMS key option is required.' TYPE 'E'.
      CATCH /aws1/cx_knskmsthrottlingex.
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
      CATCH /aws1/cx_knsprovthruputexcdex.
```

```
        MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
      CATCH /aws1/cx_knsresourcenotfoundex.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutRecord in der AWS SDK for SAP ABAP API-Referenz.

`RegisterStreamConsumer`

Das folgende Codebeispiel zeigt die Verwendung`RegisterStreamConsumer`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kns->registerstreamconsumer(       " oo_result is returned
for testing purposes. "
            iv_streamarn = iv_stream_arn
            iv_consumername = iv_consumer_name ).
        MESSAGE 'Stream consumer registered.' TYPE 'I'.
      CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourceinuse.
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie RegisterStreamConsumer in der AWS SDK for SAP ABAP API-Referenz.

AWS KMS Beispiele zur Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Aktionen ausführen und gängige Szenarien implementieren. AWS KMS

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo AWS KMS

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Key Management Service beginnen.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_kms->listkeys( ).
    MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListKeys in der AWS SDK for SAP ABAP API-Referenz.

Aktionen

`CreateAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAlias`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->createalias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_key_id
        ).
        MESSAGE 'Alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsalreadyexistsex.
        MESSAGE 'Alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidaliasnameex.
        MESSAGE 'Invalid alias name.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateAlias in der AWS SDK for SAP ABAP API-Referenz.

`CreateGrant`

Das folgende Codebeispiel zeigt die Verwendung`CreateGrant`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grantee_principal = 'arn:aws:iam::123456789012:role/my-role'
        " it_operations contains 'Encrypt', 'Decrypt', 'GenerateDataKey'
        oo_result = lo_kms->creategrant(
          iv_keyid = iv_key_id
          iv_granteeprincipal = iv_grantee_principal
          it_operations = it_operations
        ).
        MESSAGE 'Grant created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateGrant in der AWS SDK for SAP ABAP API-Referenz.

`CreateKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_description = 'Created by the AWS SDK for SAP ABAP'
        oo_result = lo_kms->createkey( iv_description = iv_description ).
        MESSAGE 'KMS key created successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
      CATCH /aws1/cx_kmslimitexceededex.
        MESSAGE 'Limit exceeded for KMS resources.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateKey in der AWS SDK for SAP ABAP API-Referenz.

`Decrypt`

Das folgende Codebeispiel zeigt die Verwendung`Decrypt`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->decrypt(
          iv_keyid = iv_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Text decrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
```

```
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie unter Decrypt in AWS SDK for SAP ABAP API-Referenz.

`DeleteAlias`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteAlias`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_alias_name = 'alias/my-key-alias'
    lo_kms->deletealias( iv_aliasname = iv_alias_name ).
    MESSAGE 'Alias deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Alias not found.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteAlias in der AWS SDK for SAP ABAP API-Referenz.

`DescribeKey`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->describekey( iv_keyid = iv_key_id ).
        DATA(lo_key) = oo_result->get_keymetadata( ).
        MESSAGE 'Retrieved key information successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeKey in der AWS SDK for SAP ABAP API-Referenz.

`DisableKey`

Das folgende Codebeispiel zeigt die Verwendung`DisableKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->disablekey( iv_keyid = iv_key_id ).
```

```
        MESSAGE 'KMS key disabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DisableKey in der AWS SDK for SAP ABAP API-Referenz.

`EnableKey`

Das folgende Codebeispiel zeigt die Verwendung`EnableKey`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekey( iv_keyid = iv_key_id ).
        MESSAGE 'KMS key enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie EnableKey in der AWS SDK for SAP ABAP API-Referenz.

`EnableKeyRotation`

Das folgende Codebeispiel zeigt die Verwendung`EnableKeyRotation`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        lo_kms->enablekeyrotation( iv_keyid = iv_key_id ).
        MESSAGE 'Key rotation enabled successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsunsupportedopex.
        MESSAGE 'Operation not supported for this key.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie EnableKeyRotation in der AWS SDK for SAP ABAP API-Referenz.

`Encrypt`

Das folgende Codebeispiel zeigt die Verwendung`Encrypt`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_plaintext contains the data to encrypt
        oo_result = lo_kms->encrypt(
          iv_keyid = iv_key_id
          iv_plaintext = iv_plaintext
        ).
        MESSAGE 'Text encrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie unter Encrypt in AWS SDK for SAP ABAP API-Referenz.

`GenerateDataKey`

Das folgende Codebeispiel zeigt die Verwendung. `GenerateDataKey`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_keyspec = 'AES_256'
        oo_result = lo_kms->generatedatakey(
          iv_keyid = iv_key_id
          iv_keyspec = 'AES_256'
        ).
        MESSAGE 'Data key generated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
```

```
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GenerateDataKey in der AWS SDK for SAP ABAP API-Referenz.

`GetKeyPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetKeyPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->getkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
        ).
        MESSAGE 'Retrieved key policy successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetKeyPolicy in der AWS SDK for SAP ABAP API-Referenz.

`ListAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAliases`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kms->listaliases( ).
        MESSAGE 'Retrieved KMS aliases list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListAliases in der AWS SDK for SAP ABAP API-Referenz.

`ListGrants`

Das folgende Codebeispiel zeigt die Verwendung`ListGrants`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listgrants( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved grants list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListGrants in der AWS SDK for SAP ABAP API-Referenz.

`ListKeyPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListKeyPolicies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        oo_result = lo_kms->listkeypolicies( iv_keyid = iv_key_id ).
        MESSAGE 'Retrieved key policies list.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListKeyPolicies in der AWS SDK for SAP ABAP API-Referenz.

`ListKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListKeys`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_kms->listkeys( ).
        MESSAGE 'Retrieved KMS keys list.' TYPE 'I'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListKeys in der AWS SDK for SAP ABAP API-Referenz.

`PutKeyPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutKeyPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_policy = '{"Version":"2012-10-17",        "Statement": [...]}'
        lo_kms->putkeypolicy(
          iv_keyid = iv_key_id
          iv_policyname = 'default'
          iv_policy = iv_policy
        ).
        MESSAGE 'Key policy updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsmalformedplydocex.
        MESSAGE 'Malformed policy document.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutKeyPolicy in der AWS SDK for SAP ABAP API-Referenz.

`ReEncrypt`

Das folgende Codebeispiel zeigt die Verwendung`ReEncrypt`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_source_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_destination_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        " iv_ciphertext_blob contains the encrypted data
        oo_result = lo_kms->reencrypt(
          iv_sourcekeyid = iv_source_key_id
          iv_destinationkeyid = iv_destination_key_id
          iv_ciphertextblob = iv_ciphertext_blob
        ).
        MESSAGE 'Ciphertext reencrypted successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsincorrectkeyex.
        MESSAGE 'Incorrect source key for decryption.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ReEncrypt in der AWS SDK for SAP ABAP API-Referenz.

`RetireGrant`

Das folgende Codebeispiel zeigt die Verwendung`RetireGrant`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_grant_token = 'AQpAM2RhZ...'
    lo_kms->retiregrant( iv_granttoken = iv_grant_token ).
    MESSAGE 'Grant retired successfully.' TYPE 'I'.
  CATCH /aws1/cx_kmsnotfoundexception.
    MESSAGE 'Grant not found.' TYPE 'E'.
  CATCH /aws1/cx_kmsinvgranttokenex.
    MESSAGE 'Invalid grant token.' TYPE 'E'.
  CATCH /aws1/cx_kmskmsinternalex.
    MESSAGE 'An internal error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie RetireGrant in der AWS SDK for SAP ABAP API-Referenz.

`RevokeGrant`

Das folgende Codebeispiel zeigt die Verwendung`RevokeGrant`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_grant_id = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
        lo_kms->revokegrant(
          iv_keyid = iv_key_id
          iv_grantid = iv_grant_id
        ).
        MESSAGE 'Grant revoked successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Grant or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidgrantidex.
        MESSAGE 'Invalid grant ID.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie RevokeGrant in der AWS SDK for SAP ABAP API-Referenz.

`ScheduleKeyDeletion`

Das folgende Codebeispiel zeigt die Verwendung`ScheduleKeyDeletion`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_pending_window_days = 7
        oo_result = lo_kms->schedulekeydeletion(
          iv_keyid = iv_key_id
          iv_pendingwindowindays = iv_pending_window_days
        ).
        MESSAGE 'Key scheduled for deletion.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
```

```
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ScheduleKeyDeletion in der AWS SDK for SAP ABAP API-Referenz.

`Sign`

Das folgende Codebeispiel zeigt die Verwendung`Sign`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the message to sign
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->sign(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signingalgorithm = iv_signing_algorithm
        ).
        MESSAGE 'Message signed successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmsinvalidkeyusageex.
        MESSAGE 'Key cannot be used for signing.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie unter Sign in AWS SDK for SAP ABAP API-Referenz.

`TagResource`

Das folgende Codebeispiel zeigt die Verwendung`TagResource`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_tags TYPE /aws1/cl_kmstag=>tt_taglist.

    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab'
        " iv_tag_key = 'Environment'
        " iv_tag_value = 'Production'
        APPEND NEW /aws1/cl_kmstag(
          iv_tagkey = iv_tag_key
          iv_tagvalue = iv_tag_value
        ) TO lt_tags.

        lo_kms->tagresource(
          iv_keyid = iv_key_id
          it_tags = lt_tags
        ).
        MESSAGE 'Tag added to KMS key successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmstagexception.
        MESSAGE 'Invalid tag format.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie TagResource in der AWS SDK for SAP ABAP API-Referenz.

`UpdateAlias`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAlias`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_alias_name = 'alias/my-key-alias'
        " iv_target_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/5678dcba-56cd-78ef-90ab-5678901234cd'
        lo_kms->updatealias(
          iv_aliasname = iv_alias_name
          iv_targetkeyid = iv_target_key_id
        ).
        MESSAGE 'Alias updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Alias or key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie UpdateAlias in der AWS SDK for SAP ABAP API-Referenz.

`Verify`

Das folgende Codebeispiel zeigt die Verwendung`Verify`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_key_id = 'arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab' (asymmetric key)
        " iv_message contains the original message
        " iv_signature contains the signature to verify
        " iv_signing_algorithm = 'RSASSA_PSS_SHA_256'
        oo_result = lo_kms->verify(
          iv_keyid = iv_key_id
          iv_message = iv_message
          iv_signature = iv_signature
          iv_signingalgorithm = iv_signing_algorithm
        ).
        DATA(lv_valid) = oo_result->get_signaturevalid( ).
        IF lv_valid = abap_true.
          MESSAGE 'Signature is valid.' TYPE 'I'.
        ELSE.
          MESSAGE 'Signature is invalid.' TYPE 'I'.
        ENDIF.
      CATCH /aws1/cx_kmsdisabledexception.
        MESSAGE 'The key is disabled.' TYPE 'E'.
      CATCH /aws1/cx_kmsnotfoundexception.
        MESSAGE 'Key not found.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinvalidsigex.
        MESSAGE 'Invalid signature.' TYPE 'E'.
      CATCH /aws1/cx_kmskmsinternalex.
        MESSAGE 'An internal error occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie unter Verify in der AWS SDK for SAP ABAP API-Referenz.

Lamdba-Beispiele unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Lambda Aktionen ausführen und gängige Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       "Create an AWS Identity and Access Management (IAM) role that grants AWS
Lambda permission to write to logs."
       DATA(lv_policy_document) = `{` &&
           `"Version":"2012-10-17",        ` &&
                 `"Statement": [` &&
```

```
                    `{` &&
                      `"Effect": "Allow",` &&
                      `"Action": [` &&
                        `"sts:AssumeRole"` &&
                      `],` &&
                      `"Principal": {` &&
                        `"Service": [` &&
                          `"lambda.amazonaws.com"` &&
                        `]` &&
                      `}` &&
                    `}` &&
                  `]` &&
                `}`.
        TRY.
            DATA(lo_create_role_output) = lo_iam->createrole(
                    iv_rolename = iv_role_name
                    iv_assumerolepolicydocument = lv_policy_document
                    iv_description = 'Grant lambda permission to write to logs' ).
            DATA(lv_role_arn) = lo_create_role_output->get_role( )->get_arn( ).
            MESSAGE 'IAM role created.' TYPE 'I'.
            WAIT UP TO 10 SECONDS.            " Make sure that the IAM role is ready
 for use. "
          CATCH /aws1/cx_iamentityalrdyexex.
            DATA(lo_role) = lo_iam->getrole( iv_rolename = iv_role_name ).
            lv_role_arn = lo_role->get_role( )->get_arn( ).
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iammalformedplydocex.
            MESSAGE 'Policy document in the request is malformed.' TYPE 'E'.
        ENDTRY.

        TRY.
            lo_iam->attachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Attached policy to the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'E'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
```

```
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.

        " Create a Lambda function and upload handler code. "
        " Lambda function performs 'increment' action on a number. "
        TRY.
            lo_lmd->createfunction(
                 iv_functionname = iv_function_name
                 iv_runtime = `python3.9`
                 iv_role = lv_role_arn
                 iv_handler = iv_handler
                 io_code = io_initial_zip_file
                 iv_description = 'AWS Lambda code example' ).
            MESSAGE 'Lambda function created.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        " Verify the function is in Active state "
        WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
          IF sy-index = 10.
            EXIT.               " Maximum 10 seconds. "
          ENDIF.
          WAIT UP TO 1 SECONDS.
        ENDWHILE.

        "Invoke the function with a single parameter and get results."
        TRY.
            DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "increment",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_initial_invoke_output) = lo_lmd->invoke(
                       iv_functionname = iv_function_name
                       iv_payload = lv_json ).
```

```
            ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
        " ov_initial_invoke_payload is returned for testing purposes. "
            DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT XML
 lo_writer_json.
            DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " Update the function code and configure its Lambda environment with an
 environment variable. "
        " Lambda function is updated to perform 'decrement' action also. "
        TRY.
            lo_lmd->updatefunctioncode(
                  iv_functionname = iv_function_name
                  iv_zipfile = io_updated_zip_file ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function code updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdcodestorageexcdex.
            MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        TRY.
            DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
            DATA ls_variable LIKE LINE OF lt_variables.
            ls_variable-key = 'LOG_LEVEL'.
```

```
            ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
 'info' ).
            INSERT ls_variable INTO TABLE lt_variables.

            lo_lmd->updatefunctionconfiguration(
                  iv_functionname = iv_function_name
                  io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
 lt_variables ) ).
            WAIT UP TO 10 SECONDS.            " Make sure that the update is
 completed. "
            MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourceconflictex.
            MESSAGE 'Resource already exists or another operation is in progress.'
 TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
        ENDTRY.

        "Invoke the function with new parameters and get results. Display the
 execution log that's returned from the invocation."
        TRY.
            lv_json = /aws1/cl_rt_util=>string_to_xstring(
              `{`  &&
                `"action": "decrement",`  &&
                `"number": 10` &&
              `}` ).
            DATA(lo_updated_invoke_output) = lo_lmd->invoke(
                        iv_functionname = iv_function_name
                        iv_payload = lv_json ).
            ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
        " ov_updated_invoke_payload is returned for testing purposes. "
            lo_writer_json = cl_sxml_string_writer=>create( type =
 if_sxml=>co_xt_json ).
            CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT XML
 lo_writer_json.
            lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
            MESSAGE 'Lambda function invoked.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
```

```
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'E'.
          CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type.'
 TYPE 'E'.
        ENDTRY.

        " List the functions for your account. "
        TRY.
            DATA(lo_list_output) = lo_lmd->listfunctions( ).
            DATA(lt_functions) = lo_list_output->get_functions( ).
            MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
        ENDTRY.

        " Delete the Lambda function. "
        TRY.
            lo_lmd->deletefunction( iv_functionname = iv_function_name ).
            MESSAGE 'Lambda function deleted.' TYPE 'I'.
          CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist.' TYPE 'W'.
        ENDTRY.

        " Detach role policy. "
        TRY.
            lo_iam->detachrolepolicy(
                iv_rolename  = iv_role_name
                iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole' ).
            MESSAGE 'Detached policy from the IAM role.' TYPE 'I'.
          CATCH /aws1/cx_iaminvalidinputex.
            MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamplynotattachableex.
            MESSAGE 'Service role policies can only be attached to the service-
linked role for their service.' TYPE 'E'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
 modifiable.' TYPE 'E'.
        ENDTRY.
```

```
        " Delete the IAM role. "
        TRY.
            lo_iam->deleterole( iv_rolename = iv_role_name ).
            MESSAGE 'IAM role deleted.' TYPE 'I'.
          CATCH /aws1/cx_iamnosuchentityex.
            MESSAGE 'The requested resource entity does not exist.' TYPE 'W'.
          CATCH /aws1/cx_iamunmodableentityex.
            MESSAGE 'Service that depends on the service-linked role is not
modifiable.' TYPE 'E'.
        ENDTRY.

      CATCH /aws1/cx_rt_service_generic INTO lo_exception.
        DATA(lv_error) = lo_exception->get_longtext( ).
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_lmd->createfunction(
           iv_functionname = iv_function_name
           iv_runtime = `python3.9`
           iv_role = iv_role_arn
           iv_handler = iv_handler
           io_code = io_zip_file
           iv_description = 'AWS Lambda code example' ).
       MESSAGE 'Lambda function created.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreateFunction in der AWS SDK for SAP ABAP API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       lo_lmd->deletefunction( iv_functionname = iv_function_name ).
       MESSAGE 'Lambda function deleted.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie DeleteFunction in der AWS SDK for SAP ABAP API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).
" oo_result is returned for testing purposes. "
       MESSAGE 'Lambda function information retrieved.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetFunction in der AWS SDK for SAP ABAP API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die Verwendung`Invoke`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
         `{`  &&
```

```
            `"action": "increment",`  &&
            `"number": 10` &&
          `}` ).
        oo_result = lo_lmd->invoke(                  " oo_result is returned for
testing purposes. "
                iv_functionname = iv_function_name
                iv_payload = lv_json ).
        MESSAGE 'Lambda function invoked.' TYPE 'I'.
      CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvrequestcontex.
        MESSAGE 'Unable to parse request body as JSON.' TYPE 'E'.
      CATCH /aws1/cx_lmdinvalidzipfileex.
        MESSAGE 'The deployment package could not be unzipped.' TYPE 'E'.
      CATCH /aws1/cx_lmdrequesttoolargeex.
        MESSAGE 'Invoke request body JSON input limit was exceeded by the request
payload.' TYPE 'E'.
      CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
      CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist.' TYPE 'E'.
      CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
      CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
      CATCH /aws1/cx_lmdunsuppedmediatyp00.
        MESSAGE 'Invoke request body does not have JSON as its content type.' TYPE
'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Invoke (Aufrufen) in der API-Referenz für das AWS -SDK für SAP ABAP.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->listfunctions( ).       " oo_result is returned for
testing purposes. "
       DATA(lt_functions) = oo_result->get_functions( ).
       MESSAGE 'Retrieved list of Lambda functions.' TYPE 'I'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie ListFunctions in der AWS SDK for SAP ABAP API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->updatefunctioncode(     " oo_result is returned for
testing purposes. "
```

```
              iv_functionname = iv_function_name
              iv_zipfile = io_zip_file ).

       MESSAGE 'Lambda function code updated.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodestorageexcdex.
       MESSAGE 'Maximum total code size per account exceeded.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK for SAP ABAP API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_lmd->updatefunctionconfiguration(     " oo_result is returned
for testing purposes. "
             iv_functionname = iv_function_name
             iv_runtime = iv_runtime
             iv_description  = 'Updated Lambda function'
             iv_memorysize  = iv_memory_size ).

       MESSAGE 'Lambda function configuration/settings updated.' TYPE 'I'.
     CATCH /aws1/cx_lmdcodesigningcfgno00.
       MESSAGE 'Code signing configuration does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdcodeverification00.
       MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvalidcodesigex.
       MESSAGE 'Code signature failed the integrity check.' TYPE 'E'.
     CATCH /aws1/cx_lmdinvparamvalueex.
       MESSAGE 'The request contains a non-valid parameter.' TYPE 'E'.
     CATCH /aws1/cx_lmdresourceconflictex.
       MESSAGE 'Resource already exists or another operation is in progress.' TYPE
'E'.
     CATCH /aws1/cx_lmdresourcenotfoundex.
       MESSAGE 'The requested resource does not exist.' TYPE 'E'.
     CATCH /aws1/cx_lmdserviceexception.
       MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
     CATCH /aws1/cx_lmdtoomanyrequestsex.
       MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Organisationen, die SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP with Organizations Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachPolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_org->attachpolicy(
      iv_policyid = iv_policy_id
      iv_targetid = iv_target_id ).
    MESSAGE 'Policy attached to target.' TYPE 'I'.
  CATCH /aws1/cx_orgaccessdeniedex.
    MESSAGE 'You do not have permission to attach the policy.' TYPE 'E'.
  CATCH /aws1/cx_orgpolicynotfoundex.
    MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgtargetnotfoundex.
    MESSAGE 'The specified target does not exist.' TYPE 'E'.
  CATCH /aws1/cx_orgduplicateplyatta00.
    MESSAGE 'The policy is already attached to the target.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie AttachPolicy in der AWS SDK for SAP ABAP API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_org->createpolicy(       " oo_result is returned for testing
purposes. "
         iv_name        = iv_policy_name
         iv_description = iv_policy_description
         iv_content     = iv_policy_content
         iv_type        = iv_policy_type ).
       MESSAGE 'Policy created.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to create a policy.' TYPE 'E'.
     CATCH /aws1/cx_orgduplicatepolicyex.
       MESSAGE 'A policy with this name already exists.' TYPE 'E'.
     CATCH /aws1/cx_orgmalformedplydocex.
       MESSAGE 'The policy content is malformed.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreatePolicy in der AWS SDK for SAP ABAP API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_org->deletepolicy(
          iv_policyid = iv_policy_id ).
        MESSAGE 'Policy deleted.' TYPE 'I'.
      CATCH /aws1/cx_orgaccessdeniedex.
        MESSAGE 'You do not have permission to delete the policy.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicynotfoundex.
        MESSAGE 'The specified policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_orgpolicyinuseex.
        MESSAGE 'The policy is still attached to one or more targets.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeletePolicy in der AWS SDK for SAP ABAP API-Referenz.

`DescribePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DescribePolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_org->describepolicy(     " oo_result is returned for testing
purposes. "
          iv_policyid = iv_policy_id ).
```

```
      DATA(lo_policy) = oo_result->get_policy( ).
      MESSAGE 'Retrieved policy details.' TYPE 'I'.
    CATCH /aws1/cx_orgaccessdeniedex.
      MESSAGE 'You do not have permission to describe the policy.' TYPE 'E'.
    CATCH /aws1/cx_orgpolicynotfoundex.
      MESSAGE 'The specified policy does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribePolicy in der AWS SDK for SAP ABAP API-Referenz.

`DetachPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_org->detachpolicy(
        iv_policyid = iv_policy_id
        iv_targetid = iv_target_id ).
      MESSAGE 'Policy detached from target.' TYPE 'I'.
    CATCH /aws1/cx_orgaccessdeniedex.
      MESSAGE 'You do not have permission to detach the policy.' TYPE 'E'.
    CATCH /aws1/cx_orgpolicynotfoundex.
      MESSAGE 'The specified policy does not exist.' TYPE 'E'.
    CATCH /aws1/cx_orgtargetnotfoundex.
      MESSAGE 'The specified target does not exist.' TYPE 'E'.
    CATCH /aws1/cx_orgpolicynotattex.
      MESSAGE 'The policy is not attached to the target.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DetachPolicy in der AWS SDK for SAP ABAP API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_org->listpolicies(       " oo_result is returned for testing
purposes. "
         iv_filter = iv_filter ).
       DATA(lt_policies) = oo_result->get_policies( ).
       MESSAGE 'Retrieved list of policies.' TYPE 'I'.
     CATCH /aws1/cx_orgaccessdeniedex.
       MESSAGE 'You do not have permission to list policies.' TYPE 'E'.
     CATCH /aws1/cx_orgawsorgsnotinuseex.
       MESSAGE 'Your account is not a member of an organization.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie ListPolicies in der AWS SDK for SAP ABAP API-Referenz.

Amazon Pinpoint-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Pinpoint Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`SendMessages`

Das folgende Codebeispiel zeigt die Verwendung. `SendMessages`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine E-Mail-Nachricht.

```
" Build the addresses map from the list of to_addresses
DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
LOOP AT it_to_addresses INTO DATA(lo_address).
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = lo_address->get_value( )
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
  ) INTO TABLE lt_addresses.
ENDLOOP.

" Send the email message
DATA(lo_result) = lo_ppt->sendmessages(
  iv_applicationid = iv_app_id
  io_messagerequest = NEW /aws1/cl_pptmessagerequest(
    it_addresses = lt_addresses
    io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
      io_emailmessage = NEW /aws1/cl_pptemailmessage(
        iv_fromaddress = iv_sender
        io_simpleemail = NEW /aws1/cl_pptsimpleemail(
          io_subject = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
            iv_data = iv_subject
          )
          io_htmlpart = NEW /aws1/cl_pptsimpleemailpart(
            iv_charset = iv_char_set
```

```
                iv_data = iv_html_message
              )
              io_textpart = NEW /aws1/cl_pptsimpleemailpart(
                iv_charset = iv_char_set
                iv_data = iv_text_message
              )
            )
          )
        )
      )
    ).

    " Extract message IDs from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    ot_message_ids = lo_message_response->get_result( ).

    MESSAGE 'Email message sent successfully.' TYPE 'I'.
```

Senden Sie eine SMS-Nachricht.

```
    " Build the addresses map for the destination number
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = iv_destination_number
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
    ) INTO TABLE lt_addresses.

    " Send the SMS message
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_smsmessage = NEW /aws1/cl_pptsmsmessage(
            iv_body = iv_message
            iv_messagetype = iv_message_type
            iv_originationnumber = iv_origination_number
          )
        )
      )
    ).
```

```
" Extract message ID from response
DATA(lo_message_response) = lo_result->get_messageresponse( ).
DATA(lt_results) = lo_message_response->get_result( ).
LOOP AT lt_results INTO DATA(ls_result).
  IF ls_result-key = iv_destination_number.
    ov_message_id = ls_result-value->get_messageid( ).
    EXIT.
  ENDIF.
ENDLOOP.

MESSAGE 'SMS message sent successfully.' TYPE 'I'.
```

Senden Sie eine E-Mail-Nachricht mit einer vorhandenen E-Mail-Vorlage.

```
" Build the addresses map from the list of to_addresses
DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
LOOP AT it_to_addresses INTO DATA(lo_address).
  INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
    key = lo_address->get_value( )
    value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'EMAIL' )
  ) INTO TABLE lt_addresses.
ENDLOOP.

" Send the email message using a template
DATA(lo_result) = lo_ppt->sendmessages(
  iv_applicationid = iv_app_id
  io_messagerequest = NEW /aws1/cl_pptmessagerequest(
    it_addresses = lt_addresses
    io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
      io_emailmessage = NEW /aws1/cl_pptemailmessage(
        iv_fromaddress = iv_sender
      )
    )
    io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
      io_emailtemplate = NEW /aws1/cl_ppttemplate(
        iv_name = iv_template_name
        iv_version = iv_template_version
      )
    )
  )
).
```

```
    " Extract message IDs from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    ot_message_ids = lo_message_response->get_result( ).

    MESSAGE 'Templated email message sent successfully.' TYPE 'I'.
```

Senden Sie eine Textnachricht mit einer vorhandenen SMS-Vorlage.

```
    " Build the addresses map for the destination number
    DATA lt_addresses TYPE /aws1/cl_pptaddressconf=>tt_mapofaddressconfiguration.
    INSERT VALUE /aws1/cl_pptaddressconf=>ts_mapofaddressconf_maprow(
      key = iv_destination_number
      value = NEW /aws1/cl_pptaddressconf( iv_channeltype = 'SMS' )
    ) INTO TABLE lt_addresses.

    " Send the SMS message using a template
    DATA(lo_result) = lo_ppt->sendmessages(
      iv_applicationid = iv_app_id
      io_messagerequest = NEW /aws1/cl_pptmessagerequest(
        it_addresses = lt_addresses
        io_messageconfiguration = NEW /aws1/cl_pptdirectmessageconf(
          io_smsmessage = NEW /aws1/cl_pptsmsmessage(
            iv_messagetype = iv_message_type
            iv_originationnumber = iv_origination_number
          )
        )
        io_templateconfiguration = NEW /aws1/cl_ppttemplateconf(
          io_smstemplate = NEW /aws1/cl_ppttemplate(
            iv_name = iv_template_name
            iv_version = iv_template_version
          )
        )
      )
    ).

    " Extract message ID from response
    DATA(lo_message_response) = lo_result->get_messageresponse( ).
    DATA(lt_results) = lo_message_response->get_result( ).
    LOOP AT lt_results INTO DATA(ls_result).
      IF ls_result-key = iv_destination_number.
        ov_message_id = ls_result-value->get_messageid( ).
        EXIT.
```

```
      ENDIF.
    ENDLOOP.

    MESSAGE 'Templated SMS message sent successfully.' TYPE 'I'.
```

- API-Details finden Sie SendMessages in der AWS SDK for SAP ABAP API-Referenz.

Amazon Pinpoint SMS- und Voice-API-Beispiele unter Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Pinpoint SMS und Voice API Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateConfigurationSet`

Das folgende Codebeispiel zeigt die Verwendung. `CreateConfigurationSet`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        " Create a new configuration set
        lo_pps->createconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        MESSAGE 'Configuration set created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie CreateConfigurationSet in der AWS SDK for SAP ABAP API-Referenz.

`CreateConfigurationSetEventDestination`

Das folgende Codebeispiel zeigt die Verwendung`CreateConfigurationSetEventDestination`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        " Create event destination for the configuration set
        lo_pps->createconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination created successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsalreadyexistsex INTO DATA(lo_already_exists_ex).
        MESSAGE lo_already_exists_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_already_exists_ex.
      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppslimitexceededex INTO DATA(lo_limit_exceeded_ex).
        MESSAGE lo_limit_exceeded_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_limit_exceeded_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie CreateConfigurationSetEventDestination in der AWS SDK for SAP ABAP API-Referenz.

`DeleteConfigurationSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteConfigurationSet`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Delete the configuration set
        lo_pps->deleteconfigurationset(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        MESSAGE 'Configuration set deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie DeleteConfigurationSet in der AWS SDK for SAP ABAP API-Referenz.

`DeleteConfigurationSetEventDestination`

Das folgende Codebeispiel zeigt die Verwendung`DeleteConfigurationSetEventDestination`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Delete the event destination
        lo_pps->deleteconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
        ).

        MESSAGE 'Event destination deleted successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie DeleteConfigurationSetEventDestination in der AWS SDK for SAP ABAP API-Referenz.

`GetConfigurationSetEventDestinations`

Das folgende Codebeispiel zeigt die Verwendung`GetConfigurationSetEventDestinations`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Get event destinations for the configuration set
        oo_result = lo_pps->getconfseteventdestinations(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
        ).

        " Process the event destinations
        LOOP AT oo_result->get_eventdestinations( ) INTO DATA(lo_event_dest).
          DATA(lv_dest_name) = lo_event_dest->get_name( ).
          DATA(lv_enabled) = lo_event_dest->get_enabled( ).

          MESSAGE |Event destination: { lv_dest_name }, Enabled: { lv_enabled }|
 TYPE 'I'.

          " Check for CloudWatch Logs destination
          DATA(lo_cloudwatch_dest) = lo_event_dest->get_cloudwatchlogsdst( ).
          IF lo_cloudwatch_dest IS NOT INITIAL.
            DATA(lv_log_group_arn) = lo_cloudwatch_dest->get_loggrouparn( ).
            MESSAGE |  CloudWatch Logs destination: { lv_log_group_arn }| TYPE 'I'.
          ENDIF.

          " Check for Kinesis Firehose destination
          DATA(lo_firehose_dest) = lo_event_dest->get_kinesisfirehosedst( ).
          IF lo_firehose_dest IS NOT INITIAL.
            DATA(lv_delivery_stream) = lo_firehose_dest->get_deliverystreamarn( ).
            MESSAGE |  Kinesis Firehose destination: { lv_delivery_stream }| TYPE
 'I'.
          ENDIF.

          " Check for SNS destination
          DATA(lo_sns_dest) = lo_event_dest->get_snsdestination( ).
          IF lo_sns_dest IS NOT INITIAL.
            DATA(lv_topic_arn) = lo_sns_dest->get_topicarn( ).
```

```
            MESSAGE |  SNS destination: { lv_topic_arn }| TYPE 'I'.
          ENDIF.
        ENDLOOP.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie GetConfigurationSetEventDestinations in der AWS SDK for SAP ABAP API-Referenz.

`ListConfigurationSets`

Das folgende Codebeispiel zeigt die Verwendung`ListConfigurationSets`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " List all configuration sets
        oo_result = lo_pps->listconfigurationsets(
          iv_nexttoken = iv_next_token    " Optional: Token for pagination
          iv_pagesize = iv_page_size      " Optional: Number of results per page,
e.g., '10'
        ).
```

```
      " Process the configuration sets
      LOOP AT oo_result->get_configurationsets( ) INTO DATA(lo_config_set).
        DATA(lv_config_set_name) = lo_config_set->get_value( ).
        MESSAGE |Configuration set: { lv_config_set_name }| TYPE 'I'.
      ENDLOOP.

      " Check if there are more results
      DATA(lv_next_token) = oo_result->get_nexttoken( ).
      IF lv_next_token IS NOT INITIAL.
        MESSAGE |More results available. Next token: { lv_next_token }| TYPE 'I'.
      ENDIF.

    CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
      MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_internal_error_ex.
    CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
      MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many_requests_ex.
  ENDTRY.
```

- API-Details finden Sie ListConfigurationSets in der AWS SDK for SAP ABAP API-Referenz.

`SendVoiceMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendVoiceMessage`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Create SSML message type object with voice parameters
```

```
      DATA(lo_ssml_message) = NEW /aws1/cl_ppsssmlmessagetype(
        iv_languagecode = iv_language_code    " e.g., 'en-US'
        iv_voiceid = iv_voice_id              " e.g., 'Matthew'
        iv_text = iv_ssml_message             " SSML formatted message text
      ).

      " Create voice message content with the SSML message
      DATA(lo_content) = NEW /aws1/cl_ppsvoicemessagecont(
        io_ssmlmessage = lo_ssml_message
      ).

      " Send the voice message
      DATA(lo_result) = lo_pps->sendvoicemessage(
        iv_originationphonenumber = iv_origination_number  " e.g., '+12065550110'
        iv_callerid = iv_caller_id                         " e.g., '+12065550199'
        iv_destinationphonenumber = iv_destination_number  " e.g., '+12065550142'
        io_content = lo_content
      ).

      " Retrieve the message ID from the response
      ov_message_id = lo_result->get_messageid( ).

      MESSAGE 'Voice message sent successfully.' TYPE 'I'.

    CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
      MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_internal_error_ex.
    CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
      MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
      RAISE EXCEPTION lo_too_many_requests_ex.
  ENDTRY.
```

- API-Details finden Sie SendVoiceMessage in der AWS SDK for SAP ABAP API-Referenz.

`UpdateConfigurationSetEventDestination`

Das folgende Codebeispiel zeigt die Verwendung`UpdateConfigurationSetEventDestination`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Update the event destination
        lo_pps->updateconfseteventdst(
          iv_configurationsetname = iv_configuration_set_name    " e.g., 'my-config-
set'
          iv_eventdestinationname = iv_event_destination_name    " e.g., 'my-event-
dest'
          io_eventdestination = io_event_destination
        ).

        MESSAGE 'Event destination updated successfully.' TYPE 'I'.

      CATCH /aws1/cx_ppsnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_not_found_ex.
      CATCH /aws1/cx_ppsbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_ppsinternalsvcerrorex INTO DATA(lo_internal_error_ex).
        MESSAGE lo_internal_error_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_internal_error_ex.
      CATCH /aws1/cx_ppstoomanyrequestsex INTO DATA(lo_too_many_requests_ex).
        MESSAGE lo_too_many_requests_ex->get_text( ) TYPE 'I'.
        RAISE EXCEPTION lo_too_many_requests_ex.
    ENDTRY.
```

- API-Details finden Sie UpdateConfigurationSetEventDestination in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Polly, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Polly Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteLexicon`

Das folgende Codebeispiel zeigt die Verwendung. `DeleteLexicon`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ply->deletelexicon( iv_name ).
    MESSAGE 'Lexicon deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteLexicon in der AWS SDK for SAP ABAP API-Referenz.

`DescribeVoices`

Das folgende Codebeispiel zeigt die Verwendung`DescribeVoices`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_engine IS NOT INITIAL AND iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine
            iv_languagecode = iv_language ).
        ELSEIF iv_engine IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_engine = iv_engine ).
        ELSEIF iv_language IS NOT INITIAL.
          oo_result = lo_ply->describevoices(
            iv_languagecode = iv_language ).
        ELSE.
          oo_result = lo_ply->describevoices( ).
        ENDIF.
        MESSAGE 'Retrieved voice metadata.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidnexttokenex.
        MESSAGE 'The NextToken is invalid.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeVoices in der AWS SDK for SAP ABAP API-Referenz.

`GetLexicon`

Das folgende Codebeispiel zeigt die Verwendung`GetLexicon`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ply->getlexicon( iv_name ).
    DATA(lo_lexicon) = oo_result->get_lexicon( ).
    IF lo_lexicon IS BOUND.
      DATA(lv_lex_name) = lo_lexicon->get_name( ).
      MESSAGE |Retrieved lexicon: { lv_lex_name }| TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_plylexiconnotfoundex.
    MESSAGE 'Lexicon not found.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetLexicon in der AWS SDK for SAP ABAP API-Referenz.

`GetSpeechSynthesisTask`

Das folgende Codebeispiel zeigt die Verwendung`GetSpeechSynthesisTask`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ply->getspeechsynthesistask( iv_task_id ).
    DATA(lo_task) = oo_result->get_synthesistask( ).
    IF lo_task IS BOUND.
```

```
        DATA(lv_status) = lo_task->get_taskstatus( ).
        MESSAGE |Task status: { lv_status }| TYPE 'I'.
      ENDIF.
    CATCH /aws1/cx_plyinvalidtaskidex.
      MESSAGE 'Invalid task ID.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
    CATCH /aws1/cx_plysynthesistsknotf00.
      MESSAGE 'Synthesis task not found.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie GetSpeechSynthesisTask in der AWS SDK for SAP ABAP API-Referenz.

`ListLexicons`

Das folgende Codebeispiel zeigt die Verwendung`ListLexicons`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_ply->listlexicons( ).
      DATA(lt_lexicons) = oo_result->get_lexicons( ).
      DATA(lv_count) = lines( lt_lexicons ).
      MESSAGE |Found { lv_count } lexicons| TYPE 'I'.
    CATCH /aws1/cx_plyinvalidnexttokenex.
      MESSAGE 'Invalid NextToken.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListLexicons in der AWS SDK for SAP ABAP API-Referenz.

`ListSpeechSynthesisTasks`

Das folgende Codebeispiel zeigt die Verwendung`ListSpeechSynthesisTasks`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_max_results IS NOT INITIAL AND iv_status IS NOT INITIAL.
          oo_result = lo_ply->listspeechsynthesistasks(
            iv_maxresults = iv_max_results
            iv_status = iv_status ).
        ELSEIF iv_max_results IS NOT INITIAL.
          oo_result = lo_ply->listspeechsynthesistasks(
            iv_maxresults = iv_max_results ).
        ELSEIF iv_status IS NOT INITIAL.
          oo_result = lo_ply->listspeechsynthesistasks(
            iv_status = iv_status ).
        ELSE.
          oo_result = lo_ply->listspeechsynthesistasks( ).
        ENDIF.
        DATA(lt_tasks) = oo_result->get_synthesistasks( ).
        DATA(lv_count) = lines( lt_tasks ).
        MESSAGE |Found { lv_count } synthesis tasks| TYPE 'I'.
      CATCH /aws1/cx_plyinvalidnexttokenex.
        MESSAGE 'Invalid NextToken.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListSpeechSynthesisTasks in der AWS SDK for SAP ABAP API-Referenz.

`PutLexicon`

Das folgende Codebeispiel zeigt die Verwendung`PutLexicon`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ply->putlexicon(
      iv_name = iv_name
      iv_content = iv_content ).
    MESSAGE 'Lexicon created successfully.' TYPE 'I'.
  CATCH /aws1/cx_plyinvalidlexiconex.
    MESSAGE 'Invalid lexicon.' TYPE 'E'.
  CATCH /aws1/cx_plylexiconsizeexcdex.
    MESSAGE 'Lexicon size exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plymaxlexemelengthe00.
    MESSAGE 'Maximum lexeme length exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plymaxlexiconsnoexc00.
    MESSAGE 'Maximum number of lexicons exceeded.' TYPE 'E'.
  CATCH /aws1/cx_plyservicefailureex.
    MESSAGE 'Service failure occurred.' TYPE 'E'.
  CATCH /aws1/cx_plyunsuppedplsalpha00.
    MESSAGE 'Unsupported PLS alphabet.' TYPE 'E'.
  CATCH /aws1/cx_plyunsuppedplslangu00.
    MESSAGE 'Unsupported PLS language.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie PutLexicon in der AWS SDK for SAP ABAP API-Referenz.

`StartSpeechSynthesisTask`

Das folgende Codebeispiel zeigt die Verwendung`StartSpeechSynthesisTask`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Only pass optional parameters if they have values
        IF iv_lang_code IS NOT INITIAL AND iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_lang_code IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id
            iv_languagecode = iv_lang_code ).
        ELSEIF iv_s3_key_prefix IS NOT INITIAL.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_outputs3keyprefix = iv_s3_key_prefix
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ELSE.
          oo_result = lo_ply->startspeechsynthesistask(
            iv_engine = iv_engine
            iv_outputformat = iv_audio_format
            iv_outputs3bucketname = iv_s3_bucket
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
```

```
      ENDIF.
      MESSAGE 'Speech synthesis task started.' TYPE 'I'.
    CATCH /aws1/cx_plyinvalids3bucketex.
      MESSAGE 'Invalid S3 bucket.' TYPE 'E'.
    CATCH /aws1/cx_plyinvalidssmlex.
      MESSAGE 'Invalid SSML.' TYPE 'E'.
    CATCH /aws1/cx_plylexiconnotfoundex.
      MESSAGE 'Lexicon not found.' TYPE 'E'.
    CATCH /aws1/cx_plyservicefailureex.
      MESSAGE 'Service failure occurred.' TYPE 'E'.
    CATCH /aws1/cx_plytextlengthexcdex.
      MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie StartSpeechSynthesisTask in der AWS SDK for SAP ABAP API-Referenz.

`SynthesizeSpeech`

Das folgende Codebeispiel zeigt die Verwendung`SynthesizeSpeech`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Only pass optional language code if it has a value
      IF iv_lang_code IS NOT INITIAL.
        oo_result = lo_ply->synthesizespeech(
          iv_engine = iv_engine
          iv_outputformat = iv_output_fmt
          iv_text = iv_text
          iv_voiceid = iv_voice_id
          iv_languagecode = iv_lang_code ).
      ELSE.
        oo_result = lo_ply->synthesizespeech(
```

```
            iv_engine = iv_engine
            iv_outputformat = iv_output_fmt
            iv_text = iv_text
            iv_voiceid = iv_voice_id ).
        ENDIF.
        MESSAGE 'Speech synthesized successfully.' TYPE 'I'.
      CATCH /aws1/cx_plyinvalidssmlex.
        MESSAGE 'Invalid SSML.' TYPE 'E'.
      CATCH /aws1/cx_plylexiconnotfoundex.
        MESSAGE 'Lexicon not found.' TYPE 'E'.
      CATCH /aws1/cx_plyservicefailureex.
        MESSAGE 'Service failure occurred.' TYPE 'E'.
      CATCH /aws1/cx_plytextlengthexcdex.
        MESSAGE 'Text length exceeded maximum.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SynthesizeSpeech in der AWS SDK for SAP ABAP API-Referenz.

Amazon RDS-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname   = 'mydbparametergroup'
" iv_dbparametergroupfamily = 'mysql8.0'
" iv_description            = 'My custom DB parameter group for MySQL 8.0'
TRY.
    oo_result = lo_rds->createdbparametergroup(
      iv_dbparametergroupname   = iv_dbparametergroupname
      iv_dbparametergroupfamily = iv_dbparametergroupfamily
      iv_description            = iv_description ).
    MESSAGE 'DB parameter group created.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbparmgralrexfault.
    MESSAGE 'DB parameter group already exists.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrquotaexcd00.
    MESSAGE 'DB parameter group quota exceeded.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname = 'mydbparametergroup'
```

```
    TRY.
        lo_rds->deletedbparametergroup(
          iv_dbparametergroupname = iv_dbparametergroupname ).
        MESSAGE 'DB parameter group deleted.' TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
      CATCH /aws1/cx_rdsinvdbprmgrstatef00.
        MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_engine                 = 'mysql'
    " iv_dbparametergroupfamily = 'mysql8.0' (optional - filters by parameter group
family)
    TRY.
        oo_result = lo_rds->describedbengineversions(
          iv_engine                 = iv_engine
          iv_dbparametergroupfamily = iv_dbparametergroupfamily ).
        DATA(lv_version_count) = lines( oo_result->get_dbengineversions( ) ).
        MESSAGE |Retrieved { lv_version_count } engine versions.| TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname = 'mydbparametergroup'
TRY.
    oo_result = lo_rds->describedbparametergroups(
      iv_dbparametergroupname = iv_dbparametergroupname ).
    MESSAGE 'DB parameter group retrieved.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname = 'mydbparametergroup'
" iv_source               = 'user' (optional - filters by parameter source)
```

```
    TRY.
        oo_result = lo_rds->describedbparameters(
          iv_dbparametergroupname = iv_dbparametergroupname
          iv_source               = iv_source ).
        DATA(lv_param_count) = lines( oo_result->get_parameters( ) ).
        MESSAGE |Retrieved { lv_param_count } parameters.| TYPE 'I'.
      CATCH /aws1/cx_rdsdbprmgrnotfndfault.
        MESSAGE 'DB parameter group not found.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie unter DescribeDBParameters in der AWS SDK for SAP ABAP API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung. `DescribeOrderableDBInstanceOptions`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " iv_engine        = 'mysql'
    " iv_engineversion = '8.0.35'
    TRY.
        oo_result = lo_rds->descrorderabledbinstoptions(
          iv_engine        = iv_engine
          iv_engineversion = iv_engineversion ).
        DATA(lv_option_count) = lines( oo_result->get_orderabledbinstoptions( ) ).
        MESSAGE |Retrieved { lv_option_count } orderable DB instance options.| TYPE
 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK for SAP ABAP API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_dbparametergroupname = 'mydbparametergroup'
" it_parameters - table containing parameter objects with:
"   - parametername = 'max_connections'
"   - parametervalue = '100'
"   - applymethod = 'immediate' or 'pending-reboot'
TRY.
    oo_result = lo_rds->modifydbparametergroup(
      iv_dbparametergroupname = iv_dbparametergroupname
      it_parameters           = it_parameters ).
    MESSAGE 'DB parameter group modified.' TYPE 'I'.
  CATCH /aws1/cx_rdsdbprmgrnotfndfault.
    MESSAGE 'DB parameter group not found.' TYPE 'I'.
  CATCH /aws1/cx_rdsinvdbprmgrstatef00.
    MESSAGE 'DB parameter group is in an invalid state.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK for SAP ABAP API-Referenz.

Amazon Redshift-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Redshift Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateCluster`

Das folgende Codebeispiel zeigt die Verwendung. `CreateCluster`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie den -Cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        " Example values: iv_node_type = 'ra3.4xlarge'
        " Example values: iv_master_username = 'awsuser'
        " Example values: iv_master_password = 'AwsUser1000'
        " Example values: iv_publicly_accessible = abap_true
        " Example values: iv_number_of_nodes = 2
        oo_result = lo_rsh->createcluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_nodetype = iv_node_type
          iv_masterusername = iv_master_username
          iv_masteruserpassword = iv_master_password
          iv_publiclyaccessible = iv_publicly_accessible
          iv_numberofnodes = iv_number_of_nodes
        ).
        MESSAGE 'Redshift cluster created successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustalrdyexfault.
        MESSAGE 'Cluster already exists.' TYPE 'I'.
      CATCH /aws1/cx_rshclstquotaexcdfault.
```

```
        MESSAGE 'Cluster quota exceeded.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie CreateCluster in der AWS SDK for SAP ABAP API-Referenz.

`DeleteCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCluster`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Löschen Sie den Cluster.

```
    TRY.
        " Example values: iv_cluster_identifier = 'my-redshift-cluster'
        lo_rsh->deletecluster(
          iv_clusteridentifier = iv_cluster_identifier
          iv_skipfinalclustersnapshot = abap_true
        ).
        MESSAGE 'Redshift cluster deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rshclustnotfoundfault.
        MESSAGE 'Cluster not found.' TYPE 'I'.
      CATCH /aws1/cx_rshinvcluststatefault.
        MESSAGE 'Invalid cluster state for deletion.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteCluster in der AWS SDK for SAP ABAP API-Referenz.

`DescribeClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeClusters`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Beschreiben Sie den Cluster.

```
TRY.
    " Example values: iv_cluster_identifier = 'my-redshift-cluster' (optional)
    oo_result = lo_rsh->describeclusters(
      iv_clusteridentifier = iv_cluster_identifier
    ).
    lt_clusters = oo_result->get_clusters( ).
    lv_cluster_count = lines( lt_clusters ).
    MESSAGE |Retrieved { lv_cluster_count } cluster(s).| TYPE 'I'.
  CATCH /aws1/cx_rshclustnotfoundfault.
    MESSAGE 'Cluster not found.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie DescribeClusters in der AWS SDK for SAP ABAP API-Referenz.

`DescribeStatement`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStatement`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
    oo_result = lo_rsd->describestatement(
```

```
          iv_id = iv_statement_id
        ).
        lv_status = oo_result->get_status( ).
        MESSAGE |Statement status: { lv_status }| TYPE 'I'.
      CATCH /aws1/cx_rsdresourcenotfoundex.
        MESSAGE 'Statement not found.' TYPE 'I'.
      CATCH /aws1/cx_rsdinternalserverex.
        MESSAGE 'Internal server error.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeStatement in der AWS SDK for SAP ABAP API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
        " Example values: iv_database_name = 'dev'
        " Example values: iv_user_name = 'awsuser'
        " Example values: iv_sql = 'SELECT * FROM movies WHERE year = :year'
        " Example values: it_parameter_list - SQL parameters for parameterized
queries

        " Only pass parameters if the list is not empty
        IF it_parameter_list IS NOT INITIAL.
          oo_result = lo_rsd->executestatement(
            iv_clusteridentifier = iv_cluster_identifier
            iv_database = iv_database_name
            iv_dbuser = iv_user_name
            iv_sql = iv_sql
            it_parameters = it_parameter_list
          ).
```

```
      ELSE.
        oo_result = lo_rsd->executestatement(
          iv_clusteridentifier = iv_cluster_identifier
          iv_database = iv_database_name
          iv_dbuser = iv_user_name
          iv_sql = iv_sql
        ).
      ENDIF.

      lv_statement_id = oo_result->get_id( ).
      MESSAGE |Statement executed. ID: { lv_statement_id }| TYPE 'I'.
    CATCH /aws1/cx_rsdexecutestatementex.
      MESSAGE 'Statement execution error.' TYPE 'I'.
    CATCH /aws1/cx_rsdresourcenotfoundex.
      MESSAGE 'Resource not found.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie ExecuteStatement in der AWS SDK for SAP ABAP API-Referenz.

`GetStatementResult`

Das folgende Codebeispiel zeigt die Verwendung`GetStatementResult`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Überprüfen Sie das Ergebnis der Anweisung.

```
  TRY.
      " Example values: iv_statement_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
      " Handle pagination for large result sets

      DO.
        lo_result_page = lo_rsd->getstatementresult(
          iv_id = iv_statement_id
          iv_nexttoken = lv_next_token
```

```
        ).

        " Collect records from this page
        lt_page_records = lo_result_page->get_records( ).
        APPEND LINES OF lt_page_records TO lt_all_records.

        " Check if there are more pages
        lv_next_token = lo_result_page->get_nexttoken( ).
        IF lv_next_token IS INITIAL.
          EXIT. " No more pages
        ENDIF.
      ENDDO.

      " For the last call, set oo_result for return value
      oo_result = lo_result_page.
      lv_record_count = lines( lt_all_records ).
      MESSAGE |Retrieved { lv_record_count } record(s).| TYPE 'I'.
    CATCH /aws1/cx_rsdresourcenotfoundex.
      MESSAGE 'Statement not found or results not available.' TYPE 'I'.
    CATCH /aws1/cx_rsdinternalserverex.
      MESSAGE 'Internal server error.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie GetStatementResult in der AWS SDK for SAP ABAP API-Referenz.

`ListDatabases`

Das folgende Codebeispiel zeigt die Verwendung`ListDatabases`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Example values: iv_cluster_identifier = 'redshift-cluster-movies'
      " Example values: iv_database_name = 'dev'
      " Example values: iv_database_user = 'awsuser'
```

```
      oo_result = lo_rsd->listdatabases(
        iv_clusteridentifier = iv_cluster_identifier
        iv_database = iv_database_name
        iv_dbuser = iv_database_user
      ).
      lt_databases = oo_result->get_databases( ).
      lv_db_count = lines( lt_databases ).
      MESSAGE |Retrieved { lv_db_count } database(s).| TYPE 'I'.
    CATCH /aws1/cx_rsddatabaseconnex.
      MESSAGE 'Database connection error.' TYPE 'I'.
    CATCH /aws1/cx_rsdresourcenotfoundex.
      MESSAGE 'Cluster not found.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie ListDatabases in der AWS SDK for SAP ABAP API-Referenz.

`ModifyCluster`

Das folgende Codebeispiel zeigt die Verwendung`ModifyCluster`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Ändern Sie einen Cluster.

```
  TRY.
      " Example values: iv_cluster_identifier = 'my-redshift-cluster'
      " Example values: iv_pref_maintenance_wn = 'wed:07:30-wed:08:00'
      lo_rsh->modifycluster(
        iv_clusteridentifier = iv_cluster_identifier
        iv_preferredmaintenancewin00 = iv_pref_maintenance_wn
      ).
      MESSAGE 'Redshift cluster modified successfully.' TYPE 'I'.
    CATCH /aws1/cx_rshclustnotfoundfault.
      MESSAGE 'Cluster not found.' TYPE 'I'.
    CATCH /aws1/cx_rshinvcluststatefault.
```

```
        MESSAGE 'Invalid cluster state for modification.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ModifyCluster in der AWS SDK for SAP ABAP API-Referenz.

Amazon Rekognition-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Rekognition Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CompareFaces`

Das folgende Codebeispiel zeigt die Verwendung. `CompareFaces`

Weitere Informationen finden Sie unter Vergleich von Gesichtern in Bildern.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
      " Create S3 object reference for the source image
      DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
        iv_bucket = iv_source_s3_bucket
        iv_name = iv_source_s3_key ).

      " Create source image object
      DATA(lo_source_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_source_s3obj ).

      " Create S3 object reference for the target image
      DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
        iv_bucket = iv_target_s3_bucket
        iv_name = iv_target_s3_key ).

      " Create target image object
      DATA(lo_target_image) = NEW /aws1/cl_rekimage(
        io_s3object = lo_target_s3obj ).

      " Compare faces
      oo_result = lo_rek->comparefaces(
        io_sourceimage = lo_source_image
        io_targetimage = lo_target_image
        iv_similaritythreshold = iv_similarity ).

      DATA(lt_face_matches) = oo_result->get_facematches( ).
      DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

      " Get counts of matched and unmatched faces
      DATA(lv_matched_count) = lines( lt_face_matches ).
      DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).

      " Output detailed comparison results
      DATA(lv_message) = |Face comparison completed: | &&
                         |{ lv_matched_count } matched face(s), | &&
                         |{ lv_unmatched_count } unmatched face(s).|.
      MESSAGE lv_message TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie CompareFaces in der AWS SDK for SAP ABAP API-Referenz.

`CreateCollection`

Das folgende Codebeispiel zeigt die Verwendung`CreateCollection`.

Weitere Informationen finden Sie unter Erstellen einer Sammlung.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->createcollection(
      iv_collectionid = iv_collection_id ).
    MESSAGE 'Collection created successfully.' TYPE 'I'.
  CATCH /aws1/cx_rekresrcalrdyexistsex.
    MESSAGE 'Collection already exists.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateCollection in der AWS SDK for SAP ABAP API-Referenz.

`DeleteCollection`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCollection`.

Weitere Informationen finden Sie unter Löschen einer Sammlung.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_rek->deletecollection(
          iv_collectionid = iv_collection_id ).
        MESSAGE 'Collection deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteCollection in der AWS SDK for SAP ABAP API-Referenz.

`DeleteFaces`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFaces`.

Weitere Informationen finden Sie unter Löschen von Gesichtern aus einer Sammlung.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_rek->deletefaces(
          iv_collectionid = iv_collection_id
          it_faceids = it_face_ids ).

        DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
        DATA(lv_deleted_count) = lines( lt_deleted_faces ).
        DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
        MESSAGE lv_msg6 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteFaces in der AWS SDK for SAP ABAP API-Referenz.

`DescribeCollection`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCollection`.

Weitere Informationen finden Sie unter Beschreiben einer Sammlung.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->describecollection(
      iv_collectionid = iv_collection_id ).
    DATA(lv_face_count) = oo_result->get_facecount( ).
    DATA(lv_msg) = |Collection described: { lv_face_count } face(s) indexed.|.
    MESSAGE lv_msg TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DescribeCollection in der AWS SDK for SAP ABAP API-Referenz.

`DetectFaces`

Das folgende Codebeispiel zeigt die Verwendung`DetectFaces`.

Weitere Informationen finden Sie unter Erkennen von Gesichtern in einem Bild.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect faces in the image with all attributes
        DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
        DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
        INSERT lo_attr_wrapper INTO TABLE lt_attributes.

        oo_result = lo_rek->detectfaces(
          io_image = lo_image
          it_attributes = lt_attributes ).

        DATA(lt_face_details) = oo_result->get_facedetails( ).
        DATA(lv_detected_count) = lines( lt_face_details ).
        DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
        MESSAGE lv_msg8 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DetectFaces in der AWS SDK for SAP ABAP API-Referenz.

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect labels in the image
    oo_result = lo_rek->detectlabels(
      io_image = lo_image
      iv_maxlabels = iv_max_labels ).

    DATA(lt_labels) = oo_result->get_labels( ).
    DATA(lv_label_count) = lines( lt_labels ).
    DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
    MESSAGE lv_msg9 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetectLabels in der AWS SDK for SAP ABAP API-Referenz.

`DetectModerationLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectModerationLabels`.

Weitere Informationen finden Sie unter Erkennen von unangemessenen Bildern.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect moderation labels
    oo_result = lo_rek->detectmoderationlabels(
      io_image = lo_image ).

    DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
    DATA(lv_mod_count) = lines( lt_moderation_labels ).
    DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
    MESSAGE lv_msg10 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetectModerationLabels in der AWS SDK for SAP ABAP API-Referenz.

`DetectText`

Das folgende Codebeispiel zeigt die Verwendung`DetectText`.

Weitere Informationen finden Sie unter Erkennen von Text in einem Bild.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Detect text in the image
    oo_result = lo_rek->detecttext(
      io_image = lo_image ).

    DATA(lt_text_detections) = oo_result->get_textdetections( ).
    DATA(lv_text_count) = lines( lt_text_detections ).
    DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
    MESSAGE lv_msg11 TYPE 'I'.
  CATCH /aws1/cx_rekinvalids3objectex.
    MESSAGE 'Invalid S3 object.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DetectText in der AWS SDK for SAP ABAP API-Referenz.

`IndexFaces`

Das folgende Codebeispiel zeigt die Verwendung`IndexFaces`.

Weitere Informationen finden Sie unter Hinzufügen von Gesichtern zu einer Sammlung.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie IndexFaces in der AWS SDK for SAP ABAP API-Referenz.

`ListCollections`

Das folgende Codebeispiel zeigt die Verwendung`ListCollections`.

Weitere Informationen finden Sie unter Sammlungen auflisten.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListCollections in der AWS SDK for SAP ABAP API-Referenz.

`ListFaces`

Das folgende Codebeispiel zeigt die Verwendung`ListFaces`.

Weitere Informationen finden Sie unter Gesichter in einer Sammlung auflisten.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie ListFaces in der AWS SDK for SAP ABAP API-Referenz.

`RecognizeCelebrities`

Das folgende Codebeispiel zeigt die Verwendung`RecognizeCelebrities`.

Weitere Informationen finden Sie unter Erkennen von Prominenten in einem Bild.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Recognize celebrities
        oo_result = lo_rek->recognizecelebrities(
          io_image = lo_image ).

        DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
        DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
        DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
        MESSAGE lv_msg12 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie RecognizeCelebrities in der AWS SDK for SAP ABAP API-Referenz.

`SearchFaces`

Das folgende Codebeispiel zeigt die Verwendung`SearchFaces`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Gesichts-ID).

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_rek->searchfaces(
          iv_collectionid = iv_collection_id
          iv_faceid = iv_face_id
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count2) = lines( lt_face_matches ).
        DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
        MESSAGE lv_msg5 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection or face not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SearchFaces in der AWS SDK for SAP ABAP API-Referenz.

`SearchFacesByImage`

Das folgende Codebeispiel zeigt die Verwendung`SearchFacesByImage`.

Weitere Informationen finden Sie unter Nach einem Gesicht suchen (Bild).

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).
```

```
        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Search for matching faces
        oo_result = lo_rek->searchfacesbyimage(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_facematchthreshold = iv_threshold
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_matches) = oo_result->get_facematches( ).
        DATA(lv_match_count) = lines( lt_face_matches ).
        DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
        MESSAGE lv_msg4 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SearchFacesByImage in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon S3 unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
    DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

    " Create an Amazon Simple Storage Service (Amazon S3) bucket. "
    TRY.
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
```

```
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.


    "Upload an object to an S3 bucket."
    TRY.
        "Get contents of file from application server."
        DATA lv_file_content TYPE xstring.
        OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
        READ DATASET iv_key INTO lv_file_content.
        CLOSE DATASET iv_key.

        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
            iv_body = lv_file_content ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                   iv_bucket = iv_bucket_name
                   iv_key = iv_key ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
```

```
        lo_s3->copyobject(
          iv_bucket = iv_dest_bucket
          iv_key = iv_dest_object
          iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
        MESSAGE 'Object copied to another bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CopyObject in der AWS SDK for SAP ABAP API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " determine our region from our session
        DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
        DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
        " When in the us-east-1 region, you must not specify a constraint
        " In all other regions, specify the region as the constraint
        IF lv_region = 'us-east-1'.
          CLEAR lo_constraint.
        ELSE.
          lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
        ENDIF.

        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
```

```
            io_createbucketconfiguration  = lo_constraint ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateBucket in der AWS SDK for SAP ABAP API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteBucket in der AWS SDK for SAP ABAP API-Referenz.

`DeleteBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketCors`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_s3->deletebucketcors(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteBucketCors in der AWS SDK for SAP ABAP API-Referenz.

`DeleteBucketLifecycle`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketLifecycle`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_s3->deletebucketlifecycle(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteBucketLifecycle in der AWS SDK for SAP ABAP API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_s3->deletebucketpolicy(
      iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteBucketPolicy in der AWS SDK for SAP ABAP API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteObject in der AWS SDK for SAP ABAP API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->deleteobjects(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteObjects in der AWS SDK for SAP ABAP API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketacl(         " oo_result is returned for testing
purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetBucketAcl in der AWS SDK for SAP ABAP API-Referenz.

`GetBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketCors`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketcors(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetBucketCors in der AWS SDK for SAP ABAP API-Referenz.

`GetBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketLifecycleConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getbucketlifecycleconf(         " oo_result is returned
for testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetBucketLifecycleConfiguration in der AWS SDK for SAP ABAP API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->getbucketpolicy(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name ).
        DATA(lv_policy) = oo_result->get_policy( ).
        MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetBucketPolicy in der AWS SDK for SAP ABAP API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->getobject(           " oo_result is returned for testing
purposes. "
                  iv_bucket = iv_bucket_name
                  iv_key = iv_object_key ).
        DATA(lv_object_data) = oo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetObject in der AWS SDK for SAP ABAP API-Referenz.

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAcl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->getobjectacl(         " oo_result is returned for testing
purposes. "
         iv_bucket = iv_bucket_name
         iv_key = iv_object_key ).
       MESSAGE 'Retrieved object ACL.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie GetObjectAcl in der AWS SDK for SAP ABAP API-Referenz.

`GetObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLegalHold`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
```

```
        oo_result = lo_s3->getobjectlegalhold(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key ).
        MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetObjectLegalHold in der AWS SDK for SAP ABAP API-Referenz.

`GetObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectLockConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->getobjectlockconfiguration(         " oo_result is
returned for testing purposes. "
          iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetObjectLockConfiguration in der AWS SDK for SAP ABAP API-Referenz.

`HeadBucket`

Das folgende Codebeispiel zeigt die Verwendung`HeadBucket`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->headbucket(         " oo_result is returned for testing
purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie HeadBucket in der AWS SDK for SAP ABAP API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->listbuckets(         " oo_result is returned for testing
purposes. "
```

```
        ).
        DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
        MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list buckets.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListBuckets in der AWS SDK for SAP ABAP API-Referenz.

`ListObjectVersions`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectVersions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3->listobjectversions(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
          iv_prefix = iv_prefix ).
        MESSAGE 'Retrieved object versions.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListObjectVersions in der AWS SDK for SAP ABAP API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name ).
       MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK for SAP ABAP API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       " Example: Grant log delivery access to a bucket
       " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
       lo_s3->putbucketacl(
         iv_bucket = iv_bucket_name
         iv_grantwrite = iv_grantwrite ).
       MESSAGE 'Bucket ACL updated.' TYPE 'I'.
```

```
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie PutBucketAcl in der AWS SDK for SAP ABAP API-Referenz.

`PutBucketCors`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketCors`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Example: Allow PUT, POST, DELETE methods from http://www.example.com
      lo_s3->putbucketcors(
        iv_bucket = iv_bucket_name
        io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
          it_corsrules = it_cors_rules ) ).
      MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie PutBucketCors in der AWS SDK for SAP ABAP API-Referenz.

`PutBucketLifecycleConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketLifecycleConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Expire objects with prefix 'logs/' after 30 days
        lo_s3->putbucketlifecycleconf(
          iv_bucket = iv_bucket_name
          io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
        MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutBucketLifecycleConfiguration in der AWS SDK for SAP ABAP API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketPolicy`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example policy JSON string
        " iv_policy = '{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
```

```
      lo_s3->putbucketpolicy(
        iv_bucket = iv_bucket_name
        iv_policy = iv_policy ).
      MESSAGE 'Bucket policy set.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie PutBucketPolicy in der AWS SDK for SAP ABAP API-Referenz.

`PutBucketVersioning`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketVersioning`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Example: Enable versioning on a bucket
      " iv_status = 'Enabled'
      lo_s3->putbucketversioning(
        iv_bucket = iv_bucket_name
        io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(
          iv_status = iv_status ) ).
      MESSAGE 'Bucket versioning enabled.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
      MESSAGE 'Bucket does not exist.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie PutBucketVersioning in der AWS SDK for SAP ABAP API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie PutObject in der AWS SDK for SAP ABAP API-Referenz.

`PutObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectAcl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Grant read access to an AWS user
        " iv_grantread = 'emailAddress=user@example.com'
        lo_s3->putobjectacl(
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          iv_grantread = iv_grantread ).
        MESSAGE 'Object ACL updated.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutObjectAcl in der AWS SDK for SAP ABAP API-Referenz.

`PutObjectLegalHold`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLegalHold`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Set legal hold status to ON
        " iv_status = 'ON'
        lo_s3->putobjectlegalhold(
          iv_bucket = iv_bucket_name
          iv_key = iv_object_key
          io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
            iv_status = iv_status ) ).
        MESSAGE 'Object legal hold status set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
```

```
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutObjectLegalHold in der AWS SDK for SAP ABAP API-Referenz.

`PutObjectLockConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectLockConfiguration`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Example: Enable object lock with default retention
        " iv_enabled = 'Enabled'
        lo_s3->putobjectlockconfiguration(
          iv_bucket = iv_bucket_name
          io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(
            iv_objectlockenabled = iv_enabled ) ).
        MESSAGE 'Object lock configuration set.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie PutObjectLockConfiguration in der AWS SDK for SAP ABAP API-Referenz.

`PutObjectRetention`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectRetention`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " Example: Set retention mode to GOVERNANCE for 30 days
    " iv_mode = 'GOVERNANCE'
    " iv_retain_date should be a timestamp in the future
    lo_s3->putobjectretention(
      iv_bucket = iv_bucket_name
      iv_key = iv_object_key
      io_retention = NEW /aws1/cl_s3_objectlockret(
        iv_mode = iv_mode
        iv_retainuntildate = iv_retain_date )
      iv_bypassgovernanceretention = abap_true ).
    MESSAGE 'Object retention set.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie PutObjectRetention in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie vorsignierte Anforderungen für GET-S3-Objekte.

```
" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service.  It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
  iv_bucket = iv_bucket_name
  iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file.  The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```

Beispiele für Amazon S3 Control mit SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon S3 Control Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    " iv_manifest_arn  = 'arn:aws:s3:::my-bucket/job-manifest.csv'
    " iv_manifest_etag = 'abc123def456'
    " iv_report_bucket = 'arn:aws:s3:::my-report-bucket'
    DATA(lo_result) = lo_s3c->createjob(
      iv_accountid            = iv_account_id
      iv_rolearn              = iv_role_arn
      iv_confirmationrequired = abap_true
      iv_priority             = 10
      iv_description          = 'Batch job for tagging objects'
      io_operation            = NEW /aws1/cl_s3cjoboperation(
        io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
```

```
              it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                ( NEW /aws1/cl_s3cs3tag(
                    iv_key   = 'BatchTag'
                    iv_value = 'BatchValue' ) )
              )
            )
          )
          io_manifest             = NEW /aws1/cl_s3cjobmanifest(
            io_spec     = NEW /aws1/cl_s3cjobmanifestspec(
              iv_format = 'S3BatchOperations_CSV_20180820'
              it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
                ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
              )
            )
            io_location = NEW /aws1/cl_s3cjobmanifestloc(
              iv_objectarn = iv_manifest_arn
              iv_etag      = iv_manifest_etag
            )
          )
          io_report               = NEW /aws1/cl_s3cjobreport(
            iv_bucket      = iv_report_bucket
            iv_format      = 'Report_CSV_20180820'
            iv_enabled     = abap_true
            iv_prefix      = 'batch-op-reports'
            iv_reportscope = 'AllTasks'
          )
        ).
        ov_job_id = lo_result->get_jobid( ).
        MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie CreateJob in der AWS SDK for SAP ABAP API-Referenz.

`DeleteJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJobTagging`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_s3c->deletejobtagging(
      iv_accountid = iv_account_id
      iv_jobid     = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
  CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_nf.
  CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_cli.
  CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
      EXPORTING previous = lo_ex_srv.
ENDTRY.
```

- API-Details finden Sie DeleteJobTagging in der AWS SDK for SAP ABAP API-Referenz.

`DescribeJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_s3c->describejob(         " oo_result is returned for testing
purposes.
         iv_accountid = iv_account_id
         iv_jobid     = iv_job_id
       ).
       DATA(lo_job) = oo_result->get_job( ).
       DATA(lv_status) = lo_job->get_status( ).
       DATA(lv_priority) = lo_job->get_priority( ).
       DATA(lo_progress) = lo_job->get_progresssummary( ).
       IF lo_progress IS NOT INITIAL.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                 |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                 |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
                 |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
       ELSE.
         MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
       ENDIF.
     CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
       MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_nf.
     CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
       MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
         EXPORTING previous = lo_ex_cli.
     CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
       MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
       RAISE EXCEPTION TYPE /aws1/cx_rt_generic
```

```
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie DescribeJob in der AWS SDK for SAP ABAP API-Referenz.

`GetJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`GetJobTagging`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3c->getjobtagging(       " oo_result is returned for testing
purposes.
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
        ).
        DATA(lt_tags) = oo_result->get_tags( ).
        MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }| TYPE
'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie GetJobTagging in der AWS SDK for SAP ABAP API-Referenz.

`PutJobTagging`

Das folgende Codebeispiel zeigt die Verwendung`PutJobTagging`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_s3c->putjobtagging(
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          it_tags      = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Environment'
                iv_value = 'Development' ) )
            ( NEW /aws1/cl_s3cs3tag(
                iv_key   = 'Team'
                iv_value = 'DataProcessing' ) )
          )
        ).
        MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
        MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_tags.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie PutJobTagging in der AWS SDK for SAP ABAP API-Referenz.

`UpdateJobPriority`

Das folgende Codebeispiel zeigt die Verwendung`UpdateJobPriority`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_s3c->updatejobpriority(   " oo_result is returned for testing
 purposes.
          iv_accountid = iv_account_id
          iv_jobid     = iv_job_id
          iv_priority  = 60
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
        MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_bad.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
```

```
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie UpdateJobPriority in der AWS SDK for SAP ABAP API-Referenz.

`UpdateJobStatus`

Das folgende Codebeispiel zeigt die Verwendung`UpdateJobStatus`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_requested_status = 'Cancelled'
        oo_result = lo_s3c->updatejobstatus(     " oo_result is returned for testing
 purposes.
          iv_accountid          = iv_account_id
          iv_jobid              = iv_job_id
          iv_requestedjobstatus = iv_requested_status
        ).
        MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
      CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
        MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_js.
      CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_nf.
      CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_cli.
      CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
```

```
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
          EXPORTING previous = lo_ex_srv.
    ENDTRY.
```

- API-Details finden Sie UpdateJobStatus in der AWS SDK for SAP ABAP API-Referenz.

SageMaker KI-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit SageMaker KI Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`CreateEndpoint`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.

    "Create a production variant as an ABAP object."
    "Identifies a model that you want to host and the resources chosen to deploy for
 hosting it."
    lo_production_variants = NEW #( iv_variantname = iv_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
 iv_initial_instance_count
                                    iv_instancetype = iv_instance_type ).

    INSERT lo_production_variants INTO TABLE lt_production_variants.

    "Create an endpoint configuration."
    TRY.
        oo_ep_config_result = lo_sgm->createendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name
          it_productionvariants = lt_production_variants ).
        MESSAGE 'Endpoint configuration created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.

    "Create an endpoint."
    TRY.
        oo_result = lo_sgm->createendpoint(     " oo_result is returned for testing
 purposes. "
            iv_endpointconfigname = iv_endpoint_config_name
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateEndpoint in der AWS SDK for SAP ABAP API-Referenz.

`CreateModel`

Das folgende Codebeispiel zeigt die Verwendung`CreateModel`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

   "Create an ABAP object for the container image based on input variables."
   lo_primarycontainer = NEW #( iv_image = iv_container_image
                                iv_modeldataurl = iv_model_data_url ).

   "Create an Amazon SageMaker model."
   TRY.
       oo_result = lo_sgm->createmodel(        " oo_result is returned for testing
purposes. "
         iv_executionrolearn = iv_execution_role_arn
         iv_modelname = iv_model_name
         io_primarycontainer = lo_primarycontainer ).
       MESSAGE 'Model created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreateModel in der AWS SDK for SAP ABAP API-Referenz.

`CreateTrainingJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateTrainingJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
   DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
   DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
   DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
   DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
   DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
   DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
   DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
   DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
   DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
   DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
   DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
   DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.

   "Create ABAP internal table for hyperparameters based on input variables."
   "These hyperparameters are based on the Amazon SageMaker built-in algorithm,
XGBoost."
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
   INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
   INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
   INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
   INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
TABLE lt_hyperparameters.
```

```
   lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
   INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
   INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
   INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
lt_hyperparameters.

   "Create ABAP objects for training data sources."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).

   INSERT lo_trn_channel INTO TABLE lt_input_data_config.

   "Create ABAP objects for validation data sources."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).

   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
```

```
                                          iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).

   "Create a training job."
   TRY.
       oo_result = lo_sgm->createtrainingjob(    " oo_result is returned for
testing purposes. "
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreateTrainingJob in der AWS SDK for SAP ABAP API-Referenz.

`CreateTransformJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateTransformJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
   DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
   DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
   DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
   DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

   "Create an ABAP object for an Amazon Simple Storage Service (Amazon S3) data
source."
   lo_s3datasource = NEW #( iv_s3uri = iv_tf_data_s3uri
                            iv_s3datatype = iv_tf_data_s3datatype ).

   "Create an ABAP object for data source."
   lo_datasource = NEW #( io_s3datasource = lo_s3datasource ).

   "Create an ABAP object for transform data source."
   lo_transforminput = NEW #( io_datasource = lo_datasource
                              iv_contenttype = iv_tf_data_contenttype
                              iv_compressiontype = iv_tf_data_compressiontype ).

   "Create an ABAP object for resource configuration."
   lo_transformresources = NEW #( iv_instancecount = iv_instance_count
                                  iv_instancetype = iv_instance_type ).

   "Create an ABAP object for output data configuration."
   lo_transformoutput = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create a transform job."
   TRY.
       oo_result = lo_sgm->createtransformjob(     " oo_result is returned for
testing purposes. "
           iv_modelname = iv_tf_model_name
           iv_transformjobname = iv_tf_job_name
           io_transforminput = lo_transforminput
```

```
            io_transformoutput = lo_transformoutput
            io_transformresources = lo_transformresources ).
        MESSAGE 'Transform job created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourceinuse.
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateTransformJob in der AWS SDK for SAP ABAP API-Referenz.

`DeleteEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEndpoint`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Delete an endpoint."
    TRY.
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_endpoint_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
        DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
{ lo_endpoint_exception->av_err_msg }|.
        MESSAGE lv_endpoint_error TYPE 'E'.
    ENDTRY.

    "Delete an endpoint configuration."
    TRY.
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_endpoint_config_name ).
```

```
        MESSAGE 'Endpoint deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteEndpoint in der AWS SDK for SAP ABAP API-Referenz.

`DeleteModel`

Das folgende Codebeispiel zeigt die Verwendung`DeleteModel`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteModel in der AWS SDK for SAP ABAP API-Referenz.

`DescribeTrainingJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTrainingJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->describetrainingjob(      " oo_result is returned for
 testing purposes. "
          iv_trainingjobname = iv_training_job_name ).
        MESSAGE 'Retrieved description of training job.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DescribeTrainingJob in der AWS SDK for SAP ABAP API-Referenz.

`ListAlgorithms`

Das folgende Codebeispiel zeigt die Verwendung`ListAlgorithms`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listalgorithms(         " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of algorithms.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListAlgorithms in der AWS SDK for SAP ABAP API-Referenz.

`ListModels`

Das folgende Codebeispiel zeigt die Verwendung`ListModels`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listmodels(           " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of models.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListModels in der AWS SDK for SAP ABAP API-Referenz.

`ListNotebookInstances`

Das folgende Codebeispiel zeigt die Verwendung`ListNotebookInstances`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listnotebookinstances(        " oo_result is returned
 for testing purposes. "
          iv_namecontains = iv_name_contains ).
        MESSAGE 'Retrieved list of notebook instances.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListNotebookInstances in der AWS SDK for SAP ABAP API-Referenz.

`ListTrainingJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTrainingJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sgm->listtrainingjobs(       " oo_result is returned for
 testing purposes. "
          iv_namecontains = iv_name_contains
          iv_maxresults = iv_max_results ).
        MESSAGE 'Retrieved list of training jobs.' TYPE 'I'.
```

```
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListTrainingJobs in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Erste Schritte mit Modellen und Endpunkten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen Trainingsjob und erstellen Sie ein SageMaker KI-Modell.
- Eine Endpunktkonfiguration erstellen.
- Erstellen Sie einen Endpunkt und bereinigen Sie dann die Ressourcen.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
    DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
    DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
    DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
    DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
    DATA lo_resource_config  TYPE REF TO /aws1/cl_sgmresourceconfig.
    DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
    DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
    DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
```

```
    DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
    DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
    DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescrtrnjobrsp.
    DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
    DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
    DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
    DATA lv_model_data_url TYPE /aws1/sgmurl.

    lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

    "Create ABAP internal table for hyperparameters based on input variables."
    "These hyperparameters are based on Amazon SageMaker built-in algorithm -
 XGBoost"
    lo_hyperparameters_w = NEW #( iv_value = iv_hp_max_depth ).
    INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_eta ).
    INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_eval_metric ).
    INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_scale_pos_weight ).
    INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w )  INTO
 TABLE lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_subsample ).
    INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_objective ).
    INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    lo_hyperparameters_w = NEW #( iv_value = iv_hp_num_round ).
    INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w )  INTO TABLE
 lt_hyperparameters.

    "Create ABAP internal table for data based on input variables."
```

```
   "Training data."
   lo_trn_s3datasource = NEW #( iv_s3datatype = iv_trn_data_s3datatype
                                iv_s3datadistributiontype =
iv_trn_data_s3datadistribution
                                iv_s3uri = iv_trn_data_s3uri ).

   lo_trn_datasource = NEW #( io_s3datasource = lo_trn_s3datasource ).

   lo_trn_channel = NEW #( iv_channelname = 'train'
                           io_datasource = lo_trn_datasource
                           iv_compressiontype = iv_trn_data_compressiontype
                           iv_contenttype = iv_trn_data_contenttype ).
   INSERT lo_trn_channel INTO TABLE lt_input_data_config.

   "Validation data."
   lo_val_s3datasource = NEW #( iv_s3datatype = iv_val_data_s3datatype
                                iv_s3datadistributiontype =
iv_val_data_s3datadistribution
                                iv_s3uri = iv_val_data_s3uri ).

   lo_val_datasource = NEW #( io_s3datasource = lo_val_s3datasource ).

   lo_val_channel = NEW #( iv_channelname = 'validation'
                           io_datasource = lo_val_datasource
                           iv_compressiontype = iv_val_data_compressiontype
                           iv_contenttype = iv_val_data_contenttype ).
   INSERT lo_val_channel INTO TABLE lt_input_data_config.

   "Create an ABAP object for algorithm specification based on input variables."
   lo_algorithm_specification = NEW #( iv_trainingimage = iv_training_image
                                       iv_traininginputmode =
iv_training_input_mode ).

   "Create an ABAP object for resource configuration."
   lo_resource_config = NEW #( iv_instancecount = iv_instance_count
                               iv_instancetype = iv_instance_type
                               iv_volumesizeingb = iv_volume_sizeingb ).

   "Create an ABAP object for output data configuration."
   lo_output_data_config = NEW #( iv_s3outputpath = iv_s3_output_path ).

   "Create an ABAP object for stopping condition."
   lo_stopping_condition = NEW #( iv_maxruntimeinseconds =
iv_max_runtime_in_seconds ).
```

```
   TRY.
       lo_sgm->createtrainingjob(
         iv_trainingjobname           = iv_training_job_name
         iv_rolearn                   = iv_role_arn
         it_hyperparameters           = lt_hyperparameters
         it_inputdataconfig           = lt_input_data_config
         io_algorithmspecification    = lo_algorithm_specification
         io_outputdataconfig          = lo_output_data_config
         io_resourceconfig            = lo_resource_config
         io_stoppingcondition         = lo_stopping_condition ).
       MESSAGE 'Training job created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourceinuse.
       MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcenotfound.
       MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
       MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
   ENDTRY.

   "Wait for training job to be completed."
   lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
   WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
     IF sy-index = 30.
       EXIT.               "Maximum 900 seconds."
     ENDIF.
     WAIT UP TO 30 SECONDS.
     lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
   ENDWHILE.

   "Create ABAP object for the container image based on input variables."
   lo_primarycontainer = NEW #( iv_image = iv_training_image
                                iv_modeldataurl = lv_model_data_url ).

   "Create an Amazon SageMaker model."
   TRY.
       lo_sgm->createmodel(
         iv_executionrolearn = iv_role_arn
         iv_modelname = iv_model_name
         io_primarycontainer = lo_primarycontainer ).
       MESSAGE 'Model created.' TYPE 'I'.
     CATCH /aws1/cx_sgmresourcelimitexcd.
```

```
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.

    "Create an endpoint production variant."
    lo_production_variants = NEW #( iv_variantname = iv_ep_variant_name
                                    iv_modelname = iv_model_name
                                    iv_initialinstancecount =
iv_ep_initial_instance_count
                                    iv_instancetype = iv_ep_instance_type ).
    INSERT lo_production_variants INTO TABLE lt_production_variants.

    TRY.
        "Create an endpoint configuration."
        lo_ep_config_result = lo_sgm->createendpointconfig(
          iv_endpointconfigname = iv_ep_cfg_name
          it_productionvariants = lt_production_variants ).
        MESSAGE 'Endpoint configuration created.' TYPE 'I'.

        "Create an endpoint."
        oo_ep_output = lo_sgm->createendpoint(        " oo_ep_output is returned for
testing purposes. "
            iv_endpointconfigname = iv_ep_cfg_name
            iv_endpointname = iv_ep_name ).
        MESSAGE 'Endpoint created.' TYPE 'I'.
      CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources.' TYPE 'E'.
    ENDTRY.

    "Wait for endpoint creation to be completed."
    DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
    WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname = iv_ep_name ).
    ENDWHILE.

    TRY.
        "Delete an endpoint."
        lo_sgm->deleteendpoint(
            iv_endpointname = iv_ep_name ).
        MESSAGE 'Endpoint deleted' TYPE 'I'.
```

```
        "Delete an endpoint configuration."
        lo_sgm->deleteendpointconfig(
          iv_endpointconfigname = iv_ep_cfg_name ).
        MESSAGE 'Endpoint configuration deleted.' TYPE 'I'.

        "Delete model."
        lo_sgm->deletemodel(
                  iv_modelname = iv_model_name ).
        MESSAGE 'Model deleted.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - CreateEndpoint
  - CreateEndpointConfig
  - CreateModel
  - CreateTrainingJob
  - DeleteEndpoint
  - DeleteEndpointConfig
  - DeleteModel
  - DescribeEndpoint
  - DescribeTrainingJob

Secrets Manager-Beispiele, die SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Secrets Manager Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`BatchGetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetSecretValue`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " iv_filter_name = 'mySecret'
        DATA(lo_result) = lo_smr->batchgetsecretvalue(
          it_filters = VALUE /aws1/cl_smrfilter=>tt_filterslisttype(
            (
              NEW /aws1/cl_smrfilter(
                iv_key = 'name'
                it_values = VALUE /aws1/
cl_smrfiltvalsstrlist_w=>tt_filtervaluesstringlist(
                  ( NEW /aws1/cl_smrfiltvalsstrlist_w( iv_value = iv_filter_name ) )
                )
              )
            )
          )
        ).
        ot_secret_values = lo_result->get_secretvalues( ).
        MESSAGE 'Secrets retrieved successfully.' TYPE 'I'.
      CATCH /aws1/cx_smrresourcenotfoundex.
        MESSAGE 'One or more requested secrets were not found.' TYPE 'E'.
      CATCH /aws1/cx_smrdecryptionfailure.
        MESSAGE 'Failed to decrypt one or more secrets.' TYPE 'E'.
```

```
    CATCH /aws1/cx_smrinvalidparameterex.
      MESSAGE 'Invalid parameter provided.' TYPE 'E'.
    CATCH /aws1/cx_smrinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie BatchGetSecretValue in der AWS SDK for SAP ABAP API-Referenz.

`GetSecretValue`

Das folgende Codebeispiel zeigt die Verwendung`GetSecretValue`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " iv_secret_name = 'MySecretName'
      DATA(lo_result) = lo_smr->getsecretvalue( iv_secretid = iv_secret_name ).
      ov_secret_value = lo_result->get_secretstring( ).
      MESSAGE 'Secret value retrieved successfully.' TYPE 'I'.
    CATCH /aws1/cx_smrresourcenotfoundex.
      MESSAGE 'The requested secret was not found.' TYPE 'E'.
    CATCH /aws1/cx_smrdecryptionfailure.
      MESSAGE 'Failed to decrypt the secret.' TYPE 'E'.
    CATCH /aws1/cx_smrinvalidparameterex.
      MESSAGE 'Invalid parameter provided.' TYPE 'E'.
    CATCH /aws1/cx_smrinvalidrequestex.
      MESSAGE 'Invalid request.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie GetSecretValue in der AWS SDK for SAP ABAP API-Referenz.

Amazon SES-Beispiele, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon SES Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptFilter`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" iv_allow = abap_true means 'Allow', abap_false means 'Block'
DATA(lv_policy) = COND /aws1/sesreceiptfilterpolicy(
  WHEN iv_allow = abap_true THEN 'Allow'
  ELSE 'Block'
).

DATA(lo_ip_filter) = NEW /aws1/cl_sesreceiptipfilter(
  iv_policy = lv_policy
  iv_cidr = iv_ip_address_or_range
).

DATA(lo_filter) = NEW /aws1/cl_sesreceiptfilter(
```

```
      iv_name = iv_filter_name
      io_ipfilter = lo_ip_filter
    ).

    TRY.
        lo_ses->createreceiptfilter( io_filter = lo_filter ).
        MESSAGE 'Receipt filter created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Filter already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- API-Details finden Sie CreateReceiptFilter in der AWS SDK for SAP ABAP API-Referenz.

`CreateReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    " Create S3 action for copying emails to S3
    DATA(lo_s3_action) = NEW /aws1/cl_sess3action(
      iv_bucketname = iv_bucket_name
      iv_objectkeyprefix = iv_prefix
    ).

    " Create receipt action with S3 action
    DATA(lo_action) = NEW /aws1/cl_sesreceiptaction(
      io_s3action = lo_s3_action
    ).
```

```
" Create list of actions
DATA lt_actions TYPE /aws1/cl_sesreceiptaction=>tt_receiptactionslist.
APPEND lo_action TO lt_actions.

" Create receipt rule
DATA(lo_rule) = NEW /aws1/cl_sesreceiptrule(
  iv_name = iv_rule_name
  iv_enabled = abap_true
  it_recipients = it_recipients
  it_actions = lt_actions
).

TRY.
    lo_ses->createreceiptrule(
      iv_rulesetname = iv_rule_set_name
      io_rule = lo_rule
    ).
    MESSAGE 'Receipt rule created successfully' TYPE 'I'.
  CATCH /aws1/cx_sesinvalids3confex INTO DATA(lo_ex1).
    DATA(lv_error) = |Invalid S3 configuration: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- API-Details finden Sie CreateReceiptRule in der AWS SDK for SAP ABAP API-Referenz.

`CreateReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRuleSet`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ses->createreceiptruleset( iv_rulesetname = iv_rule_set_name ).
        MESSAGE 'Receipt rule set created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Rule set already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- API-Details finden Sie CreateReceiptRuleSet in der AWS SDK for SAP ABAP API-Referenz.

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA(lo_template) = NEW /aws1/cl_sestemplate(
      iv_templatename = iv_name
      iv_subjectpart = iv_subject
      iv_textpart = iv_text
      iv_htmlpart = iv_html
    ).

    TRY.
        lo_ses->createtemplate( io_template = lo_template ).
        MESSAGE 'Template created successfully' TYPE 'I'.
      CATCH /aws1/cx_sesalreadyexistsex INTO DATA(lo_ex1).
        DATA(lv_error) = |Template already exists: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
```

```
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
      lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex2.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie CreateTemplate in der AWS SDK for SAP ABAP API-Referenz.

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_ses->deleteidentity( iv_identity = iv_identity ).
      MESSAGE 'Identity deleted successfully' TYPE 'I'.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
      DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- API-Details finden Sie DeleteIdentity in der AWS SDK for SAP ABAP API-Referenz.

`DeleteReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptFilter`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletereceiptfilter( iv_filtername = iv_filter_name ).
    MESSAGE 'Receipt filter deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- API-Details finden Sie DeleteReceiptFilter in der AWS SDK for SAP ABAP API-Referenz.

`DeleteReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRule`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletereceiptrule(
      iv_rulesetname = iv_rule_set_name
      iv_rulename = iv_rule_name
    ).
    MESSAGE 'Receipt rule deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
```

```
      DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie DeleteReceiptRule in der AWS SDK for SAP ABAP API-Referenz.

`DeleteReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRuleSet`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_ses->deletereceiptruleset( iv_rulesetname = iv_rule_set_name ).
      MESSAGE 'Receipt rule set deleted successfully' TYPE 'I'.
    CATCH /aws1/cx_sescannotdeleteex INTO DATA(lo_ex1).
      DATA(lv_error) = |Cannot delete rule set: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie DeleteReceiptRuleSet in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ses->deletetemplate( iv_templatename = iv_template_name ).
    MESSAGE 'Template deleted successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- API-Details finden Sie DeleteTemplate in der AWS SDK for SAP ABAP API-Referenz.

`DescribeReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`DescribeReceiptRuleSet`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ses->describereceiptruleset(
      iv_rulesetname = iv_rule_set_name
    ).
```

```
        MESSAGE 'Receipt rule set described successfully' TYPE 'I'.
      CATCH /aws1/cx_sesrulesetdoesnotexex INTO DATA(lo_ex1).
        DATA(lv_error) = |Rule set does not exist: { lo_ex1->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex1.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
        lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex_generic.
    ENDTRY.
```

- API-Details finden Sie DescribeReceiptRuleSet in der AWS SDK for SAP ABAP API-Referenz.

`GetIdentityVerificationAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetIdentityVerificationAttributes`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   DATA lt_identities TYPE /aws1/cl_sesidentitylist_w=>tt_identitylist.
   APPEND NEW /aws1/cl_sesidentitylist_w( iv_value = iv_identity ) TO
lt_identities.

   TRY.
       DATA(lo_result) = lo_ses->getidentityverificationattrs(
         it_identities = lt_identities
       ).

       DATA(lt_attrs) = lo_result->get_verificationattributes( ).
       IF lt_attrs IS NOT INITIAL.
         LOOP AT lt_attrs ASSIGNING FIELD-SYMBOL(<ls_attr>).
           ov_status = <ls_attr>-value->get_verificationstatus( ).
           EXIT.
         ENDLOOP.
       ELSE.
```

```
        ov_status = 'NotFound'.
      ENDIF.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
      DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex.
  ENDTRY.
```

- API-Details finden Sie GetIdentityVerificationAttributes in der AWS SDK for SAP ABAP API-Referenz.

`GetTemplate`

Das folgende Codebeispiel zeigt die Verwendung`GetTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      DATA(lo_result) = lo_ses->gettemplate( iv_templatename = iv_template_name ).
      oo_template = lo_result->get_template( ).
      MESSAGE 'Template retrieved successfully' TYPE 'I'.
    CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
      DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie GetTemplate in der AWS SDK for SAP ABAP API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_ses->listidentities(
      iv_identitytype = iv_identity_type
      iv_maxitems = iv_max_items
    ).
    ot_identities = lo_result->get_identities( ).
    MESSAGE 'Identities retrieved successfully' TYPE 'I'.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
    DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex.
ENDTRY.
```

- API-Details finden Sie ListIdentities in der AWS SDK for SAP ABAP API-Referenz.

`ListReceiptFilters`

Das folgende Codebeispiel zeigt die Verwendung`ListReceiptFilters`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ses->listreceiptfilters( ).
        ot_filters = lo_result->get_filters( ).
        MESSAGE 'Receipt filters retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie ListReceiptFilters in der AWS SDK for SAP ABAP API-Referenz.

`ListTemplates`

Das folgende Codebeispiel zeigt die Verwendung`ListTemplates`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_ses->listtemplates( iv_maxitems = iv_max_items ).
        ot_templates = lo_result->get_templatesmetadata( ).
        MESSAGE 'Templates retrieved successfully' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie ListTemplates in der AWS SDK for SAP ABAP API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
" Create message object
DATA(lo_subject) = NEW /aws1/cl_sescontent( iv_data = iv_subject ).
DATA(lo_text_body) = NEW /aws1/cl_sescontent( iv_data = iv_text ).
DATA(lo_html_body) = NEW /aws1/cl_sescontent( iv_data = iv_html ).
DATA(lo_body) = NEW /aws1/cl_sesbody(
  io_text = lo_text_body
  io_html = lo_html_body
).
DATA(lo_message) = NEW /aws1/cl_sesmessage(
  io_subject = lo_subject
  io_body = lo_body
).

TRY.
    " Send email
    DATA(lo_result) = lo_ses->sendemail(
      iv_source = iv_source
      io_destination = io_destination
      io_message = lo_message
      it_replytoaddresses = it_reply_tos
    ).
    ov_msg_id = lo_result->get_messageid( ).
    MESSAGE 'Email sent successfully' TYPE 'I'.
  CATCH /aws1/cx_sesacctsendingpause00 INTO DATA(lo_ex1).
    DATA(lv_error) = |Account sending paused: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesmessagerejected INTO DATA(lo_ex2).
    lv_error = |Message rejected: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
```

```
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie SendEmail in der AWS SDK for SAP ABAP API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      " Send templated email
      DATA(lo_result) = lo_ses->sendtemplatedemail(
        iv_source = iv_source
        io_destination = io_destination
        iv_template = iv_template_name
        iv_templatedata = iv_template_data
        it_replytoaddresses = it_reply_tos
      ).
      ov_msg_id = lo_result->get_messageid( ).
      MESSAGE 'Templated email sent successfully' TYPE 'I'.
    CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
      DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex1.
    CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
      lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
      MESSAGE lv_error TYPE 'I'.
      RAISE EXCEPTION lo_ex_generic.
  ENDTRY.
```

- API-Details finden Sie SendTemplatedEmail in der AWS SDK for SAP ABAP API-Referenz.

`UpdateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
DATA(lo_template) = NEW /aws1/cl_sestemplate(
  iv_templatename = iv_name
  iv_subjectpart = iv_subject
  iv_textpart = iv_text
  iv_htmlpart = iv_html
).

TRY.
    lo_ses->updatetemplate( io_template = lo_template ).
    MESSAGE 'Template updated successfully' TYPE 'I'.
  CATCH /aws1/cx_sestmpldoesnotexistex INTO DATA(lo_ex1).
    DATA(lv_error) = |Template does not exist: { lo_ex1->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex1.
  CATCH /aws1/cx_sesinvalidtemplateex INTO DATA(lo_ex2).
    lv_error = |Invalid template: { lo_ex2->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex2.
  CATCH /aws1/cx_rt_generic INTO DATA(lo_ex_generic).
    lv_error = |An error occurred: { lo_ex_generic->get_text( ) }|.
    MESSAGE lv_error TYPE 'I'.
    RAISE EXCEPTION lo_ex_generic.
ENDTRY.
```

- API-Details finden Sie UpdateTemplate in der AWS SDK for SAP ABAP API-Referenz.

`VerifyDomainIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyDomainIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       DATA(lo_result) = lo_ses->verifydomainidentity( iv_domain =
iv_domain_name ).
       ov_token = lo_result->get_verificationtoken( ).
       MESSAGE 'Domain verification initiated' TYPE 'I'.
     CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
       DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
       MESSAGE lv_error TYPE 'I'.
       RAISE EXCEPTION lo_ex.
   ENDTRY.
```

- API-Details finden Sie VerifyDomainIdentity in der AWS SDK for SAP ABAP API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ses->verifyemailidentity( iv_emailaddress = iv_email_address ).
        MESSAGE 'Email verification initiated' TYPE 'I'.
      CATCH /aws1/cx_rt_generic INTO DATA(lo_ex).
        DATA(lv_error) = |An error occurred: { lo_ex->get_text( ) }|.
        MESSAGE lv_error TYPE 'I'.
        RAISE EXCEPTION lo_ex.
    ENDTRY.
```

- API-Details finden Sie VerifyEmailIdentity in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon SES API v2 unter Verwendung des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon SES API v2 Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateContact`

Das folgende Codebeispiel zeigt die Verwendung`CreateContact`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->createcontact(
      iv_contactlistname = iv_contact_list_name
      iv_emailaddress = iv_email_address ).
    MESSAGE 'Contact created successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2alreadyexistsex.
    MESSAGE 'Contact already exists.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex.
    MESSAGE 'Bad request.' TYPE 'E'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateContact in der AWS SDK for SAP ABAP API-Referenz.

`CreateContactList`

Das folgende Codebeispiel zeigt die Verwendung`CreateContactList`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->createcontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list created successfully.' TYPE 'I'.
```

```
    CATCH /aws1/cx_se2alreadyexistsex.
      MESSAGE 'Contact list already exists.' TYPE 'I'.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE 'Bad request - contact list limit may be reached.' TYPE 'I'.
      " Re-raise the exception so the caller can handle it
      RAISE EXCEPTION lo_bad_request.
    CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
      MESSAGE 'Limit exceeded - contact list limit reached.' TYPE 'I'.
      " Re-raise the exception so the caller can handle it
      RAISE EXCEPTION lo_limit_exceeded.
  ENDTRY.
```

- API-Details finden Sie CreateContactList in der AWS SDK for SAP ABAP API-Referenz.

`CreateEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_se2->createemailidentity(
        iv_emailidentity = iv_email_identity ).
      MESSAGE 'Email identity created successfully.' TYPE 'I'.
    CATCH /aws1/cx_se2alreadyexistsex.
      MESSAGE 'Email identity already exists.' TYPE 'I'.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
    CATCH /aws1/cx_se2limitexceededex INTO DATA(lo_limit_exceeded).
      MESSAGE lo_limit_exceeded TYPE 'I' DISPLAY LIKE 'E'.
  ENDTRY.
```

- API-Details finden Sie CreateEmailIdentity in der AWS SDK for SAP ABAP API-Referenz.

`CreateEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateEmailTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_template_content) = NEW /aws1/cl_se2emailtmplcontent(
          iv_subject = iv_subject
          iv_html = iv_html
          iv_text = iv_text ).

        lo_se2->createemailtemplate(
          iv_templatename = iv_template_name
          io_templatecontent = lo_template_content ).
        MESSAGE 'Email template created successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2alreadyexistsex.
        MESSAGE 'Email template already exists.' TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex.
        MESSAGE 'Bad request.' TYPE 'E'.
      CATCH /aws1/cx_se2limitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateEmailTemplate in der AWS SDK for SAP ABAP API-Referenz.

`DeleteContactList`

Das folgende Codebeispiel zeigt die Verwendung`DeleteContactList`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->deletecontactlist(
      iv_contactlistname = iv_contact_list_name ).
    MESSAGE 'Contact list deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2notfoundexception.
    MESSAGE 'Contact list not found.' TYPE 'I'.
  CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
    MESSAGE 'Bad request.' TYPE 'I'.
    RAISE EXCEPTION lo_bad_request.
ENDTRY.
```

- API-Details finden Sie DeleteContactList in der AWS SDK for SAP ABAP API-Referenz.

`DeleteEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_se2->deleteemailidentity(
      iv_emailidentity = iv_email_identity ).
    MESSAGE 'Email identity deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_se2notfoundexception.
```

```
      MESSAGE 'Email identity not found.' TYPE 'I'.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE 'Bad request.' TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
  ENDTRY.
```

- API-Details finden Sie DeleteEmailIdentity in der AWS SDK for SAP ABAP API-Referenz.

`DeleteEmailTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteEmailTemplate`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      lo_se2->deleteemailtemplate(
        iv_templatename = iv_template_name ).
      MESSAGE 'Email template deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_se2notfoundexception.
      MESSAGE 'Email template not found.' TYPE 'I'.
    CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
      MESSAGE 'Bad request.' TYPE 'I'.
      RAISE EXCEPTION lo_bad_request.
  ENDTRY.
```

- API-Details finden Sie DeleteEmailTemplate in der AWS SDK for SAP ABAP API-Referenz.

`GetEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`GetEmailIdentity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_se2->getemailidentity(
          iv_emailidentity = iv_email_identity ).
        MESSAGE |Identity type: { oo_result->get_identitytype( ) }, | &&
                |verified for sending: { oo_result-
>get_verifiedforsendingstatus( ) }| TYPE 'I'.
      CATCH /aws1/cx_se2notfoundexception.
        MESSAGE |Email identity { iv_email_identity } not found.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- API-Details finden Sie GetEmailIdentity in der AWS SDK for SAP ABAP API-Referenz.

`ListContacts`

Das folgende Codebeispiel zeigt die Verwendung`ListContacts`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_se2->listcontacts(
          iv_contactlistname = iv_contact_list_name ).
        DATA(lv_count) = lines( oo_result->get_contacts( ) ).
```

```
        MESSAGE |Retrieved { lv_count } contacts from list.| TYPE 'I'.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE 'Contact list not found.' TYPE 'I'.
        RAISE EXCEPTION lo_not_found.
    ENDTRY.
```

- API-Details finden Sie ListContacts in der AWS SDK for SAP ABAP API-Referenz.

`SendBulkEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendBulkEmail`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Build the default template content used for all bulk recipients
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
          iv_templatedata = iv_template_data ).

        DATA(lo_default_content) = NEW /aws1/cl_se2bulkemailcontent(
          io_template = lo_template ).

        DATA(lo_result) = lo_se2->sendbulkemail(
          iv_fromemailaddress = iv_from_address
          io_defaultcontent   = lo_default_content
          it_bulkemailentries = it_bulk_entries ).

        ot_results = lo_result->get_bulkemailentryresults( ).
        MESSAGE |Bulk email sent to { lines( it_bulk_entries ) } recipient(s).| TYPE
'I'.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_rejected).
```

```
        MESSAGE lo_rejected TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_rejected.
      CATCH /aws1/cx_se2mailfrmdomnotver00 INTO DATA(lo_not_verified).
        MESSAGE lo_not_verified TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_verified.
      CATCH /aws1/cx_se2notfoundexception INTO DATA(lo_not_found).
        MESSAGE lo_not_found TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_not_found.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE lo_bad_request TYPE 'I' DISPLAY LIKE 'E'.
        RAISE EXCEPTION lo_bad_request.
    ENDTRY.
```

- API-Details finden Sie SendBulkEmail in der AWS SDK for SAP ABAP API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Sendet eine Nachricht.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value = iv_to_email_address )
 TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create message content
        DATA(lo_subject) = NEW /aws1/cl_se2content( iv_data = iv_subject ).
        DATA(lo_text_body) = NEW /aws1/cl_se2content( iv_data = iv_text_body ).
```

```
        DATA(lo_html_body) = NEW /aws1/cl_se2content( iv_data = iv_html_body ).
        DATA(lo_body) = NEW /aws1/cl_se2body(
          io_text = lo_text_body
          io_html = lo_html_body ).
        DATA(lo_message) = NEW /aws1/cl_se2message(
          io_subject = lo_subject
          io_body = lo_body ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_simple = lo_message ).

        " Send the email
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content ).
        MESSAGE 'Email sent successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

Sendet eine Nachricht unter Verwendung einer Vorlage.

```
    TRY.
        " Create destination with recipient address
        DATA lt_to_addresses TYPE /aws1/
cl_se2emailaddresslist_w=>tt_emailaddresslist.
        APPEND NEW /aws1/cl_se2emailaddresslist_w( iv_value = iv_to_email_address )
 TO lt_to_addresses.
        DATA(lo_destination) = NEW /aws1/cl_se2destination(
          it_toaddresses = lt_to_addresses ).

        " Create template reference
        DATA(lo_template) = NEW /aws1/cl_se2template(
          iv_templatename = iv_template_name
```

```
          iv_templatedata = iv_template_data ).

        DATA(lo_content) = NEW /aws1/cl_se2emailcontent(
          io_template = lo_template ).

        " Create list management options
        DATA(lo_list_mgmt) = NEW /aws1/cl_se2listmanagementopts(
          iv_contactlistname = iv_contact_list_name ).

        " Send the email using template
        lo_se2->sendemail(
          iv_fromemailaddress = iv_from_email_address
          io_destination = lo_destination
          io_content = lo_content
          io_listmanagementoptions = lo_list_mgmt ).
        MESSAGE 'Email sent using template successfully.' TYPE 'I'.
      CATCH /aws1/cx_se2accountsuspendedex INTO DATA(lo_account_suspended).
        MESSAGE 'Account suspended.' TYPE 'I'.
        RAISE EXCEPTION lo_account_suspended.
      CATCH /aws1/cx_se2badrequestex INTO DATA(lo_bad_request).
        MESSAGE 'Bad request.' TYPE 'I'.
        RAISE EXCEPTION lo_bad_request.
      CATCH /aws1/cx_se2messagerejected INTO DATA(lo_message_rejected).
        MESSAGE 'Message rejected - check email verification.' TYPE 'I'.
        RAISE EXCEPTION lo_message_rejected.
    ENDTRY.
```

- API-Details finden Sie SendEmail in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon SNS unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon SNS Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung. `CreateTopic`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result is
returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum number
of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie CreateTopic in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteTopic in der AWS SDK for SAP ABAP API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetTopicAttributes in der AWS SDK for SAP ABAP API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
       DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
       MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list subscribers.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie ListSubscriptions in der AWS SDK for SAP ABAP API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
```

```
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ListTopics in der AWS SDK for SAP ABAP API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

Veröffentlichen Sie eine Nachricht mit Attributen zu einem Thema.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
```

```
          iv_message = iv_message
          it_messageattributes = it_msg_attrs ).
        MESSAGE 'Message with attributes published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

Veröffentlichen Sie eine Nachricht in mehreren Formaten zu einem Thema.

```
    " Build JSON message structure for multi-format message
    DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.

    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = lv_json_message
          iv_subject = iv_subject
          iv_messagestructure = 'json' ).
        MESSAGE 'Multi-format message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für SAP ABAP.

`SetSubscriptionAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetSubscriptionAttributes`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->setsubscriptionattributes(
            iv_subscriptionarn = iv_subscription_arn
            iv_attributename  = 'FilterPolicy'
            iv_attributevalue = iv_filter_policy ).
        MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SetSubscriptionAttributes in der AWS SDK for SAP ABAP API-Referenz.

`SetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetTopicAttributes`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SetTopicAttributes in der AWS SDK for SAP ABAP API-Referenz.

Abonnieren

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is returned
for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum number
of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS SDK für SAP ABAP.

Abonnement beenden

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be deleted/
unsubscribed. Confirm subscription before performing unsubscribe operation.' TYPE
 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS SDK für SAP ABAP.

Szenarien

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value = 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum number
 of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
 topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
 an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum number
 of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
```

```
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'
                                                                    iv_stringvalue =
 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from $19
 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   " iv_phone_number = '+12065550101' - Phone number in E.164 format
   TRY.
       oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
         iv_phonenumber = iv_phone_number
         iv_message = iv_message ).
       MESSAGE 'Message published to phone number.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Phone number does not exist.' TYPE 'E'.
   ENDTRY.
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS SDK für SAP ABAP.

Beispiele für Amazon SQS unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon SQS Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

- Szenarien

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung. `CreateQueue`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie eine Standard-Warteschlange von Amazon SQS.

```
    TRY.
        oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).        "
 oo_result is returned for testing purposes. "
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
 queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

Erstellen Sie eine Amazon-SQS-Warteschlange, die auf eine Nachricht wartet.

```
    TRY.
        DATA lt_attributes TYPE /aws1/cl_sqsqueueattrmap_w=>tt_queueattributemap.
        DATA ls_attribute TYPE /aws1/
cl_sqsqueueattrmap_w=>ts_queueattributemap_maprow.
        ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.               " Time in
 seconds for long polling, such as how long the call waits for a message to arrive
 in the queue before returning. "
        ls_attribute-value = NEW /aws1/cl_sqsqueueattrmap_w( iv_value =
 iv_wait_time ).
        INSERT ls_attribute INTO TABLE lt_attributes.
```

```
        oo_result = lo_sqs->createqueue(                  " oo_result is returned
for testing purposes. "
                iv_queuename = iv_queue_name
                it_attributes = lt_attributes ).
        MESSAGE 'SQS queue created.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedeldrecently.
        MESSAGE 'After deleting a queue, wait 60 seconds before creating another
queue with the same name.' TYPE 'E'.
      CATCH /aws1/cx_sqsqueuenameexists.
        MESSAGE 'A queue with this name already exists.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateQueue in der AWS SDK for SAP ABAP API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sqs->deletemessage(
           iv_queueurl = iv_queue_url
           iv_receipthandle = iv_receipt_handle ).
        MESSAGE 'Message deleted from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsinvalididformat.
        MESSAGE 'The specified receipt handle is not valid.' TYPE 'E'.
      CATCH /aws1/cx_sqsreceipthandleisinv.
        MESSAGE 'The specified receipt handle is not valid for the current version.'
TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteMessage in der AWS SDK for SAP ABAP API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   TRY.
       oo_result = lo_sqs->deletemessagebatch(       " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_entries ).
       MESSAGE 'Messages deleted from SQS queue.' TYPE 'I'.
     CATCH /aws1/cx_sqsbtcentidsnotdist00.
       MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
     CATCH /aws1/cx_sqsemptybatchrequest.
       MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
     CATCH /aws1/cx_sqsinvbatchentryid.
       MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE 'E'.
     CATCH /aws1/cx_sqstoomanyentriesin00.
       MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK for SAP ABAP API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
        MESSAGE 'SQS queue deleted' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteQueue in der AWS SDK for SAP ABAP API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).        "
oo_result is returned for testing purposes. "
        MESSAGE 'Queue URL retrieved.' TYPE 'I'.
      CATCH /aws1/cx_sqsqueuedoesnotexist.
        MESSAGE 'The requested queue does not exist.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetQueueUrl in der AWS SDK for SAP ABAP API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->listqueues( ).        " oo_result is returned for
testing purposes. "
        MESSAGE 'Retrieved list of queues.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListQueues in der AWS SDK for SAP ABAP API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Empfangen Sie eine Nachricht von einer Amazon-SQS-Warteschlange.

```
    TRY.
        oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).    "
oo_result is returned for testing purposes. "
        DATA(lt_messages) = oo_result->get_messages( ).
        MESSAGE 'Message received from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsoverlimit.
```

```
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
    ENDTRY.
```

Empfangen Sie mithilfe der Long-Poll-Unterstützung eine Nachricht aus einer Amazon-SQS-Warteschlange.

```
    TRY.
        oo_result = lo_sqs->receivemessage(           " oo_result is returned for
testing purposes. "
                iv_queueurl = iv_queue_url
                iv_waittimeseconds = iv_wait_time ).    " Time in seconds for long
polling, such as how long the call waits for a message to arrive in the queue
before returning. " ).
        DATA(lt_messages) = oo_result->get_messages( ).
        MESSAGE 'Message received from SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsoverlimit.
        MESSAGE 'Maximum number of in-flight messages reached.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie ReceiveMessage in der AWS SDK for SAP ABAP API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->sendmessage(              " oo_result is returned for
testing purposes. "
           iv_queueurl = iv_queue_url
           iv_messagebody = iv_message ).
        MESSAGE 'Message sent to SQS queue.' TYPE 'I'.
```

```
      CATCH /aws1/cx_sqsinvalidmsgconts.
        MESSAGE 'Message contains non-valid characters.' TYPE 'E'.
      CATCH /aws1/cx_sqsunsupportedop.
        MESSAGE 'Operation not supported.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SendMessage in der AWS SDK for SAP ABAP API-Referenz.

`SendMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`SendMessageBatch`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        oo_result = lo_sqs->sendmessagebatch(         " oo_result is returned for
testing purposes. "
          iv_queueurl = iv_queue_url
          it_entries = it_messages ).
        MESSAGE 'Messages sent to SQS queue.' TYPE 'I'.
      CATCH /aws1/cx_sqsbtcentidsnotdist00.
        MESSAGE 'Two or more batch entries in the request have the same ID.' TYPE
'E'.
      CATCH /aws1/cx_sqsbatchreqtoolong.
        MESSAGE 'The length of all the messages put together is more than the
limit.' TYPE 'E'.
      CATCH /aws1/cx_sqsemptybatchrequest.
        MESSAGE 'The batch request does not contain any entries.' TYPE 'E'.
      CATCH /aws1/cx_sqsinvbatchentryid.
        MESSAGE 'The ID of a batch entry in a batch request is not valid.' TYPE 'E'.
      CATCH /aws1/cx_sqstoomanyentriesin00.
        MESSAGE 'The batch request contains more entries than allowed.' TYPE 'E'.
      CATCH /aws1/cx_sqsunsupportedop.
        MESSAGE 'Operation not supported.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie SendMessageBatch in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Erstellen und veröffentlichen zu einem FIFO-Thema

Die folgenden Code-Beispiele zeigen, wie man ein Amazon-SNS-Thema erstellt.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein FIFO-Thema, abonnieren Sie eine Amazon-SQS-FIFO-Warteschlange für das Thema und veröffentlichen Sie eine Nachricht zu einem Amazon-SNS-Thema.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value = 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum number
 of topics allowed.' TYPE 'E'.
    ENDTRY.
```

```
   " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon SNS)
 topic. "
   " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed to
 an SNS FIFO topic. "
   TRY.
       DATA(lo_subscribe_result) = lo_sns->subscribe(
              iv_topicarn = lv_topic_arn
              iv_protocol = 'sqs'
              iv_endpoint = iv_queue_arn ).
       DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
       ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
       MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum number
 of subscriptions allowed.' TYPE 'E'.
   ENDTRY.

   " Publish message to SNS topic. "
   TRY.
       DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
       DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
       ls_msg_attributes-key = 'Importance'.
       ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String'
                                                                   iv_stringvalue =
 'High' ).
       INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

       DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from $19
 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes ).
       ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
```

```
      MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
      MESSAGE 'Topic does not exist.' TYPE 'E'.
  ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - CreateTopic
  - Veröffentlichen
  - Abonnieren

Beispiele für Step Functions mit SDK für SAP ABAP

Die folgenden Codebeispiele zeigen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Step Functions Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateActivity`

Das folgende Codebeispiel zeigt die Verwendung`CreateActivity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_sfn->createactivity(
      iv_name = iv_name
    ).
    ov_activity_arn = lo_result->get_activityarn( ).
    MESSAGE 'Activity created successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivityalrdyex.
    MESSAGE 'Activity already exists.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidname.
    MESSAGE 'Invalid activity name.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivitylimitexcd.
    MESSAGE 'Activity limit exceeded.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie CreateActivity in der AWS SDK for SAP ABAP API-Referenz.

`CreateStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`CreateStateMachine`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_sfn->createstatemachine(
      iv_name = iv_name
```

```
          iv_definition = iv_definition
          iv_rolearn = iv_role_arn
        ).
        ov_state_machine_arn = lo_result->get_statemachinearn( ).
        MESSAGE 'State machine created successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinealrex.
        MESSAGE 'State machine already exists.' TYPE 'E'.
      CATCH /aws1/cx_sfninvaliddefinition.
        MESSAGE 'Invalid state machine definition.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidname.
        MESSAGE 'Invalid state machine name.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid role ARN.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie CreateStateMachine in der AWS SDK for SAP ABAP API-Referenz.

`DeleteActivity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteActivity`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_sfn->deleteactivity(
          iv_activityarn = iv_activity_arn
        ).
        MESSAGE 'Activity deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid activity ARN.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DeleteActivity in der AWS SDK for SAP ABAP API-Referenz.

`DeleteStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DeleteStateMachine`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_sfn->deletestatemachine(
      iv_statemachinearn = iv_state_machine_arn
    ).
    MESSAGE 'State machine deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnvalidationex.
    MESSAGE 'Validation error occurred.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie DeleteStateMachine in der AWS SDK for SAP ABAP API-Referenz.

`DescribeExecution`

Das folgende Codebeispiel zeigt die Verwendung`DescribeExecution`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_sfn->describeexecution(
```

```
        iv_executionarn = iv_execution_arn
      ).
      MESSAGE 'Execution described successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfnexecdoesnotexist.
      MESSAGE 'Execution does not exist.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidarn.
      MESSAGE 'Invalid execution ARN.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribeExecution in der AWS SDK for SAP ABAP API-Referenz.

`DescribeStateMachine`

Das folgende Codebeispiel zeigt die Verwendung`DescribeStateMachine`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_sfn->describestatemachine(
        iv_statemachinearn = iv_state_machine_arn
      ).
      MESSAGE 'State machine described successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfnstatemachinedoes00.
      MESSAGE 'State machine does not exist.' TYPE 'E'.
    CATCH /aws1/cx_sfninvalidarn.
      MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie DescribeStateMachine in der AWS SDK for SAP ABAP API-Referenz.

`GetActivityTask`

Das folgende Codebeispiel zeigt die Verwendung`GetActivityTask`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_sfn->getactivitytask(
      iv_activityarn = iv_activity_arn
    ).
    MESSAGE 'Activity task retrieved successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfnactivitydoesnotex.
    MESSAGE 'Activity does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidarn.
    MESSAGE 'Invalid activity ARN.' TYPE 'E'.
  CATCH /aws1/cx_sfnactivityworkerlm00.
    MESSAGE 'Activity worker limit exceeded.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie GetActivityTask in der AWS SDK for SAP ABAP API-Referenz.

`ListActivities`

Das folgende Codebeispiel zeigt die Verwendung`ListActivities`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_sfn->listactivities( ).
    DATA(lt_activities) = lo_result->get_activities( ).
    LOOP AT lt_activities INTO DATA(lo_activity).
```

```
        IF lo_activity->get_name( ) = iv_name.
          ov_activity_arn = lo_activity->get_activityarn( ).
          EXIT.
        ENDIF.
      ENDLOOP.
      MESSAGE 'Activities listed successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfninvalidtoken.
      MESSAGE 'Invalid pagination token.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListActivities in der AWS SDK for SAP ABAP API-Referenz.

`ListStateMachines`

Das folgende Codebeispiel zeigt die Verwendung`ListStateMachines`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      DATA(lo_result) = lo_sfn->liststatemachines( ).
      DATA(lt_state_machines) = lo_result->get_statemachines( ).
      LOOP AT lt_state_machines INTO DATA(lo_state_machine).
        IF lo_state_machine->get_name( ) = iv_name.
          ov_state_machine_arn = lo_state_machine->get_statemachinearn( ).
          EXIT.
        ENDIF.
      ENDLOOP.
      MESSAGE 'State machines listed successfully.' TYPE 'I'.
    CATCH /aws1/cx_sfninvalidtoken.
      MESSAGE 'Invalid pagination token.' TYPE 'E'.
  ENDTRY.
```

- API-Details finden Sie ListStateMachines in der AWS SDK for SAP ABAP API-Referenz.

`SendTaskSuccess`

Das folgende Codebeispiel zeigt die Verwendung`SendTaskSuccess`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_sfn->sendtasksuccess(
      iv_tasktoken = iv_task_token
      iv_output = iv_task_response
    ).
    MESSAGE 'Task success sent successfully.' TYPE 'I'.
  CATCH /aws1/cx_sfninvalidtoken.
    MESSAGE 'Invalid task token.' TYPE 'E'.
  CATCH /aws1/cx_sfntaskdoesnotexist.
    MESSAGE 'Task does not exist.' TYPE 'E'.
  CATCH /aws1/cx_sfninvalidoutput.
    MESSAGE 'Invalid task output.' TYPE 'E'.
  CATCH /aws1/cx_sfntasktimedout.
    MESSAGE 'Task timed out.' TYPE 'E'.
ENDTRY.
```

- API-Details finden Sie SendTaskSuccess in der AWS SDK for SAP ABAP API-Referenz.

`StartExecution`

Das folgende Codebeispiel zeigt die Verwendung`StartExecution`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        DATA(lo_result) = lo_sfn->startexecution(
          iv_statemachinearn = iv_state_machine_arn
          iv_input = iv_input
        ).
        ov_execution_arn = lo_result->get_executionarn( ).
        MESSAGE 'Execution started successfully.' TYPE 'I'.
      CATCH /aws1/cx_sfnstatemachinedoes00.
        MESSAGE 'State machine does not exist.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidarn.
        MESSAGE 'Invalid state machine ARN.' TYPE 'E'.
      CATCH /aws1/cx_sfninvalidexecinput.
        MESSAGE 'Invalid execution input.' TYPE 'E'.
      CATCH /aws1/cx_sfnexeclimitexceeded.
        MESSAGE 'Execution limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie StartExecution in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Systems Manager, die das SDK für SAP ABAP verwenden

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Systems Manager Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateDocument`

Das folgende Codebeispiel zeigt die Verwendung`CreateDocument`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->createdocument(
        iv_name = iv_name
        iv_content = iv_content
        iv_documenttype = 'Command' ).
    MESSAGE 'Document created.' TYPE 'I'.
  CATCH /aws1/cx_ssmdocalreadyexists.
    MESSAGE 'Document already exists.' TYPE 'I'.
  CATCH /aws1/cx_ssminvaliddoccontent.
    MESSAGE 'Invalid document content.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie CreateDocument in der AWS SDK for SAP ABAP API-Referenz.

`CreateMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`CreateMaintenanceWindow`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ssm->createmaintenancewindow(
        iv_name = iv_name
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window created.' TYPE 'I'.
  CATCH /aws1/cx_ssmresrclimitexcdex.
    MESSAGE 'Resource limit exceeded.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie CreateMaintenanceWindow in der AWS SDK for SAP ABAP API-Referenz.

`CreateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`CreateOpsItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    oo_result = lo_ssm->createopsitem(
        iv_title = iv_title
        iv_source = iv_source
```

```
            iv_category = iv_category
            iv_severity = iv_severity
            iv_description = iv_description ).
        MESSAGE 'OpsItem created.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemlimitexcdex.
        MESSAGE 'You have exceeded your open OpsItem limit.' TYPE 'I'.
      CATCH /aws1/cx_ssmopsitemalrdyexex.
        MESSAGE 'OpsItem already exists.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie CreateOpsItem in der AWS SDK for SAP ABAP API-Referenz.

`DeleteDocument`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDocument`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_ssm->deletedocument( iv_name = iv_name ).
        MESSAGE 'Document deleted.' TYPE 'I'.
      CATCH /aws1/cx_ssminvaliddocument.
        MESSAGE 'Invalid document.' TYPE 'I'.
      CATCH /aws1/cx_ssmassocdinstances.
        MESSAGE 'Document has associated instances.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DeleteDocument in der AWS SDK for SAP ABAP API-Referenz.

`DeleteMaintenanceWindow`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMaintenanceWindow`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->deletemaintenancewindow( iv_windowid = iv_window_id ).
    MESSAGE 'Maintenance window deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssminternalservererr.
    MESSAGE 'Internal server error occurred.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie DeleteMaintenanceWindow in der AWS SDK for SAP ABAP API-Referenz.

`DeleteOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteOpsItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->deleteopsitem( iv_opsitemid = iv_ops_item_id ).
    MESSAGE 'OpsItem deleted.' TYPE 'I'.
  CATCH /aws1/cx_ssmopsiteminvparamex.
    MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie DeleteOpsItem in der AWS SDK for SAP ABAP API-Referenz.

`DescribeDocument`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDocument`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_ssm->describedocument( iv_name = iv_name ).
    DATA(lo_document) = lo_result->get_document( ).
    IF lo_document IS BOUND.
      rv_status = lo_document->get_status( ).
      MESSAGE |Document status: { rv_status }| TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_ssminvaliddocument.
    MESSAGE 'Invalid document.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie DescribeDocument in der AWS SDK for SAP ABAP API-Referenz.

`DescribeOpsItems`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOpsItems`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        " Create filter for OpsItem ID
        DATA(lt_filters) = VALUE /aws1/cl_ssmopsitemfilter=>tt_opsitemfilters(
          ( NEW /aws1/cl_ssmopsitemfilter(
              iv_key = 'OpsItemId'
              it_values = VALUE /aws1/
cl_ssmopsitemfiltvals_w=>tt_opsitemfiltervalues(
                ( NEW /aws1/cl_ssmopsitemfiltvals_w( iv_value = iv_ops_item_id ) )
              )
              iv_operator = 'Equal'
            ) )
        ).

        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->describeopsitems(
          it_opsitemfilters = lt_filters ).

        rv_found = abap_false.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmdescropsitemsrsp( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_opsitemsummaries( ) INTO DATA(lo_item).
            DATA(lv_title) = lo_item->get_title( ).
            DATA(lv_status) = lo_item->get_status( ).
            MESSAGE |The OpsItem title is { lv_title } and the status is
 { lv_status }| TYPE 'I'.
            rv_found = abap_true.
          ENDLOOP.
        ENDWHILE.
      CATCH /aws1/cx_ssminternalservererr.
        MESSAGE 'Internal server error occurred.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie DescribeOpsItems in der AWS SDK for SAP ABAP API-Referenz.

`ListCommandInvocations`

Das folgende Codebeispiel zeigt die Verwendung`ListCommandInvocations`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        " Use paginator to get all results
        DATA(lo_paginator) = lo_ssm->get_paginator( ).
        DATA(lo_iterator) = lo_paginator->listcommandinvocations(
          iv_instanceid = iv_instance_id ).

        DATA lv_count TYPE i VALUE 0.

        WHILE lo_iterator->has_next( ).
          DATA(lo_result) = CAST /aws1/cl_ssmlistcmdinvcsresult( lo_iterator-
>get_next( ) ).
          LOOP AT lo_result->get_commandinvocations( ) INTO DATA(lo_invocation).
            lv_count = lv_count + 1.
            DATA(lv_requested_datetime) = lo_invocation->get_requesteddatetime( ).
            MESSAGE |Command invocation requested at: { lv_requested_datetime }|
 TYPE 'I'.
          ENDLOOP.
        ENDWHILE.

        MESSAGE |{ lv_count } command invocation(s) found for instance
 { iv_instance_id }.| TYPE 'I'.
      CATCH /aws1/cx_ssminvalidinstanceid.
        MESSAGE 'Invalid instance ID.' TYPE 'I'.
      CATCH /aws1/cx_ssminvalidcommandid.
        MESSAGE 'Invalid command ID.' TYPE 'I'.
    ENDTRY.
```

- API-Details finden Sie ListCommandInvocations in der AWS SDK for SAP ABAP API-Referenz.

SendCommand

Das folgende Codebeispiel zeigt die VerwendungSendCommand.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    DATA(lo_result) = lo_ssm->sendcommand(
        it_instanceids = it_instance_ids
        iv_documentname = iv_document_name
        iv_timeoutseconds = 3600 ).
    DATA(lo_command) = lo_result->get_command( ).
    IF lo_command IS BOUND.
      rv_command_id = lo_command->get_commandid( ).
      MESSAGE 'Command sent successfully.' TYPE 'I'.
    ENDIF.
  CATCH /aws1/cx_ssminvaliddocument.
    MESSAGE 'Invalid document.' TYPE 'I'.
  CATCH /aws1/cx_ssminvalidinstanceid.
    MESSAGE 'Invalid instance ID.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie SendCommand in der AWS SDK for SAP ABAP API-Referenz.

UpdateMaintenanceWindow

Das folgende Codebeispiel zeigt die VerwendungUpdateMaintenanceWindow.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_ssm->updatemaintenancewindow(
        iv_windowid = iv_window_id
        iv_name = iv_name
        iv_enabled = iv_enabled
        iv_schedule = iv_schedule
        iv_duration = iv_duration
        iv_cutoff = iv_cutoff
        iv_allowunassociatedtargets = iv_allow_unassociated_targets ).
    MESSAGE 'Maintenance window updated.' TYPE 'I'.
  CATCH /aws1/cx_ssmdoesnotexistex.
    MESSAGE 'Maintenance window does not exist.' TYPE 'I'.
ENDTRY.
```

- API-Details finden Sie UpdateMaintenanceWindow in der AWS SDK for SAP ABAP API-Referenz.

`UpdateOpsItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateOpsItem`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      lo_ssm->updateopsitem(
          iv_opsitemid = iv_ops_item_id
          iv_title = iv_title
          iv_description = iv_description
          iv_status = iv_status ).
      MESSAGE 'OpsItem updated.' TYPE 'I'.
    CATCH /aws1/cx_ssmopsitemnotfoundex.
      MESSAGE 'OpsItem not found.' TYPE 'I'.
    CATCH /aws1/cx_ssmopsiteminvparamex.
      MESSAGE 'Invalid OpsItem parameter.' TYPE 'I'.
  ENDTRY.
```

- API-Details finden Sie UpdateOpsItem in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Textract unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Textract Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AnalyzeDocument`

Das folgende Codebeispiel zeigt die Verwendung. `AnalyzeDocument`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Detects text and additional elements, such as forms or tables,"
   "in a local image file or from in-memory byte data."
   "The image must be in PNG or JPG format."


   "Create ABAP objects for feature type."
   "Add TABLES to return information about the tables."
   "Add FORMS to return detected form data."
   "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).

   "Analyze document stored in Amazon S3."
   TRY.
       oo_result = lo_tex->analyzedocument(      "oo_result is returned for testing
purposes."
         io_document        = lo_document
         it_featuretypes    = lt_featuretypes ).
       LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
         ENDIF.
       ENDLOOP.
```

```
        MESSAGE 'Analyze document completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texhlquotaexceededex.
        MESSAGE 'Human loop quota exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.

      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie AnalyzeDocument in der AWS SDK for SAP ABAP API-Referenz.

`DetectDocumentText`

Das folgende Codebeispiel zeigt die Verwendung`DetectDocumentText`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Detects text in the input document."
```

```
    "Amazon Textract can detect lines of text and the words that make up a line of
 text."
    "The input document must be in one of the following image formats: JPEG, PNG,
 PDF, or TIFF."

    "Create an ABAP object for the Amazon S3 object."
    DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
      iv_name   = iv_s3object ).

    "Create an ABAP object for the document."
    DATA(lo_document) = NEW /aws1/cl_texdocument( io_s3object = lo_s3object ).
    "Analyze document stored in Amazon S3."
    TRY.
        oo_result = lo_tex->detectdocumenttext( io_document = lo_document ).
 "oo_result is returned for testing purposes."
        LOOP AT oo_result->get_blocks( ) INTO DATA(lo_block).
          IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
            MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
          ENDIF.
        ENDLOOP.
        DATA(lo_metadata) = oo_result->get_documentmetadata( ).
        MESSAGE 'The number of pages in the document is ' && lo_metadata-
>ask_pages( ) TYPE 'I'.
        MESSAGE 'Detect document text completed.' TYPE 'I'.
      CATCH /aws1/cx_texaccessdeniedex.
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
      CATCH /aws1/cx_texbaddocumentex.
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
      CATCH /aws1/cx_texdocumenttoolargeex.
        MESSAGE 'The document is too large.' TYPE 'E'.
      CATCH /aws1/cx_texinternalservererr.
        MESSAGE 'Internal server error.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie DetectDocumentText in der AWS SDK for SAP ABAP API-Referenz.

`GetDocumentAnalysis`

Das folgende Codebeispiel zeigt die Verwendung`GetDocumentAnalysis`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets the results for an Amazon Textract"
   "asynchronous operation that analyzes text in a document."
   TRY.
       oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purposes."
       WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
         IF sy-index = 10.
           EXIT.               "Maximum 300 seconds.
         ENDIF.
         WAIT UP TO 30 SECONDS.
         oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
       ENDWHILE.

       DATA(lt_blocks) = oo_result->get_blocks( ).
       LOOP AT lt_blocks INTO DATA(lo_block).
         IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
           MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
         ENDIF.
       ENDLOOP.
       MESSAGE 'Document analysis retrieved.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
```

```
      CATCH /aws1/cx_texinvalidjobidex.
        MESSAGE 'Job ID is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidkmskeyex.
        MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texinvalidparameterex.
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie GetDocumentAnalysis in der AWS SDK for SAP ABAP API-Referenz.

`StartDocumentAnalysis`

Das folgende Codebeispiel zeigt die Verwendung`StartDocumentAnalysis`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Starts the asynchronous analysis of an input document for relationships"
    "between detected items such as key-value pairs, tables, and selection
elements."

    "Create ABAP objects for feature type."
    "Add TABLES to return information about the tables."
    "Add FORMS to return detected form data."
    "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."

    DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
      ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
      ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).
```

```
   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start async document analysis."
   TRY.
       oo_result = lo_tex->startdocumentanalysis(      "oo_result is returned for
testing purposes."
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       DATA(lv_jobid) = oo_result->get_jobid( ).

       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texlimitexceededex.
       MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
     CATCH /aws1/cx_texprovthruputexcdex.
       MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
     CATCH /aws1/cx_texthrottlingex.
       MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
     CATCH /aws1/cx_texunsupporteddocex.
       MESSAGE 'The document is not supported.' TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie StartDocumentAnalysis in der AWS SDK for SAP ABAP API-Referenz.

StartDocumentTextDetection

Das folgende Codebeispiel zeigt die VerwendungStartDocumentTextDetection.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Starts the asynchronous detection of text in a document."
   "Amazon Textract can detect lines of text and the words that make up a line of
text."

   "Create an ABAP object for the Amazon S3 object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).
   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).
   "Start document analysis."
   TRY.
       oo_result = lo_tex->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).
       DATA(lv_jobid) = oo_result->get_jobid( ).             "oo_result is returned
for testing purposes."
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
```

```
        MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
      CATCH /aws1/cx_texinvalids3objectex.
        MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
      CATCH /aws1/cx_texlimitexceededex.
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
      CATCH /aws1/cx_texprovthruputexcdex.
        MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
      CATCH /aws1/cx_texthrottlingex.
        MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
      CATCH /aws1/cx_texunsupporteddocex.
        MESSAGE 'The document is not supported.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie StartDocumentTextDetection in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Erste Schritte mit der Dokumentenanalyse

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie die asynchrone Analyse.
- Rufen Sie die Analyse von Dokumenten ab.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Create ABAP objects for feature type."
    "Add TABLES to return information about the tables."
    "Add FORMS to return detected form data."
    "To perform both types of analysis, add TABLES and FORMS to FeatureTypes."
```

```
   DATA(lt_featuretypes) = VALUE /aws1/cl_texfeaturetypes_w=>tt_featuretypes(
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'FORMS' ) )
     ( NEW /aws1/cl_texfeaturetypes_w( iv_value = 'TABLES' ) ) ).

   "Create an ABAP object for the Amazon Simple Storage Service (Amazon S3)
object."
   DATA(lo_s3object) = NEW /aws1/cl_texs3object( iv_bucket = iv_s3bucket
     iv_name   = iv_s3object ).

   "Create an ABAP object for the document."
   DATA(lo_documentlocation) = NEW /aws1/cl_texdocumentlocation( io_s3object =
lo_s3object ).

   "Start document analysis."
   TRY.
       DATA(lo_start_result) = lo_tex->startdocumentanalysis(
         io_documentlocation     = lo_documentlocation
         it_featuretypes         = lt_featuretypes ).
       MESSAGE 'Document analysis started.' TYPE 'I'.
     CATCH /aws1/cx_texaccessdeniedex.
       MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
     CATCH /aws1/cx_texbaddocumentex.
       MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
     CATCH /aws1/cx_texdocumenttoolargeex.
       MESSAGE 'The document is too large.' TYPE 'E'.
     CATCH /aws1/cx_texidempotentprmmis00.
       MESSAGE 'Idempotent parameter mismatch exception.' TYPE 'E'.
     CATCH /aws1/cx_texinternalservererr.
       MESSAGE 'Internal server error.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidkmskeyex.
       MESSAGE 'AWS KMS key is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texinvalidparameterex.
       MESSAGE 'Request has non-valid parameters.' TYPE 'E'.
     CATCH /aws1/cx_texinvalids3objectex.
       MESSAGE 'Amazon S3 object is not valid.' TYPE 'E'.
     CATCH /aws1/cx_texlimitexceededex.
       MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
     CATCH /aws1/cx_texprovthruputexcdex.
       MESSAGE 'Provisioned throughput exceeded limit.' TYPE 'E'.
     CATCH /aws1/cx_texthrottlingex.
       MESSAGE 'The request processing exceeded the limit.' TYPE 'E'.
     CATCH /aws1/cx_texunsupporteddocex.
       MESSAGE 'The document is not supported.' TYPE 'E'.
   ENDTRY.
```

```
    "Get job ID from the output."
    DATA(lv_jobid) = lo_start_result->get_jobid( ).

    "Wait for job to complete."
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).     " oo_result
is returned for testing purposes. "
    WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
      IF sy-index = 10.
        EXIT.               "Maximum 300 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
      oo_result = lo_tex->getdocumentanalysis( iv_jobid = lv_jobid ).
    ENDWHILE.

    DATA(lt_blocks) = oo_result->get_blocks( ).
    LOOP AT lt_blocks INTO DATA(lo_block).
      IF lo_block->get_text( ) = 'INGREDIENTS: POWDERED SUGAR* (CANE SUGAR,'.
        MESSAGE 'Found text in the doc: ' && lo_block->get_text( ) TYPE 'I'.
      ENDIF.
    ENDLOOP.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - GetDocumentAnalysis
  - StartDocumentAnalysis

Amazon Transcribe-Beispiele mithilfe des SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Transcribe Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung. `CreateVocabulary`

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->createvocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Custom vocabulary created.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
```

```
    ENDTRY.
```

- API-Details finden Sie CreateVocabulary in der AWS SDK for SAP ABAP API-Referenz.

`DeleteTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTranscriptionJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        lo_tnb->deletetranscriptionjob( iv_job_name ).
        MESSAGE 'Transcription job deleted.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API-Details finden Sie DeleteTranscriptionJob in der AWS SDK for SAP ABAP API-Referenz.

`DeleteVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`DeleteVocabulary`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    lo_tnb->deletevocabulary( iv_vocabulary_name ).
    MESSAGE 'Vocabulary deleted.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
  CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
    MESSAGE lo_limit_ex TYPE 'I'.
    RAISE EXCEPTION lo_limit_ex.
  CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
    MESSAGE lo_not_found_ex TYPE 'I'.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- API-Details finden Sie DeleteVocabulary in der AWS SDK for SAP ABAP API-Referenz.

`GetTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`GetTranscriptionJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
      oo_result = lo_tnb->gettranscriptionjob( iv_job_name ).
      DATA(lo_job) = oo_result->get_transcriptionjob( ).
      MESSAGE 'Retrieved transcription job details.' TYPE 'I'.
    CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
      MESSAGE lo_not_found_ex TYPE 'I'.
      RAISE EXCEPTION lo_not_found_ex.
    CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
      MESSAGE lo_internal_ex TYPE 'I'.
      RAISE EXCEPTION lo_internal_ex.
  ENDTRY.
```

- API-Details finden Sie GetTranscriptionJob in der AWS SDK for SAP ABAP API-Referenz.

`GetVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`GetVocabulary`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
  TRY.
      oo_result = lo_tnb->getvocabulary( iv_vocabulary_name ).
      MESSAGE 'Retrieved vocabulary details.' TYPE 'I'.
    CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
      MESSAGE lo_bad_request_ex TYPE 'I'.
      RAISE EXCEPTION lo_bad_request_ex.
    CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
      MESSAGE lo_not_found_ex TYPE 'I'.
      RAISE EXCEPTION lo_not_found_ex.
    CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
      MESSAGE lo_internal_ex TYPE 'I'.
      RAISE EXCEPTION lo_internal_ex.
```

```
    ENDTRY.
```

- API-Details finden Sie GetVocabulary in der AWS SDK for SAP ABAP API-Referenz.

`ListTranscriptionJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTranscriptionJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF iv_job_filter IS NOT INITIAL.
          oo_result = lo_tnb->listtranscriptionjobs( iv_jobnamecontains =
iv_job_filter ).
        ELSE.
          oo_result = lo_tnb->listtranscriptionjobs( ).
        ENDIF.
        MESSAGE 'Retrieved transcription jobs list.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
    ENDTRY.
```

- API-Details finden Sie ListTranscriptionJobs in der AWS SDK for SAP ABAP API-Referenz.

`ListVocabularies`

Das folgende Codebeispiel zeigt die Verwendung`ListVocabularies`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
    IF iv_vocab_filter IS NOT INITIAL.
      oo_result = lo_tnb->listvocabularies( iv_namecontains = iv_vocab_filter ).
    ELSE.
      oo_result = lo_tnb->listvocabularies( ).
    ENDIF.
    MESSAGE 'Retrieved vocabularies list.' TYPE 'I'.
  CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
    MESSAGE lo_bad_request_ex TYPE 'I'.
    RAISE EXCEPTION lo_bad_request_ex.
  CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
    MESSAGE lo_internal_ex TYPE 'I'.
    RAISE EXCEPTION lo_internal_ex.
ENDTRY.
```

- API-Details finden Sie ListVocabularies in der AWS SDK for SAP ABAP API-Referenz.

`StartTranscriptionJob`

Das folgende Codebeispiel zeigt die Verwendung`StartTranscriptionJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
TRY.
```

```
        DATA(lo_media) = NEW /aws1/cl_tnbmedia( iv_mediafileuri = iv_media_uri ).
        DATA(lo_settings) = NEW /aws1/cl_tnbsettings( ).
        IF iv_vocabulary_name IS NOT INITIAL.
          lo_settings = NEW /aws1/cl_tnbsettings( iv_vocabularyname =
iv_vocabulary_name ).
        ENDIF.

        oo_result = lo_tnb->starttranscriptionjob(
          iv_transcriptionjobname = iv_job_name
          io_media = lo_media
          iv_mediaformat = iv_media_format
          iv_languagecode = iv_language_code
          io_settings = lo_settings ).

        MESSAGE 'Transcription job started.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
        RAISE EXCEPTION lo_bad_request_ex.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- API-Details finden Sie StartTranscriptionJob in der AWS SDK for SAP ABAP API-Referenz.

`UpdateVocabulary`

Das folgende Codebeispiel zeigt die Verwendung`UpdateVocabulary`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    TRY.
        IF it_phrases IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            it_phrases = it_phrases ).
        ELSEIF iv_vocab_file_uri IS NOT INITIAL.
          oo_result = lo_tnb->updatevocabulary(
            iv_vocabularyname = iv_vocabulary_name
            iv_languagecode = iv_language_code
            iv_vocabularyfileuri = iv_vocab_file_uri ).
        ENDIF.
        MESSAGE 'Vocabulary updated.' TYPE 'I'.
      CATCH /aws1/cx_tnbbadrequestex INTO DATA(lo_bad_request_ex).
        MESSAGE lo_bad_request_ex TYPE 'I'.
      CATCH /aws1/cx_tnblimitexceededex INTO DATA(lo_limit_ex).
        MESSAGE lo_limit_ex TYPE 'I'.
        RAISE EXCEPTION lo_limit_ex.
      CATCH /aws1/cx_tnbnotfoundexception INTO DATA(lo_not_found_ex).
        MESSAGE lo_not_found_ex TYPE 'I'.
      CATCH /aws1/cx_tnbinternalfailureex INTO DATA(lo_internal_ex).
        MESSAGE lo_internal_ex TYPE 'I'.
        RAISE EXCEPTION lo_internal_ex.
      CATCH /aws1/cx_tnbconflictexception INTO DATA(lo_conflict_ex).
        MESSAGE lo_conflict_ex TYPE 'I'.
        RAISE EXCEPTION lo_conflict_ex.
    ENDTRY.
```

- API-Details finden Sie UpdateVocabulary in der AWS SDK for SAP ABAP API-Referenz.

Beispiele für Amazon Translate unter Verwendung von SDK für SAP ABAP

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für SAP ABAP mit Amazon Translate Aktionen ausführen und gängige Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`DescribeTextTranslationJob`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTextTranslationJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets the properties associated with an asynchronous batch translation job."
   "Includes properties such as name, ID, status, source and target languages, and
input/output Amazon Simple Storage Service (Amazon S3) buckets."
   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
for testing purposes."
         iv_jobid        = iv_jobid ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie DescribeTextTranslationJob in der AWS SDK for SAP ABAP API-Referenz.

`ListTextTranslationJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListTextTranslationJobs`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Gets a list of the batch translation jobs that you have submitted."

   DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

   "Create an ABAP object for filtering using jobname."
   lo_filter = NEW #( iv_jobname = iv_jobname ).

   TRY.
       oo_result = lo_xl8->listtexttranslationjobs(      "oo_result is returned for
testing purposes."
         io_filter        = lo_filter ).
       MESSAGE 'Jobs retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidfilterex.
       MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie ListTextTranslationJobs in der AWS SDK for SAP ABAP API-Referenz.

StartTextTranslationJob

Das folgende Codebeispiel zeigt die VerwendungStartTextTranslationJob.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Starts an asynchronous batch translation job."
    "Use batch translation jobs to translate large volumes of text across multiple
 documents at once."

    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).

    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        oo_result = lo_xl8->starttexttranslationjob(      "oo_result is returned for
 testing purposes."
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
```

```
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie StartTextTranslationJob in der AWS SDK for SAP ABAP API-Referenz.

`StopTextTranslationJob`

Das folgende Codebeispiel zeigt die Verwendung`StopTextTranslationJob`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    "Stops an asynchronous batch translation job that is in progress."

    TRY.
        oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned for
testing purposes."
          iv_jobid        = iv_jobid ).
        MESSAGE 'Translation job stopped.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
```

```
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- API-Details finden Sie StopTextTranslationJob in der AWS SDK for SAP ABAP API-Referenz.

`TranslateText`

Das folgende Codebeispiel zeigt die Verwendung`TranslateText`.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
   "Translates input text from the source language to the target language."
   TRY.
       oo_result = lo_xl8->translatetext(      "oo_result is returned for testing
purposes."
         iv_text        = iv_text
           iv_sourcelanguagecode = iv_sourcelanguagecode
           iv_targetlanguagecode = iv_targetlanguagecode ).
       MESSAGE 'Translation completed.' TYPE 'I'.
     CATCH /aws1/cx_xl8detectedlanguage00.
       MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred.' TYPE 'E'.
     CATCH /aws1/cx_xl8invalidrequestex.
       MESSAGE 'The request that you made is not valid.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8serviceunavailex.
       MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE 'E'.
     CATCH /aws1/cx_xl8textsizelmtexcdex.
       MESSAGE 'The size of the text you submitted exceeds the size limit. ' TYPE
'E'.
```

```
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language of
 the source text into the requested target language. ' TYPE 'E'.
    ENDTRY.
```

- API-Details finden Sie TranslateText in der AWS SDK for SAP ABAP API-Referenz.

Szenarien

Erste Schritte mit Übersetzungsaufträgen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Starten Sie einen asynchronen Stapelübersetzungsauftrag.
- Warten Sie, bis der asynchrone Auftrag abgeschlossen wurde.
- Beschreiben Sie den asynchronen Job.

SDK für SAP ABAP

> **Note**
>
> Es läuft noch mehr. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
    DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
    DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
    DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

    "Create an ABAP object for the input data config."
    lo_inputdataconfig = NEW #( iv_s3uri = iv_input_data_s3uri
                                iv_contenttype = iv_input_data_contenttype ).
```

```
    "Create an ABAP object for the output data config."
    lo_outputdataconfig = NEW #( iv_s3uri = iv_output_data_s3uri ).

    "Create an internal table for target languages."
    lo_targetlanguagecodes = NEW #( iv_value = iv_targetlanguagecode ).
    INSERT lo_targetlanguagecodes  INTO TABLE lt_targetlanguagecodes.

    TRY.
        DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
          io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode ).
        MESSAGE 'Translation job started.' TYPE 'I'.
      CATCH /aws1/cx_xl8internalserverex.
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
      CATCH /aws1/cx_xl8invparamvalueex.
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8invalidrequestex.
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
      CATCH /aws1/cx_xl8resourcenotfoundex.
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
      CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time. '
 TYPE 'E'.
      CATCH /aws1/cx_xl8unsuppedlanguage00.
        MESSAGE 'Amazon Translate does not support translation from the language of
 the source text into the requested target language.' TYPE 'E'.
    ENDTRY.

    "Get the job ID."
    DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

    "Wait for translate job to complete."
    DATA(lo_des_translation_result) = lo_xl8->describetexttranslationjob( iv_jobid =
 lv_jobid ).
    WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
      IF sy-index = 30.
        EXIT.               "Maximum 900 seconds."
      ENDIF.
      WAIT UP TO 30 SECONDS.
```

```
      lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
   ENDWHILE.

   TRY.
       oo_result = lo_xl8->describetexttranslationjob(      "oo_result is returned
for testing purposes."
         iv_jobid        = lv_jobid ).
       MESSAGE 'Job description retrieved.' TYPE 'I'.
     CATCH /aws1/cx_xl8internalserverex.
       MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
     CATCH /aws1/cx_xl8resourcenotfoundex.
       MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
     CATCH /aws1/cx_xl8toomanyrequestsex.
       MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
   ENDTRY.
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS - SDK für SAP ABAP.
  - DescribeTextTranslationJob
  - StartTextTranslationJob

Codebeispiele für SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie das AWS SDK für Swift mit verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder Funktionen genutzt werden können, die spezifisch für den Service sind.

Weitere -Quellen

- SDK für Swift-Entwicklerhandbuch — Weitere Informationen zur Verwendung von Swift mit AWS.
- AWS -Entwicklerzentrum: Codebeispiele, die Sie nach Kategorie oder Volltextsuche filtern können.
- AWS SDK-Beispiele — GitHub Repo mit vollständigem Code in bevorzugten Sprachen. Enthält Anleitungen zum Einrichten und Ausführen des Codes.

Dienstleistungen

- Beispiele für Amazon Bedrock unter Verwendung von SDK für Swift
- Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Swift
- Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Swift
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Swift
- DynamoDB-Beispiele unter Verwendung von SDK für Swift
- Beispiele für Amazon EC2 unter Verwendung von SDK für Swift
- AWS Glue Beispiele zur Verwendung des SDK für Swift
- IAM-Beispiele unter Verwendung von SDK für Swift
- Lambda-Beispiele unter Verwendung von SDK für Swift
- Beispiele für Amazon RDS unter Verwendung von SDK für Swift
- Beispiele für Amazon S3 unter Verwendung von SDK für Swift
- Beispiele für Amazon SNS unter Verwendung von SDK für Swift
- Beispiele für Amazon SQS unter Verwendung von SDK für Swift
- AWS STS Beispiele mit SDK für Swift
- Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für Swift

Beispiele für Amazon Bedrock unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon Bedrock Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Amazon Bedrock

Das folgende Codebeispiel zeigt, wie Sie mit Amazon Bedrock beginnen.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import Foundation

import AWSBedrock

struct ExampleCommand: ParsableCommand {
    static var configuration = CommandConfiguration(
        commandName: "ListFoundationModels",
        abstract: """
        This example demonstrates how to retrieve a list of the available
        foundation models from Amazon Bedrock.
        """,
        discussion: """
        """
    )

    /// Construct a string listing the specified modalities.
    ///
    /// - Parameter modalities: An array of the modalities to list.
```

```
    ///
    /// - Returns: A string with a human-readable list of modalities.
    func buildModalityList(modalities: [BedrockClientTypes.ModelModality]?) ->
String {
        var first = true
        var str = ""

        if modalities == nil {
            return "<none>"
        }

        for modality in modalities! {
            if !first {
                str += ", "
            }
            first = false
            str += modality.rawValue
        }

        return str
    }

    /// Construct a string listing the specified customizations.
    ///
    /// - Parameter customizations: An array of the customizations to list.
    ///
    /// - Returns: A string listing the customizations.
    func buildCustomizationList(customizations:
[BedrockClientTypes.ModelCustomization]?) -> String {
        var first = true
        var str = ""

        if customizations == nil {
            return "<none>"
        }

        for customization in customizations! {
            if !first {
                str += ", "
            }
            first = false
            str += customization.rawValue
        }
```

```
        return str
    }

    /// Construct a string listing the specified inferences.
    ///
    /// - Parameter inferences: An array of inferences to list.
    ///
    /// - Returns: A string listing the specified inferences.
    func buildInferenceList(inferences: [BedrockClientTypes.InferenceType]?) ->
 String {
        var first = true
        var str = ""

        if inferences == nil {
            return "<none>"
        }

        for inference in inferences! {
            if !first {
                str += ", "
            }
            first = false
            str += inference.rawValue
        }

        return str
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region: "us-
east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }
```

```
        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
            print(" Supported inference types: \(buildInferenceList(inferences:
 summary.inferenceTypesSupported))")
            print("------------------------------------------\n")
        }

        print("\(summaries.count) models available.")
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API-Details finden Sie ListFoundationModels in der AWS SDK for Swift API-Referenz.

Aktionen

`ListFoundationModels`

Das folgende Codebeispiel zeigt die Verwendung`ListFoundationModels`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSBedrock

        // Always use the Region "us-east-1" to have access to the most models.
        let config = try await BedrockClient.BedrockClientConfiguration(region: "us-
east-1")
        let bedrockClient = BedrockClient(config: config)

        let output = try await bedrockClient.listFoundationModels(
            input: ListFoundationModelsInput()
        )

        guard let summaries = output.modelSummaries else {
            print("No models returned.")
            return
        }

        // Output a list of the models with their details.
        for summary in summaries {
            print("==========================================")
            print(" Model ID: \(summary.modelId ?? "<unknown>")")
            print("------------------------------------------")
            print(" Name: \(summary.modelName ?? "<unknown>")")
            print(" Provider: \(summary.providerName ?? "<unknown>")")
            print(" Input modalities: \(buildModalityList(modalities:
 summary.inputModalities))")
            print(" Output modalities: \(buildModalityList(modalities:
 summary.outputModalities))")
            print(" Supported customizations:
 \(buildCustomizationList(customizations: summary.customizationsSupported ))")
```

```
            print(" Supported inference types: \(buildInferenceList(inferences:
summary.inferenceTypesSupported))")
            print("-----------------------------------------\n")
        }
```

- API-Details finden Sie ListFoundationModels in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon Bedrock Runtime unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon Bedrock Runtime Aktionen ausführen und allgemeine Szenarien implementieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Amazon Nova
- Amazon Nova Canvas
- Amazon Nova Reel
- Anthropic Claude
- Meta Llama

Amazon Nova

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Amazon Nova.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "amazon.nova-micro-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
```

```
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS -SDK für Swift.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Amazon Nova gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Amazon Nova und verarbeiten Sie den Antwortstream in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Amazon Nova and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
```

```
    let modelId = "amazon.nova-lite-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nNova Lite:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
```

```
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Swift API-Referenz.

Amazon Nova Canvas

InvokeModel

Das folgende Codebeispiel zeigt, wie Amazon Nova Canvas auf Amazon Bedrock aufgerufen wird, um ein Image zu generieren.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erstellen Sie ein Bild mit Amazon Nova Canvas.

```
// Use the native inference API to create an image with Amazon Nova Canvas

import AWSBedrockRuntime
import AWSSDKIdentity
import Foundation

struct NovaImageOutput: Decodable {
    let images: [Data]
}

func generateImage(_ textPrompt: String) async throws {
    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    config.awsCredentialIdentityResolver = try SSOAWSCredentialIdentityResolver()

    let client = BedrockRuntimeClient(config: config)
```

```
    // Set the model ID.
    let modelId = "amazon.nova-canvas-v1:0"

    //  Format the request payload using the model's native structure.
    let input = InvokeModelInput(
        accept: "application/json",
        body: """
            {
                "textToImageParams": {
                    "text": "\(textPrompt)"
                },
                "taskType": "TEXT_IMAGE",
                "imageGenerationConfig": {
                    "seed": 42,
                    "quality": "standard",
                    "width": 512,
                    "height": 512,
                    "numberOfImages": 1
                }
            }
            """.data(using: .utf8),
        modelId: modelId
    )

    // Invoke the model with the request.
    let response = try await client.invokeModel(input: input)

    // Decode the response body.
    let output = try JSONDecoder().decode(NovaImageOutput.self, from:
 response.body!)

    // Extract the image data.
    guard let data = output.images.first else {
        print("No image data found")
        return
    }

    // Save the generated image to a local folder.
    let fileURL = URL.documentsDirectory.appending(path: "nova_canvas.png")
    print(fileURL)
    try data.write(to: fileURL)
    print("Image is saved at \(fileURL)")
}
```

- API-Details finden Sie InvokeModel in der AWS SDK for Swift API-Referenz.

Amazon Nova Reel

Text-to-video

Das folgende Codebeispiel zeigt, wie Amazon Nova Reel zur Generierung eines Videos anhand eines Textprompts verwendet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Nova Reel, um ein Video aus einem Text-Prompt zu generieren.

```
// This example demonstrates how to use Amazon Nova Reel to generate a video from a
 text prompt.
// It shows how to:
// - Set up the Amazon Bedrock runtime client
// - Configure a text-to-video request
// - Submit an asynchronous job for video generation
// - Poll for job completion status
// - Access the generated video from S3

import AWSBedrockRuntime
import Foundation
import Smithy

func startTextToVideoGenerationJob(
    bedrockRuntimeClient: BedrockRuntimeClient, prompt: String, outputS3Uri: String
) async throws -> String? {
    // Specify the model ID for text-to-video generation
    let modelId = "amazon.nova-reel-v1:0"

    // Configure the video generation request with additional parameters
    let modelInputSource: [String: Any] = [
```

```
        "taskType": "TEXT_VIDEO",
        "textToVideoParams": [
            "text": "\(prompt)"
        ],
        "videoGenerationConfig": [
            "durationSeconds": 6,
            "fps": 24,
            "dimension": "1280x720",
        ],
    ]

    let modelInput = try Document.make(from: modelInputSource)

    let input = StartAsyncInvokeInput(
        modelId: modelId,
        modelInput: modelInput,
        outputDataConfig: .s3outputdataconfig(
            BedrockRuntimeClientTypes.AsyncInvokeS3OutputDataConfig(
                s3Uri: outputS3Uri
            )
        )
    )

    // Invoke the model asynchronously
    let output = try await bedrockRuntimeClient.startAsyncInvoke(input: input)
    return output.invocationArn
}

func queryJobStatus(
    bedrockRuntimeClient: BedrockRuntimeClient,
    invocationArn: String?
) async throws -> GetAsyncInvokeOutput {
    try await bedrockRuntimeClient.getAsyncInvoke(
        input: GetAsyncInvokeInput(invocationArn: invocationArn))
}

func main() async throws {
    // Create a Bedrock Runtime client
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Specify the S3 location for the output video
    let bucket = "s3://REPLACE-WITH-YOUR-S3-BUCKET-NAM"
```

```
    print("Submitting video generation job...")
    let invocationArn = try await startTextToVideoGenerationJob(
        bedrockRuntimeClient: client,
        prompt: "A pomegranate juice in a railway station",
        outputS3Uri: bucket
    )
    print("Job started with invocation ARN: \(String(describing:invocationArn))")

    // Poll for job completion
    var status: BedrockRuntimeClientTypes.AsyncInvokeStatus?
    var isReady = false
    var hasFailed = false

    while !isReady && !hasFailed {
        print("\nPolling job status...")
        status = try await queryJobStatus(
            bedrockRuntimeClient: client, invocationArn: invocationArn
        ).status
        switch status {
        case .completed:
            isReady = true
            print("Video is ready\nCheck S3 bucket: \(bucket)")
        case .failed:
            hasFailed = true
            print("Something went wrong")
        case .inProgress:
            print("Job is in progress...")
            try await Task.sleep(nanoseconds: 15 * 1_000_000_000)  // 15 seconds
        default:
            isReady = true
        }
    }
}

do {
    try await main()
} catch {
    print("An error occurred: \(error)")
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - GetAsyncInvoke
  - StartAsyncInvoke

Anthropic Claude

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Anthropic Claude.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
```

```
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Swift.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Anthropic Claude gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie eine Textnachricht an Anthropic Claude mithilfe der Converse-API von Bedrock und verarbeiten Sie den Antwortstrom in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Anthropic Claude and print the response stream

import AWSBedrockRuntime

func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "anthropic.claude-3-haiku-20240307-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
```

```
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nAnthropic Claude:")

        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Swift API-Referenz.

Meta Llama

Converse

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama.

```
// An example demonstrating how to use the Conversation API to send
// a text message to Meta Llama.

import AWSBedrockRuntime

func converse(_ textPrompt: String) async throws -> String {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseInput to send to the model
    let input = ConverseInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)
```

```
    // Send the ConverseInput to the model
    let response = try await client.converse(input: input)

    // Extract and return the response text.
    if case let .message(msg) = response.output {
        if case let .text(textResponse) = msg.content![0] {
            return textResponse
        } else {
            return "No text response found in message content"
        }
    } else {
        return "No message found in converse output"
    }
}
```

- Weitere API-Informationen finden Sie unter Converse in der API-Referenz zum AWS SDK für Swift.

ConverseStream

Das folgende Codebeispiel zeigt, wie mit der Converse-API von Bedrock eine Textnachricht an Meta Llama gesendet und der Antwortstream in Echtzeit verarbeitet wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Senden Sie mithilfe der Converse-API von Bedrock eine Textnachricht an Meta Llama und verarbeiten Sie den Antwortstream in Echtzeit.

```
// An example demonstrating how to use the Conversation API to send a text message
// to Meta Llama and print the response stream.

import AWSBedrockRuntime
```

```
func printConverseStream(_ textPrompt: String) async throws {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    let config =
        try await BedrockRuntimeClient.BedrockRuntimeClientConfiguration()
    let client = BedrockRuntimeClient(config: config)

    // Set the model ID.
    let modelId = "meta.llama3-8b-instruct-v1:0"

    // Start a conversation with the user message.
    let message = BedrockRuntimeClientTypes.Message(
        content: [.text(textPrompt)],
        role: .user
    )

    // Optionally use inference parameters.
    let inferenceConfig =
        BedrockRuntimeClientTypes.InferenceConfiguration(
            maxTokens: 512,
            stopSequences: ["END"],
            temperature: 0.5,
            topp: 0.9
        )

    // Create the ConverseStreamInput to send to the model.
    let input = ConverseStreamInput(
        inferenceConfig: inferenceConfig, messages: [message], modelId: modelId)

    // Send the ConverseStreamInput to the model.
    let response = try await client.converseStream(input: input)

    // Extract the streaming response.
    guard let stream = response.stream else {
        print("No stream available")
        return
    }

    // Extract and print the streamed response text in real-time.
    for try await event in stream {
        switch event {
        case .messagestart(_):
            print("\nMeta Llama:")
```

```
        case .contentblockdelta(let deltaEvent):
            if case .text(let text) = deltaEvent.delta {
                print(text, terminator: "")
            }

        default:
            break
        }
    }
}
```

- API-Details finden Sie ConverseStream in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon Cognito Identity unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon Cognito Identity Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung`CreateIdentityPool`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        do {
            let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
 "com.exampleco.CognitoIdentityDemo",
                                                           identityPoolName: name)

            let result = try await cognitoIdentityClient.createIdentityPool(input:
 cognitoInputCall)
            guard let poolId = result.identityPoolId else {
                return nil
            }

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- API-Details finden Sie CreateIdentityPool in der AWS SDK for Swift API-Referenz.

`DeleteIdentityPool`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentityPool`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        do {
            let input = DeleteIdentityPoolInput(
                identityPoolId: id
            )

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- API-Details finden Sie DeleteIdentityPool in der AWS SDK for Swift API-Referenz.

`ListIdentityPools`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentityPools`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
 "ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
 listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
 contents.")
                    continue
                }

                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
```

```
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

Rufen Sie ID eines vorhandenen Identitätspools ab oder erstellen Sie sie, wenn sie nicht bereits vorhanden ist.

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        do {
            guard let poolId = try await getIdentityPoolID(name: name) else {
                return try await createIdentityPool(name: name)
            }

            return poolId
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- Weitere Informationen finden Sie im Entwicklerhandbuch zum AWS -SDK für Swift.
- API-Details finden Sie ListIdentityPools in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
```

```
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }
```

- API-Details finden Sie AdminGetUser in der AWS SDK for Swift API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die Verwendung`AdminInitiateAuth`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
 CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
```

```
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

- API-Details finden Sie AdminInitiateAuth in der AWS SDK for Swift API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die Verwendung`AdminRespondToAuthChallenge`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
```

```
   ///   - cipClient: The `CognitoIdentityProviderClient` to use.
   ///   - userName: The user's username.
   ///   - clientId: The app client ID.
   ///   - userPoolId: The user pool to sign into.
   ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
   ///     authenticator.
   ///   - session: The authentication session to continue processing.
   func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                    clientId: String, userPoolId: String, mfaCode:
String,
                                    session: String) async {
       print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

       var challengeResponsesOb: [String: String] = [:]
       challengeResponsesOb["USERNAME"] = userName
       challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

       do {
           let output = try await cipClient.adminRespondToAuthChallenge(
               input: AdminRespondToAuthChallengeInput(
                   challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                   challengeResponses: challengeResponsesOb,
                   clientId: clientId,
                   session: session,
                   userPoolId: userPoolId
               )
           )

           guard let authenticationResult = output.authenticationResult else {
               print("*** Unable to get authentication result.")
               return
           }

           print("=====> Authentication result (JWTs are redacted):")
           print(authenticationResult)
       } catch _ as SoftwareTokenMFANotFoundException {
           print("*** The specified user pool isn't configured for MFA.")
           return
       } catch _ as CodeMismatchException {
           print("*** The specified MFA code doesn't match the expected value.")
           return
       } catch _ as UserNotFoundException {
```

```
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
 "<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- API-Details finden Sie AdminRespondToAuthChallenge in der AWS SDK for Swift API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
```

```
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient, authSession:
String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the app's
MFA.")
            return nil
        }
    }
```

- API-Details finden Sie AssociateSoftwareToken in der AWS SDK for Swift API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise `false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId: String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }
```

- API-Details finden Sie ConfirmSignUp in der AWS SDK for Swift API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        do {
            let output = try await cognitoClient.listUsers(
                input: ListUsersInput(
                    userPoolId: poolId
                )
            )

            guard let users = output.users else {
                print("No users found.")
                return
            }

            print("\(users.count) user(s) found.")
            for user in users {
                print("  \(user.username ?? "<unknown>")")
            }
        } catch _ as NotAuthorizedException {
            print("*** Please authenticate with AWS before using this command.")
            return
        } catch _ as ResourceNotFoundException {
            print("*** The specified User Pool was not found.")
            return
        } catch {
            print("*** An unexpected type of error occurred.")
            return
        }
```

- API-Details finden Sie ListUsers in der AWS SDK for Swift API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient, clientId:
 String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
                print("*** Unable to get the delivery method for the resent code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user \(userName).")
```

```
            return false
        }
    }
```

- API-Details finden Sie ResendConfirmationCode in der AWS SDK for Swift API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die Verwendung`SignUp`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]
```

```
        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException {
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- API-Details finden Sie SignUp in der AWS SDK for Swift API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
```

```
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- API-Details finden Sie VerifySoftwareToken in der AWS SDK for Swift API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
```

```
// build this package.

import PackageDescription

let package = Package(
    name: "cognito-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
            dependencies: [
                .product(name: "AWSCognitoIdentityProvider", package: "aws-sdk-
swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei.

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
```

```
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
```

```
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
```

```
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
```

```
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException {
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
```

```
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient, clientId:
 String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
                print("*** Unable to get the delivery method for the resent code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
            print("*** Unable to resend the confirmation code to user \(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise `false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId: String,
                       userName: String, code: String) async -> Bool {
```

```
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
```

```
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient, authSession:
 String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
```

```
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the app's
MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
```

```
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                     clientId: String, userPoolId: String, mfaCode:
String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
```

```
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that user
in the
```

```
             specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
             user by associating an authenticator application (such as Google
Authenticator
             or a password manager that supports TOTP). Then, the user uses a code
from their
             authenticator application to sign in.

             """)

       let userName = stringRequest("Please enter a new username: ")
       let password = stringRequest("Enter a password: ")
       let email = stringRequest("Enter your email address: ", minLength: 5)

       // Submit the sign-up request to AWS.

       print("==> Signing up user \(userName)...")
       if await signUp(cipClient: cipClient, clientId: clientId,
                       userName: userName, password: password,
                       email: email) == false {
           return
       }

       // Check the user's status. This time, it should come back "unconfirmed".

       print("==> Getting the status of user \(userName) from the user pool (should
be 'unconfirmed')...")
       if await adminGetUser(cipClient: cipClient, userName: userName, userPoolId:
poolId) == false {
           return
       }

       // Ask the user if they want a replacement code sent, such as if the
       // code hasn't arrived yet. If the user responds with a "yes," send a
       // new code.

       if yesNoRequest("==> A confirmation code was sent to \(userName). Would you
like to send a new code (Y/N)? ") {
           print("==> Sending a new confirmation code...")
           if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
               return
           }
       }
```

```
        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId, userName:
userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should be
'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName, userPoolId:
poolId) == false {
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient, authSession:
authSession)
        if newSession == nil {
```

```
            return
        }

        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in your
 authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
 mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")

        let session2 = await initiateAuth(cipClient: cipClient, clientId: clientId,
                                     userName: userName, password: password,
 userPoolId: poolId)
        guard let session2 else {
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a new
 6-digit code, then enter it: ",
                                     minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName: userName,
                                          clientId: clientId, userPoolId: poolId,
                                          mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

DynamoDB-Beispiele unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Eine Swift-Klasse, die DynamoDB-Aufrufe an das SDK für Swift verarbeitet

```
import AWSDynamoDB
import Foundation

/// An enumeration of error codes representing issues that can arise when using
/// the `MovieTable` class.
enum MoviesError: Error {
    /// The specified table wasn't found or couldn't be created.
    case TableNotFound
```

```
    /// The specified item wasn't found or couldn't be created.
    case ItemNotFound
    /// The Amazon DynamoDB client is not properly initialized.
    case UninitializedClient
    /// The table status reported by Amazon DynamoDB is not recognized.
    case StatusUnknown
    /// One or more specified attribute values are invalid or missing.
    case InvalidAttributes
}

/// A class representing an Amazon DynamoDB table containing movie
/// information.
public class MovieTable {
    var ddbClient: DynamoDBClient?
    let tableName: String

    /// Create an object representing a movie table in an Amazon DynamoDB
    /// database.
    ///
    /// - Parameters:
    ///   - region: The optional Amazon Region to create the database in.
    ///   - tableName: The name to assign to the table. If not specified, a
    ///     random table name is generated automatically.
    ///
    /// > Note: The table is not necessarily available when this function
    /// returns. Use `tableExists()` to check for its availability, or
    /// `awaitTableActive()` to wait until the table's status is reported as
    /// ready to use by Amazon DynamoDB.
    ///
    init(region: String? = nil, tableName: String) async throws {
        do {
            let config = try await DynamoDBClient.DynamoDBClientConfiguration()
            if let region = region {
                config.region = region
            }

            self.ddbClient = DynamoDBClient(config: config)
            self.tableName = tableName

            try await self.createTable()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing Amazon DynamoDBClient
 client"))
            throw error
```

```
        }
    }

    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "year",
attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "title",
attributeType: .s)
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }


    /// Check to see if the table exists online yet.
    ///
    /// - Returns: `true` if the table exists, or `false` if not.
    ///
    func tableExists() async throws -> Bool {
```

```
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }

            return description.tableName == self.tableName
        } catch {
            print("ERROR: tableExists:", dump(error))
            throw error
        }
    }


    ///
    /// Waits for the table to exist and for its status to be active.
    ///
    func awaitTableActive() async throws {
        while try (await self.tableExists() == false) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25 seconds
to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }

        while try (await self.getTableStatus() != .active) {
            do {
                let duration = UInt64(0.25 * 1_000_000_000) // Convert .25 seconds
to nanoseconds.
                try await Task.sleep(nanoseconds: duration)
            } catch {
                print("Sleep error:", dump(error))
            }
        }
```

```
    }


    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
        }
    }


    /// Get the table's status.
    ///
    /// - Returns: The table status, as defined by the
    ///   `DynamoDBClientTypes.TableStatus` enum.
    ///
    func getTableStatus() async throws -> DynamoDBClientTypes.TableStatus {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DescribeTableInput(
                tableName: self.tableName
            )
            let output = try await client.describeTable(input: input)
            guard let description = output.table else {
                throw MoviesError.TableNotFound
            }
            guard let status = description.tableStatus else {
                throw MoviesError.StatusUnknown
            }
```

```
            return status
        } catch {
            print("ERROR: getTableStatus:", dump(error))
            throw error
        }
    }


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from: jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and populate
            // them with `PutRequest` requests, each specifying one movie from the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
```

```
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
```

```
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    /// Given a movie's details, add a movie to the Amazon DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title as a `String`.
    ///   - year: The release year of the movie (`Int`).
    ///   - rating: The movie's rating if available (`Double`; default is
    ///     `nil`).
    ///   - plot: A summary of the movie's plot (`String`; default is `nil`,
    ///     indicating no plot summary is available).
    ///
    func add(title: String, year: Int, rating: Double? = nil,
             plot: String? = nil) async throws
    {
        do {
            let movie = Movie(title: title, year: year, rating: rating, plot: plot)
            try await self.add(movie: movie)
        } catch {
            print("ERROR: add with fields:", dump(error))
            throw error
        }
    }


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
```

```
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
```

```
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
"QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
```

```
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]? =
nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []

            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = ScanInput(
                consistentRead: true,
                exclusiveStartKey: startKey,
                expressionAttributeNames: [
                    "#y": "year" // `year` is a reserved word, so use `#y` instead.
                ],
                expressionAttributeValues: [
                    ":y1": .n(String(firstYear)),
                    ":y2": .n(String(lastYear))
                ],
                filterExpression: "#y BETWEEN :y1 AND :y2",
                tableName: self.tableName
            )

            let pages = client.scanPaginated(input: input)

            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Build an array of `Movie` objects for the returned items.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
```

```
            return movieList

        } catch {
            print("ERROR: getMovies with scan:", dump(error))
            throw error
        }
    }


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.

            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] = [:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
```

```
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String: DynamoDBClientTypes.AttributeValue]
= output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }


    /// Delete a movie, given its title and release year.
    ///
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
```

```
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
}
```

Die Strukturen, die von der MovieTable Klasse zur Darstellung von Filmen verwendet werden.

```
import Foundation
import AWSDynamoDB

/// The optional details about a movie.
public struct Details: Codable {
    /// The movie's rating, if available.
    var rating: Double?
    /// The movie's plot, if available.
    var plot: String?
}

/// A structure describing a movie. The `year` and `title` properties are
/// required and are used as the key for Amazon DynamoDB operations. The
/// `info` sub-structure's two properties, `rating` and `plot`, are optional.
public struct Movie: Codable {
    /// The year in which the movie was released.
    var year: Int
    /// The movie's title.
    var title: String
    /// A `Details` object providing the optional movie rating and plot
    /// information.
    var info: Details

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
```

```
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - rating: The movie's rating (optional `Double`).
    ///   - plot: The movie's plot (optional `String`)
    init(title: String, year: Int, rating: Double? = nil, plot: String? = nil) {
        self.title = title
        self.year = year

        self.info = Details(rating: rating, plot: plot)
    }

    /// Create a `Movie` object representing a movie, given the movie's
    /// details.
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The year in which the movie was released (`Int`).
    ///   - info: The optional rating and plot information for the movie in a
    ///     `Details` object.
    init(title: String, year: Int, info: Details?){
        self.title = title
        self.year = year

        if info != nil {
            self.info = info!
        } else {
            self.info = Details(rating: nil, plot: nil)
        }
    }

    ///
    /// Return a new `MovieTable` object, given an array mapping string to Amazon
    /// DynamoDB attribute values.
    ///
    /// - Parameter item: The item information provided to the form used by
    ///   DynamoDB. This is an array of strings mapped to
    ///   `DynamoDBClientTypes.AttributeValue` values.
    init(withItem item: [Swift.String:DynamoDBClientTypes.AttributeValue]) throws  {
        // Read the attributes.

        guard let titleAttr = item["title"],
              let yearAttr = item["year"] else {
            throw MoviesError.ItemNotFound
```

```
        }
        let infoAttr = item["info"] ?? nil

        // Extract the values of the title and year attributes.

        if case .s(let titleVal) = titleAttr {
            self.title = titleVal
        } else {
            throw MoviesError.InvalidAttributes
        }

        if case .n(let yearVal) = yearAttr {
            self.year = Int(yearVal)!
        } else {
            throw MoviesError.InvalidAttributes
        }

        // Extract the rating and/or plot from the `info` attribute, if
        // they're present.

        var rating: Double? = nil
        var plot: String? = nil

        if infoAttr != nil, case .m(let infoVal) = infoAttr {
            let ratingAttr = infoVal["rating"] ?? nil
            let plotAttr = infoVal["plot"] ?? nil

            if ratingAttr != nil, case .n(let ratingVal) = ratingAttr {
                rating = Double(ratingVal) ?? nil
            }
            if plotAttr != nil, case .s(let plotVal) = plotAttr {
                plot = plotVal
            }
        }

        self.info = Details(rating: rating, plot: plot)
    }

    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
```

```
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
 [Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]

        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
 }
```

Ein Programm, das die MovieTable Klasse für den Zugriff auf eine DynamoDB-Datenbank verwendet.

```
import ArgumentParser
import ClientRuntime
import Foundation

import AWSDynamoDB


@testable import MovieList
```

```
extension String {
    // Get the directory if the string is a file path.
    func directory() -> String {
        guard let lastIndex = lastIndex(of: "/") else {
            print("Error: String directory separator not found.")
            return ""
        }
        return String(self[...lastIndex])
    }
}

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The path of the sample movie data JSON file.")
    var jsonPath: String = #file.directory() + "../../../../../resources/
sample_files/movies.json"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion: String?

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    /// Configuration details for the command.
    static var configuration = CommandConfiguration(
        commandName: "basics",
        abstract: "A basic scenario demonstrating the usage of Amazon DynamoDB.",
        discussion: """
        An example showing how to use Amazon DynamoDB to perform a series of
        common database activities on a simple movie database.
        """
    )

    /// Called by ``main()`` to asynchronously run the AWS example.
```

```
    func runAsync() async throws {
        print("Welcome to the AWS SDK for Swift basic scenario for Amazon
DynamoDB!")

        //=====================================================================
        // 1. Create the table. The Amazon DynamoDB table is represented by
        //    the `MovieTable` class.
        //=====================================================================

        let tableName = "ddb-movies-sample-\(Int.random(in: 1 ... Int.max))"

        print("Creating table \"\(tableName)\"...")

        let movieDatabase = try await MovieTable(region: awsRegion,
                                                 tableName: tableName)

        print("\nWaiting for table to be ready to use...")
        try await movieDatabase.awaitTableActive()

        //=====================================================================
        // 2. Add a movie to the table.
        //=====================================================================

        print("\nAdding a movie...")
        try await movieDatabase.add(title: "Avatar: The Way of Water", year: 2022)
        try await movieDatabase.add(title: "Not a Real Movie", year: 2023)

        //=====================================================================
        // 3. Update the plot and rating of the movie using an update
        //    expression.
        //=====================================================================

        print("\nAdding details to the added movie...")
        _ = try await movieDatabase.update(title: "Avatar: The Way of Water", year:
2022,
                                           rating: 9.2, plot: "It's a sequel.")

        //=====================================================================
        // 4. Populate the table from the JSON file.
        //=====================================================================

        print("\nPopulating the movie database from JSON...")
        try await movieDatabase.populate(jsonPath: jsonPath)
```

```
        //=====================================================================
        // 5. Get a specific movie by key. In this example, the key is a
        //    combination of `title` and `year`.
        //=====================================================================

        print("\nLooking for a movie in the table...")
        let gotMovie = try await movieDatabase.get(title: "This Is the End", year:
2013)

        print("Found the movie \"\(gotMovie.title)\", released in
\(gotMovie.year).")
        print("Rating: \(gotMovie.info.rating ?? 0.0).")
        print("Plot summary: \(gotMovie.info.plot ?? "None.")")

        //=====================================================================
        // 6. Delete a movie.
        //=====================================================================

        print("\nDeleting the added movie...")
        try await movieDatabase.delete(title: "Avatar: The Way of Water", year:
2022)

        //=====================================================================
        // 7. Use a query with a key condition expression to return all movies
        //    released in a given year.
        //=====================================================================

        print("\nGetting movies released in 1994...")
        let movieList = try await movieDatabase.getMovies(fromYear: 1994)
        for movie in movieList {
            print("    \(movie.title)")
        }

        //=====================================================================
        // 8. Use `scan()` to return movies released in a range of years.
        //=====================================================================

        print("\nGetting movies released between 1993 and 1997...")
        let scannedMovies = try await movieDatabase.getMovies(firstYear: 1993,
lastYear: 1997)
        for movie in scannedMovies {
            print("    \(movie.title) (\(movie.year))")
        }
```

```
        //=====================================================================
        // 9. Delete the table.
        //=====================================================================

        print("\nDeleting the table...")
        try await movieDatabase.deleteTable()
    }
}

@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

Aktionen

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung. `BatchGetItem`

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Gets an array of `Movie` objects describing all the movies in the
    /// specified list. Any movies that aren't found in the list have no
    /// corresponding entry in the resulting array.
    ///
    /// - Parameters
    ///     - keys: An array of tuples, each of which specifies the title and
    ///       release year of a movie to fetch from the table.
    ///
    /// - Returns:
    ///     - An array of `Movie` objects describing each match found in the
    ///     table.
    ///
    /// - Throws:
    ///     - `MovieError.ClientUninitialized` if the DynamoDB client has not
    ///     been initialized.
    ///     - DynamoDB errors are thrown without change.
    func batchGet(keys: [(title: String, year: Int)]) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MovieError.ClientUninitialized
            }

            var movieList: [Movie] = []
            var keyItems: [[Swift.String: DynamoDBClientTypes.AttributeValue]] = []

            // Convert the list of keys into the form used by DynamoDB.
```

```
            for key in keys {
                let item: [Swift.String: DynamoDBClientTypes.AttributeValue] = [
                    "title": .s(key.title),
                    "year": .n(String(key.year))
                ]
                keyItems.append(item)
            }

            // Create the input record for `batchGetItem()`. The list of requested
            // items is in the `requestItems` property. This array contains one
            // entry for each table from which items are to be fetched. In this
            // example, there's only one table containing the movie data.
            //
            // If we wanted this program to also support searching for matches
            // in a table of book data, we could add a second `requestItem`
            // mapping the name of the book table to the list of items we want to
            // find in it.
            let input = BatchGetItemInput(
                requestItems: [
                    self.tableName: .init(
                        consistentRead: true,
                        keys: keyItems
                    )
                ]
            )

            // Fetch the matching movies from the table.

            let output = try await client.batchGetItem(input: input)

            // Get the set of responses. If there aren't any, return the empty
            // movie list.

            guard let responses = output.responses else {
                return movieList
            }

            // Get the list of matching items for the table with the name
            // `tableName`.

            guard let responseList = responses[self.tableName] else {
                return movieList
            }
```

```
            // Create `Movie` items for each of the matching movies in the table
            // and add them to the `MovieList` array.

            for response in responseList {
                try movieList.append(Movie(withItem: response))
            }

            return movieList
        } catch {
            print("ERROR: batchGet", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie BatchGetItem in der AWS SDK for Swift API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Populate the movie database from the specified JSON file.
    ///
    /// - Parameter jsonPath: Path to a JSON file containing movie data.
    ///
    func populate(jsonPath: String) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }
```

```
            // Create a Swift `URL` and use it to load the file into a `Data`
            // object. Then decode the JSON into an array of `Movie` objects.

            let fileUrl = URL(fileURLWithPath: jsonPath)
            let jsonData = try Data(contentsOf: fileUrl)

            var movieList = try JSONDecoder().decode([Movie].self, from: jsonData)

            // Truncate the list to the first 200 entries or so for this example.

            if movieList.count > 200 {
                movieList = Array(movieList[...199])
            }

            // Before sending records to the database, break the movie list into
            // 25-entry chunks, which is the maximum size of a batch item request.

            let count = movieList.count
            let chunks = stride(from: 0, to: count, by: 25).map {
                Array(movieList[$0 ..< Swift.min($0 + 25, count)])
            }

            // For each chunk, create a list of write request records and populate
            // them with `PutRequest` requests, each specifying one movie from the
            // chunk. Once the chunk's items are all in the `PutRequest` list,
            // send them to Amazon DynamoDB using the
            // `DynamoDBClient.batchWriteItem()` function.

            for chunk in chunks {
                var requestList: [DynamoDBClientTypes.WriteRequest] = []

                for movie in chunk {
                    let item = try await movie.getAsItem()
                    let request = DynamoDBClientTypes.WriteRequest(
                        putRequest: .init(
                            item: item
                        )
                    )
                    requestList.append(request)
                }

                let input = BatchWriteItemInput(requestItems: [tableName:
requestList])
```

```
                _ = try await client.batchWriteItem(input: input)
            }
        } catch {
            print("ERROR: populate:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie BatchWriteItem in der AWS SDK for Swift API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    ///
    /// Create a movie table in the Amazon DynamoDB data store.
    ///
    private func createTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = CreateTableInput(
                attributeDefinitions: [
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "year",
 attributeType: .n),
                    DynamoDBClientTypes.AttributeDefinition(attributeName: "title",
 attributeType: .s)
```

```
                ],
                billingMode: DynamoDBClientTypes.BillingMode.payPerRequest,
                keySchema: [
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "year",
keyType: .hash),
                    DynamoDBClientTypes.KeySchemaElement(attributeName: "title",
keyType: .range)
                ],
                tableName: self.tableName
            )
            let output = try await client.createTable(input: input)
            if output.tableDescription == nil {
                throw MoviesError.TableNotFound
            }
        } catch {
            print("ERROR: createTable:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreateTable in der AWS SDK for Swift API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Delete a movie, given its title and release year.
    ///
```

```
    /// - Parameters:
    ///   - title: The movie's title.
    ///   - year: The movie's release year.
    ///
    func delete(title: String, year: Int) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            _ = try await client.deleteItem(input: input)
        } catch {
            print("ERROR: delete:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteItem in der AWS SDK for Swift API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB
```

```
    ///
    /// Deletes the table from Amazon DynamoDB.
    ///
    func deleteTable() async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = DeleteTableInput(
                tableName: self.tableName
            )
            _ = try await client.deleteTable(input: input)
        } catch {
            print("ERROR: deleteTable:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteTable in der AWS SDK for Swift API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Return a `Movie` record describing the specified movie from the Amazon
    /// DynamoDB table.
```

```
    ///
    /// - Parameters:
    ///   - title: The movie's title (`String`).
    ///   - year: The movie's release year (`Int`).
    ///
    /// - Throws: `MoviesError.ItemNotFound` if the movie isn't in the table.
    ///
    /// - Returns: A `Movie` record with the movie's details.
    func get(title: String, year: Int) async throws -> Movie {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = GetItemInput(
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                tableName: self.tableName
            )
            let output = try await client.getItem(input: input)
            guard let item = output.item else {
                throw MoviesError.ItemNotFound
            }

            let movie = try Movie(withItem: item)
            return movie
        } catch {
            print("ERROR: get:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie GetItem in der AWS SDK for Swift API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Get a list of the DynamoDB tables available in the specified Region.
    ///
    /// - Returns: An array of strings listing all of the tables available
    ///   in the Region specified when the session was created.
    public func getTableList() async throws -> [String] {
        let input = ListTablesInput(
        )
        return try await session.listTables(input: input)
    }
```

- API-Details finden Sie ListTables in der AWS SDK for Swift API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Add a movie specified as a `Movie` structure to the Amazon DynamoDB
```

```
    /// table.
    ///
    /// - Parameter movie: The `Movie` to add to the table.
    ///
    func add(movie: Movie) async throws {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Get a DynamoDB item containing the movie data.
            let item = try await movie.getAsItem()

            // Send the `PutItem` request to Amazon DynamoDB.

            let input = PutItemInput(
                item: item,
                tableName: self.tableName
            )
            _ = try await client.putItem(input: input)
        } catch {
            print("ERROR: add movie:", dump(error))
            throw error
        }
    }


    ///
    /// Return an array mapping attribute names to Amazon DynamoDB attribute
    /// values, representing the contents of the `Movie` record as a DynamoDB
    /// item.
    ///
    /// - Returns: The movie item as an array of type
    ///   `[Swift.String:DynamoDBClientTypes.AttributeValue]`.
    ///
    func getAsItem() async throws ->
[Swift.String:DynamoDBClientTypes.AttributeValue]  {
        // Build the item record, starting with the year and title, which are
        // always present.

        var item: [Swift.String:DynamoDBClientTypes.AttributeValue] = [
            "year": .n(String(self.year)),
            "title": .s(self.title)
        ]
```

```
        // Add the `info` field with the rating and/or plot if they're
        // available.

        var details: [Swift.String:DynamoDBClientTypes.AttributeValue] = [:]
        if (self.info.rating != nil || self.info.plot != nil) {
            if self.info.rating != nil {
                details["rating"] = .n(String(self.info.rating!))
            }
            if self.info.plot != nil {
                details["plot"] = .s(self.info.plot!)
            }
        }
        item["info"] = .m(details)

        return item
    }
```

- API-Details finden Sie PutItem in der AWS SDK for Swift API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Get all the movies released in the specified year.
    ///
    /// - Parameter year: The release year of the movies to return.
    ///
    /// - Returns: An array of `Movie` objects describing each matching movie.
    ///
```

```
    func getMovies(fromYear year: Int) async throws -> [Movie] {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            let input = QueryInput(
                expressionAttributeNames: [
                    "#y": "year"
                ],
                expressionAttributeValues: [
                    ":y": .n(String(year))
                ],
                keyConditionExpression: "#y = :y",
                tableName: self.tableName
            )
            // Use "Paginated" to get all the movies.
            // This lets the SDK handle the 'lastEvaluatedKey' property in
"QueryOutput".

            let pages = client.queryPaginated(input: input)

            var movieList: [Movie] = []
            for try await page in pages {
                guard let items = page.items else {
                    print("Error: no items returned.")
                    continue
                }

                // Convert the found movies into `Movie` objects and return an array
                // of them.

                for item in items {
                    let movie = try Movie(withItem: item)
                    movieList.append(movie)
                }
            }
            return movieList
        } catch {
            print("ERROR: getMovies:", dump(error))
            throw error
        }
    }
```

- Detaillierte API-Informationen finden Sie unter Abfragen in der API-Referenz zum AWS SDK für Swift.

`Scan`

Das folgende Codebeispiel zeigt die Verwendung`Scan`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Return an array of `Movie` objects released in the specified range of
    /// years.
    ///
    /// - Parameters:
    ///   - firstYear: The first year of movies to return.
    ///   - lastYear: The last year of movies to return.
    ///   - startKey: A starting point to resume processing; always use `nil`.
    ///
    /// - Returns: An array of `Movie` objects describing the matching movies.
    ///
    /// > Note: The `startKey` parameter is used by this function when
    ///   recursively calling itself, and should always be `nil` when calling
    ///   directly.
    ///
    func getMovies(firstYear: Int, lastYear: Int,
                   startKey: [Swift.String: DynamoDBClientTypes.AttributeValue]? =
 nil)
        async throws -> [Movie]
    {
        do {
            var movieList: [Movie] = []
```

```
        guard let client = self.ddbClient else {
            throw MoviesError.UninitializedClient
        }

        let input = ScanInput(
            consistentRead: true,
            exclusiveStartKey: startKey,
            expressionAttributeNames: [
                "#y": "year" // `year` is a reserved word, so use `#y` instead.
            ],
            expressionAttributeValues: [
                ":y1": .n(String(firstYear)),
                ":y2": .n(String(lastYear))
            ],
            filterExpression: "#y BETWEEN :y1 AND :y2",
            tableName: self.tableName
        )

        let pages = client.scanPaginated(input: input)

        for try await page in pages {
            guard let items = page.items else {
                print("Error: no items returned.")
                continue
            }

            // Build an array of `Movie` objects for the returned items.

            for item in items {
                let movie = try Movie(withItem: item)
                movieList.append(movie)
            }
        }
        return movieList

    } catch {
        print("ERROR: getMovies with scan:", dump(error))
        throw error
    }
}
```

- Detaillierte API-Informationen finden Sie unter Scan in der API-Referenz zum AWS -SDK für Swift.

`UpdateItem`

Das folgende Codebeispiel zeigt die Verwendung`UpdateItem`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSDynamoDB


    /// Update the specified movie with new `rating` and `plot` information.
    ///
    /// - Parameters:
    ///   - title: The title of the movie to update.
    ///   - year: The release year of the movie to update.
    ///   - rating: The new rating for the movie.
    ///   - plot: The new plot summary string for the movie.
    ///
    /// - Returns: An array of mappings of attribute names to their new
    ///   listing each item actually changed. Items that didn't need to change
    ///   aren't included in this list. `nil` if no changes were made.
    ///
    func update(title: String, year: Int, rating: Double? = nil, plot: String? =
 nil) async throws
        -> [Swift.String: DynamoDBClientTypes.AttributeValue]?
    {
        do {
            guard let client = self.ddbClient else {
                throw MoviesError.UninitializedClient
            }

            // Build the update expression and the list of expression attribute
            // values. Include only the information that's changed.
```

```
            var expressionParts: [String] = []
            var attrValues: [Swift.String: DynamoDBClientTypes.AttributeValue] = [:]

            if rating != nil {
                expressionParts.append("info.rating=:r")
                attrValues[":r"] = .n(String(rating!))
            }
            if plot != nil {
                expressionParts.append("info.plot=:p")
                attrValues[":p"] = .s(plot!)
            }
            let expression = "set \(expressionParts.joined(separator: ", "))"

            let input = UpdateItemInput(
                // Create substitution tokens for the attribute values, to ensure
                // no conflicts in expression syntax.
                expressionAttributeValues: attrValues,
                // The key identifying the movie to update consists of the release
                // year and title.
                key: [
                    "year": .n(String(year)),
                    "title": .s(title)
                ],
                returnValues: .updatedNew,
                tableName: self.tableName,
                updateExpression: expression
            )
            let output = try await client.updateItem(input: input)

            guard let attributes: [Swift.String: DynamoDBClientTypes.AttributeValue]
 = output.attributes else {
                throw MoviesError.InvalidAttributes
            }
            return attributes
        } catch {
            print("ERROR: update:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie UpdateItem in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon EC2 unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon EC2 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Amazon EC2

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon EC2.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
```

```
// build this package.

import PackageDescription

let package = Package(
    name: "hello-ec2",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "hello-ec2",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Die Datei `entry.swift`

```
// An example that shows how to use the AWS SDK for Swift to perform a simple
// operation using Amazon Elastic Compute Cloud (EC2).
//
```

```
import ArgumentParser
import Foundation

import AWSEC2

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "hello-ec2",
        abstract: """
        Demonstrates a simple operation using Amazon EC2.
        """,
        discussion: """
        An example showing how to make a call to Amazon EC2 using the AWS SDK for
 Swift.
        """
    )

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
```

```
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let groupNames = await getSecurityGroupNames(ec2Client: ec2Client)

        print("Found \(groupNames.count) security group(s):")

        for group in groupNames {
            print("    \(group)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
```

```
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Swift API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie anschließend eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Stellen Sie über SSH eine Verbindung zu Ihrer Instance her und bereinigen Sie dann die Ressourcen.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
```

```
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "ec2-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "ec2-scenario",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "AWSSSM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Die Datei `entry.swift`

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation
import AWSEC2

// Allow waiters to be used.

import class SmithyWaitersAPI.Waiter
import struct SmithyWaitersAPI.WaiterOptions

import AWSSSM

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "ec2-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon EC2 using the
        AWS SDK for Swift.
        """,
        discussion: """
        """
    )
```

```
    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ssmConfig = try await SSMClient.SSMClientConfiguration(region:
 awsRegion)
        let ssmClient = SSMClient(config: ssmConfig)

        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
 awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let example = Example(ec2Client: ec2Client, ssmClient: ssmClient)

        await example.run()
    }
}

class Example {
    let ec2Client: EC2Client
    let ssmClient: SSMClient

    // Storage for AWS EC2 properties.

    var keyName: String? = nil
    var securityGroupId: String? = nil
    var instanceId: String? = nil
    var allocationId: String? = nil
    var associationId: String? = nil

    init(ec2Client: EC2Client, ssmClient: SSMClient) {
        self.ec2Client = ec2Client
        self.ssmClient = ssmClient
    }

    /// The example's main body.
    func run() async {
        //=====================================================================
        // 1. Create an RSA key pair, saving the private key as a `.pem` file.
        //    Create a `defer` block that will delete the private key when the
        //    program exits.
        //=====================================================================

        print("Creating an RSA key pair...")

        keyName = self.tempName(prefix: "ExampleKeyName")
```

```
        let keyUrl = await self.createKeyPair(name: keyName!)

        guard let keyUrl else {
            print("*** Failed to create the key pair!")
            return
        }

        print("Created the private key at: \(keyUrl.absoluteString)")

        // Schedule deleting the private key file to occur automatically when
        // the program exits, no matter how it exits.

        defer {
            do {
                try FileManager.default.removeItem(at: keyUrl)
            } catch {
                print("*** Failed to delete the private key at
\(keyUrl.absoluteString)")
            }
        }

        //=====================================================================
        // 2. List the key pairs by calling `DescribeKeyPairs`.
        //=====================================================================

        print("Describing available key pairs...")
        await self.describeKeyPairs()

        //=====================================================================
        // 3. Create a security group for the default VPC, and add an inbound
        //    rule to allow SSH from the current computer's public IPv4
        //    address.
        //=====================================================================

        print("Creating the security group...")

        let secGroupName = self.tempName(prefix: "ExampleSecurityGroup")
        let ipAddress = self.getMyIPAddress()

        guard let ipAddress else {
            print("*** Unable to get the device's IP address.")
            return
        }
```

```
        print("IP address is: \(ipAddress)")

        securityGroupId = await self.createSecurityGroup(
            name: secGroupName,
            description: "An example security group created using the AWS SDK for
Swift"
        )

        if securityGroupId == nil {
            await cleanUp()
            return
        }

        print("Created security group: \(securityGroupId ?? "<unknown>")")

        if !(await self.authorizeSecurityGroupIngress(groupId: securityGroupId!,
ipAddress: ipAddress)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 4. Display security group information for the new security group
        //    using DescribeSecurityGroups.
        //=====================================================================

        if !(await self.describeSecurityGroups(groupId: securityGroupId!)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 5. Get a list of Amazon Linux 2023 AMIs and pick one (SSM is the
        //    best practice), using path and then filter the list after the
        //    fact to include "al2023" in the Name field
        //    (ssm.GetParametersByPath). Paginate to get all images.
        //=====================================================================

        print("Searching available images for Amazon Linux 2023 images...")

        let options = await self.findAMIsMatchingFilter("al2023")

        //=====================================================================
        // 6. The information in the AMI options isn't great, so make a list
```

```
        //    of the image IDs (the "Value" field in the AMI options) and get
        //    more information about them from EC2. Display the Description
        //    field and select one of them (DescribeImages with ImageIds
        //    filter).
        //=====================================================================

        print("Images matching Amazon Linux 2023:")

        var imageIds: [String] = []
        for option in options {
            guard let id = option.value else {
                continue
            }
            imageIds.append(id)
        }

        let images = await self.describeImages(imageIds)

        // This is where you would normally let the user choose which AMI to
        // use. However, for this example, we're just going to use the first
        // one, whatever it is.

        let chosenImage = images[0]

        //=====================================================================
        // 7. Get a list of instance types that are compatible with the
        //    selected AMI's architecture (such as "x86_64") and are either
        //    small or micro. Select one (DescribeInstanceTypes).
        //=====================================================================

        print("Getting the instance types compatible with the selected image...")

        guard let arch = chosenImage.architecture else {
            print("*** The selected image doesn't have a valid architecture.")
            await cleanUp()
            return
        }

        let imageTypes = await self.getMatchingInstanceTypes(architecture: arch)

        for type in imageTypes {
            guard let instanceType = type.instanceType else {
                continue
            }
```

```
            print("    \(instanceType.rawValue)")
        }

        // This example selects the first returned instance type. A real-world
        // application would probably ask the user to select one here.

        let chosenInstanceType = imageTypes[0]

        //=====================================================================
        // 8. Create an instance with the key pair, security group, AMI, and
        //    instance type (RunInstances).
        //=====================================================================

        print("Creating an instance...")

        guard let imageId = chosenImage.imageId else {
            print("*** Cannot start image without a valid image ID.")
            await cleanUp()
            return
        }
        guard let instanceType = chosenInstanceType.instanceType else {
            print("*** Unable to start image without a valid image type.")
            await cleanUp()
            return
        }

        let instance = await self.runInstance(
            imageId: imageId,
            instanceType: instanceType,
            keyPairName: keyName!,
            securityGroups: [securityGroupId!]
        )

        guard let instance else {
            await cleanUp()
            return
        }

        instanceId = instance.instanceId
        if instanceId == nil {
            print("*** Instance is missing an ID. Canceling.")
            await cleanUp()
            return
        }
```

```
        //=====================================================================
        // 9. Wait for the instance to be ready and then display its
        //    information (DescribeInstances).
        //=====================================================================

        print("Waiting a few seconds to let the instance come up...")

        do {
            try await Task.sleep(for: .seconds(20))
        } catch {
            print("*** Error pausing the task.")
        }
        print("Success! Your new instance is ready:")

        //=====================================================================
        // 10. Display SSH connection info for the instance.
        //=====================================================================

        var runningInstance = await self.describeInstance(instanceId: instanceId!)

        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command:")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 11. Stop the instance and wait for it to stop (StopInstances).
        //=====================================================================

        print("Stopping the instance...")

        if !(await self.stopInstance(instanceId: instanceId!, waitUntilStopped:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Start the instance and wait for it to start (StartInstances).
        //=====================================================================

        print("Starting the instance again...")
```

```
        if !(await self.startInstance(instanceId: instanceId!, waitUntilStarted:
 true)) {
            await cleanUp()
            return
        }

        //=====================================================================
        // 13. Display SSH connection info for the instance. Note that it's
        //     changed.
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command.")
            print("This is probably different from when the instance was running
 before.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // 14. Allocate an elastic IP and associate it with the instance
        //     (AllocateAddress and AssociateAddress).
        //=====================================================================

        allocationId = await self.allocateAddress()

        if allocationId == nil {
            await cleanUp()
            return
        }

        associationId = await self.associateAddress(instanceId: instanceId!,
 allocationId: allocationId)

        if associationId == nil {
            await cleanUp()
            return
        }

        //=====================================================================
        // 15. Display SSH connection info for the connection. Note that the
        //     public IP is now the Elastic IP, which stays constant.
```

```
        //=====================================================================

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil) {
            print("\nYou can SSH to this instance using the following command.")
            print("This has changed again, and is now the Elastic IP.")
            print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
        }

        //=====================================================================
        // Handle all cleanup tasks
        //=====================================================================

        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items:
    ///
    /// * Elastic IP allocation and association
    /// * Terminate the instance
    /// * Delete the security group
    /// * Delete the key pair
    func cleanUp() async {
        //=====================================================================
        // 16. Disassociate and delete the Elastic IP (DisassociateAddress and
        //     ReleaseAddress).
        //=====================================================================

        if associationId != nil {
            await self.disassociateAddress(associationId: associationId!)
        }

        if allocationId != nil {
            await self.releaseAddress(allocationId: allocationId!)
        }

        //=====================================================================
        // 17. Terminate the instance and wait for it to terminate
        //     (TerminateInstances).
        //=====================================================================

        if instanceId != nil {
            print("Terminating the instance...")
```

```
            _ = await self.terminateInstance(instanceId: instanceId!,
waitUntilTerminated: true)
        }

        //=====================================================================
        // 18. Delete the security group (DeleteSecurityGroup).
        //=====================================================================

        if securityGroupId != nil {
            print("Deleting the security group...")
            _ = await self.deleteSecurityGroup(groupId: securityGroupId!)
        }

        //=====================================================================
        // 19. Delete the key pair (DeleteKeyPair).
        //=====================================================================

        if keyName != nil {
            print("Deleting the key pair...")
            _ = await self.deleteKeyPair(keyPair: keyName!)
        }
    }

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.
```

```
            let fileURL = URL.temporaryDirectory
                                .appendingPathComponent(name)
                                .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )

            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":", keyPair.keyFingerprint ??
"<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
```

```
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
            print("*** Error deleting the key pair: \(error.localizedDescription)")
            return false
        }
    }

    /// Return a list of AMI names that contain the specified string.
    ///
    /// - Parameter filter: A string that must be contained in all returned
    ///   AMI names.
    ///
    /// - Returns: An array of the parameters matching the specified substring.
    func findAMIsMatchingFilter(_ filter: String) async ->
[SSMClientTypes.Parameter] {
        var parameterList: [SSMClientTypes.Parameter] = []
        var matchingAMIs: [SSMClientTypes.Parameter] = []

        do {
            let pages = ssmClient.getParametersByPathPaginated(
                input: GetParametersByPathInput(
                    path: "/aws/service/ami-amazon-linux-latest"
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return matchingAMIs
                }

                for parameter in parameters {
                    parameterList.append(parameter)
                }
            }
```

```
            print("Found \(parameterList.count) images total:")
            for parameter in parameterList {
                guard let name = parameter.name else {
                    continue
                }
                print("    \(name)")

                if name.contains(filter) {
                    matchingAMIs.append(parameter)
                }
            }
        } catch {
            return matchingAMIs
        }

        return matchingAMIs
    }

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues =
EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )
```

```
        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }

    /// Get the latest information about the specified instance and output it
    /// to the screen, returning the instance details to the caller.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to provide details about.
    ///   - stateFilter: The state to require the instance to be in.
    ///
    /// - Returns: The instance's details as an `EC2ClientTypes.Instance` object.
    func describeInstance(instanceId: String,
                          stateFilter: EC2ClientTypes.InstanceStateName? =
EC2ClientTypes.InstanceStateName.running) async
                          -> EC2ClientTypes.Instance? {
        do {
            let pages = ec2Client.describeInstancesPaginated(
                input: DescribeInstancesInput(
                    instanceIds: [instanceId]
                )
            )

            for try await page in pages {
                guard let reservations = page.reservations else {
```

```
                    continue
                }

                for reservation in reservations {
                    guard let instances = reservation.instances else {
                        continue
                    }

                    for instance in instances {
                        guard let state = instance.state else {
                            print("*** Instance is missing its state...")
                            continue
                        }
                        let instanceState = state.name

                        if stateFilter != nil && (instanceState != stateFilter) {
                            continue
                        }

                        let instanceTypeName: String
                        if instance.instanceType == nil {
                            instanceTypeName = "<N/A>"
                        } else {
                            instanceTypeName = instance.instanceType?.rawValue ??
"<N/A>"
                        }

                        let instanceStateName: String
                        if instanceState == nil {
                            instanceStateName = "<N/A>"
                        } else {
                            instanceStateName = instanceState?.rawValue ?? "<N/A>"
                        }

                        print("""
                        Instance: \(instance.instanceId ?? "<N/A>")
                                • Image ID: \(instance.imageId ?? "<N/A>")
                                • Instance type: \(instanceTypeName)
                                • Key name: \(instance.keyName ?? "<N/A>")
                                • VPC ID: \(instance.vpcId ?? "<N/A>")
                                • Public IP: \(instance.publicIpAddress ?? "N/A")
                                • State: \(instanceStateName)
                        """)
```

```
                        return instance
                    }
                }
            }
        } catch {
            print("*** Error retrieving instance information to display:
\(error.localizedDescription)")
            return nil
        }

        return nil
    }

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
```

```
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance: \(error.localizedDescription)")
            return false
        }
    }

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
```

```
                        instanceIds: instanceList
                    )
                )
                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance: \(error.localizedDescription)")
            return false
        }
    }

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
```

```
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
\(error.localizedDescription)")
            return false
        }
    }

    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
                    continue
```

```
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
\(error.localizedDescription)")
            return []
        }
    }

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
```

```
            print("*** Error creating the instance: \(error.localizedDescription)")
            return nil
        }
    }

    /// Return the device's external IP address.
    ///
    /// - Returns: A string containing the device's IP address.
    func getMyIPAddress() -> String? {
        guard let url = URL(string: "http://checkip.amazonaws.com") else {
            print("Couldn't create the URL")
            return nil
        }

        do {
            print("Getting the IP address...")
            return try String(contentsOf: url, encoding:
String.Encoding.utf8).trim()
        } catch {
            print("*** Unable to get your public IP address.")
            return nil
        }
    }

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description groupDescription:
String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )

            return output.groupId
        } catch {
```

```
            print("*** Error creating the security group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async ->
Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
```

```
    }

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
\(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
\(error.localizedDescription)")
            return false
        }
    }

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
\(error.localizedDescription)")
```

```
            return false
        }
    }

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )
```

```
            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
            print("*** Unable to disassociate the IP address:
\(error.localizedDescription)")
        }
    }

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
\(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
```

```
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup

- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Allocate an Elastic IP address.
    ///
    /// - Returns: A string containing the ID of the Elastic IP.
    func allocateAddress() async -> String? {
        do {
            let output = try await ec2Client.allocateAddress(
                input: AllocateAddressInput(
```

```
                    domain: EC2ClientTypes.DomainType.vpc
                )
            )

            guard let allocationId = output.allocationId else {
                return nil
            }

            return allocationId
        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie AllocateAddress in der AWS SDK for Swift API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
```

```
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie AssociateAddress in der AWS SDK for Swift API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Authorize ingress of connections for the security group.
    ///
    /// - Parameters:
    ///   - groupId: The group ID of the security group to authorize access for.
    ///   - ipAddress: The IP address of the device to grant access to.
    ///
```

```
    /// - Returns: `true` if access is successfully granted; otherwise `false`.
    func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async ->
Bool {
        let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
        let httpPermission = EC2ClientTypes.IpPermission(
            fromPort: 80,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 80
        )

        let sshPermission = EC2ClientTypes.IpPermission(
            fromPort: 22,
            ipProtocol: "tcp",
            ipRanges: [ipRange],
            toPort: 22
        )

        do {
            _ = try await ec2Client.authorizeSecurityGroupIngress(
                input: AuthorizeSecurityGroupIngressInput(
                    groupId: groupId,
                    ipPermissions: [httpPermission, sshPermission]
                )
            )

            return true
        } catch {
            print("*** Error authorizing ingress for the security group:
\(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie AuthorizeSecurityGroupIngress in der AWS SDK for Swift API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create a new RSA key pair and save the private key to a randomly-named
    /// file in the temporary directory.
    ///
    /// - Parameter name: The name of the key pair to create.
    ///
    /// - Returns: The URL of the newly created `.pem` file or `nil` if unable
    ///   to create the key pair.
    func createKeyPair(name: String) async -> URL? {
        do {
            let output = try await ec2Client.createKeyPair(
                input: CreateKeyPairInput(
                    keyName: name
                )
            )

            guard let keyMaterial = output.keyMaterial else {
                return nil
            }

            // Build the URL of the temporary private key file.

            let fileURL = URL.temporaryDirectory
                                 .appendingPathComponent(name)
                                 .appendingPathExtension("pem")

            do {
                try keyMaterial.write(to: fileURL, atomically: true, encoding:
 String.Encoding.utf8)
                return fileURL
            } catch {
                print("*** Failed to write the private key.")
                return nil
```

```
            }
        } catch {
            print("*** Unable to create the key pair.")
            return nil
        }
    }
```

- API-Details finden Sie CreateKeyPair in der AWS SDK for Swift API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create a new security group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the group to create.
    ///   - groupDescription: A description of the new security group.
    ///
    /// - Returns: The ID string of the new security group.
    func createSecurityGroup(name groupName: String, description groupDescription:
 String) async -> String? {
        do {
            let output = try await ec2Client.createSecurityGroup(
                input: CreateSecurityGroupInput(
                    description: groupDescription,
                    groupName: groupName
                )
            )
```

```
            return output.groupId
        } catch {
            print("*** Error creating the security group:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie CreateSecurityGroup in der AWS SDK for Swift API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Delete an EC2 key pair.
    ///
    /// - Parameter keyPair: The name of the key pair to delete.
    ///
    /// - Returns: `true` if the key pair is deleted successfully; otherwise
    ///   `false`.
    func deleteKeyPair(keyPair: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteKeyPair(
                input: DeleteKeyPairInput(
                    keyName: keyPair
                )
            )

            return true
        } catch {
```

```
            print("*** Error deleting the key pair: \(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie DeleteKeyPair in der AWS SDK for Swift API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Delete a security group.
    ///
    /// - Parameter groupId: The ID of the security group to delete.
    ///
    /// - Returns: `true` on successful deletion; `false` on error.
    func deleteSecurityGroup(groupId: String) async -> Bool {
        do {
            _ = try await ec2Client.deleteSecurityGroup(
                input: DeleteSecurityGroupInput(
                    groupId: groupId
                )
            )

            return true
        } catch {
            print("*** Error deleting the security group:
 \(error.localizedDescription)")
            return false
        }
```

```
    }
```

- API-Details finden Sie DeleteSecurityGroup in der AWS SDK for Swift API-Referenz.

`DescribeImages`

Das folgende Codebeispiel zeigt die Verwendung`DescribeImages`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Return an array of `EC2ClientTypes.Image` objects describing all of
    /// the images in the specified array.
    ///
    /// - Parameter idList: A list of image ID strings indicating the images
    ///   to return details about.
    ///
    /// - Returns: An array of the images.
    func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
        do {
            let output = try await ec2Client.describeImages(
                input: DescribeImagesInput(
                    imageIds: idList
                )
            )

            guard let images = output.images else {
                print("*** No images found.")
                return []
            }

            for image in images {
                guard let id = image.imageId else {
```

```
                    continue
                }
                print("   \(id): \(image.description ?? "<no description>")")
            }

            return images
        } catch {
            print("*** Error getting image descriptions:
 \(error.localizedDescription)")
            return []
        }
    }
```

- API-Details finden Sie DescribeImages in der AWS SDK for Swift API-Referenz.

`DescribeInstanceTypes`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstanceTypes`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Return a list of instance types matching the specified architecture
    /// and instance sizes.
    ///
    /// - Parameters:
    ///   - architecture: The architecture of the instance types to return, as
    ///     a member of `EC2ClientTypes.ArchitectureValues`.
    ///   - sizes: An array of one or more strings identifying sizes of
    ///     instance type to accept.
    ///
    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
```

```
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues =
EC2ClientTypes.ArchitectureValues.x8664,
                          sizes: [String] = ["*.micro", "*.small"]) async
                          -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }
```

- API-Details finden Sie DescribeInstanceTypes in der AWS SDK for Swift API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Describe the key pairs associated with the user by outputting each key
    /// pair's name and fingerprint.
    func describeKeyPairs() async {
        do {
            let output = try await ec2Client.describeKeyPairs(
                input: DescribeKeyPairsInput()
            )

            guard let keyPairs = output.keyPairs else {
                print("*** No key pairs list available.")
                return
            }

            for keyPair in keyPairs {
                print(keyPair.keyName ?? "<unknown>", ":", keyPair.keyFingerprint ??
 "<unknown>")
            }
        } catch {
            print("*** Error: Unable to obtain a key pair list.")
        }
    }
```

- API-Details finden Sie DescribeKeyPairs in der AWS SDK for Swift API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie die Paginierung mit `describeSecurityGroupsPaginated()`.

```
import AWSEC2

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
    ///   group the user is a member of.
    func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
        let pages = ec2Client.describeSecurityGroupsPaginated(
            input: DescribeSecurityGroupsInput()
        )

        var groupNames: [String] = []

        do {
            for try await page in pages {
                guard let groups = page.securityGroups else {
                    print("*** Error: No groups returned.")
                    continue
                }

                for group in groups {
                    groupNames.append(group.groupName ?? "<unknown>")
                }
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
```

```
    }
```

Ohne Paginierung.

```
import AWSEC2

    func describeSecurityGroups(groupId: String) async -> Bool {
        do {
            let output = try await ec2Client.describeSecurityGroups(
                input: DescribeSecurityGroupsInput(
                    groupIds: [groupId]
                )
            )

            guard let securityGroups = output.securityGroups else {
                print("No security groups found.")
                return true
            }

            for group in securityGroups {
                print("Group \(group.groupId ?? "<unknown>") found with VPC
 \(group.vpcId ?? "<unknown>")")
            }
            return true
        } catch {
            print("*** Error getting security group details:
 \(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie DescribeSecurityGroups in der AWS SDK for Swift API-Referenz.

`DisassociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`DisassociateAddress`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {
            print("*** Unable to disassociate the IP address:
 \(error.localizedDescription)")
        }
    }
```

- API-Details finden Sie DisassociateAddress in der AWS SDK for Swift API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Release an allocated Elastic IP.
    ///
    /// - Parameter allocationId: The allocation ID of the Elastic IP to
    ///   release.
    func releaseAddress(allocationId: String?) async {
        do {
            _ = try await ec2Client.releaseAddress(
                input: ReleaseAddressInput(
                    allocationId: allocationId
                )
            )
        } catch {
            print("*** Unable to release the IP address:
 \(error.localizedDescription)")
        }
    }
```

- API-Details finden Sie ReleaseAddress in der AWS SDK for Swift API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Create and return a new EC2 instance.
    ///
    /// - Parameters:
    ///   - imageId: The image ID of the AMI to use when creating the instance.
    ///   - instanceType: The type of instance to create.
```

```
    ///   - keyPairName: The RSA key pair's name to use to secure the instance.
    ///   - securityGroups: The security group or groups to add the instance
    ///     to.
    ///
    /// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
    func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                     keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
        do {
            let output = try await ec2Client.runInstances(
                input: RunInstancesInput(
                    imageId: imageId,
                    instanceType: instanceType,
                    keyName: keyPairName,
                    maxCount: 1,
                    minCount: 1,
                    securityGroupIds: securityGroups
                )
            )

            guard let instances = output.instances else {
                print("*** Unable to create the instance.")
                return nil
            }

            return instances[0]
        } catch {
            print("*** Error creating the instance: \(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie RunInstances in der AWS SDK for Swift API-Referenz.

StartInstances

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance
    ///     has started. Otherwise, execution continues and the instance starts
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully started (or is left to
    ///   start asynchronously). `false` if the instance doesn't start.
    func startInstance(instanceId: String, waitUntilStarted: Bool = false) async ->
 Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.startInstances(
                input: StartInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStarted {
                print("Waiting for the instance to start...")

                let waitOptions = WaiterOptions(maxWaitTime: 60.0)
                let output = try await ec2Client.waitUntilInstanceRunning(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )
                switch output.result {
```

```
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to start the instance: \(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie StartInstances in der AWS SDK for Swift API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Stop the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to stop.
    ///   - waitUntilStopped: If `true`, execution waits until the instance
    ///     has stopped. Otherwise, execution continues and the instance stops
    ///     asynchronously.
    ///
    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
```

```
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async ->
Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance: \(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie StopInstances in der AWS SDK for Swift API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSEC2

    /// Terminate the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to terminate.
    ///   - waitUntilTerminated: Whether or not to wait until the instance is
    ///     terminated before returning.
    ///
    /// - Returns: `true` if terminated successfully. `false` if not or if an
    ///   error occurs.
    func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
 async -> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.terminateInstances(
                input: TerminateInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilTerminated {
                print("Waiting for the instance to terminate...")

                let waitOptions = WaiterOptions(maxWaitTime: 600.0)
                let output = try await ec2Client.waitUntilInstanceTerminated(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
```

```
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to terminate the instance:
\(error.localizedDescription)")
            return false
        }
    }
```

- API-Details finden Sie TerminateInstances in der AWS SDK for Swift API-Referenz.

AWS Glue Beispiele zur Verwendung des SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift with Aktionen ausführen und allgemeine Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon S3-Bucket crawlt und eine Datenbank mit Metadaten generiert. CSV-formatted
- Listen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf. AWS Glue Data Catalog
- Erstellen Sie einen Job, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten zu transformieren und die JSON-formatted Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "glue-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
```

```
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "glue-scenario",
            dependencies: [
                .product(name: "AWSGlue", package: "aws-sdk-swift"),
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to demonstrate
// creating and using crawlers and jobs using AWS Glue.
//
// 0. Upload the Python job script to Amazon S3 so it can be used when
//    calling `startJobRun()` later.
// 1. Create a crawler, pass it the IAM role and the URL of the public Amazon
//    S3 bucket that contains the source data:
//    s3://crawler-public-us-east-1/flight/2016/csv.
// 2. Start the crawler. This takes time, so after starting it, use a loop
//    that calls `getCrawler()` until the state is "READY".
// 3. Get the database created by the crawler, and the tables in the
//    database. Display them to the user.
// 4. Create a job. Pass it the IAM role and the URL to a Python ETL script
```

```
//    previously uploaded to the user's S3 bucket.
// 5. Start a job run, passing the following custom arguments. These are
//    expected by the ETL script, so must exactly match.
//    * `--input_database: <name of the database created by the crawler>`
//    * `--input_table: <name of the table created by the crawler>`
//    * `--output_bucket_url: <URL to the scaffold bucket created for the
//      user>`
// 6. Loop and get the job run until it returns one of the following states:
//    "SUCCEEDED", "STOPPED", "FAILED", or "TIMEOUT".
// 7. Output data is stored in a group of files in the user's S3 bucket.
//    Either direct the user to their location or download a file and display
//    the results inline.
// 8. List the jobs for the user's account.
// 9. Get job run details for a job run.
// 10. Delete the demo job.
// 11. Delete the database and tables created by the example.
// 12. Delete the crawler created by the example.

import ArgumentParser
import AWSS3
import Foundation
import Smithy

import AWSClientRuntime
import AWSGlue

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS IAM role to use for AWS Glue calls.")
    var role: String

    @Option(help: "The Amazon S3 bucket to use for this example.")
    var bucket: String

    @Option(help: "The Amazon S3 URL of the data to crawl.")
    var s3url: String = "s3://crawler-public-us-east-1/flight/2016/csv"

    @Option(help: "The Python script to run as a job with AWS Glue.")
    var script: String = "./flight_etl_job_script.py"

    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(help: "A prefix string to use when naming tables.")
    var tablePrefix = "swift-glue-basics-table"
```

```
    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "glue-scenario",
        abstract: """
        Demonstrates various features of AWS Glue.
        """,
        discussion: """
        An example showing how to use AWS Glue to create, run, and monitor
        crawlers and jobs.
        """
    )

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Upload a file to an Amazon S3 bucket.
    ///
```

```
    /// - Parameters:
    ///   - s3Client: The S3 client to use when uploading the file.
    ///   - path: The local path of the source file to upload.
    ///   - toBucket: The name of the S3 bucket into which to upload the file.
    ///   - key: The key (name) to give the file in the S3 bucket.
    ///
    /// - Returns: `true` if the file is uploaded successfully, otherwise `false`.
    func uploadFile(s3Client: S3Client, path: String, toBucket: String, key: String)
async -> Bool {
        do {
            let fileData: Data = try Data(contentsOf: URL(fileURLWithPath: path))
            let dataStream = ByteStream.data(fileData)
            _ = try await s3Client.putObject(
                input: PutObjectInput(
                    body: dataStream,
                    bucket: toBucket,
                    key: key
                )
            )
        } catch {
            print("*** An unexpected error occurred uploading the script to the
Amazon S3 bucket \"\(bucket)\".")
            return false
        }

        return true
    }

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise `false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole: String,
                       s3Path: String, cronSchedule: String, databaseName: String)
async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])
```

```
        do {
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario Example
for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue crawler:
\(error.localizedDescription)")
            return false
        }

        return true
    }

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
```

```
        return true
    }

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
```

```
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
        }
    }

    /// Wait until the specified crawler is ready to run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to wait for.
    ///
    /// - Returns: `true` if the crawler is ready, `false` if the client is
    ///   stopping (and will therefore never be ready).
    func waitUntilCrawlerReady(glueClient: GlueClient, name: String) async -> Bool {
        while true {
            let state = await getCrawlerState(glueClient: glueClient, name: name)

            if state == .ready {
                return true
            } else if state == .stopping {
                return false
            }

            // Wait four seconds before trying again.

            do {
                try await Task.sleep(for: .seconds(4))
            } catch {
                print("*** Error pausing the task.")
            }
        }
    }

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
```

```
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
```

```
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
    }

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }

    /// Create an AWS Glue database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name to give the new database.
```

```
   ///   - location: The URL of the source data to use with AWS Glue.
   ///
   /// - Returns: `true` if the database is created successfully, otherwise
`false`.
   func createDatabase(glueClient: GlueClient, name databaseName: String, location:
String) async -> Bool {
       let databaseInput = GlueClientTypes.DatabaseInput(
           description: "Created by the AWS SDK for Swift Glue basic scenario
example.",
           locationUri: location,
           name: databaseName
       )

       do {
           _ = try await glueClient.createDatabase(
               input: CreateDatabaseInput(
                   databaseInput: databaseInput
               )
           )
       } catch {
           return false
       }

       return true
   }

   /// Get the AWS Glue database with the specified name.
   ///
   /// - Parameters:
   ///   - glueClient: The AWS Glue client to use.
   ///   - name: The name of the database to return.
   ///
   /// - Returns: The `GlueClientTypes.Database` object describing the
   ///   specified database, or `nil` if an error occurs or the database
   ///   isn't found.
   func getDatabase(glueClient: GlueClient, name: String) async ->
GlueClientTypes.Database? {
       do {
           let output = try await glueClient.getDatabase(
               input: GetDatabaseInput(name: name)
           )

           return output.database
       } catch {
```

```
            return nil
        }
    }

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async ->
[GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }

    /// Delete the specified database.
```

```
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.

            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
```

```
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }

        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job against.
    ///   - tableName: The name of the table in the database to run the job against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
```

```
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
            )

            guard let id = output.jobRunId else {
                return nil
            }

            return id
        } catch {
            return nil
        }
    }

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults: Int? =
nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
                return []
```

```
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String) async -
> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        // A name to give the Python script upon upload to the Amazon S3
        // bucket.
        let scriptName = "jobscript.py"
```

```
        // Schedule string in `cron` format, as described here:
        // https://docs.aws.amazon.com/glue/latest/dg/monitor-data-warehouse-
schedule.html
        let cron = "cron(15 12 * * ? *)"

        let glueConfig = try await GlueClient.GlueClientConfiguration(region:
 awsRegion)
        let glueClient = GlueClient(config: glueConfig)

        let s3Config = try await S3Client.S3ClientConfiguration(region: awsRegion)
        let s3Client = S3Client(config: s3Config)

        // Create random names for things that need them.

        let crawlerName = tempName(prefix: "swift-glue-basics-crawler")
        let databaseName = tempName(prefix: "swift-glue-basics-db")

        // Create a name for the AWS Glue job.

        let jobName = tempName(prefix: "scenario-job")

        // The URL of the Python script on S3.

        let scriptURL = "s3://\(bucket)/\(scriptName)"

        print("Welcome to the AWS SDK for Swift basic scenario for AWS Glue!")

        //======================================================================
        // 0. Upload the Python script to the target bucket so it's available
        //    for use by the Amazon Glue service.
        //======================================================================

        print("Uploading the Python script: \(script) as key \(scriptName)")
        print("Destination bucket: \(bucket)")
        if !(await uploadFile(s3Client: s3Client, path: script, toBucket: bucket,
 key: scriptName)) {
            return
        }

        //======================================================================
        // 1. Create the database and crawler using the randomized names
        //    generated previously.
        //======================================================================
```

```
        print("Creating database \"\(databaseName)\"...")
        if !(await createDatabase(glueClient: glueClient, name: databaseName,
location: s3url)) {
            print("*** Unable to create the database.")
            return
        }

        print("Creating crawler \"\(crawlerName)\"...")
        if !(await createCrawler(glueClient: glueClient, crawlerName: crawlerName,
                                 iamRole: role, s3Path: s3url, cronSchedule: cron,
                                 databaseName: databaseName)) {
            return
        }

        //=====================================================================
        // 2. Start the crawler, then wait for it to be ready.
        //=====================================================================

        print("Starting the crawler and waiting until it's ready...")
        if !(await startCrawler(glueClient: glueClient, name: crawlerName)) {
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        if !(await waitUntilCrawlerReady(glueClient: glueClient, name: crawlerName))
{
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
        }

        //=====================================================================
        // 3. Get the database and table created by the crawler.
        //=====================================================================

        print("Getting the crawler's database...")
        let database = await getDatabase(glueClient: glueClient, name: databaseName)

        guard let database else {
            print("*** Unable to get the database.")
            return
        }
        print("Database URI: \(database.locationUri ?? "<unknown>")")

        let tableList = await getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)
```

```
        print("Found \(tableList.count) table(s):")
        for table in tableList {
            print("  \(table.name ?? "<unnamed>")")
        }

        if tableList.count != 1 {
            print("*** Incorrect number of tables found. There should only be one.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        guard let tableName = tableList[0].name else {
            print("*** Table is unnamed.")
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 4. Create a job.
        //=====================================================================

        print("Creating a job...")
        if !(await createJob(glueClient: glueClient, name: jobName, role: role,
                             scriptLocation: scriptURL)) {
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 5. Start a job run.
        //=====================================================================

        print("Starting the job...")

        // Construct the Amazon S3 URL for the job run's output. This is in
        // the bucket specified on the command line, with a folder name that's
        // unique for this job run.
```

```
        let timeStamp = Date().timeIntervalSince1970
        let jobPath = "\(jobName)-\(Int(timeStamp))"
        let outputURL = "s3://\(bucket)/\(jobPath)"

        // Start the job run.

        let jobRunID = await startJobRun(glueClient: glueClient, name: jobName,
                                         databaseName: databaseName,
                                         tableName: tableName,
                                         outputURL: outputURL)

        guard let jobRunID else {
            print("*** Job run ID is invalid.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        //=====================================================================
        // 6. Wait for the job run to indicate that the run is complete.
        //=====================================================================

        print("Waiting for job run to end...")

        var jobRunFinished = false
        var jobRunState: GlueClientTypes.JobRunState

        repeat {
            let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)
            guard let jobRun else {
                print("*** Unable to get the job run.")
                _ = await deleteJob(glueClient: glueClient, name: jobName)
                _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
                _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
                return
            }
            jobRunState = jobRun.jobRunState ?? .failed

            //=====================================================================
```

```
            // 7. Output where to find the data if the job run was successful.
            //    If the job run failed for any reason, output an appropriate
            //    error message.
            //=====================================================================

            switch jobRunState {
                case .succeeded:
                    print("Job run succeeded. JSON files are in the Amazon S3
path:")
                    print("    \(outputURL)")
                    jobRunFinished = true
                case .stopped:
                    jobRunFinished = true
                case .error:
                    print("*** Error: Job run ended in an error.
\(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .failed:
                    print("*** Error: Job run failed. \(jobRun.errorMessage ?? "")")
                    jobRunFinished = true
                case .timeout:
                    print("*** Warning: Job run timed out.")
                    jobRunFinished = true
                default:
                    do {
                        try await Task.sleep(for: .milliseconds(250))
                    } catch {
                        print("*** Error pausing the task.")
                    }
            }
        } while jobRunFinished != true

        //=====================================================================
        // 7.5. List the job runs for this job, showing each job run's ID and
        // its execution time.
        //=====================================================================

        print("Getting all job runs for the job \(jobName):")
        let jobRuns = await getJobRuns(glueClient: glueClient, name: jobName)

        if jobRuns.count == 0 {
            print("    <no job runs found>")
        } else {
            print("Found \(jobRuns.count) job runs... listing execution times:")
```

```
            for jobRun in jobRuns {
                print("    \(jobRun.id ?? "<unnamed>"): \(jobRun.executionTime)
seconds")
            }
        }

        //=====================================================================
        // 8. List the jobs for the user's account.
        //=====================================================================

        print("\nThe account has the following jobs:")
        let jobs = await listJobs(glueClient: glueClient)

        if jobs.count == 0 {
            print("    <no jobs found>")
        } else {
            for job in jobs {
                print("    \(job)")
            }
        }

        //=====================================================================
        // 9. Get the job run details for a job run.
        //=====================================================================

        print("Information about the job run:")
        let jobRun = await getJobRun(glueClient: glueClient, name: jobName, id:
jobRunID)

        guard let jobRun else {
            print("*** Unable to retrieve the job run.")
            _ = await deleteJob(glueClient: glueClient, name: jobName)
            _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
withTables: true)
            _ = await deleteCrawler(glueClient: glueClient, name: crawlerName)
            return
        }

        let startDate = jobRun.startedOn ?? Date(timeIntervalSince1970: 0)
        let endDate = jobRun.completedOn ?? Date(timeIntervalSince1970: 0)
        let dateFormatter: DateFormatter = DateFormatter()
        dateFormatter.dateStyle = .long
        dateFormatter.timeStyle = .long
```

```
        print("    Started at: \(dateFormatter.string(from: startDate))")
        print("  Completed at: \(dateFormatter.string(from: endDate))")

        //=====================================================================
        // 10. Delete the job.
        //=====================================================================

        print("\nDeleting the job...")
        _ = await deleteJob(glueClient: glueClient, name: jobName)

        //=====================================================================
        // 11. Delete the database and tables created by this example.
        //=====================================================================

        print("Deleting the database...")
        _ = await deleteDatabase(glueClient: glueClient, name: databaseName,
 withTables: true)

        //=====================================================================
        // 12. Delete the crawler.
        //=====================================================================

        print("Deleting the crawler...")
        if !(await deleteCrawler(glueClient: glueClient, name: crawlerName)) {
            return
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt die Verwendung`CreateCrawler`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: An AWS Glue client to use for the crawler.
    ///   - crawlerName: A name for the new crawler.
    ///   - iamRole: The name of an Amazon IAM role for the crawler to use.
    ///   - s3Path: The path of an Amazon S3 folder to use as a target location.
    ///   - cronSchedule: A `cron` schedule indicating when to run the crawler.
    ///   - databaseName: The name of an AWS Glue database to operate on.
    ///
    /// - Returns: `true` if the crawler is created successfully, otherwise `false`.
    func createCrawler(glueClient: GlueClient, crawlerName: String, iamRole: String,
                       s3Path: String, cronSchedule: String, databaseName: String)
 async -> Bool {
        let s3Target = GlueClientTypes.S3Target(path: s3url)
        let targetList = GlueClientTypes.CrawlerTargets(s3Targets: [s3Target])

        do {
            _ = try await glueClient.createCrawler(
                input: CreateCrawlerInput(
                    databaseName: databaseName,
                    description: "Created by the AWS SDK for Swift Scenario Example
 for AWS Glue.",
                    name: crawlerName,
                    role: iamRole,
                    schedule: cronSchedule,
                    tablePrefix: tablePrefix,
                    targets: targetList
                )
            )
        } catch _ as AlreadyExistsException {
            print("*** A crawler named \"\(crawlerName)\" already exists.")
            return false
        } catch _ as OperationTimeoutException {
            print("*** The attempt to create the AWS Glue crawler timed out.")
            return false
        } catch {
            print("*** An unexpected error occurred creating the AWS Glue crawler:
 \(error.localizedDescription)")
            return false
```

```
        }

        return true
    }
```

- API-Details finden Sie CreateCrawler in der AWS SDK for Swift API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Create a new AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name to give the new job.
    ///   - role: The IAM role for the job to use when accessing AWS services.
    ///   - scriptLocation: The AWS S3 URI of the script to be run by the job.
    ///
    /// - Returns: `true` if the job is created successfully, otherwise `false`.
    func createJob(glueClient: GlueClient, name jobName: String, role: String,
                   scriptLocation: String) async -> Bool {
        let command = GlueClientTypes.JobCommand(
            name: "glueetl",
            pythonVersion: "3",
            scriptLocation: scriptLocation
        )

        do {
            _ = try await glueClient.createJob(
```

```
                input: CreateJobInput(
                    command: command,
                    description: "Created by the AWS SDK for Swift Glue basic
 scenario example.",
                    glueVersion: "3.0",
                    name: jobName,
                    numberOfWorkers: 10,
                    role: role,
                    workerType: .g1x
                )
            )
        } catch {
            return false
        }
        return true
    }
```

- API-Details finden Sie CreateJob in der AWS SDK for Swift API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler to delete.
    ///
    /// - Returns: `true` if successful, otherwise `false`.
```

```
    func deleteCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.deleteCrawler(
                input: DeleteCrawlerInput(name: name)
            )
        } catch {
            return false
        }
        return true
    }
```

- API-Details finden Sie DeleteCrawler in der AWS SDK for Swift API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database to delete.
    ///   - deleteTables: A Bool indicating whether or not to delete the
    ///     tables in the database before attempting to delete the database.
    ///
    /// - Returns: `true` if the database (and optionally its tables) are
    ///   deleted, otherwise `false`.
    func deleteDatabase(glueClient: GlueClient, name databaseName: String,
                        withTables deleteTables: Bool = false) async -> Bool {
        if deleteTables {
```

```
            var tableNames: [String] = []

            // Get a list of the names of all of the tables in the database.

            let tableList = await self.getTablesInDatabase(glueClient: glueClient,
databaseName: databaseName)
            for table in tableList {
                guard let name = table.name else {
                    continue
                }
                tableNames.append(name)
            }

            // Delete the tables. If there's only one table, use
            // `deleteTable()`, otherwise, use `batchDeleteTable()`. You can
            // use `batchDeleteTable()` for a single table, but this
            // demonstrates the use of `deleteTable()`.

            if tableNames.count == 1 {
                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
            } else {
                do {
                    print("    Deleting tables...")
                    _ = try await glueClient.batchDeleteTable(
                        input: BatchDeleteTableInput(
                            databaseName: databaseName,
                            tablesToDelete: tableNames
                        )
                    )
                } catch {
                    print("*** Unable to delete the tables.")
                }
            }
        }
```

```
        // Delete the database itself.

        do {
            print("    Deleting the database itself...")
            _ = try await glueClient.deleteDatabase(
                input: DeleteDatabaseInput(name: databaseName)
            )
        } catch {
            print("*** Unable to delete the database.")
            return false
        }
        return true
    }
```

- API-Details finden Sie DeleteDatabase in der AWS SDK for Swift API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Delete an AWS Glue job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to delete.
    ///
    /// - Returns: `true` if the job is successfully deleted, otherwise `false`.
    func deleteJob(glueClient: GlueClient, name jobName: String) async -> Bool {
        do {
```

```
            _ = try await glueClient.deleteJob(
                input: DeleteJobInput(jobName: jobName)
            )
        } catch {
            return false
        }
        return true
    }
```

- API-Details finden Sie DeleteJob in der AWS SDK for Swift API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

                do {
                    print("    Deleting table...")
                    _ = try await glueClient.deleteTable(
                        input: DeleteTableInput(
                            databaseName: databaseName,
                            name: tableNames[0]
                        )
                    )
                } catch {
                    print("*** Unable to delete the table.")
                }
```

- API-Details finden Sie DeleteTable in der AWS SDK for Swift API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Get the state of the specified AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the crawler whose state should be returned.
    ///
    /// - Returns: A `GlueClientTypes.CrawlerState` value describing the
    ///   state of the crawler.
    func getCrawlerState(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.CrawlerState {
        do {
            let output = try await glueClient.getCrawler(
                input: GetCrawlerInput(name: name)
            )

            // If the crawler or its state is `nil`, report that the crawler
            // is stopping. This may not be what you want for your
            // application but it works for this one!

            guard let crawler = output.crawler else {
                return GlueClientTypes.CrawlerState.stopping
            }
            guard let state = crawler.state else {
                return GlueClientTypes.CrawlerState.stopping
            }
            return state
        } catch {
            return GlueClientTypes.CrawlerState.stopping
```

```
        }
    }
```

- API-Details finden Sie GetCrawler in der AWS SDK for Swift API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Get the AWS Glue database with the specified name.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - name: The name of the database to return.
    ///
    /// - Returns: The `GlueClientTypes.Database` object describing the
    ///   specified database, or `nil` if an error occurs or the database
    ///   isn't found.
    func getDatabase(glueClient: GlueClient, name: String) async ->
 GlueClientTypes.Database? {
        do {
            let output = try await glueClient.getDatabase(
                input: GetDatabaseInput(name: name)
            )

            return output.database
        } catch {
            return nil
        }
    }
```

- API-Details finden Sie GetDatabase in der AWS SDK for Swift API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Get information about a specific AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to return job run data for.
    ///   - id: The run ID of the specific job run to return.
    ///
    /// - Returns: A `GlueClientTypes.JobRun` object describing the state of
    ///   the job run, or `nil` if an error occurs.
    func getJobRun(glueClient: GlueClient, name jobName: String, id: String) async -
> GlueClientTypes.JobRun? {
        do {
            let output = try await glueClient.getJobRun(
                input: GetJobRunInput(
                    jobName: jobName,
                    runId: id
                )
            )

            return output.jobRun
        } catch {
            return nil
        }
```

```
    }
```

- API-Details finden Sie GetJobRun in der AWS SDK for Swift API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the job runs for the specified job.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job for which to return its job runs.
    ///   - maxResults: The maximum number of job runs to return (default:
    ///     1000).
    ///
    /// - Returns: An array of `GlueClientTypes.JobRun` objects describing
    ///   each job run.
    func getJobRuns(glueClient: GlueClient, name jobName: String, maxResults: Int? =
 nil) async -> [GlueClientTypes.JobRun] {
        do {
            let output = try await glueClient.getJobRuns(
                input: GetJobRunsInput(
                    jobName: jobName,
                    maxResults: maxResults
                )
            )

            guard let jobRuns = output.jobRuns else {
                print("*** No job runs found.")
```

```
                return []
            }

            return jobRuns
        } catch is EntityNotFoundException {
            print("*** The specified job name, \(jobName), doesn't exist.")
            return []
        } catch {
            print("*** Unexpected error getting job runs:")
            dump(error)
            return []
        }
    }
```

- API-Details finden Sie GetJobRuns in der AWS SDK for Swift API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Returns a list of the tables in the specified database.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - databaseName: The name of the database whose tables are to be
    ///     returned.
    ///
    /// - Returns: An array of `GlueClientTypes.Table` objects, each
    ///   describing one table in the named database. An empty array indicates
    ///   that there are either no tables in the database, or an error
```

```
    ///   occurred before any tables could be found.
    func getTablesInDatabase(glueClient: GlueClient, databaseName: String) async ->
[GlueClientTypes.Table] {
        var tables: [GlueClientTypes.Table] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.getTables(
                    input: GetTablesInput(
                        databaseName: databaseName,
                        nextToken: nextToken
                    )
                )

                guard let tableList = output.tableList else {
                    return tables
                }

                tables = tables + tableList
                nextToken = output.nextToken
            } catch {
                return tables
            }
        } while nextToken != nil

        return tables
    }
```

- API-Details finden Sie GetTables in der AWS SDK for Swift API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Return a list of the AWS Glue jobs listed on the user's account.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - maxJobs: The maximum number of jobs to return (default: 100).
    ///
    /// - Returns: An array of strings listing the names of all available AWS
    ///   Glue jobs.
    func listJobs(glueClient: GlueClient, maxJobs: Int = 100) async -> [String] {
        var jobList: [String] = []
        var nextToken: String?

        repeat {
            do {
                let output = try await glueClient.listJobs(
                    input: ListJobsInput(
                        maxResults: maxJobs,
                        nextToken: nextToken
                    )
                )

                guard let jobs = output.jobNames else {
                    return jobList
                }

                jobList = jobList + jobs
                nextToken = output.nextToken
            } catch {
                return jobList
            }
        } while (nextToken != nil)

        return jobList
    }
```

- API-Details finden Sie ListJobs in der AWS SDK for Swift API-Referenz.

StartCrawler

Das folgende Codebeispiel zeigt die VerwendungStartCrawler.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Start running an AWS Glue crawler.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use when starting the crawler.
    ///   - name: The name of the crawler to start running.
    ///
    /// - Returns: `true` if the crawler is started successfully, otherwise `false`.
    func startCrawler(glueClient: GlueClient, name: String) async -> Bool {
        do {
            _ = try await glueClient.startCrawler(
                input: StartCrawlerInput(name: name)
            )
        } catch {
            print("*** An unexpected error occurred starting the crawler.")
            return false
        }

        return true
    }
```

- API-Details finden Sie StartCrawler in der AWS SDK for Swift API-Referenz.

StartJobRun

Das folgende Codebeispiel zeigt die VerwendungStartJobRun.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSGlue

    /// Start an AWS Glue job run.
    ///
    /// - Parameters:
    ///   - glueClient: The AWS Glue client to use.
    ///   - jobName: The name of the job to run.
    ///   - databaseName: The name of the AWS Glue database to run the job against.
    ///   - tableName: The name of the table in the database to run the job against.
    ///   - outputURL: The AWS S3 URI of the bucket location into which to
    ///     write the resulting output.
    ///
    /// - Returns: `true` if the job run is started successfully, otherwise `false`.
    func startJobRun(glueClient: GlueClient, name jobName: String, databaseName:
 String,
                     tableName: String, outputURL: String) async -> String? {
        do {
            let output = try await glueClient.startJobRun(
                input: StartJobRunInput(
                    arguments: [
                        "--input_database": databaseName,
                        "--input_table": tableName,
                        "--output_bucket_url": outputURL
                    ],
                    jobName: jobName,
                    numberOfWorkers: 10,
                    workerType: .g1x
                )
            )

            guard let id = output.jobRunId else {
                return nil
            }
```

```
            return id
        } catch {
            return nil
        }
    }
```

- API-Details finden Sie StartJobRun in der AWS SDK for Swift API-Referenz.

IAM-Beispiele unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`AttachRolePolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
```

```
import AWSS3


    public func attachRolePolicy(role: String, policyArn: String) async throws {
        let input = AttachRolePolicyInput(
            policyArn: policyArn,
            roleName: role
        )
        do {
            _ = try await client.attachRolePolicy(input: input)
        } catch {
            print("ERROR: Attaching a role policy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie AttachRolePolicy in der AWS SDK for Swift API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createAccessKey(userName: String) async throws ->
 IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
        do {
```

```
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            print("ERROR: createAccessKey:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreateAccessKey in der AWS SDK for Swift API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createPolicy(name: String, policyDocument: String) async throws ->
 IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
```

```
            }
            return policy
        } catch {
            print("ERROR: createPolicy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreatePolicy in der AWS SDK for Swift API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createRole(name: String, policyDocument: String) async throws ->
 String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
```

```
            }
            return id
        } catch {
            print("ERROR: createRole:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreateRole in der AWS SDK for Swift API-Referenz.

`CreateServiceLinkedRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateServiceLinkedRole`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createServiceLinkedRole(service: String, suffix: String? = nil,
 description: String?)
                    async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
```

```
            print("ERROR: createServiceLinkedRole:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreateServiceLinkedRole in der AWS SDK for Swift API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func createUser(name: String) async throws -> String {
        let input = CreateUserInput(
            userName: name
        )
        do {
            let output = try await client.createUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            guard let id = user.userId else {
                throw ServiceHandlerError.noSuchUser
            }
            return id
        } catch {
            print("ERROR: createUser:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie CreateUser in der AWS SDK for Swift API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws
    {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
            print("ERROR: deleteAccessKey:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteAccessKey in der AWS SDK for Swift API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            print("ERROR: deletePolicy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeletePolicy in der AWS SDK for Swift API-Referenz.

`DeleteRole`

Das folgende Codebeispiel zeigt die Verwendung`DeleteRole`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            print("ERROR: deleteRole:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteRole in der AWS SDK for Swift API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func deleteUser(user: IAMClientTypes.User) async throws {
        let input = DeleteUserInput(
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUser(input: input)
        } catch {
            print("ERROR: deleteUser:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteUser in der AWS SDK for Swift API-Referenz.

`DeleteUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUserPolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
 throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
```

```
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            print("ERROR: deleteUserPolicy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteUserPolicy in der AWS SDK for Swift API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
 IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            print("ERROR: detachRolePolicy:", dump(error))
            throw error
```

```
        }
    }
```

- API-Details finden Sie DetachRolePolicy in der AWS SDK for Swift API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: getPolicy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie GetPolicy in der AWS SDK for Swift API-Referenz.

`GetRole`

Das folgende Codebeispiel zeigt die Verwendung`GetRole`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: getRole:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie GetRole in der AWS SDK for Swift API-Referenz.

`GetUser`

Das folgende Codebeispiel zeigt die Verwendung`GetUser`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func getUser(name: String? = nil) async throws -> IAMClientTypes.User {
        let input = GetUserInput(
            userName: name
        )
        do {
            let output = try await iamClient.getUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            return user
        } catch {
            print("ERROR: getUser:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie GetUser in der AWS SDK for Swift API-Referenz.

`ListAttachedRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListAttachedRolePolicies`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3



    /// Returns a list of AWS Identity and Access Management (IAM) policies
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
 [IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []

        // Use "Paginated" to get all the attached role polices.
        // This lets the SDK handle the 'isTruncated' in
 "ListAttachedRolePoliciesOutput".
        let input = ListAttachedRolePoliciesInput(
            roleName: role
        )
        let output = client.listAttachedRolePoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let attachedPolicies = page.attachedPolicies else {
                    print("Error: no attached policies returned.")
                    continue
                }
                for attachedPolicy in attachedPolicies {
                    policyList.append(attachedPolicy)
                }
            }
```

```
        } catch {
            print("ERROR: listAttachedRolePolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- API-Details finden Sie ListAttachedRolePolicies in der AWS SDK for Swift API-Referenz.

`ListGroups`

Das folgende Codebeispiel zeigt die Verwendung`ListGroups`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listGroups() async throws -> [String] {
        var groupList: [String] = []

        // Use "Paginated" to get all the groups.
        // This lets the SDK handle the 'isTruncated' property in
 "ListGroupsOutput".
        let input = ListGroupsInput()

        let pages = client.listGroupsPaginated(input: input)
        do {
            for try await page in pages {
                guard let groups = page.groups else {
                    print("Error: no groups returned.")
                    continue
                }
```

```
                for group in groups {
                    if let name = group.groupName {
                        groupList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listGroups:", dump(error))
            throw error
        }
        return groupList
    }
```

- API-Details finden Sie ListGroups in der AWS SDK for Swift API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []

        // Use "Paginated" to get all the policies.
        // This lets the SDK handle the 'isTruncated' in "ListPoliciesOutput".
        let input = ListPoliciesInput()
        let output = client.listPoliciesPaginated(input: input)

        do {
```

```
            for try await page in output {
                guard let policies = page.policies else {
                    print("Error: no policies returned.")
                    continue
                }

                for policy in policies {
                    guard let name = policy.policyName,
                          let id = policy.policyId,
                          let arn = policy.arn
                    else {
                        throw ServiceHandlerError.noSuchPolicy
                    }
                    policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
                }
            }
        } catch {
            print("ERROR: listPolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- API-Details finden Sie ListPolicies in der AWS SDK for Swift API-Referenz.

`ListRolePolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListRolePolicies`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3
```

```
    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []

        // Use "Paginated" to get all the role policies.
        // This lets the SDK handle the 'isTruncated' in "ListRolePoliciesOutput".
        let input = ListRolePoliciesInput(
            roleName: role
        )
        let pages = client.listRolePoliciesPaginated(input: input)

        do {
            for try await page in pages {
                guard let policies = page.policyNames else {
                    print("Error: no role policies returned.")
                    continue
                }

                for policy in policies {
                    policyList.append(policy)
                }
            }
        } catch {
            print("ERROR: listRolePolicies:", dump(error))
            throw error
        }
        return policyList
    }
```

- API-Details finden Sie ListRolePolicies in der AWS SDK for Swift API-Referenz.

`ListRoles`

Das folgende Codebeispiel zeigt die Verwendung`ListRoles`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listRoles() async throws -> [String] {
        var roleList: [String] = []

        // Use "Paginated" to get all the roles.
        // This lets the SDK handle the 'isTruncated' in "ListRolesOutput".
        let input = ListRolesInput()
        let pages = client.listRolesPaginated(input: input)

        do {
            for try await page in pages {
                guard let roles = page.roles else {
                    print("Error: no roles returned.")
                    continue
                }

                for role in roles {
                    if let name = role.roleName {
                        roleList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listRoles:", dump(error))
            throw error
        }
        return roleList
    }
```

- API-Details finden Sie ListRoles in der AWS SDK for Swift API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    public func listUsers() async throws -> [MyUserRecord] {
        var userList: [MyUserRecord] = []

        // Use "Paginated" to get all the users.
        // This lets the SDK handle the 'isTruncated' in "ListUsersOutput".
        let input = ListUsersInput()
        let output = client.listUsersPaginated(input: input)

        do {
            for try await page in output {
                guard let users = page.users else {
                    continue
                }
                for user in users {
                    if let id = user.userId, let name = user.userName {
                        userList.append(MyUserRecord(id: id, name: name))
                    }
                }
            }
        }
        catch {
            print("ERROR: listUsers:", dump(error))
            throw error
        }
       return userList
    }
```

- API-Details finden Sie ListUsers in der AWS SDK for Swift API-Referenz.

`PutUserPolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutUserPolicy`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSIAM
import AWSS3


    func putUserPolicy(policyDocument: String, policyName: String, user:
 IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            print("ERROR: putUserPolicy:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie PutUserPolicy in der AWS SDK for Swift API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Definieren Sie die erste Lambda-Funktion, die einfach den angegebenen Wert erhöht.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "increment",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "increment",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)

import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
```

```
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function body.
///
/// - Parameters:
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let incrementLambdaRuntime = LambdaRuntime {
        (event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?

    if action != "increment" {
        context.logger.error("Unrecognized operation: \"\(action)\". The only
 supported action is \"increment\".")
    } else {
        answer = event.number + 1
        context.logger.info("The calculated answer is \(answer!).")
    }

    let response = Response(answer: answer)
    return response
}

// Run the Lambda runtime code.
```

```
try await incrementLambdaRuntime.run()
```

Definieren Sie die zweite Lambda-Funktion, die eine arithmetische Operation mit zwei Zahlen ausführt.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "calculator",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/swift-server/swift-aws-lambda-runtime.git",
            branch: "main"),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "calculator",
            dependencies: [
                .product(name: "AWSLambdaRuntime", package: "swift-aws-lambda-
runtime"),
            ],
            path: "Sources"
        )
    ]
)
```

```
import Foundation
import AWSLambdaRuntime

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct Request: Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

/// A dictionary mapping operation names to closures that perform that
/// operation and return the result.
let actions = [
    "plus": { (x: Int, y: Int) -> Int in
        return x + y
    },
    "minus": { (x: Int, y: Int) -> Int in
        return x - y
    },
    "times": { (x: Int, y: Int) -> Int in
        return x * y
    },
    "divided-by": { (x: Int, y: Int) -> Int in
        return x / y
    }
]

/// The contents of the response sent back to the client. This must be
/// `Encodable`.
struct Response: Encodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}


/// The Lambda function's entry point. Called by the Lambda runtime.
///
/// - Parameters:
```

```
///   - event: The `Request` describing the request made by the
///     client.
///   - context: A `LambdaContext` describing the context in
///     which the lambda function is running.
///
/// - Returns: A `Response` object that will be encoded to JSON and sent
///   to the client by the Lambda runtime.
let calculatorLambdaRuntime = LambdaRuntime {
        (_ event: Request, context: LambdaContext) -> Response in
    let action = event.action
    var answer: Int?
    var actionFunc: ((Int, Int) -> Int)?

    // Get the closure to run to perform the calculation.

    actionFunc = await actions[action]

    guard let actionFunc else {
        context.logger.error("Unrecognized operation '\(action)\'")
        return Response(answer: nil)
    }

    // Perform the calculation and return the answer.

    answer = actionFunc(event.x, event.y)

    guard let answer else {
        context.logger.error("Error computing \(event.x) \(action) \(event.y)")
    }
    context.logger.info("\(event.x) \(action) \(event.y) = \(answer)")

    return Response(answer: answer)
}

try await calculatorLambdaRuntime.run()
```

Definieren Sie das Hauptprogramm, das die beiden Lambda-Funktionen aufruft.

```
// swift-tools-version: 5.9
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
//
```

```
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "lambda-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "lambda-basics",
            dependencies: [
                .product(name: "AWSLambda", package: "aws-sdk-swift"),
                .product(name: "AWSIAM", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources"
        )
    ]
)

//
/// An example demonstrating a variety of important AWS Lambda functions.

import ArgumentParser
import AWSIAM
import SmithyWaitersAPI
```

```
import AWSClientRuntime
import AWSLambda
import Foundation

/// Represents the contents of the requests being received from the client.
/// This structure must be `Decodable` to indicate that its initializer
/// converts an external representation into this type.
struct IncrementRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The number to act upon.
    let number: Int
}

struct Response: Encodable, Decodable, Sendable {
    /// The resulting value after performing the action.
    let answer: Int?
}

struct CalculatorRequest: Encodable, Decodable, Sendable {
    /// The action to perform.
    let action: String
    /// The first number to act upon.
    let x: Int
    /// The second number to act upon.
    let y: Int
}

let exampleName = "SwiftLambdaRoleExample"
let basicsFunctionName = "lambda-basics-function"

/// The ARN of the standard IAM policy for execution of Lambda functions.
let policyARN = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Option(help: "Name of the IAM Role to use for the Lambda functions")
    var role = exampleName
    @Option(help: "Zip archive containing the 'increment' lambda function")
    var incpath: String
    @Option(help: "Zip archive containing the 'calculator' lambda function")
    var calcpath: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"
```

```
    static var configuration = CommandConfiguration(
        commandName: "lambda-basics",
        abstract: """
        This example demonstrates several common operations using AWS Lambda.
        """,
        discussion: """
        """
    )

    /// Returns the specified IAM role object.
    ///
    /// - Parameters:
    ///   - iamClient: `IAMClient` to use when looking for the role.
    ///   - roleName: The name of the role to check.
    ///
    /// - Returns: The `IAMClientTypes.Role` representing the specified role.
    func getRole(iamClient: IAMClient, roleName: String) async throws
                 -> IAMClientTypes.Role {
        do {
            let roleOutput = try await iamClient.getRole(
                input: GetRoleInput(
                    roleName: roleName
                )
            )

            guard let role = roleOutput.role else {
                throw ExampleError.roleNotFound
            }
            return role
        } catch {
            throw ExampleError.roleNotFound
        }
    }

    /// Create the AWS IAM role that will be used to access AWS Lambda.
    ///
    /// - Parameters:
    ///   - iamClient: The AWS `IAMClient` to use.
    ///   - roleName: The name of the AWS IAM role to use for Lambda.
    ///
    /// - Throws: `ExampleError.roleCreateError`
    ///
    /// - Returns: The `IAMClientTypes.Role` struct that describes the new role.
```

```
    func createRoleForLambda(iamClient: IAMClient, roleName: String) async throws ->
IAMClientTypes.Role {
        let output = try await iamClient.createRole(
            input: CreateRoleInput(
                assumeRolePolicyDocument:
                """
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"Service": "lambda.amazonaws.com"},
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
                """,
                roleName: roleName
            )
        )

        guard let role = output.role else {
            throw ExampleError.roleCreateError
        }

        // Wait for the role to be ready for use.

        _ = try await iamClient.waitUntilRoleExists(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetRoleInput(roleName: roleName)
        )

        return role
    }

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
```

```
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async ->
Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
    }

    /// Create the specified AWS Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to create.
    ///   - roleArn: The ARN of the role to apply to the function.
    ///   - path: The path of the Zip archive containing the function.
    ///
    /// - Returns: `true` if the AWS Lambda was successfully created; `false`
    ///   if it wasn't.
    func createFunction(lambdaClient: LambdaClient, functionName: String,
                                roleArn: String?, path: String) async throws -> Bool
{
        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
```

```
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }

        // Wait for a while to be sure the function is done being created.

        let output = try await lambdaClient.waitUntilFunctionActiveV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(functionName: functionName)
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Update the AWS Lambda function with new code to run when the function
    /// is invoked.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to update.
    ///   - path: The pathname of the Zip file containing the packaged Lambda
    ///     function.
    /// - Throws: `ExampleError.zipFileReadError`
    /// - Returns: `true` if the function's code is updated successfully.
    ///   Otherwise, returns `false`.
    func updateFunctionCode(lambdaClient: LambdaClient, functionName: String,
                            path: String) async throws -> Bool {
        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data
```

```
        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }

        let output = try await lambdaClient.waitUntilFunctionUpdatedV2(
            options: WaiterOptions(
                maxWaitTime: 20,
                minDelay: 0.5,
                maxDelay: 2
            ),
            input: GetFunctionInput(
                functionName: functionName
            )
        )

        switch output.result {
            case .success:
                return true
            case .failure:
                return false
        }
    }

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
```

```
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String) async
throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
```

```
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
Int {
```

```
        do {
            let incRequest = IncrementRequest(action: "increment", number: number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }

    /// Invoke the calculator Lambda function.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - action: Which arithmetic operation to perform: "plus", "minus",
    ///     "times", or "divided-by".
    ///   - x: The first number to use in the computation.
    ///   - y: The second number to use in the computation.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: The computed answer as an `Int`.
    func invokeCalculator(lambdaClient: LambdaClient, action: String, x: Int, y:
Int) async throws -> Int {
        do {
            let calcRequest = CalculatorRequest(action: action, x: x, y: y)
            let calcData = try! JSONEncoder().encode(calcRequest)
```

```
            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: calcData
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }

    }

    /// Perform the example's tasks.
    func basics() async throws {
        let iamClient = try await IAMClient(
            config: IAMClient.IAMClientConfiguration(region: region)
        )

        let lambdaClient = try await LambdaClient(
            config: LambdaClient.LambdaClientConfiguration(region: region)
        )

        /// The IAM role to use for the example.
        var iamRole: IAMClientTypes.Role

        // Look for the specified role. If it already exists, use it. If not,
        // create it and attach the desired policy to it.

        do {
            iamRole = try await getRole(iamClient: iamClient, roleName: role)
        } catch ExampleError.roleNotFound {
            // The role wasn't found, so create it and attach the needed
            // policy.
```

```
            iamRole = try await createRoleForLambda(iamClient: iamClient, roleName:
role)

            do {
                _ = try await iamClient.attachRolePolicy(
                    input: AttachRolePolicyInput(policyArn: policyARN, roleName:
role)
                )
            } catch {
                throw ExampleError.policyError
            }
        }

        // Give the policy time to attach to the role.

        sleep(5)

        // Look to see if the function already exists. If it does, throw an
        // error.

        if await doesLambdaFunctionExist(lambdaClient: lambdaClient, name:
basicsFunctionName) {
            throw ExampleError.functionAlreadyExists
        }

        // Create, then invoke, the "increment" version of the calculator
        // function.

        print("Creating the increment Lambda function...")
        if try await createFunction(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    roleArn: iamRole.arn, path: incpath) {
            print("Running increment function calls...")
            for number in 0...4 {
                do {
                    let answer = try await invokeIncrement(lambdaClient:
lambdaClient, number: number)
                    print("Increment \(number) = \(answer)")
                } catch {
                    print("Error incrementing \(number): ",
error.localizedDescription)
                }
            }
```

```
        } else {
            print("*** Failed to create the increment function.")
        }

        // Enable debug logging.

        print("\nEnabling debug logging...")
        try await enableDebugLogging(lambdaClient: lambdaClient, functionName:
basicsFunctionName)

        // Change it to a basic arithmetic calculator. Then invoke it a few
        // times.

        print("\nReplacing the Lambda function with a calculator...")

        if try await updateFunctionCode(lambdaClient: lambdaClient, functionName:
basicsFunctionName,
                                    path: calcpath) {
            print("Running calculator function calls...")
            for x in [6, 10] {
                for y in [2, 4] {
                    for action in ["plus", "minus", "times", "divided-by"] {
                        do {
                            let answer = try await invokeCalculator(lambdaClient:
lambdaClient, action: action, x: x, y: y)
                            print("\(x) \(action) \(y) = \(answer)")
                        } catch {
                            print("Error calculating \(x) \(action) \(y): ",
error.localizedDescription)
                        }
                    }
                }
            }
        }

        // List all lambda functions.

        let functionNames = try await getFunctionNames(lambdaClient: lambdaClient)

        if functionNames.count > 0 {
            print("\nAWS Lambda functions available on your account:")
            for name in functionNames {
                print("  \(name)")
            }
```

```
        }

        // Delete the lambda function.

        print("Deleting lambda function...")

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
        } catch {
            print("Error: Unable to delete the function.")
        }

        // Detach the role from the policy, then delete the role.

        print("Deleting the AWS IAM role...")

        do {
            _ = try await iamClient.detachRolePolicy(
                input: DetachRolePolicyInput(
                    policyArn: policyARN,
                    roleName: role
                )
            )
            _ = try await iamClient.deleteRole(
                input: DeleteRoleInput(
                    roleName: role
                )
            )
        } catch {
            throw ExampleError.deleteRoleError
        }
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
```

```
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.basics()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}


/// Errors thrown by the example's functions.
enum ExampleError: Error {
    /// An AWS Lambda function with the specified name already exists.
    case functionAlreadyExists
    /// The specified role doesn't exist.
    case roleNotFound
    /// Unable to create the role.
    case roleCreateError
    /// Unable to delete the role.
    case deleteRoleError
    /// Unable to attach a policy to the role.
    case policyError
    /// Unable to get the executable directory.
    case executableNotFound
    /// An error occurred creating a lambda function.
    case createLambdaError
    /// An error occurred invoking the lambda function.
    case invokeError
    /// No answer received from the invocation.
    case noAnswerReceived
    /// Unable to list the AWS Lambda functions.
    case listFunctionsError
    /// Unable to update the AWS Lambda function.
    case updateFunctionError
    /// Unable to update the function configuration.
    case updateFunctionConfigurationError
    /// The environment response is missing after an
    /// UpdateEnvironmentConfiguration attempt.
    case environmentResponseMissingError
    /// The environment variables are missing from the EnvironmentResponse and
    /// no errors occurred.
    case environmentVariablesMissingError
```

```
    /// The log level is incorrect after attempting to set it.
    case logLevelIncorrectError
    /// Unable to load the AWS Lambda function's Zip file.
    case zipFileReadError

    var errorDescription: String? {
        switch self {
        case .functionAlreadyExists:
            return "An AWS Lambda function with that name already exists."
        case .roleNotFound:
            return "The specified role doesn't exist."
        case .deleteRoleError:
            return "Unable to delete the AWS IAM role."
        case .roleCreateError:
            return "Unable to create the specified role."
        case .policyError:
            return "An error occurred attaching the policy to the role."
        case .executableNotFound:
            return "Unable to find the executable program directory."
        case .createLambdaError:
            return "An error occurred creating a lambda function."
        case .invokeError:
            return "An error occurred invoking a lambda function."
        case .noAnswerReceived:
            return "No answer received from the lambda function."
        case .listFunctionsError:
            return "Unable to list the AWS Lambda functions."
        case .updateFunctionError:
            return "Unable to update the AWS lambda function."
        case .updateFunctionConfigurationError:
            return "Unable to update the AWS lambda function configuration."
        case .environmentResponseMissingError:
            return "The environment is missing from the response after updating the
 function configuration."
        case .environmentVariablesMissingError:
            return "While no error occurred, no environment variables were returned
 following function configuration."
        case .logLevelIncorrectError:
            return "The log level is incorrect after attempting to set it to DEBUG."
        case .zipFileReadError:
            return "Unable to read the AWS Lambda function."
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt die Verwendung`CreateFunction`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            // Read the Zip archive containing the AWS Lambda function.

            let zipUrl = URL(fileURLWithPath: path)
            let zipData = try Data(contentsOf: zipUrl)

            // Create the AWS Lambda function that runs the specified code,
            // using the name given on the command line. The Lambda function
```

```
            // will run using the Amazon Linux 2 runtime.

            _ = try await lambdaClient.createFunction(
                input: CreateFunctionInput(
                    code: LambdaClientTypes.FunctionCode(zipFile: zipData),
                    functionName: functionName,
                    handler: "handle",
                    role: roleArn,
                    runtime: .providedal2
                )
            )
        } catch {
            print("*** Error creating Lambda function:")
            dump(error)
            return false
        }
```

- API-Details finden Sie CreateFunction in der AWS SDK for Swift API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

        do {
            _ = try await lambdaClient.deleteFunction(
                input: DeleteFunctionInput(
                    functionName: "lambda-basics-function"
                )
            )
```

```
        } catch {
            print("Error: Unable to delete the function.")
        }
```

- API-Details finden Sie DeleteFunction in der AWS SDK for Swift API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Detect whether or not the AWS Lambda function with the specified name
    /// exists, by requesting its function information.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - name: The name of the AWS Lambda function to find.
    ///
    /// - Returns: `true` if the Lambda function exists. Otherwise `false`.
    func doesLambdaFunctionExist(lambdaClient: LambdaClient, name: String) async ->
 Bool {
        do {
            _ = try await lambdaClient.getFunction(
                input: GetFunctionInput(functionName: name)
            )
        } catch {
            return false
        }

        return true
```

```
    }
```

- API-Details finden Sie GetFunction in der AWS SDK for Swift API-Referenz.

Invoke

Das folgende Codebeispiel zeigt die VerwendungInvoke.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Invoke the Lambda function to increment a value.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `IAMClient` to use.
    ///   - number: The number to increment.
    ///
    /// - Throws: `ExampleError.noAnswerReceived`, `ExampleError.invokeError`
    ///
    /// - Returns: An integer number containing the incremented value.
    func invokeIncrement(lambdaClient: LambdaClient, number: Int) async throws ->
 Int {
        do {
            let incRequest = IncrementRequest(action: "increment", number: number)
            let incData = try! JSONEncoder().encode(incRequest)

            // Invoke the lambda function.

            let invokeOutput = try await lambdaClient.invoke(
                input: InvokeInput(
                    functionName: "lambda-basics-function",
                    payload: incData
```

```
                )
            )

            let response = try! JSONDecoder().decode(Response.self,
from:invokeOutput.payload!)

            guard let answer = response.answer else {
                throw ExampleError.noAnswerReceived
            }
            return answer

        } catch {
            throw ExampleError.invokeError
        }
    }
```

- Weitere API-Informationen finden Sie unter Invoke in der API-Referenz zum AWS -SDK für Swift.

`ListFunctions`

Das folgende Codebeispiel zeigt die Verwendung`ListFunctions`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation

    /// Returns an array containing the names of all AWS Lambda functions
    /// available to the user.
    ///
    /// - Parameter lambdaClient: The `IAMClient` to use.
    ///
```

```
    /// - Throws: `ExampleError.listFunctionsError`
    ///
    /// - Returns: An array of lambda function name strings.
    func getFunctionNames(lambdaClient: LambdaClient) async throws -> [String] {
        let pages = lambdaClient.listFunctionsPaginated(
            input: ListFunctionsInput()
        )

        var functionNames: [String] = []

        for try await page in pages {
            guard let functions = page.functions else {
                throw ExampleError.listFunctionsError
            }

            for function in functions {
                functionNames.append(function.functionName ?? "<unknown>")
            }
        }

        return functionNames
    }
```

- API-Details finden Sie ListFunctions in der AWS SDK for Swift API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
import Foundation
```

```
        let zipUrl = URL(fileURLWithPath: path)
        let zipData: Data

        // Read the function's Zip file.

        do {
            zipData = try Data(contentsOf: zipUrl)
        } catch {
            throw ExampleError.zipFileReadError
        }

        // Update the function's code and wait for the updated version to be
        // ready for use.

        do {
            _ = try await lambdaClient.updateFunctionCode(
                input: UpdateFunctionCodeInput(
                    functionName: functionName,
                    zipFile: zipData
                )
            )
        } catch {
            return false
        }
```

- API-Details finden Sie UpdateFunctionCode in der AWS SDK for Swift API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSClientRuntime
import AWSLambda
```

```
import Foundation

    /// Tell the server-side component to log debug output by setting its
    /// environment's `LOG_LEVEL` to `DEBUG`.
    ///
    /// - Parameters:
    ///   - lambdaClient: The `LambdaClient` to use.
    ///   - functionName: The name of the AWS Lambda function to enable debug
    ///     logging for.
    ///
    /// - Throws: `ExampleError.environmentResponseMissingError`,
    ///   `ExampleError.updateFunctionConfigurationError`,
    ///   `ExampleError.environmentVariablesMissingError`,
    ///   `ExampleError.logLevelIncorrectError`,
    ///   `ExampleError.updateFunctionConfigurationError`
    func enableDebugLogging(lambdaClient: LambdaClient, functionName: String) async
 throws {
        let envVariables = [
            "LOG_LEVEL": "DEBUG"
        ]
        let environment = LambdaClientTypes.Environment(variables: envVariables)

        do {
            let output = try await lambdaClient.updateFunctionConfiguration(
                input: UpdateFunctionConfigurationInput(
                    environment: environment,
                    functionName: functionName
                )
            )

            guard let response = output.environment else {
                throw ExampleError.environmentResponseMissingError
            }

            if response.error != nil {
                throw ExampleError.updateFunctionConfigurationError
            }

            guard let retVariables = response.variables else {
                throw ExampleError.environmentVariablesMissingError
            }

            for envVar in retVariables {
                if envVar.key == "LOG_LEVEL" && envVar.value != "DEBUG" {
```

```
                    print("*** Log level is not set to DEBUG!")
                    throw ExampleError.logLevelIncorrectError
                }
            }
        } catch {
            throw ExampleError.updateFunctionConfigurationError
        }
    }
```

- API-Details finden Sie UpdateFunctionConfiguration in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon RDS unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon RDS Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.

- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "rds-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
```

```
        // from dependencies.
        .executableTarget(
            name: "rds-scenario",
            dependencies: [
                .product(name: "AWSRDS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Die Swift-Codedatei, `entry.swift`.

```
// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Relational Database Service (RDS).
//

import ArgumentParser
import Foundation
import AWSRDS

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"
    @Option(help: "The username to use for the database administrator.")
    var dbUsername = "admin"
    @Option(help: "The password to use for the database administrator.")
    var dbPassword: String

    static var configuration = CommandConfiguration(
        commandName: "rds-scenario",
        abstract: """
        Performs various operations to demonstrate the use of Amazon RDS Instances
        using the AWS SDK for Swift.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
```

```
        let example = try await Example(region: awsRegion, username: dbUsername,
 password: dbPassword)

        await example.run()
    }
}

class Example {
    let rdsClient: RDSClient

    // Storage for AWS RDS properties

    let dbUsername: String
    let dbPassword: String
    var dbInstanceIdentifier: String
    var dbSnapshotIdentifier: String
    var dbParameterGroupName: String
    var dbParameterGroup: RDSClientTypes.DBParameterGroup?
    var selectedEngineVersion: String?

    init(region: String, username: String, password: String) async throws{
        let rdsConfig = try await RDSClient.RDSClientConfiguration(region: region)
        rdsClient = RDSClient(config: rdsConfig)

        dbUsername = username
        dbPassword = password
        dbParameterGroupName = ""
        dbInstanceIdentifier = ""
        dbSnapshotIdentifier = ""
    }

    /// The example's main body.
    func run() async {
        var parameterGroupFamilies: Set<String> = []

        //=====================================================================
        // 1. Get available database engine families for MySQL.
        //=====================================================================

        let engineVersions = await getDBEngineVersions(engineName: "mysql")

        for version in engineVersions {
            if version.dbParameterGroupFamily != nil {
                parameterGroupFamilies.insert(version.dbParameterGroupFamily!)
```

```
            }
        }

        if engineVersions.count > 0 {
            selectedEngineVersion = engineVersions.last!.engineVersion
        } else {
            print("*** Unable to find a valid database engine version. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Found \(parameterGroupFamilies.count) parameter group families:")
        for family in parameterGroupFamilies {
            print("    \(family)")
        }

        //=====================================================================
        // 2. Select an engine family and create a custom DB parameter group.
        //    We select a family by sorting the set of family names, then
        //    choosing the last one.
        //=====================================================================

        let sortedFamilies = parameterGroupFamilies.sorted()

        guard let selectedFamily = sortedFamilies.last else {
            print("*** Unable to find a database engine family. Canceling
operations.")
            await cleanUp()
            return
        }

        print("Selected database engine family \(selectedFamily)")

        dbParameterGroupName = tempName(prefix: "rds-example")
        print("Creating a database parameter group named \(dbParameterGroupName)
using \(selectedFamily)")
        dbParameterGroup = await createDBParameterGroup(groupName:
dbParameterGroupName,
                                                        familyName: selectedFamily)

        //=====================================================================
        // 3. Get the parameter group's details.
        //=====================================================================
```

```
        print("Getting the database parameter group list...")
        let dbParameterGroupList = await describeDBParameterGroups(groupName:
dbParameterGroupName)
        guard let dbParameterGroupList else {
            await cleanUp()
            return
        }

        print("Found \(dbParameterGroupList.count) parameter groups...")
        for group in dbParameterGroupList {
            print("    \(group.dbParameterGroupName ?? "<unknown>")")
        }
        print()

        //=====================================================================
        // 4. Get a list of the parameter group's parameters. This list is
        //    likely to be long, so use pagination. Find the
        //    auto_increment_offset and auto_increment_increment parameters.
        //=====================================================================

        let parameters = await describeDBParameters(groupName: dbParameterGroupName)

        //=====================================================================
        // 5. Parse and display each parameter's name, description, and
        //    allowed values.
        //=====================================================================

        for parameter in parameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            if name == "auto_increment_offset" || name ==
"auto_increment_increment" {
                print("Parameter \(name):")
                print("          Value: \(parameter.parameterValue ??
"<undefined>")")
                print("      Data type: \(parameter.dataType ?? "<unknown>")")
                print("    Description: \(parameter.description ?? "")")
                print(" Allowed values: \(parameter.allowedValues ??
"<unspecified")")
```

```
                print(String(repeating: "=", count: 78))
            }
        }

        //=====================================================================
        // 6. Modify both the auto_increment_offset and
        //    auto_increment_increment parameters in one call in the custom
        //    parameter group. Set their parameterValue fields to a new
        //    permitted value.
        //=====================================================================

        print("Setting auto_increment_offset and auto_increment_increment both to
5...")
        await modifyDBParameters(groupName: dbParameterGroupName)

        //=====================================================================
        // 7. Get and display the updated parameters, specifying a source of
        //    "user" to get only the modified parameters.
        //=====================================================================

        let updatedParameters = await describeDBParameters(groupName:
dbParameterGroupName, source: "user")

        for parameter in updatedParameters {
            let name = parameter.parameterName
            guard let name else {
                print("*** Unable to get parameter name!")
                continue
            }

            print("Parameter \(name):")
            print("          Value: \(parameter.parameterValue ?? "<undefined>")")
            print("      Data type: \(parameter.dataType ?? "<unknown>")")
            print("    Description: \(parameter.description ?? "")")
            print(" Allowed values: \(parameter.allowedValues ?? "<unspecified")")
            print(String(repeating: "=", count: 78))
        }

        //=====================================================================
        // 8. Get a list of allowed engine versions using
        //    DescribeRDSEngineVersions.
        //=====================================================================

        await listAllowedEngines(family: selectedFamily)
```

```
        //=====================================================================
        // 9. Get a list of micro instance classes available for the selected
        //    engine and engine version.
        //=====================================================================

        let dbInstanceClass = await chooseMicroInstance(engine: "mysql",
engineVersion: selectedEngineVersion)
        guard let dbInstanceClass else {
            print("Did not get a valid instance class. Canceling operations.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 10. Create an RDS database that contains a MySQL database and uses
        //     the parameter group we created.
        //=====================================================================

        print("Creating the database instance...")

        guard let instanceClass = dbInstanceClass.dbInstanceClass else {
            print("Instance class name is unknown. Canceling operations.")
            await cleanUp()
            return
        }

        dbInstanceIdentifier = tempName(prefix: "sample-identifier")
        let dbInstanceArn = await createDBInstance(
            name: "SampleDatabase\(Int.random(in: 1000000..<1000000000))",
            instanceIdentifier: dbInstanceIdentifier,
            parameterGroupName: dbParameterGroupName,
            engine: "mysql",
            engineVersion: selectedEngineVersion!,
            instanceClass: instanceClass,
            username: dbUsername,
            password: dbPassword
        )

        if dbInstanceArn == nil {
            await cleanUp()
            return
        }
```

```
        //=====================================================================
        // 11. Wait for the database instance to be ready by calling
        //     DescribeDBInstances repeatedly until it reports
        //     dbInstanceStatus as "available". This can take upwards of 10
        //     minutes, let the user know that.
        //=====================================================================

        guard let endpoint = await waitUntilDBInstanceReady(instanceIdentifier:
dbInstanceIdentifier) else {
            print("\nDid not get a valid endpoint from AWS RDS.")
            await cleanUp()
            return
        }

        guard let endpointAddress = endpoint.address else {
            print("\nNo endpoint address returned.")
            await cleanUp()
            return
        }
        guard let endpointPort = endpoint.port else {
            print("\nNo endpoint port returned.")
            await cleanUp()
            return
        }

        //=====================================================================
        // 12. Display connection information for the database instance.
        //=====================================================================

        print("\nTo connect to the new database instance using 'mysql' from the
shell:")
        print("    mysql -h \(endpointAddress) -P \(endpointPort) -u
\(self.dbUsername)")

        //=====================================================================
        // 13. Create a snapshot of the database instance.
        //=====================================================================

        dbSnapshotIdentifier = tempName(prefix: "sample-snapshot")
        await createDBSnapshot(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        //=====================================================================
        // 14. Wait for the snapshot to be ready.
```

```
        //=====================================================================

        await waitUntilDBSnapshotReady(instanceIdentifier: dbInstanceIdentifier,
snapshotIdentifier: dbSnapshotIdentifier)

        // That's it! Clean up and exit!

        print("Example complete! Cleaning up...")
        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items.
    func cleanUp() async {
        print("Deleting the database instance \(dbInstanceIdentifier)...")
        await deleteDBInstance(instanceIdentifier: dbInstanceIdentifier)
        await waitUntilDBInstanceDeleted(instanceIdentifier: dbInstanceIdentifier)

        print("Deleting the database parameter group \(dbParameterGroupName)...")
        await deleteDBParameterGroup(groupName: dbParameterGroupName)
    }

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
[RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }

    /// Create a new database parameter group with the specified name.
```

```
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
    func describeDBParameterGroups(groupName: String) async ->
[RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
\(error.localizedDescription)")
            return nil
        }
```

```
    }

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
 [RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
 \(error.localizedDescription)")
            return []
        }

        return parameterList
    }

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
```

```
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]

        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Output a list of the database engine versions supported by the
    /// specified family.
    ///
    /// - Parameter family: The family for which to list allowed database
    ///   engines.
    func listAllowedEngines(family: String?) async {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    dbParameterGroupFamily: family,
                    engine: "mysql"
                )
            )
```

```
            guard let engineVersions = output.dbEngineVersions else {
                print("No engine versions returned.")
                return
            }

            print("Found \(engineVersions.count) database engine versions:")
            for version in engineVersions {
                print("    \(version.engineVersion ?? "<unknown>"):
\(version.dbEngineDescription ?? "")")
            }
        } catch {
            print("*** Error getting database engine version list:
\(error.localizedDescription)")
            return
        }
    }

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
    ///   - engineVersion: The database version for which to list instances.
    ///
    /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
    ///   the selected instance type.
    func chooseMicroInstance(engine: String = "mysql", engineVersion: String? = nil)
async -> RDSClientTypes.OrderableDBInstanceOption? {
        do {
            let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
                input: DescribeOrderableDBInstanceOptionsInput(
                    engine: engine,
                    engineVersion: engineVersion
                )
            )

            var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

            for try await page in pages {
                guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                    continue
                }
```

```
                for dbInstanceOption in orderableDBInstanceOptions {
                    guard let className = dbInstanceOption.dbInstanceClass else {
                        continue
                    }
                    if className.contains("micro") {
                        optionsList.append(dbInstanceOption)
                    }
                }
            }

            print("Found \(optionsList.count) database instances of 'micro' class
types:")
            for dbInstanceOption in optionsList {
                print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
            }

            return optionsList[0]
        } catch {
            print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
```

```
   /// - Returns: A string indicating the ARN of the newly created database
   ///   instance, or nil if the instance couldn't be created.
   func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                         engine: String, engineVersion: String, instanceClass:
String,
                         username: String, password: String) async -> String? {
       do {
           let output = try await rdsClient.createDBInstance(
               input: CreateDBInstanceInput(
                   allocatedStorage: 100,
                   dbInstanceClass: instanceClass,
                   dbInstanceIdentifier: instanceIdentifier,
                   dbName: name,
                   dbParameterGroupName: parameterGroupName,
                   engine: engine,
                   engineVersion: engineVersion,
                   masterUserPassword: password,
                   masterUsername: username,
                   storageType: "gp2"
               )
           )

           guard let dbInstance = output.dbInstance else {
               print("*** Unable to get the database instance.")
               return nil
           }

           return dbInstance.dbInstanceArn
       } catch {
           print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
           return nil
       }
   }

   /// Wait until the specified database is available to use.
   ///
   /// - Parameter instanceIdentifier: The database instance identifier of the
   ///   database to wait for.
   func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
RDSClientTypes.Endpoint? {
       do {
```

```
            putString("Waiting for the database instance to be ready to use. This
may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
```

```
   ///     snapshot.
   ///   - snapshotIdentifier: A unique identifier to give the newly-created
   ///     snapshot.
   func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
async {
       do {
           let output = try await rdsClient.createDBSnapshot(
               input: CreateDBSnapshotInput(
                   dbInstanceIdentifier: instanceIdentifier,
                   dbSnapshotIdentifier: snapshotIdentifier
               )
           )

           guard let snapshot = output.dbSnapshot else {
               print("No snapshot returned.")
               return
           }

           print("The snapshot has been created with ID \(snapshot.dbiResourceId ??
"<unknown>")")
       } catch {
           print("*** Unable to create the database snapshot named
\(snapshotIdentifier): \(error.localizedDescription)")
       }
   }

   /// Wait until the specified database snapshot is available to use.
   ///
   /// - Parameters:
   ///   - instanceIdentifier: The identifier of the database for which the
   ///     snapshot was taken.
   ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
   func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
String) async {
       var snapshotReady = false

       putString("Waiting for the snapshot to be ready...")

       do {
           while !snapshotReady {
               let output = try await rdsClient.describeDBSnapshots(
                   input: DescribeDBSnapshotsInput(
                       dbInstanceIdentifier: instanceIdentifier,
                       dbSnapshotIdentifier: snapshotIdentifier
```

```
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }

                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
```

```
        } catch {
            print("*** Error deleting the database instance \(instanceIdentifier): 
\(error.localizedDescription)")
        }
    }

    /// Wait until the specified database instance has been deleted.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to wait for.
    func waitUntilDBInstanceDeleted(instanceIdentifier: String) async {
        putString("Waiting for the database instance to be deleted. This may take a 
few minutes...")
        do {
            var isDatabaseDeleted = false
            var foundInstance = false

            while !isDatabaseDeleted {
                let output = try await rdsClient.describeDBInstances(input: 
DescribeDBInstancesInput())
                guard let instanceList = output.dbInstances else {
                    return
                }

                foundInstance = false

                for instance in instanceList {
                    guard let foundInstanceIdentifier = 
instance.dbInstanceIdentifier else {
                        continue
                    }

                    if instanceIdentifier == foundInstanceIdentifier {
                        foundInstance = true
                        break
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
```

```
                if !foundInstance {
                    isDatabaseDeleted = true
                    print()
                }
            }
        } catch {
            print("\n*** Error waiting for the database instance to be deleted:
\(error.localizedDescription)")
        }
    }

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
\(error.localizedDescription)")
        }
    }

    /// Generate and return a unique file name that begins with the specified
    /// string.
    ///
    /// - Parameters:
    ///   - prefix: Text to use at the beginning of the returned name.
    ///
    /// - Returns: A string containing a unique filename that begins with the
    ///   specified `prefix`.
    ///
    /// The returned name uses a random number between 1 million and 1 billion to
    /// provide reasonable certainty of uniqueness for the purposes of this
    /// example.
    func tempName(prefix: String) -> String {
        return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
    }

    /// Print a string to standard output without a trailing newline, and
```

```
    /// without buffering.
    ///
    /// - Parameter str: The string to output.
    func putString(_ str: String = "") {
        if str.length >= 1 {
            let data = str.data(using: .utf8)
            guard let data else {
                return
            }
            FileHandle.standardOutput.write(data)
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups

- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a new database instance.
    ///
    /// - Parameters:
    ///   - name: The name of the database to create.
    ///   - instanceIdentifier: The identifier to give the new database
    ///     instance.
    ///   - parameterGroupName: The name of the parameter group to associate
    ///     with the new database instance.
    ///   - engine: The database engine to use.
    ///   - engineVersion: The version of the database given by `engine` to
    ///     use.
    ///   - instanceClass: The memory and compute capacity of the database
    ///     instance, such as `db.m5.large``.
    ///   - username: The admin user's username to establish for the new
    ///     instance.
    ///   - password: The password to use for the specified user's access.
    ///
    /// - Returns: A string indicating the ARN of the newly created database
    ///   instance, or nil if the instance couldn't be created.
```

```
   func createDBInstance(name: String, instanceIdentifier: String,
parameterGroupName: String,
                         engine: String, engineVersion: String, instanceClass:
String,
                         username: String, password: String) async -> String? {
       do {
           let output = try await rdsClient.createDBInstance(
               input: CreateDBInstanceInput(
                   allocatedStorage: 100,
                   dbInstanceClass: instanceClass,
                   dbInstanceIdentifier: instanceIdentifier,
                   dbName: name,
                   dbParameterGroupName: parameterGroupName,
                   engine: engine,
                   engineVersion: engineVersion,
                   masterUserPassword: password,
                   masterUsername: username,
                   storageType: "gp2"
               )
           )

           guard let dbInstance = output.dbInstance else {
               print("*** Unable to get the database instance.")
               return nil
           }

           return dbInstance.dbInstanceArn
       } catch {
           print("*** An error occurred while creating the database instance:
\(error.localizedDescription)")
           return nil
       }
   }
```

- Weitere API-Informationen finden Sie unter CreateDBInstance in der API-Referenz zum AWS SDK für Swift.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a new database parameter group with the specified name.
    ///
    /// - Parameters:
    ///   - groupName: The name of the new parameter group.
    ///   - familyName: The name of the parameter group family.
    /// - Returns:
    func createDBParameterGroup(groupName: String, familyName: String) async ->
 RDSClientTypes.DBParameterGroup? {
        do {
            let output = try await rdsClient.createDBParameterGroup(
                input: CreateDBParameterGroupInput(
                    dbParameterGroupFamily: familyName,
                    dbParameterGroupName: groupName,
                    description: "Created using the AWS SDK for Swift"
                )
            )
            return output.dbParameterGroup
        } catch {
            print("*** Error creating the parameter group:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie CreateDBParameterGroup in der AWS SDK for Swift API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Create a snapshot of the specified name.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database instance to
    ///     snapshot.
    ///   - snapshotIdentifier: A unique identifier to give the newly-created
    ///     snapshot.
    func createDBSnapshot(instanceIdentifier: String, snapshotIdentifier: String)
 async {
        do {
            let output = try await rdsClient.createDBSnapshot(
                input: CreateDBSnapshotInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    dbSnapshotIdentifier: snapshotIdentifier
                )
            )

            guard let snapshot = output.dbSnapshot else {
                print("No snapshot returned.")
                return
            }

            print("The snapshot has been created with ID \(snapshot.dbiResourceId ??
 "<unknown>")")
        } catch {
            print("*** Unable to create the database snapshot named
 \(snapshotIdentifier): \(error.localizedDescription)")
        }
    }
```

- Weitere API-Informationen finden Sie unter CreateDBSnapshot in der API-Referenz zum AWS -SDK für Swift.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Delete the specified database instance.
    ///
    /// - Parameter instanceIdentifier: The identifier of the database
    ///   instance to delete.
    func deleteDBInstance(instanceIdentifier: String) async {
        do {
            _ = try await rdsClient.deleteDBInstance(
                input: DeleteDBInstanceInput(
                    dbInstanceIdentifier: instanceIdentifier,
                    deleteAutomatedBackups: true,
                    skipFinalSnapshot: true
                )
            )
        } catch {
            print("*** Error deleting the database instance \(instanceIdentifier):
 \(error.localizedDescription)")
        }
    }
```

- Weitere API-Informationen finden Sie unter DeleteDBInstance in der API-Referenz zum AWS SDK für Swift.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Delete the specified database parameter group.
    ///
    /// - Parameter groupName: The name of the parameter group to delete.
    func deleteDBParameterGroup(groupName: String) async {
        do {
            _ = try await rdsClient.deleteDBParameterGroup(
                input: DeleteDBParameterGroupInput(
                    dbParameterGroupName: groupName
                )
            )
        } catch {
            print("*** Error deleting the database parameter group \(groupName):
 \(error.localizedDescription)")
        }
    }
```

- API-Details finden Sie DeleteDBParameterGroup in der AWS SDK for Swift API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Get all the database engine versions available for the specified
    /// database engine.
    ///
    /// - Parameter engineName: The name of the database engine to query.
    ///
    /// - Returns: An array of `RDSClientTypes.DBEngineVersion` structures,
    ///   each describing one supported version of the specified database.
    func getDBEngineVersions(engineName: String) async ->
 [RDSClientTypes.DBEngineVersion] {
        do {
            let output = try await rdsClient.describeDBEngineVersions(
                input: DescribeDBEngineVersionsInput(
                    engine: engineName
                )
            )

            return output.dbEngineVersions ?? []
        } catch {
            return []
        }
    }
```

- API-Details finden Sie DescribeDBEngineVersions in der AWS SDK for Swift API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Wait until the specified database is available to use.
    ///
    /// - Parameter instanceIdentifier: The database instance identifier of the
    ///   database to wait for.
    func waitUntilDBInstanceReady(instanceIdentifier: String) async ->
 RDSClientTypes.Endpoint? {
        do {
            putString("Waiting for the database instance to be ready to use. This
 may take 10 minutes or more...")
            while true {
                let output = try await rdsClient.describeDBInstances(
                    input: DescribeDBInstancesInput(
                        dbInstanceIdentifier: instanceIdentifier
                    )
                )

                guard let instanceList = output.dbInstances else {
                    continue
                }

                for instance in instanceList {
                    let status = instance.dbInstanceStatus

                    guard let status else {
                        print("\nUnable to determine the status.")
                        continue
                    }

                    if status.contains("available") {
                        return instance.endpoint
                    } else {
                        putString(".")
```

```
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("*** Unable to wait until the database is ready:
\(error.localizedDescription)")
            return nil
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBInstances in der API-Referenz zum AWS SDK für Swift.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Get descriptions of the database parameter groups matching the given
    /// name.
    ///
    /// - Parameter groupName: The name of the parameter group to describe.
    ///
    /// - Returns: An array of [RDSClientTypes.DBParameterGroup] objects
    ///   describing the parameter group.
```

```
    func describeDBParameterGroups(groupName: String) async ->
 [RDSClientTypes.DBParameterGroup]? {
        do {
            let output = try await rdsClient.describeDBParameterGroups(
                input: DescribeDBParameterGroupsInput(
                    dbParameterGroupName: groupName
                )
            )
            return output.dbParameterGroups
        } catch {
            print("*** Error getting the database parameter group's details:
 \(error.localizedDescription)")
            return nil
        }
    }
```

- API-Details finden Sie DescribeDBParameterGroups in der AWS SDK for Swift API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Returns the detailed parameter list for the specified database
    /// parameter group.
    ///
    /// - Parameters:
    ///   - groupName: The name of the parameter group to return parameters for.
    ///   - source: The types of parameters to return (`user`, `system`, or
    ///     `engine-default`).
    ///
```

```
    /// - Returns: An array of `RdSClientTypes.Parameter` objects, each
    ///   describing one of the group's parameters.
    func describeDBParameters(groupName: String, source: String? = nil) async ->
[RDSClientTypes.Parameter] {
        var parameterList: [RDSClientTypes.Parameter] = []

        do {
            let pages = rdsClient.describeDBParametersPaginated(
                input: DescribeDBParametersInput(
                    dbParameterGroupName: groupName,
                    source: source
                )
            )

            for try await page in pages {
                guard let parameters = page.parameters else {
                    return []
                }

                parameterList += parameters
            }
        } catch {
            print("*** Error getting database parameters:
\(error.localizedDescription)")
            return []
        }

        return parameterList
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBParameters in der API-Referenz zum AWS SDK für Swift.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Wait until the specified database snapshot is available to use.
    ///
    /// - Parameters:
    ///   - instanceIdentifier: The identifier of the database for which the
    ///     snapshot was taken.
    ///   - snapshotIdentifier: The identifier of the snapshot to wait for.
    func waitUntilDBSnapshotReady(instanceIdentifier: String, snapshotIdentifier:
 String) async {
        var snapshotReady = false

        putString("Waiting for the snapshot to be ready...")

        do {
            while !snapshotReady {
                let output = try await rdsClient.describeDBSnapshots(
                    input: DescribeDBSnapshotsInput(
                        dbInstanceIdentifier: instanceIdentifier,
                        dbSnapshotIdentifier: snapshotIdentifier
                    )
                )

                guard let snapshotList = output.dbSnapshots else {
                    return
                }

                for snapshot in snapshotList {
                    guard let snapshotReadyStr = snapshot.status else {
                        return
                    }

                    if snapshotReadyStr.contains("available") {
                        snapshotReady = true
```

```
                        print()
                    } else {
                        putString(".")
                        do {
                            try await Task.sleep(for: .seconds(15))
                        } catch {
                            print("\n*** Error pausing the task!")
                        }
                    }
                }
            }
        } catch {
            print("\n*** Unable to wait for the database snapshot to be ready:
\(error.localizedDescription)")
        }
    }
```

- Weitere API-Informationen finden Sie unter DescribeDBSnapshots in der API-Referenz zum AWS SDK für Swift.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Print a list of available database instances with "micro" in the class
    /// name, then return one of them to be used by other code.
    ///
    /// - Parameters:
    ///   - engine: The database engine for which to list database instance
    ///     classes.
```

```
   ///   - engineVersion: The database version for which to list instances.
   ///
   /// - Returns: An `RDSClientTypes.OrderableDBInstanceOption` describing
   ///   the selected instance type.
   func chooseMicroInstance(engine: String = "mysql", engineVersion: String? = nil)
async -> RDSClientTypes.OrderableDBInstanceOption? {
       do {
           let pages = rdsClient.describeOrderableDBInstanceOptionsPaginated(
               input: DescribeOrderableDBInstanceOptionsInput(
                   engine: engine,
                   engineVersion: engineVersion
               )
           )

           var optionsList: [RDSClientTypes.OrderableDBInstanceOption] = []

           for try await page in pages {
               guard let orderableDBInstanceOptions =
page.orderableDBInstanceOptions else {
                   continue
               }

               for dbInstanceOption in orderableDBInstanceOptions {
                   guard let className = dbInstanceOption.dbInstanceClass else {
                       continue
                   }
                   if className.contains("micro") {
                       optionsList.append(dbInstanceOption)
                   }
               }
           }

           print("Found \(optionsList.count) database instances of 'micro' class
types:")
           for dbInstanceOption in optionsList {
               print("    \(dbInstanceOption.engine ??
"<unknown>") \(dbInstanceOption.engineVersion ?? "<unknown>")
(\(dbInstanceOption.dbInstanceClass ?? "<unknown class>"))")
           }

           return optionsList[0]
       } catch {
           print("*** Error getting a list of orderable instance options:
\(error.localizedDescription)")
```

```
            return nil
        }
    }
```

- API-Details finden Sie DescribeOrderableDBInstanceOptions in der AWS SDK for Swift API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSRDS

    /// Demonstrates modifying two of the specified database parameter group's
    /// parameters.
    ///
    /// - Parameter groupName: The name of the parameter group to change
    ///   parameters for.
    func modifyDBParameters(groupName: String) async {
        let parameter1 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_offset",
            parameterValue: "5"
        )
        let parameter2 = RDSClientTypes.Parameter(
            applyMethod: RDSClientTypes.ApplyMethod.immediate,
            parameterName: "auto_increment_increment",
            parameterValue: "5"
        )

        let parameterList = [parameter1, parameter2]
```

```
        do {
            _ = try await rdsClient.modifyDBParameterGroup(
                input: ModifyDBParameterGroupInput(
                    dbParameterGroupName: groupName,
                    parameters: parameterList
                )
            )

            print("Successfully modified the parameter group \(groupName).")
        } catch {
            print("*** Error modifying the parameter group \(groupName):
\(error.localizedDescription)")
        }
    }
```

- API-Details finden Sie ModifyDBParameterGroup in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon S3 unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon S3 Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Grundlagen
- Aktionen
- Szenarien

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
```

```
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async throws {
        do {
            configuration = try await S3Client.S3ClientConfiguration()
         //   configuration.region = "us-east-2" // Uncomment this to set the region
 programmatically.
            client = S3Client(config: configuration)
        }
        catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            throw error
        }
    }


    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the locationConstraint
 in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
```

```
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You may
wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
```

```
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data) async
throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
```

```
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
```

```
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
 data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }


    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
        }
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
```

```
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
"ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
```

```
            }


            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:
```

```
        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = try await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName, file:
uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName, to:
downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket: bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
            for name in fileList {
```

```
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CopyObject

- CreateBucket
- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Aktionen

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object."))
            throw error
```

```
        }
    }
```

- API-Details finden Sie CopyObject in der AWS SDK for Swift API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func createBucket(name: String) async throws {
        var input = CreateBucketInput(
            bucket: name
        )

        // For regions other than "us-east-1", you must set the locationConstraint
 in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
 S3ClientTypes.CreateBucketConfiguration(locationConstraint:
 S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
```

```
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You may
 wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
```

- API-Details finden Sie CreateBucket in der AWS SDK for Swift API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        do {
            _ = try await client.deleteBucket(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a bucket"))
            throw error
        }
    }
```

- API-Details finden Sie DeleteBucket in der AWS SDK for Swift API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Deleting a file."))
            throw error
        }
    }
```

- API-Details finden Sie DeleteObject in der AWS SDK for Swift API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
                quiet: true
            )
        )

        do {
            _ = try await client.deleteObjects(input: input)
        } catch {
            print("ERROR: deleteObjects:", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie DeleteObjects in der AWS SDK for Swift API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func downloadFile(bucket: String, key: String, to: String) async throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
 data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file."))
            throw error
        }
    }
```

```
import AWSS3

    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }
```

```
            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to read
data.")
            }

            return data
        }
        catch {
            print("ERROR: ", dump(error, name: "Reading a file."))
            throw error
        }
    }
```

- API-Details finden Sie GetObject in der AWS SDK for Swift API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    /// Return an array containing information about every available bucket.
    ///
    /// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
    ///   each bucket.
    public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
        return try await client.listBuckets(input: ListBucketsInput())
    }
```

- API-Details finden Sie ListBuckets in der AWS SDK for Swift API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3

    public func listBucketFiles(bucket: String) async throws -> [String] {
        do {
            let input = ListObjectsV2Input(
                bucket: bucket
            )

            // Use "Paginated" to get all the objects.
            // This lets the SDK handle the 'continuationToken' in
 "ListObjectsV2Output".
            let output = client.listObjectsV2Paginated(input: input)
            var names: [String] = []

            for try await page in output {
                guard let objList = page.contents else {
                    print("ERROR: listObjectsV2Paginated returned nil contents.")
                    continue
                }

                for obj in objList {
                    if let objName = obj.key {
                        names.append(objName)
                    }
                }
            }
```

```
            return names
        }
        catch {
            print("ERROR: ", dump(error, name: "Listing objects."))
            throw error
        }
    }
```

- API-Details finden Sie unter ListObjects V2 in der AWS SDK for Swift API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSS3
import Smithy

    public func uploadFile(bucket: String, key: String, file: String) async throws {
        let fileUrl = URL(fileURLWithPath: file)
        do {
            let fileData = try Data(contentsOf: fileUrl)
            let dataStream = ByteStream.data(fileData)

            let input = PutObjectInput(
                body: dataStream,
                bucket: bucket,
                key: key
            )

            _ = try await client.putObject(input: input)
        }
        catch {
```

```
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

```
import AWSS3
import Smithy

    public func createFile(bucket: String, key: String, withData data: Data) async
 throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.putObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Putting an object."))
            throw error
        }
    }
```

- API-Details finden Sie PutObject in der AWS SDK for Swift API-Referenz.

Szenarien

Herunterladen von Streams unbekannter Größe

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe aus einem Amazon S3-Objekt herunterladen.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Download a file from the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to get the file from.
    ///   - key: The name (or path) of the file to download from the bucket.
    ///   - destPath: The pathname on the local filesystem at which to store
    ///     the downloaded file.
    func downloadFile(bucket: String, key: String, destPath: String?) async throws {
        let fileURL: URL

        // If no destination path was provided, use the key as the name to use
        // for the file in the downloads folder.

        if destPath == nil {
            do {
                try fileURL = FileManager.default.url(
                    for: .downloadsDirectory,
                    in: .userDomainMask,
                    appropriateFor: URL(string: key),
                    create: true
                ).appendingPathComponent(key)
            } catch {
                throw TransferError.directoryError
            }
        } else {
```

```
            fileURL = URL(fileURLWithPath: destPath!)
        }

        let config = try await S3Client.S3ClientConfiguration(region: region)
        let s3Client = S3Client(config: config)

        // Create a `FileHandle` referencing the local destination. Then
        // create a `ByteStream` from that.

        FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
        let fileHandle = try FileHandle(forWritingTo: fileURL)

        // Download the file using `GetObject`.

        let getInput = GetObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            let getOutput = try await s3Client.getObject(input: getInput)

            guard let body = getOutput.body else {
                throw TransferError.downloadError("Error: No data returned for
download")
            }

            // If the body is returned as a `Data` object, write that to the
            // file. If it's a stream, read the stream chunk by chunk,
            // appending each chunk to the destination file.

            switch body {
            case .data:
                guard let data = try await body.readData() else {
                    throw TransferError.downloadError("Download error")
                }

                // Write the `Data` to the file.

                do {
                    try data.write(to: fileURL)
                } catch {
                    throw TransferError.writeError
```

```
                }
                break

            case .stream(let stream as ReadableStream):
                while (true) {
                    let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)
                    guard let chunk = chunk else {
                        break
                    }

                    // Write the chunk to the destination file.

                    do {
                        try fileHandle.write(contentsOf: chunk)
                    } catch {
                        throw TransferError.writeError
                    }
                }

                break
            default:
                throw TransferError.downloadError("Received data is unknown object
type")
            }
        } catch {
            throw TransferError.downloadError("Error downloading the file:
\(error)")
        }

        print("File downloaded to \(fileURL.path).")
    }
```

Stream unbekannter Größe hochladen

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe in ein Amazon S3-Objekt hochladen.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams


    /// Upload a file to the specified bucket.
    ///
    /// - Parameters:
    ///   - bucket: The Amazon S3 bucket name to store the file into.
    ///   - key: The name (or path) of the file to upload to in the `bucket`.
    ///   - sourcePath: The pathname on the local filesystem of the file to
    ///     upload.
    func uploadFile(sourcePath: String, bucket: String, key: String?) async throws {
        let fileURL: URL = URL(fileURLWithPath: sourcePath)
        let fileName: String

        // If no key was provided, use the last component of the filename.

        if key == nil {
            fileName = fileURL.lastPathComponent
        } else {
            fileName = key!
        }

        let s3Client = try await S3Client()

        // Create a FileHandle for the source file.

        let fileHandle = FileHandle(forReadingAtPath: sourcePath)
        guard let fileHandle = fileHandle else {
```

```
            throw TransferError.readError
        }

        // Create a byte stream to retrieve the file's contents. This uses the
        // Smithy FileStream and ByteStream types.

        let stream = FileStream(fileHandle: fileHandle)
        let body = ByteStream.stream(stream)

        // Create a `PutObjectInput` with the ByteStream as the body of the
        // request's data. The AWS SDK for Swift will handle sending the
        // entire file in chunks, regardless of its size.

        let putInput = PutObjectInput(
            body: body,
            bucket: bucket,
            key: fileName
        )

        do {
            _ = try await s3Client.putObject(input: putInput)
        } catch {
            throw TransferError.uploadError("Error uploading the file: \(error)")
        }

        print("File uploaded to \(fileURL.path).")
    }
```

Beispiele für Amazon SNS unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SNS.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Package.swift Akte.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
```

```
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Das Hauptprogramm von Swift.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sns-basics",
        abstract: """
        This example shows how to list all of your available Amazon SNS topics.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
```

```
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("   \(topic)")
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
```

```
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API-Details finden Sie ListTopics in der AWS SDK for Swift API-Referenz.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(name: name)
        )

        guard let arn = output.topicArn else {
            print("No topic ARN returned by Amazon SNS.")
            return
        }
```

- API-Details finden Sie CreateTopic in der AWS SDK for Swift API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: arn)
        )
```

- API-Details finden Sie DeleteTopic in der AWS SDK for Swift API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }
```

- API-Details finden Sie ListTopics in der AWS SDK for Swift API-Referenz.

`Veröffentlichen`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS
```

```
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.publish(
            input: PublishInput(
                message: message,
                topicArn: arn
            )
        )

        guard let messageId = output.messageId else {
            print("No message ID received from Amazon SNS.")
            return
        }

        print("Published message with ID \(messageId)")
```

- Weitere API-Informationen finden Sie unter Publish in der API-Referenz zum AWS -SDK für Swift.

`Abonnieren`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)
```

```
        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: email,
                protocol: "email",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

Abonnieren Sie eine Telefonnummer für ein Thema, um Benachrichtigungen per SMS zu erhalten.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        let output = try await snsClient.subscribe(
            input: SubscribeInput(
                endpoint: phone,
                protocol: "sms",
                returnSubscriptionArn: true,
                topicArn: arn
            )
        )

        guard let subscriptionArn = output.subscriptionArn else {
            print("No subscription ARN received from Amazon SNS.")
            return
        }

        print("Subscription \(subscriptionArn) created.")
```

- Weitere API-Informationen finden Sie unter Subscribe in der API-Referenz zum AWS -SDK für Swift.

`Abonnement beenden`

Das folgende Codebeispiel zeigt die Verwendung`Unsubscribe`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSNS

        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(
                subscriptionArn: arn
            )
        )

        print("Unsubscribed.")
```

- Weitere API-Informationen finden Sie unter Unsubscribe in der API-Referenz zum AWS -SDK für Swift.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
            str = readLine()
```

```
            if str != nil && str?.count == 0 {
                str = nil
            }
        }

        return str!
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt: prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Display a menu of options then request a selection.
    ///
    /// - Parameters:
    ///   - prompt: A prompt string to display before the menu.
    ///   - options: An array of strings giving the menu options.
    ///
    /// - Returns: The index number of the selected option or 0 if no item was
    ///   selected.
    func menuRequest(prompt: String, options: [String]) -> Int {
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
```

```
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]

    /// Ask the user to choose one of the attribute values to use as a filter.
    ///
    /// - Parameters:
    ///   - message: A message to display before the menu of values.
    ///   - attrValues: An array of strings giving the values to choose from.
    ///
    /// - Returns: The string corresponding to the selected option.
    func askForFilter(message: String, attrValues: [String]) -> String? {
        print(message)
        for (index, value) in attrValues.enumerated() {
            print("  [\(index)] \(value)")
        }
```

```
        var answer: Int?
        repeat {
            answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
        } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

        if answer == 0 {
            return nil
        }
        return attrValues[answer!]
    }

    /// Prompts the user for filter terms and constructs the attribute
    /// record that specifies them.
    ///
    /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
    ///   the user-defined filter.
    func buildFilterAttributes() -> [String:String] {
        var attr: [String:String] = [:]
        var filterString = ""

        var first = true

        while let ans = askForFilter(message: "Choose a value to apply to the 'tone'
attribute.",
                                     attrValues: attrValues) {
            if !first {
                filterString += ","
            }
            first = false

            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
```

```
   ///   - isFIFO: Whether or not to create a FIFO queue.
   ///
   /// - Returns: The URL of the queue.
   func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
       repeat {
           var queueName = stringRequest(prompt: prompt)
           var attributes: [String: String] = [:]

           if isFIFO {
               queueName += ".fifo"
               attributes["FifoQueue"] = "true"
           }

           do {
               let output = try await sqsClient.createQueue(
                   input: CreateQueueInput(
                       attributes: attributes,
                       queueName: queueName
                   )
               )
               guard let url = output.queueUrl else {
                   return nil
               }

               return url
           } catch _ as QueueDeletedRecently {
               print("You need to use a different queue name. A queue by that name
was recently deleted.")
               continue
           }
       } while true
   }

   /// Return the ARN of a queue given its URL.
   ///
   /// - Parameter queueUrl: The URL of the queue for which to return the
   ///   ARN.
   ///
   /// - Returns: The ARN of the specified queue.
   func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws -> String?
{
       let output = try await sqsClient.getQueueAttributes(
           input: GetQueueAttributesInput(
```

```
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
```

```
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async throws
                                -> [SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
```

```
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl: queueUrl)
        )

        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown ID>")
with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)
```

```
        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well as
              the queue subscriptions. You can then post messages to the topic and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/N)?
")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
                  supported.

                  Deduplication IDs are either set in the message or are
automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic, any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.
```

```
                For more information about deduplication, see:
                https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
                """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
```

```
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print("""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print("""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
```

```
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
messages will
                be received in the queue. For information about message filtering,
see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's subscription
to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
```

```
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil

        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
```

```
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a message
group ID for this message: ")

                if !isContentBasedDeduplication {
```

```
                    if first {
                        print("""
                              Because you're not using content-based deduplication,
you
                              must enter a deduplication ID. If other messages with
the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue] =
[:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType: "String",
stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes

            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false
```

```
            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )
```

```
        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen

- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Beispiele für Amazon SQS unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt, wie Sie mit Amazon SQS beginnen.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Die Datei `Package.swift`

```
import PackageDescription

let package = Package(
    name: "sqs-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or
 a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sqs-basics",
            dependencies: [
                .product(name: "AWSSQS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-parser")
            ],
            path: "Sources")

    ]
)
```

Der Swift-Quellcode, `entry.swift`.

```
import ArgumentParser
import AWSClientRuntime
import AWSSQS
```

```
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "sqs-basics",
        abstract: """
        This example shows how to list all of your available Amazon SQS queues.
        """,
        discussion: """
        """
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
            }
        }

        print("You have \(queues.count) queues:")
        for queue in queues {
            print("   \(queue)")
```

```
        }
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API-Details finden Sie ListQueues in der AWS SDK for Swift API-Referenz.

Aktionen

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.createQueue(
```

```
            input: CreateQueueInput(
                queueName: queueName
            )
        )

        guard let queueUrl = output.queueUrl else {
            print("No queue URL returned.")
            return
        }
```

- API-Details finden Sie CreateQueue in der AWS SDK for Swift API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        // Create the list of message entries.

        var entries: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []
        var messageNumber = 1

        for handle in handles {
            let entry = SQSClientTypes.DeleteMessageBatchRequestEntry(
                id: "\(messageNumber)",
                receiptHandle: handle
            )
            entries.append(entry)
            messageNumber += 1
```

```
        }

        // Delete the messages.

        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(
                entries: entries,
                queueUrl: queue
            )
        )

        // Get the lists of failed and successful deletions from the output.

        guard let failedEntries = output.failed else {
            print("Failed deletion list is missing!")
            return
        }
        guard let successfulEntries = output.successful else {
            print("Successful deletion list is missing!")
            return
        }

        // Display a list of the failed deletions along with their
        // corresponding explanation messages.

        if failedEntries.count != 0 {
            print("Failed deletions:")

            for entry in failedEntries {
                print("Message #\(entry.id ?? "<unknown>") failed:
\(entry.message ?? "<unknown>")")
            }
        } else {
            print("No failed deletions.")
        }

        // Output a list of the message numbers that were successfully deleted.

        if successfulEntries.count != 0 {
            var successes = ""

            for entry in successfulEntries {
                if successes.count == 0 {
                    successes = entry.id ?? "<unknown>"
```

```
                } else {
                    successes = "\(successes), \(entry.id ?? "<unknown>")"
                }
            }
            print("Succeeded: ", successes)
        } else {
            print("No successful deletions.")
        }
```

- API-Details finden Sie DeleteMessageBatch in der AWS SDK for Swift API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.deleteQueue(
                input: DeleteQueueInput(
                    queueUrl: queueUrl
                )
            )
        } catch _ as AWSSQS.QueueDoesNotExist {
            print("Error: The specified queue doesn't exist.")
            return
        }
```

- API-Details finden Sie DeleteQueue in der AWS SDK for Swift API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [
                    .approximatenumberofmessages,
                    .maximummessagesize
                ],
                queueUrl: url
            )
        )

        guard let attributes = output.attributes else {
            print("No queue attributes returned.")
            return
        }

        for (attr, value) in attributes {
            switch(attr) {
            case "ApproximateNumberOfMessages":
                print("Approximate message count: \(value)")
            case "MaximumMessageSize":
                print("Maximum message size: \(value)kB")
            default:
                continue
```

```
            }
        }
```

- API-Details finden Sie GetQueueAttributes in der AWS SDK for Swift API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        var queues: [String] = []
        let outputPages = sqsClient.listQueuesPaginated(
            input: ListQueuesInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let urls = output.queueUrls else {
                print("No queues found.")
                return
            }

            // Iterate over the queue URLs listed on this page, adding them
            // to the `queues` array.

            for queueUrl in urls {
                queues.append(queueUrl)
```

```
            }
        }
```

- API-Details finden Sie ListQueues in der AWS SDK for Swift API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: maxMessages,
                queueUrl: url
            )
        )

        guard let messages = output.messages else {
            print("No messages received.")
            return
        }

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print(message.body ?? "<body missing>")
            print("---")
        }
```

- API-Details finden Sie ReceiveMessage in der AWS SDK for Swift API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSQS

        let config = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: config)

        do {
            _ = try await sqsClient.setQueueAttributes(
                input: SetQueueAttributesInput(
                    attributes: [
                        "MaximumMessageSize": "\(maxSize)"
                    ],
                    queueUrl: url
                )
            )
        } catch _ as AWSSQS.InvalidAttributeValue {
            print("Invalid maximum message size: \(maxSize) kB.")
        }
```

- API-Details finden Sie SetQueueAttributes in der AWS SDK for Swift API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: """
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        """,
        discussion: """
        Supports filtering using a "tone" attribute.
        """
    )
```

```
/// Prompt for an input string. Only non-empty strings are allowed.
///
/// - Parameter prompt: The prompt to display.
///
/// - Returns: The string input by the user.
func stringRequest(prompt: String) -> String {
    var str: String?

    while str == nil {
        print(prompt, terminator: "")
        str = readLine()

        if str != nil && str?.count == 0 {
            str = nil
        }
    }

    return str!
}

/// Ask a yes/no question.
///
/// - Parameter prompt: A prompt string to print.
///
/// - Returns: `true` if the user answered "Y", otherwise `false`.
func yesNoRequest(prompt: String) -> Bool {
    while true {
        let answer = stringRequest(prompt: prompt).lowercased()
        if answer == "y" || answer == "n" {
            return answer == "y"
        }
    }
}

/// Display a menu of options then request a selection.
///
/// - Parameters:
///   - prompt: A prompt string to display before the menu.
///   - options: An array of strings giving the menu options.
///
/// - Returns: The index number of the selected option or 0 if no item was
///   selected.
func menuRequest(prompt: String, options: [String]) -> Int {
```

```
        let numOptions = options.count

        if numOptions == 0 {
            return 0
        }

        print(prompt)

        for (index, value) in options.enumerated() {
            print("(\(index)) \(value)")
        }

        repeat {
            print("Enter your selection (0 - \(numOptions-1)): ", terminator: "")
            if let answer = readLine() {
                guard let answer = Int(answer) else {
                    print("Please enter the number matching your selection.")
                    continue
                }

                if answer >= 0 && answer < numOptions {
                    return answer
                } else {
                    print("Please enter the number matching your selection.")
                }
            }
        } while true
    }

    /// Ask the user too press RETURN. Accepts any input but ignores it.
    ///
    /// - Parameter prompt: The text prompt to display.
    func returnRequest(prompt: String) {
        print(prompt, terminator: "")
        _ = readLine()
    }

    var attrValues = [
        "<none>",
        "cheerful",
        "funny",
        "serious",
        "sincere"
    ]
```

```
   /// Ask the user to choose one of the attribute values to use as a filter.
   ///
   /// - Parameters:
   ///   - message: A message to display before the menu of values.
   ///   - attrValues: An array of strings giving the values to choose from.
   ///
   /// - Returns: The string corresponding to the selected option.
   func askForFilter(message: String, attrValues: [String]) -> String? {
       print(message)
       for (index, value) in attrValues.enumerated() {
           print("  [\(index)] \(value)")
       }

       var answer: Int?
       repeat {
           answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
       } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

       if answer == 0 {
           return nil
       }
       return attrValues[answer!]
   }

   /// Prompts the user for filter terms and constructs the attribute
   /// record that specifies them.
   ///
   /// - Returns: A mapping of "FilterPolicy" to a JSON string representing
   ///   the user-defined filter.
   func buildFilterAttributes() -> [String:String] {
       var attr: [String:String] = [:]
       var filterString = ""

       var first = true

       while let ans = askForFilter(message: "Choose a value to apply to the 'tone'
attribute.",
                                    attrValues: attrValues) {
           if !first {
               filterString += ","
           }
           first = false
```

```
            filterString += "\"\(ans)\""
        }

        let filterJSON = "{ \"tone\": [\(filterString)]}"
        attr["FilterPolicy"] = filterJSON

        return attr
    }
    /// Create a queue, returning its URL string.
    ///
    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that name
was recently deleted.")
                continue
            }
```

```
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws -> String?
{
        let output = try await sqsClient.getQueueAttributes(
            input: GetQueueAttributesInput(
                attributeNames: [.queuearn],
                queueUrl: queueUrl
            )
        )

        guard let attributes = output.attributes else {
            return nil
        }

        return attributes["QueueArn"]
    }

    /// Applies the needed policy to the specified queue.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The queue to apply the policy to.
    ///   - queueArn: The ARN of the queue to apply the policy to.
    ///   - topicArn: The topic that should have access via the policy.
    ///
    /// - Throws: Errors from the SQS `SetQueueAttributes` action.
    func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                        queueArn: String, topicArn: String) async throws {
        _ = try await sqsClient.setQueueAttributes(
            input: SetQueueAttributesInput(
                attributes: [
                    "Policy":
                        """
                        {
                            "Statement": [
                                {
```

```
                                    "Effect": "Allow",
                                    "Principal": {
                                        "Service": "sns.amazonaws.com"
                                    },
                                    "Action": "sqs:SendMessage",
                                    "Resource": "\(queueArn)",
                                    "Condition": {
                                        "ArnEquals": {
                                            "aws:SourceArn": "\(topicArn)"
                                        }
                                    }
                                }
                            ]
                        }
                        """

                ],
                queueUrl: queueUrl
            )
        )
    }

    /// Receive the available messages on a queue, outputting them to the
    /// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
    /// all the received messages.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue on which to receive messages.
    ///
    /// - Throws: Errors from `SQSClient.receiveMessage()`
    ///
    /// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
    ///   items, each describing one received message in the format needed to
    ///   delete it.
    func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async throws
                                -> [SQSClientTypes.DeleteMessageBatchRequestEntry] {
        let output = try await sqsClient.receiveMessage(
            input: ReceiveMessageInput(
                maxNumberOfMessages: 10,
                queueUrl: queueUrl
            )
        )
```

```
        guard let messages = output.messages else {
            print("No messages received.")
            return []
        }

        var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

        // Print out all the messages that were received, including their
        // attributes, if any.

        for message in messages {
            print("Message ID:     \(message.messageId ?? "<unknown>")")
            print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
            print("Message JSON:   \(message.body ?? "<body missing>")")

            if message.receiptHandle != nil {
                deleteList.append(
                    SQSClientTypes.DeleteMessageBatchRequestEntry(
                        id: message.messageId,
                        receiptHandle: message.receiptHandle
                    )
                )
            }
        }

        return deleteList
    }

    /// Delete all the messages in the specified list.
    ///
    /// - Parameters:
    ///   - sqsClient: The Amazon SQS client to use.
    ///   - queueUrl: The SQS queue to delete messages from.
    ///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
    ///     describing the messages to delete.
    ///
    /// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
    func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                           deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
        let output = try await sqsClient.deleteMessageBatch(
            input: DeleteMessageBatchInput(entries: deleteList, queueUrl: queueUrl)
        )
```

```
        if let failed = output.failed {
            print("\(failed.count) errors occurred deleting messages from the
queue.")
            for message in failed {
                print("---> Failed to delete message \(message.id ?? "<unknown ID>")
with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
            }
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
              \(rowOfStars)
              Welcome to the cross-service messaging with topics and queues example.
              In this workflow, you'll create an SNS topic, then create two SQS
              queues which will be subscribed to that topic.

              You can specify several options for configuring the topic, as well as
              the queue subscriptions. You can then post messages to the topic and
              receive the results on the queues.
              \(rowOfStars)\n
              """
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region: region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //    to use content-based deduplication instead of requiring a
        //    deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/N)?
")
        var isContentBasedDeduplication = false

        if isFIFO {
```

```
            print("""
                  \(rowOfStars)
                  Because you've chosen to create a FIFO topic, deduplication is
                  supported.

                  Deduplication IDs are either set in the message or are
automatically
                  generated from the content using a hash function.

                  If a message is successfully published to an SNS FIFO topic, any
                  message published and found to have the same deduplication ID
                  (within a five-minute deduplication interval), is accepted but
                  not delivered.

                  For more information about deduplication, see:
                  https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.
                  """
            )

            isContentBasedDeduplication = yesNoRequest(
                prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
            print(rowOfStars)
        }

        var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

        // 2. Create the topic. Append ".fifo" to the name if FIFO was
        //    requested, and set the "FifoTopic" attribute to "true" if so as
        //    well. Set the "ContentBasedDeduplication" attribute to "true" if
        //    content-based deduplication was requested.

        if isFIFO {
            topicName += ".fifo"
        }

        print("Topic name: \(topicName)")

        var attributes = [
            "FifoTopic": (isFIFO ? "true" : "false")
        ]

        // If it's a FIFO topic with content-based deduplication, set the
```

```
        // "ContentBasedDeduplication" attribute.

        if isContentBasedDeduplication {
            attributes["ContentBasedDeduplication"] = "true"
        }

        // Create the topic and retrieve the ARN.

        let output = try await snsClient.createTopic(
            input: CreateTopicInput(
                attributes: attributes,
                name: topicName
            )
        )

        guard let topicArn = output.topicArn else {
            print("No topic ARN returned!")
            return
        }

        print(""""
              Topic '\(topicName) has been created with the
              topic ARN \(topicArn)."
              """
        )

        print(rowOfStars)

        // 3. Create an SQS queue. Append ".fifo" to the name if one of the
        //    FIFO topic configurations was chosen, and set "FifoQueue" to
        //    "true" if the topic is FIFO.

        print(""""
              Next, you will create two SQS queues that will be subscribed
              to the topic you just created.\n
              """
        )

        let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                          sqsClient: sqsClient, isFIFO: isFIFO)
        guard let q1Url else {
            print("Unable to create queue 1!")
            return
```

```
        }

        // 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

        let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

        guard let q1Arn else {
            print("Unable to get ARN of queue 1!")
            return
        }
        print("Got queue 1 ARN: \(q1Arn)")

        // 5. Attach an AWS IAM policy to the queue using
        //    `SetQueueAttributes`.

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                                 queueArn: q1Arn, topicArn: topicArn)

        // 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
        //    the request. Set the protocol to "sqs". Set the queue ARN to the
        //    ARN just received in step 5. For FIFO topics, give the option to
        //    apply a filter. A filter allows only matching messages to enter
        //    the queue.

        var q1Attributes: [String:String]? = nil

        if isFIFO {
            print(
                """

                If you add a filter to this subscription, then only the filtered
messages will
                be received in the queue. For information about message filtering,
see
                https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html
                For this example, you can filter messages by a 'tone' attribute.

                """
            )

            let subPrompt = """
                Would you like to filter messages for the first queue's subscription
to the
                topic \(topicName) (Y/N)?
```

```
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q1Attributes = buildFilterAttributes()
            }
        }

        let sub1Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q1Attributes,
                endpoint: q1Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 7. Repeat steps 3-6 for the second queue.

        let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                  sqsClient: sqsClient, isFIFO: isFIFO)

        guard let q2Url else {
            print("Unable to create queue 2!")
            return
        }

        let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

        guard let q2Arn else {
            print("Unable to get ARN of queue 2!")
            return
        }
        print("Got queue 2 ARN: \(q2Arn)")

        try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                                 queueArn: q2Arn, topicArn: topicArn)

        var q2Attributes: [String:String]? = nil
```

```
        if isFIFO {
            let subPrompt = """
                Would you like to filter messages for the second queue's
subscription to the
                topic \(topicName) (Y/N)?
                """
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //    body for each message. Handle the types of topic correctly (SEE
        //    MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\(rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.
```

```
            if isFIFO {
                if first {
                    print("""
                        Because you're using a FIFO topic, you must set a message
                        group ID. All messages within the same group will be
                        received in the same order in which they were published.\n
                        """
                    )
                }
                publishInput.messageGroupId = stringRequest(prompt: "Enter a message
group ID for this message: ")

                if !isContentBasedDeduplication {
                    if first {
                        print("""
                              Because you're not using content-based deduplication,
you
                              must enter a deduplication ID. If other messages with
the
                              same deduplication ID are published within the same
                              deduplication interval, they will not be delivered.
                              """
                        )
                    }
                    publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
                }
            }

            // Allow the user to add a value for the "tone" attribute if they
            // wish to do so.

            var messageAttributes: [String:SNSClientTypes.MessageAttributeValue] =
[:]
            let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

            if attrValSelection != 0 {
                let val = SNSClientTypes.MessageAttributeValue(dataType: "String",
stringValue: attrValues[attrValSelection])
                messageAttributes["tone"] = val
            }

            publishInput.messageAttributes = messageAttributes
```

```
            // Publish the message and display its ID.

            let publishOutput = try await snsClient.publish(input: publishInput)

            guard let messageID = publishOutput.messageId else {
                print("Unable to get the published message's ID!")
                return
            }

            print("Message published with ID \(messageID).")
            first = false

            // 9. Repeat step 8 until the user says they don't want to post
            //    another.

        } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

        // 10. Display a list of the messages in each queue by using
        //     `ReceiveMessage`. Show at least the body and the attributes.

        print(rowOfStars)
        print("Contents of queue 1:")
        let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
        print("\n\nContents of queue 2:")
        let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
        print(rowOfStars)

        returnRequest(prompt: "\nPress return to clean up: ")

        // 11. Delete the received messages using `DeleteMessageBatch`.

        print("Deleting the messages from queue 1...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
        print("\nDeleting the messages from queue 2...")
        try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

        // 12. Unsubscribe and delete both queues.

        print("\nUnsubscribing from queue 1...")
```

```
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

AWS STS Beispiele mit SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift with Aktionen ausführen und allgemeine Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt die Verwendung`AssumeRole`.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
import AWSSTS

    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- API-Details finden Sie AssumeRole in der AWS SDK for Swift API-Referenz.

Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für Swift

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe des AWS SDK für Swift mit Amazon Transcribe Streaming Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`StartStreamTranscription`

Das folgende Codebeispiel zeigt die Verwendung. `StartStreamTranscription`

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
```

```
                languageCode: TranscribeStreamingClientTypes.LanguageCode(rawValue:
 lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK for Swift API-Referenz.

Szenarien

Transkribieren einer Audiodatei

Das folgende Code-Beispiel zeigt, wie mit Amazon Transcribe-Streaming eine Transkription einer Quell-Audiodatei generiert wird.

SDK für Swift

> **Note**
>
> Es läuft noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Amazon Transcribe Streaming, um die gesprochene Sprache in eine Audiodatei zu transkribieren.

```
/// An example that demonstrates how to watch an transcribe event stream to
/// transcribe audio from a file to the console.

import ArgumentParser
import AWSClientRuntime
import AWSTranscribeStreaming
import Foundation

/// Identify one of the media file formats supported by Amazon Transcribe.
enum TranscribeFormat: String, ExpressibleByArgument {
    case ogg = "ogg"
    case pcm = "pcm"
    case flac = "flac"
}
```

```
// -MARK: - Async command line tool

struct ExampleCommand: ParsableCommand {
    // -MARK: Command arguments
    @Flag(help: "Show partial results")
    var showPartial = false
    @Option(help: "Language code to transcribe into")
    var lang: String = "en-US"
    @Option(help: "Format of the source audio file")
    var format: TranscribeFormat
    @Option(help: "Sample rate of the source audio file in Hertz")
    var sampleRate: Int = 16000
    @Option(help: "Path of the source audio file")
    var path: String
    @Option(help: "Name of the Amazon S3 Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "tsevents",
        abstract: """
        This example shows how to use event streaming with Amazon Transcribe.
        """,
        discussion: """
        """
    )

    /// Create and return an Amazon Transcribe audio stream from the file
    /// specified in the arguments.
    ///
    /// - Throws: Errors from `TranscribeError`.
    ///
    /// - Returns: `AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
 Error>`
    func createAudioStream() async throws
                -> AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
 Error> {

        let fileURL: URL = URL(fileURLWithPath: path)
        let audioData = try Data(contentsOf: fileURL)

        // Properties defining the size of audio chunks and the total size of
        // the audio file in bytes. You should try to send chunks that last on
        // average 125 milliseconds.
```

```
        let chunkSizeInMilliseconds = 125.0
        let chunkSize = Int(chunkSizeInMilliseconds  / 1000.0 * Double(sampleRate) *
2.0)
        let audioDataSize = audioData.count

        // Create an audio stream from the source data. The stream's job is
        // to send the audio in chunks to Amazon Transcribe as
        // `AudioStream.audioevent` events.

        let audioStream =
AsyncThrowingStream<TranscribeStreamingClientTypes.AudioStream,
                                Error> { continuation in
            Task {
                var currentStart = 0
                var currentEnd = min(chunkSize, audioDataSize - currentStart)

                // Generate and send chunks of audio data as `audioevent`
                // events until the entire file has been sent. Each event is
                // yielded to the SDK after being created.

                while currentStart < audioDataSize {
                    let dataChunk = audioData[currentStart ..< currentEnd]

                    let audioEvent =
TranscribeStreamingClientTypes.AudioStream.audioevent(
                        .init(audioChunk: dataChunk)
                    )
                    let yieldResult = continuation.yield(audioEvent)
                    switch yieldResult {
                        case .enqueued(_):
                            // The chunk was successfully enqueued into the
                            // stream. The `remaining` parameter estimates how
                            // much room is left in the queue, but is ignored here.
                            break
                        case .dropped(_):
                            // The chunk was dropped because the queue buffer
                            // is full. This will cause transcription errors.
                            print("Warning: Dropped audio! The transcription will be
incomplete.")
                        case .terminated:
                            print("Audio stream terminated.")
                            continuation.finish()
                            return
```

```
                        default:
                            print("Warning: Unrecognized response during audio
streaming.")
                    }

                    currentStart = currentEnd
                    currentEnd = min(currentStart + chunkSize, audioDataSize)
                }

                // Let the SDK's continuation block know the stream is over.

                continuation.finish()
            }
        }

        return audioStream
    }

    /// Run the transcription process.
    ///
    /// - Throws: An error from `TranscribeError`.
    func transcribe(encoding: TranscribeStreamingClientTypes.MediaEncoding) async
throws {
        // Create the Transcribe Streaming client.

        let client = TranscribeStreamingClient(
            config: try await
TranscribeStreamingClient.TranscribeStreamingClientConfiguration(
                region: region
            )
        )

        // Start the transcription running on the audio stream.

        let output = try await client.startStreamTranscription(
            input: StartStreamTranscriptionInput(
                audioStream: try await createAudioStream(),
                languageCode: TranscribeStreamingClientTypes.LanguageCode(rawValue:
lang),
                mediaEncoding: encoding,
                mediaSampleRateHertz: sampleRate
            )
        )
```

```
        // Iterate over the events in the returned transcript result stream.
        // Each `transcriptevent` contains a list of result fragments which
        // need to be concatenated together to build the final transcript.
        for try await event in output.transcriptResultStream! {
            switch event {
            case .transcriptevent(let event):
            for result in event.transcript?.results ?? [] {
                guard let transcript = result.alternatives?.first?.transcript else {
                    continue
                }

                // If showing partial results is enabled and the result is
                // partial, show it. Partial results may be incomplete, and
                // may be inaccurate, with upcoming audio making the
                // transcription complete or by giving more context to make
                // transcription make more sense.

                if (result.isPartial && showPartial) {
                    print("[Partial] \(transcript)")
                }

                // When the complete fragment of transcribed text is ready,
                // print it. This could just as easily be used to draw the
                // text as a subtitle over a playing video, though timing
                // would need to be managed.

                if !result.isPartial {
                    if (showPartial) {
                        print("[Final  ] ", terminator: "")
                    }
                    print(transcript)
                }
            }
            default:
                print("Error: Unexpected message from Amazon Transcribe:")
            }
        }
    }

    /// Convert the value of the `--format` command line option into the
    /// corresponding Transcribe Streaming `MediaEncoding` type.
    ///
    /// - Returns: The `MediaEncoding` equivalent of the format specified on
    ///   the command line.
```

```
    func getMediaEncoding() -> TranscribeStreamingClientTypes.MediaEncoding {
        let mediaEncoding: TranscribeStreamingClientTypes.MediaEncoding

        switch format {
        case .flac:
            mediaEncoding = .flac
        case .ogg:
            mediaEncoding = .oggOpus
        case .pcm:
            mediaEncoding = .pcm
        }

        return mediaEncoding
    }
}

// -MARK: - Entry point

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.transcribe(encoding: command.getMediaEncoding())
        } catch let error as TranscribeError {
            print("ERROR: \(error.errorDescription ?? "Unknown error")")
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

/// Errors thrown by the example's functions.
enum TranscribeError: Error {
    /// No transcription stream available.
    case noTranscriptionStream
    /// The source media file couldn't be read.
    case readError

    var errorDescription: String? {
        switch self {
```

```
        case .noTranscriptionStream:
            return "No transcription stream returned by Amazon Transcribe."
        case .readError:
            return "Unable to read the source audio file."
        }
    }
}
```

- API-Details finden Sie StartStreamTranscription in der AWS SDK for Swift API-Referenz.